

IBM[®] Net.Data



Manual de consulta

Versión 7

IBM[®] Net.Data



Manual de consulta

Versión 7

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general incluida en el apartado “Avisos” en la página 403. .

Este manual es la traducción del original inglés *IBM Net.Data Reference Version 7*.

Este documento contiene información sobre productos patentados de IBM. Se proporciona de acuerdo con un contrato de licencia y está protegido por la ley de la propiedad intelectual. La presente publicación no incluye garantías de producto y las declaraciones que contiene no deben interpretarse como tales.

Puede solicitar publicaciones a través del representante de IBM o sucursal de IBM de su localidad, o bien llamando a los números de teléfono 1-800-879-2755, en los Estados Unidos, o 1-800-IBM-4YOU, en Canadá.

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo para utilizar o distribuir dicha información en la forma en que IBM considere adecuada, sin contraer ninguna obligación con el remitente.

© Copyright International Business Machines Corporation 1997, 2000. Reservados todos los derechos.

Contenido

Prefacio	vii
Acerca de Net.Data	vii
Acerca de este manual	viii
A quién va dirigido este manual	ix
Acerca de los ejemplos de este manual	ix
Cómo leer los diagramas de sintaxis	ix

Capítulo 1. Construcciones de lenguaje de macro de Net.Data	1
Sintaxis de macros de Net.Data	1
Elementos de sintaxis comunes	5
Nombre de variable	5
Referencia variable	5
Series	7
Construcciones de lenguaje de macro	8
Bloque Comment	10
Bloque o sentencia DEFINE	12
Sentencia ENVVAR	17
Bloque o sentencia EXEC	18
Bloque FUNCTION	21
Llamada de función (@)	30
Bloque HTML	34
Bloque IF	37
Sentencia INCLUDE	44
Sentencia LIST	47
Bloque MACRO_FUNCTION	49
Bloque MESSAGE	54
Bloque REPORT	59
Bloque ROW	62
Sentencia TABLE	65
Bloque WHILE	67
Bloque XML	71

Capítulo 2. Variables	75
Variables definidas por el usuario	76
Variables condicionales	76
Variables de entorno	78
Variables ejecutables	78
Variables ocultas	79
Variables de lista	80
Variables de tabla	82
Variables Net.Data de proceso de tablas	83
Nn	84
NLIST	85
NUM_COLUMNS	86

NUM_ROWS	87
ROW_NUM	88
TOTAL_ROWS	89
V_Nombrecolumna	90
VLIST	91
Vn	92
Variables Net.Data de informes	93
ALIGN	94
DTW_DEFAULT_REPORT	95
DTW_HTML_TABLE	96
RPT_MAX_ROWS	97
START_ROW_NUM	99
Variables Net.Data de entorno de lenguaje	103
DATABASE	104
DB_CASE	106
DB2PLAN	107
DB2SSID	108
DTW_APPLET_ALTTEXT	109
DTW_EDIT_CODES	110
DTW_PAD_PGM_PARMS	112
DTW_SAVE_TABLE_IN	114
DTW_SET_TOTAL_ROWS	115
DTW_USE_DB2_PREPARE_CACHE	117
LOCATION	119
LOGIN	120
NULL_RPT_FIELD	121
PASSWORD	122
SHOWSQL	123
SQL_STATE	125
TRANSACTION_SCOPE	126
Variables Net.Data diversas	128
DTW_CURRENT_FILENAME	129
DTW_CURRENT_LAST_MODIFIED	130
DTW_DEFAULT_MESSAGE	131
DTW_LOG_LEVEL	132
DTW_MACRO_FILENAME	133
DTW_MACRO_LAST_MODIFIED	134
DTW_MBMODE	135
DTW_MP_PATH	137
DTW_MP_VERSION	138
DTW_PRINT_HEADER	139
DTW_REMOVE_WS	141
RETURN_CODE	142

Capítulo 3. Funciones incorporadas de Net.Data	143
Nombres de función	143
Parámetros de entrada y salida	144
Cómo dar formato a los resultados de función	144
Normas de parámetros de función	145
Funciones generales	145
DTW_ADDQUOTE	146
DTW_CACHE_PAGE	148
DTW_DATE	153
DTW_EXIT	155
DTW_GETCOOKIE	157
DTW_GETENV	160
DTW_GETINIDATA	162
DTW_HTMLENCODE	164
DTW_QHTMLENCODE	167
DTW_SENDEMAIL	169
DTW_SETCOOKIE	177
DTW_SETENV	181
DTW_TIME	183
DTW_URLESCSEQ	185
Funciones matemáticas	188
DTW_ADD	189
DTW_DIVIDE	191
DTW_DIVREM	193
DTW_FORMAT	195
DTW_INTDIV	199
DTW_MULTIPLY	201
DTW_POWER	203
DTW_SUBTRACT	205
Funciones de serie	207
DTW_ASSIGN	208
DTW_CHARTOHEX	209
DTW_CONCAT	211
DTW_DELSTR	213
DTW_HEXTOCHAR	215
DTW_INSERT	217
DTW_LASTPOS	220
DTW_LENGTH	222
DTW_LOWERCASE	224
DTW_POS	226
DTW_REPLACE	228
DTW_REVERSE	230
DTW_STRIP	232
DTW_SUBSTR	234
DTW_TRANSLATE	236
DTW_UPPERCASE	238
Funciones de texto	240
DTW_DELWORD	241

DTW_SUBWORD	243
DTW_WORD	245
DTW_WORDINDEX	247
DTW_WORDLENGTH	249
DTW_WORDPOS	251
DTW_WORDS	253
Funciones de tabla	255
DTW_TB_APPENDROW	256
DTW_TB_COLS	258
DTW_TB_DELETECOL	260
DTW_TB_DELETEROW	262
DTW_TB_DLIST	264
DTW_TB_DUMP	267
DTW_TB_DUMPV	269
DTW_TB_GETN	271
DTW_TB_GETV	273
DTW_TB_HTMLENCODE	275
DTW_TB_INPUT_CHECKBOX	277
DTW_TB_INPUT_RADIO	279
DTW_TB_INPUT_TEXT	281
DTW_TB_INSERTCOL	284
DTW_TB_INSERTROW	286
DTW_TB_LIST	288
DTW_TB_QUERYCOLNONJ	291
DTW_TB_ROWS	293
DTW_TB_SELECT	295
DTW_TB_SETCOLS	298
DTW_TB_SETN	300
DTW_TB_SETV	302
DTW_TB_TABLE	304
DTW_TB_TEXTAREA	307
Funciones de interfaz de archivo plano	309
Cómo acceder a las fuentes de datos de archivos planos	309
Delimitadores de la interfaz de archivo plano	312
Bloqueo de archivos	313
DTWF_APPEND	314
DTWF_CLOSE	318
DTWF_DELETE	320
DTWF_INSERT	323
DTWF_OPEN	326
DTWF_READ	329
DTWF_READFILE	333
DTWF_REMOVE	336
DTWF_SEARCH	338
DTWF_UPDATE	342
DTWF_WRITE	346
Funciones de registro Web	350
DTWR_ADDENTRY	351

DTWR_CLEARREG	353
DTWR_CLOSEREG	355
DTWR_CREATEREG.	357
DTWR_DELENTY	359
DTWR_DELREG	361
DTWR_LISTREG	363
DTWR_LISTSUB	365
DTWR_OPENREG	367
DTWR_RTVENTRY	369
DTWR_UPDATEENTRY	371
Funciones de macro permanentes	373
DTW_ACCEPT	374
DTW_COMMIT	376
DTW_ROLLBACK	377
DTW_RTVHANDLE.	378
DTW_STATIC	380
DTW_TERMINATE	382

Apéndice A. Biblioteca técnica de Net.Data	383
---	------------

Apéndice B. Características desaprobadas	385
---	------------

EXEC_SQL	385
HTML_INPUT	385
HTML_REPORT	385
INCLUDE_URL	385
SQL	385
SQL_MESSAGE	387
SQL_REPORT	387
SQL_CODE.	387

Apéndice C. Referencia del sistema operativo de Net.Data.	389
--	------------

Avisos	403
Marcas registradas	406

Glosario	409
---------------------------	------------

Índice	413
-------------------------	------------

Cómo ponerse en contacto con IBM	419
Información sobre productos	419

Prefacio

¡Gracias por seleccionar Net.Data[®], la herramienta de desarrollo de IBM[®] para crear páginas Web dinámicas! Con Net.Data puede desarrollar con rapidez páginas Web con un contenido dinámico incorporando datos procedentes de una gran variedad de fuentes de datos y utilizando la capacidad de los lenguajes de programación que ya conoce.

Acerca de Net.Data

Con el producto Net.Data de IBM, puede crear páginas Web dinámicas utilizando datos de sistemas de gestión de bases de datos relacionales y no relacionales (DBMS), incluyendo DB2, IMS, las bases de datos habilitadas por ODBC y las bases de datos a las que puede accederse a través de DRDA y utilizando las aplicaciones escritas en lenguajes de programación, como por ejemplo, Java, JavaScript, Perl, C, C++ y REXX. La familia de productos de Net.Data facilita posibilidades similares en las máquinas que ejecutan los sistemas operativos Windows NT, AIX, OS/2, OS/390, OS/400, HP-UX, PTX, Linux y Sun Solaris.

Net.Data es un macroprocesador que se ejecuta como middleware en una máquina servidora de la Web. Puede escribir programas de aplicación de Net.Data, denominadas *macros*, que Net.Data interpreta para crear páginas Web dinámicas con un contenido personalizado basado en entradas del usuario, el estado actual de las bases de datos, otras fuentes de datos, la lógica comercial existente y otros factores que se hayan diseñado en la macro.

Petición, con formato de ubicación uniforme de recursos (uniform resource locator - URL), va de un navegador, como por ejemplo, Netscape Navigator o Internet Explorer, a un servidor Web que remite la petición a Net.Data para su ejecución. Net.Data ubica y ejecuta la macro y crea una página Web que lo personaliza basándose en las funciones que escriba. Estas funciones pueden:

- Encapsular la lógica comercial en las aplicaciones escritas en los lenguajes de programación C, C++, RPG, COBOL, JAVA, Perl o REXX, pero sin limitarse a los mismos.
- Acceder a las bases de datos como por ejemplo DB2
- Acceder a las demás fuentes de datos como por ejemplo los archivos planos

Net.Data pasa esta página Web al servidor Web, que, a su vez, remite la página a través de la red para que se visualice en el navegador.

Net.Data puede utilizarse en entornos de servidor que están configurados para utilizar interfaces, como por ejemplo el Protocolo de transferencia de hipertexto (HyperText Transfer Protocol - HTTP) y la Interfaz de pasarela habitual (Common Gateway Interface - CGI). HTTP es una interfaz estándar en la industria para la interacción entre un navegador y un servidor Web y CGI es una interfaz estándar en la industria para que el servidor Web invoque las aplicaciones de pasarela como Net.Data. Estas interfaces le permiten seleccionar su navegador favorito o servidor Web para que se utilice con Net.Data.

Para un mejor rendimiento, Net.Data da soporte a una gran variedad de Interfaces de Programación de Aplicaciones (API) del servidor Web. Además, Net.Data puede iniciarse como servlet de Java.

Acerca de este manual

Este manual explica la sintaxis y utilización de las funciones, variables y construcciones de lenguaje de Net.Data.

Es posible que este manual haga referencia a productos o características que ya han sido anunciados pero que todavía no están disponibles.

Puede obtenerse más información, demos y macros de Net.Data y la última copia de este manual en los siguientes sitios de la World Wide Web:

- <http://www.ibm.com/software/data/net.data>
- <http://www.as400.ibm.com/netdata>

En este manual, hay tablas que contienen unas 'X' para indicar el sistema operativo en el que está disponible una determinada característica de Net.Data.

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X			X	X	X	

Las abreviaturas de la tabla representan los siguientes sistemas operativos.

AIX	IBM Advanced Interactive Executive
HP-UX	Hewlett Packard UNIX
Linux	Open Source Linux
OS/2	IBM OS/2
OS/390	IBM OS/390
OS/400	IBM OS/400

PTX	IBM/Sequent DYNIX/ptx
SUN	Sun Solaris
Win NT	Microsoft Windows NT

A quién va dirigido este manual

Las personas implicadas en planificar y escribir aplicaciones de Net.Data pueden utilizar la información de este manual para comprender las funciones, variables y construcciones de lenguaje que facilita Net.Data.

Para comprender los conceptos que se tratan en este manual, debería estar familiarizado con los servidores Web, sentencias de SQL simples y HTML (incluyendo la utilización de formatos de HTML) y con la información de la publicación *Net.Data Administration and Programming Guide*.

Acerca de los ejemplos de este manual

Los ejemplos que se utilizan en este manual están concebidos de modo sencillo para ilustrar conceptos específicos. No se pretende que los ejemplos muestren todas las formas en las que pueden utilizarse las construcciones de Net.Data. Del mismo modo, algunos de los ejemplos son fragmentos de código que no pueden ejecutarse por sí mismos.

Cómo leer los diagramas de sintaxis

Las normas siguientes se aplican a los diagramas de sintaxis utilizados en este manual:

- Lea los diagramas de sintaxis de izquierda a derecha, de arriba abajo, siguiendo la vía de acceso de la línea.

El símbolo ►— indica el principio de una sentencia.

El símbolo —► indica que la sintaxis de sentencia continúa en la línea siguiente.

El símbolo ►— indica que continúa una sentencia procedente de la línea anterior.

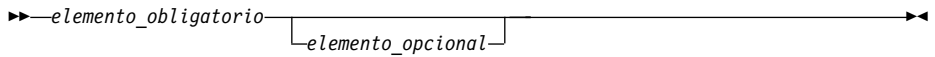
El símbolo —►◄ indica el final de una sentencia.

Los diagramas de unidades sintácticas que no sean sentencias completas comienzan por el símbolo ►— y finalizan con el símbolo —►.

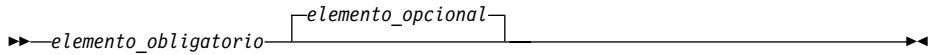
- Los elementos necesarios aparecen en la línea horizontal (la vía de acceso principal).

►—*elemento_obligatorio*—►◄

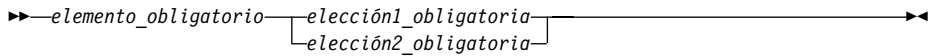
- Los elementos opcionales aparecen por debajo de la vía de acceso principal.



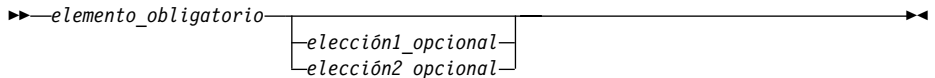
Si aparece un elemento opcional por encima de la vía de acceso principal, dicho elemento no tiene efecto sobre la ejecución de la sentencia y sólo se utiliza a efectos de legibilidad.



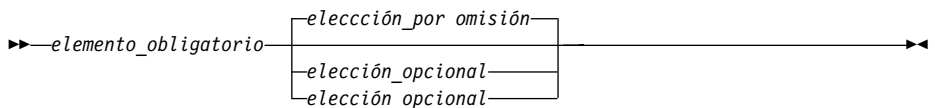
- Si puede optar entre dos o más elementos, aparecen verticalmente, en una pila.
Si *debe* optar por uno de los elementos, aparece un elemento de la pila en la vía de acceso principal.



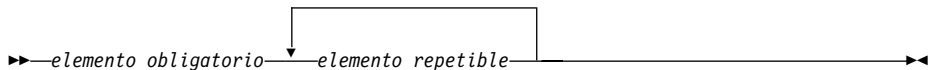
Si optar por uno de los elementos es opcional, toda la pila aparece por debajo de la vía de acceso principal.



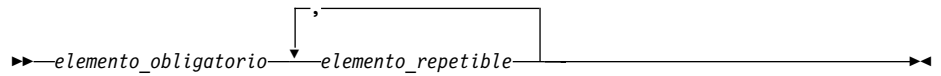
Si uno de los elementos es el valor por omisión, aparece por encima de la vía de acceso principal y las opciones restantes se muestran a continuación.



- Una flecha que vuelve hacia la izquierda, por encima de la línea principal, indica que puede repetirse un elemento.



Si la flecha de repetición contiene puntuación, debe separar los elementos repetidos con la puntuación especificada.



Una flecha de repetición por encima de una pila indica que puede repetir los elementos de la pila.

- Las palabras clave aparecen en mayúsculas (por ejemplo, FROM). En Net.Data, las palabras clave pueden estar en mayúsculas y minúsculas. Los términos que no son palabras clave aparecen en minúsculas (por ejemplo, *nombre-columna*). Representan valores o nombres proporcionados por el usuario.
- Si se muestran signos de puntuación, paréntesis, operadores aritméticos u otros símbolos parecidos, debe entrarlos como parte de la sintaxis.

Capítulo 1. Construcciones de lenguaje de macro de Net.Data

Este capítulo describe la sintaxis de macro de Net.Data y las construcciones de lenguaje utilizadas en la macro de Net.Data. Las construcciones de lenguaje constan de una palabra clave y de una sentencia o bloque en la macro de Net.Data, especifican tipos de variable diferentes y efectúan otras tareas especiales, por ejemplo la inclusión de archivos.

Este capítulo describe:

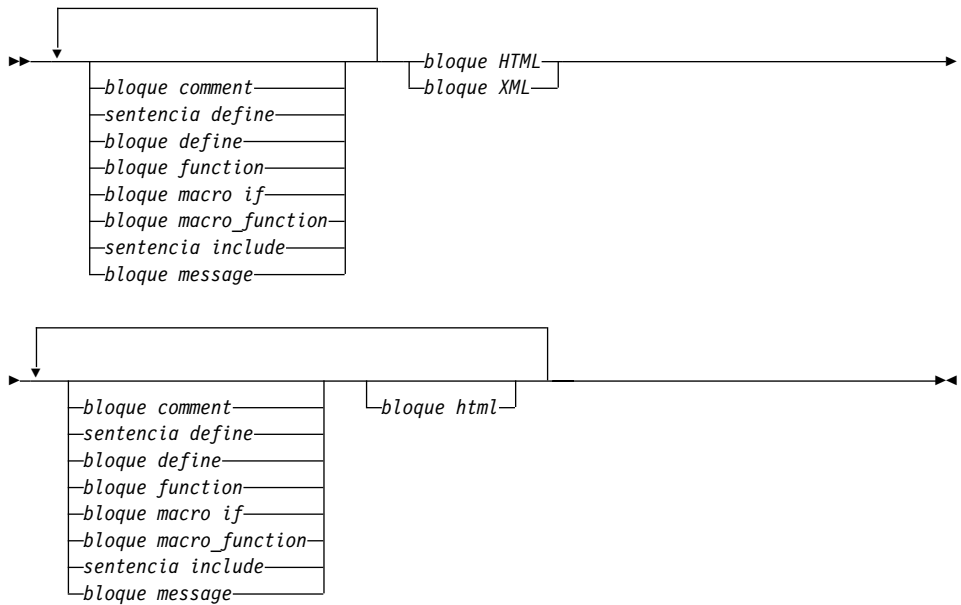
- “Sintaxis de macros de Net.Data”
- “Elementos de sintaxis comunes” en la página 5
- “Construcciones de lenguaje de macro” en la página 8

Sintaxis de macros de Net.Data

Una macro de Net.Data es un archivo de texto que consta de una serie de construcciones de lenguaje de macros de Net.Data que:

- Especifican el diseño de páginas Web
- Definen variables y funciones
- Llamam las funciones que están definidas en la macro o que Net.Data pasa a los entornos de lenguajes para el proceso

Cada sentencia se compone de una o más construcciones de lenguaje, que a su vez están compuestas de palabras clave, caracteres especiales, series, nombres y variables. El diagrama siguiente describe la estructura global de una macro de Net.Data sintácticamente válida. Consulte en el apartado “Construcciones de lenguaje de macro” en la página 8 una sintaxis detallada de cada uno de los elementos de la estructura global.



La macro Net.Data contiene dos partes: la parte de declaración y la parte de presentación. Puede utilizar estas partes varias veces y en cualquier orden.

- La *parte de declaración* contiene las definiciones de variables y funciones en la macro.
- La *parte de generación* contiene bloques HTML o XML que contienen sentencias que especifican el diseño del documento generado. Esta parte incluye la sección de informe.

La Figura 1 en la página 3 muestra las partes de declaración y presentación de la macro.

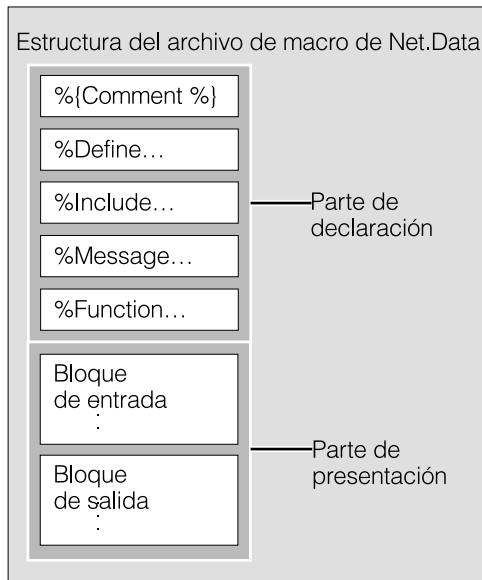


Figura 1. Estructura de macros:

Las variables y funciones que se utilizan en la parte de declaración o generación deben definirse antes de que las utilice una llamada de función o una referencia variable.

La Figura 2 en la página 4 muestra las partes de una macro. La parte de declaración contiene los bloques de definición `DEFINE` y `FUNCTION`. Los bloques HTML o XML actúan como bloques de entrada y salida.

```

%{ ***** Define block
*****%}
%DEFINE {
    page_title="Net.Data macro Template"
}%

%{ ***** Function Definition block
*****%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result)
    { %EXEC{ompsamp.cmd %}
}%

%FUNCTION(DTW_REXX) today () RETURNS(result)
    {
        result = date()
    }
}%

%{ ***** HTML Block: Input
*****%}
%HTML (INPUT) {
<html>
<head>
<title>$(page_title)<title>
</head><body>
<h1>Input Form</h1>
Today is @today()

<form method="post" action="output">
Type some data to pass to a REXX program:<br />
<input name="input_data" type="text" size="30" /><br />
<input type="submit" value="enter" /><br />
<hr />
<p>[<a href="/">Home page]
</form>
</body></html>
}%

%{ ***** HTML Block: Output
*****%}
%HTML (OUTPUT) {
<html>
<head>
<title>$(page_title)</title>
</head><body>
<h1>Output Page</h1>
<p>@rexx1(input_data)
<p><hr>
<p>[<a href="/">Home page</a> |
<a href="input">Previous page</a>]
</body></html>
}%

```

Figura 2. El formato de la plantilla de macros

El lenguaje de macros de Net.Data es un lenguaje de formato libre, proporcionándole la flexibilidad necesaria para escribir sus macros. A menos que se denote de modo específico, se ignoran los caracteres de espacio en blanco adicionales. En la sección siguiente se describe cada una de las construcciones de lenguaje de macros de Net.Data, junto con varios elementos adicionales que se utilizan para definir las construcciones. El lenguaje de macros de Net.Data da soporte a los elementos de lenguajes de DB2 WWW Connection para la compatibilidad hacia atrás. Aunque estos elementos de lenguaje se describen en el “Apéndice B. Características desaprobadas” en la página 385, resulta recomendable utilizar las construcciones de lenguaje de Net.Data.

Los ejemplos muestran algunas de las formas en que pueden utilizarse las construcciones de lenguaje, las variables, las funciones y otros elementos de las macros. Puede bajar los ejemplos y las demos de las páginas Web de Net.Data para obtener ejemplos más amplios:

- <http://www.ibm.com/software/data/net.data>
- <http://www.as400.ibm.com/netdata>

Elementos de sintaxis comunes

Los elementos de sintaxis comunes se utilizan con frecuencia en las descripciones de construcción de lenguaje:

- “Nombre de variable”
- “Referencia variable”
- “Series” en la página 7

Nombre de variable

Finalidad:

Identifica una variable. Una variable es un objeto cuyo valor puede cambiar durante la ejecución de una macro.

Los nombres de variable deben comenzar por una letra o subrayado (_) y deben contener caracteres alfanuméricos, subrayados, marcas hash (#) o puntos (.). Todos los nombres de variable son sensibles a mayúsculas y minúsculas excepto *N_Nombrecolumna* y *V_Nombrecolumna* (Consulte el apartado “Variables Net.Data de proceso de tablas” en la página 83 para obtener más información sobre estas dos excepciones.).

Referencia variable

Finalidad:

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si *VAR = 'front'*, \$(*VAR*) devuelve el valor 'front'. Las referencias variables se

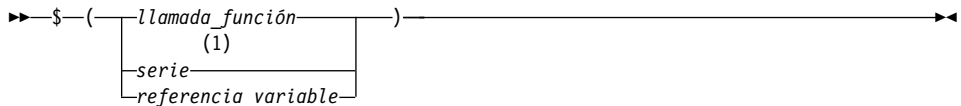
evalúan en el tiempo de ejecución. Cuando se define una variable para un bloque o sentencia EXEC, Net.Data ejecuta la acción especificada cuando lee la referencia variable.

Puede generar dinámicamente una referencia variable incluyendo referencias variables, series y llamadas de función dentro de una referencia de variable. Por ejemplo: si `frontside = 'blue'`, `$($(VAR)side)` devuelve el valor 'blue'. Si hace referencia a una variable generada dinámicamente que no siga las normas de variable, Net.Data resuelve la referencia en una serie vacía.

Restricciones::

- Las referencias variables no pueden utilizarse como parámetro OUT para una llamada de función.
- Se ignoran los espacios en blanco iniciales y de cola.
- No se admiten espacios en blanco (incluyendo los caracteres de nueva línea) entre llamadas de función, series y referencias variables.
- Una referencia variable con cualquier otro espacio en blanco devuelve una serie vacía.

Sintaxis:



Notas:

- 1 La serie sólo puede contener los caracteres que se permiten en nombres de variable: caracteres alfanuméricos, subrayados (`_`), marcas hash (`#`) o puntos (`.`).

Ejemplo 1: Referencia variable

Si ha definido una variable `homeURL`:

```
%DEFINE
homeURL="http://www.ibm.com/"
```

Puede hacer referencia a una página inicial como `$(homeURL)` y crear un enlace:

```
<a href="$(homeURL)">Home page</a>
```

Ejemplo 2: Referencia variable generada dinámicamente

Puede generar dinámicamente referencias variables que, a su vez, hacen referencia dinámicamente a un valor de campo en una fila:

```
%define{
var1="value1"
var2="value2"
var3="value3"
@DTW_ASSIGN (INDEX, "1")
%}
%WHILE (INDEX < 3) {
$(var$(INDEX))
@DTW_ADD(INDEX, "1", INDEX)
%}
```

Devuelve:

```
value1
value2
value3
```

Ejemplo 3: Una referencia variable dinámica con referencias variables anidadas y una llamada de función

```
%define my = "my"
%define u = "lower"
%define myLOWERvar = "hey"
```

```
$( $(my)@dtw_ruppercase(u)var)
```

La referencia variable devuelve el valor de hey.

Series

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación. Si la serie aparece entre dobles comillas, no se permite el carácter de línea nueva. Consulte la descripción de parámetro de serie en cada una de las construcciones de lenguaje para conocer las restricciones al utilizarse con la construcción de lenguaje.

Las series entre comillas (""), pueden contener cualquier carácter excepto el carácter de nueva línea. Si la serie está entre paréntesis, ({ %}), puede contener cualquier carácter, incluyendo el carácter de nueva línea. Por ejemplo,

```
%define multiline = {
first line
second line
%}
```

Para especificar dobles comillas dentro de una serie entre comillas, utilice dos pares de dobles comillas. Una serie utilizada como argumento de función o como término de una expresión de comparación puede contener dobles comillas. Por ejemplo, si define un valor de serie como:

```
%DEFINE result =  
" ""Hello world!"" "
```

El valor de *result* es:

```
"Hello world!"
```

Una sentencia HTML es una serie.

Las series que se utilizan como argumentos de función, términos y valores de variable pueden contener referencias variables y llamadas de función. En el ejemplo siguiente, la llamada de función `myfunc2` tiene un parámetro de serie que contiene una referencia variable y una llamada de función.

```
%html(report) {  
    @myfunc2("abc$(var1)@myfunc()")  
%}
```

`Net.Data` resuelve la referencia variable `$(var1)` y la llamada de función `@myfunc()`, en vez de interpretarlas literalmente como parte de serie, antes de pasar la serie a la función `myfunc2`.

Construcciones de lenguaje de macro

Este apartado describe las construcciones de lenguaje utilizadas en la macro de `Net.Data`.

Cada descripción de construcción de lenguaje puede contener la información siguiente:

Finalidad

Define el motivo por el que ha de utilizarse la construcción de lenguaje en la macro de `Net.Data`.

Sintaxis

Proporciona un diagrama de la estructura lógica de la construcción de lenguaje.

Parámetros

Define todos los elementos del diagrama de sintaxis y proporciona referencias cruzadas a otros ejemplos y sintaxis de construcciones de lenguaje.

Contexto

Explica el lugar en el que puede utilizarse la construcción de lenguaje de la estructura de macro de `Net.Data`.

Restricciones

Define los elementos que puede contener y especifica las restricciones de utilización.

Ejemplos

Proporciona ejemplos sencillos y explicaciones para utilizar el bloque o sentencia de palabra clave dentro de la macro de Net.Data.

Se utilizan las siguientes construcciones en la macro; por favor, consulte la sintaxis y ejemplos en la descripción de cada una de las construcciones.

- “Bloque Comment” en la página 10
- “Bloque o sentencia DEFINE” en la página 12
- “Sentencia ENVVAR” en la página 17
- “Bloque o sentencia EXEC” en la página 18
- “Bloque FUNCTION” en la página 21
- “Llamada de función (@)” en la página 30
- “Bloque HTML” en la página 34
- “Bloque IF” en la página 37
- “Sentencia INCLUDE” en la página 44
- “Sentencia LIST” en la página 47
- “Bloque MACRO_FUNCTION” en la página 49
- “Bloque MESSAGE” en la página 54
- “Bloque REPORT” en la página 59
- “Bloque ROW” en la página 62
- “Sentencia TABLE” en la página 65
- “Bloque WHILE” en la página 67
- “Bloque XML” en la página 71

Bloque Comment

Propósito

Documenta las funciones de la macro Net.Data. Puesto que el bloque COMMENT puede utilizarse en cualquier parte de la macro, no se documenta en los demás diagramas de sintaxis.

El bloque COMMENT puede utilizarse asimismo en el archivo de inicialización de Net.Data.

Sintaxis

►► `%{—texto—%}` —————►◄

Valores

texto Cualquier serie en una o más líneas. Net.Data ignora el contenido de todos los comentarios.

Contexto

Los comentarios pueden colocarse en cualquier parte entre las construcciones de lenguaje de una macro de Net.Data o del archivo de inicialización de Net.Data.

Restricciones

Se permite cualquier texto o carácter; sin embargo, los bloques comment no pueden anidarse.

Ejemplos

Ejemplo 1: Bloque comment básico

```
%{  
This is a comment block. It can contain any number of lines  
and contain any characters. Its contents are ignored by Net.Data.  
%}
```

Ejemplo 2: Comentarios en un bloque FUNCTION

```
%function(DTW_REXX) getAddress(IN name,   %{ customer name %}  
                                IN phone,  %{ customer phone number %}  
                                OUT address %{ customer address %}  
                                )  
  
{  
    ....  
%}
```

Ejemplo 3: Comentarios en un bloque HTML


```

%html(report) {

%{ run the query and save results in a table %}
@myQuery(resultTable)

%{ build a form to display a page of data %}
<form method="POST" action="report">

%{ send the table to a REXX function to send the data output %}
@displayRows(START_ROW_NUM, submit, resultTable, RPT_MAX_ROWS)

%{ pass START_ROW_NUM as a hidden variable to the next invocation %}
<input name="START_ROW_NUM" type="hidden" value="$(START_ROW_NUM)"
/>

%{ build the next and previous buttons %}
%if (submit == "both" || submit == "next_only")
  <input name="submit" type="submit" value="next" />
%endif
%if (submit == "both" || submit == "prev_only")
  <input name="submit" type="submit" value="previous" />
%endif
</form>
%}

```

Ejemplo 4: Comentarios en un bloque DEFINE

```

%define {
  START_ROW_NUM = "1"           %{ starting row number for output table %}
  RPT_MAX_ROWS = "25"           %{ maximum number of rows in the table %}
  resultTable = %table          %{ table to hold query results           %}
%}

```

Ejemplo 5: Comentarios en el archivo de inicialización de Net.Data.

```

%{ changes: removed RETURN_CODE parm and DTW_DEFAULT ENVIRONMENT
statement %}

...

ENVIRONMENT (DTW_SQL) dtwsq1 (IN LOCATION, DB2SSID, DB2PLAN, TRANSACTION_SCOPE)
ENVIRONMENT (DTW_ODBC) odbc11 (IN LOCATION, TRANSACTION_SCOPE)
ENVIRONMENT (DTW_PERL) perl11 ()
ENVIRONMENT (DTW_REXX) rexx11 ()
ENVIRONMENT (DTW_FILE) filed11 ()
ENVIRONMENT (DTW_APPLET) appld11 ()
ENVIRONMENT (DTW_SYSTEM) sysd11 ()

```

Bloque o sentencia DEFINE

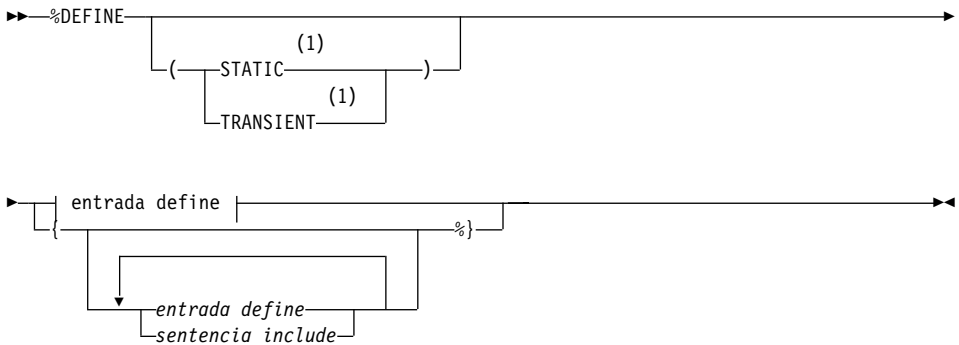
Propósito

El apartado DEFINE define nombres de variables en la parte de declaración de la macro y puede ser una sentencia o un bloque.

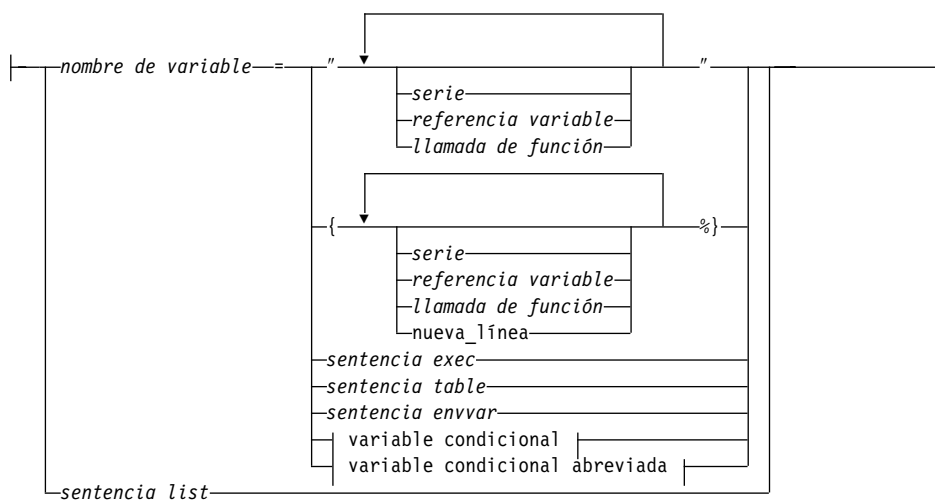
- Utilizar sentencias para definir una variable a la vez
- Utilizar bloques para definir varias variables

La definición de variable puede estar en una única línea, utilizando dobles comillas (""), o puede abarcar múltiples líneas, utilizando corchetes y un signo de porcentaje ({ %}). Después de que se defina la variable, puede hacer referencia a la misma en cualquier parte de la macro.

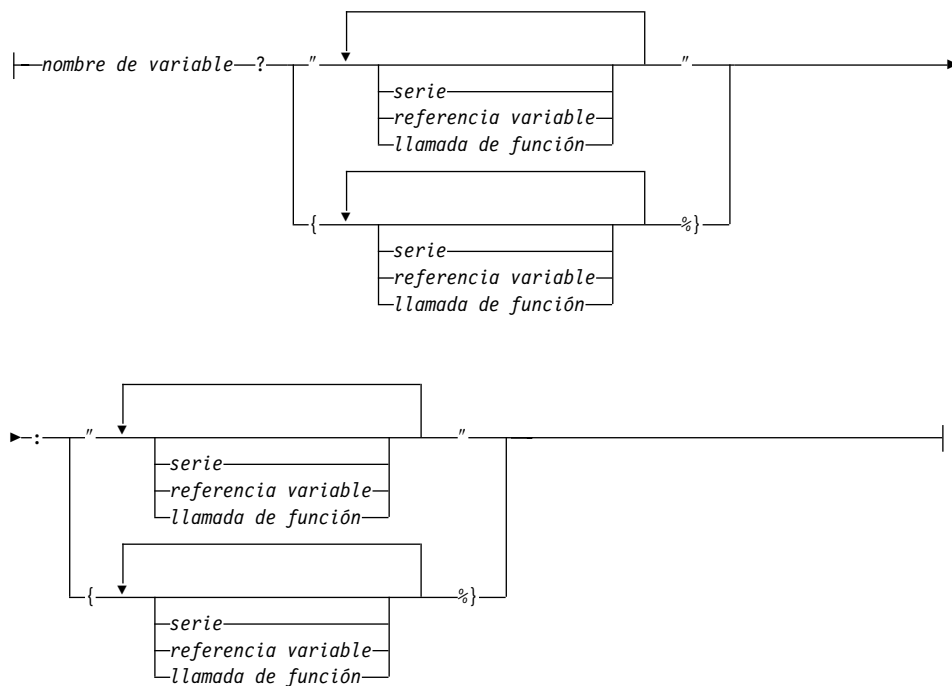
Sintaxis



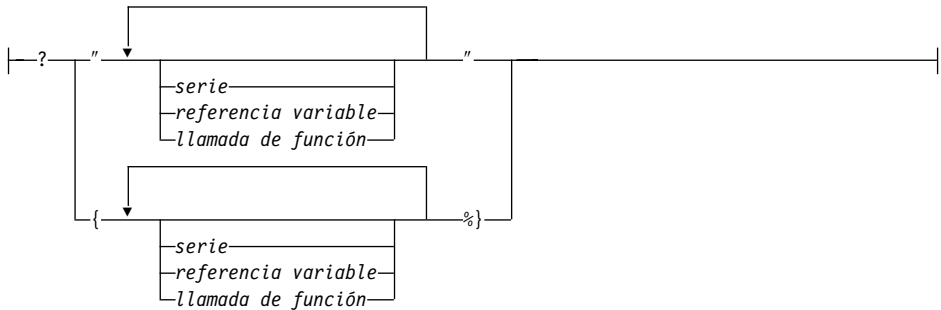
entrada define:



variable condicional:



variable condicional abreviada:



Notas:

- 1 **STATIC** y **TRANSIENT** son palabras clave para macros permanentes, que actualmente sólo están disponibles en el sistema operativo OS/400.

Valores

%DEFINE

Palabra clave que define variables.

STATIC

Palabra clave que especifica que la variable retenga su valor en las invocaciones de macro dentro de una transacción permanente. Este es el valor por omisión para las macros permanentes.

TRANSIENT

Palabra clave que especifica que esta variable no retenga su valor en las invocaciones de macro. Este es el valor por omisión para las macros no permanentes.

entrada define:

nombre de variable

Nombre que identifica una variable. Consulte el apartado “Nombre de variable” en la página 5 para obtener información de sintaxis.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación. Si la serie aparece entre dobles comillas, no se permite el carácter de línea nueva.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR='abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una función incorporada de Net.Data con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

sentencia exec

La sentencia EXEC. El nombre de un programa externo que se ejecuta cuando se hace referencia a una variable o se llama a una función. Consulte la sintaxis y ejemplos en el apartado “Bloque o sentencia EXEC” en la página 18.

sentencia table

La sentencia TABLE. Define un conjunto de datos relacionados que contiene una matriz de registros idénticos o filas y una matriz de nombres de columna que describe los campos en cada una de las filas. Consulte la sintaxis y ejemplos en el apartado “Sentencia TABLE” en la página 65.

sentencia envvar

La sentencia ENVVAR. Hace referencia a las variables de entorno. Consulte la sintaxis y ejemplos en el apartado “Sentencia ENVVAR” en la página 17.

variable condicional

Establece el valor de una variable basándose en si está o no vacía otra variable o serie.

variable condicional abreviada

Establece el valor de una variable basándose en si está o no vacía otra variable o serie. Formato abreviado de la variable condicional.

sentencia list

La sentencia LIST. Define variables que se utilizan para crear una lista delimitada de valores. Consulte la sintaxis y ejemplos en el apartado “Sentencia LIST” en la página 47.

sentencia include

La sentencia INCLUDE. Lee e incorpora un archivo en la macro de Net.Data. Consulte la sintaxis y ejemplos en el apartado “Sentencia INCLUDE” en la página 44.

Contexto

La sentencia o bloque DEFINE debe estar en un bloque IF o fuera de todos los demás bloques en la parte de declaración de la macro Net.Data.

Restricciones

- Puede contener los elementos siguientes:
 - Bloque Comment
 - Variables condicionales

- Sentencia LIST
 - Sentencia TABLE
 - Referencias variables
 - Sentencia INCLUDE
 - Sentencia EXEC
 - Llamadas de función
 - Sentencia ENVVAR
- No puede utilizar una variable en su propia definición. Por ejemplo, no se admite la definición de variable siguiente:

```
%DEFINE var = "The value is $(var)."
```

Ejemplos

Ejemplo 1: Definiciones de variable sencillas

```
%DEFINE var1 = "orders"
%DEFINE var2 = "$(var1).html"
```

Durante el tiempo de ejecución, la referencia variable `$(var2)` se evalúa como `orders.html`.

Ejemplo 2: Comillas dentro de una serie

```
%DEFINE hi = "say ""hello"""
%DEFINE empty = ""
```

Cuando se visualiza, la variable *hi* tiene el valor *say "hello"*. La variable *empty* contiene la serie vacía.

Ejemplo 3: Definición de múltiples variables

```
%DEFINE{ DATABASE = "testdb"
          home = "http://www.ibm.com/software"
          SHOWSQL = "YES"
          PI = "3.14150"
%}
```

Ejemplo 4: Definición de líneas múltiples de una variable

```
%DEFINE text = {This variable definition
                spans two lines
%}
```

Ejemplo 5: Este ejemplo de una variable condicional muestra el modo en que la variable `var` adopta el valor resultante dentro de los signos de comillas (") en el caso de que el valor resultante no contenga ningún valor NULL.

```
%DEFINE var = ? "Hello!"
$(V)@MyFunc() "
%}
```

Sentencia ENVVAR

Propósito

Define una variable como variable de entorno en el bloque DEFINE. Cuando se hace referencia a la variable ENVVAR, Net.Data devuelve el valor actual de la variable de entorno por medio del mismo nombre.

Sintaxis

►►—%ENVVAR—◄◄

Contexto

La sentencia ENVVAR puede estar en el bloque o sentencia DEFINE.

Valores

%ENVVAR

La palabra clave para definir una variable como variable de entorno en un bloque DEFINE. Esta variable obtiene el valor de una variable de entorno procedente de cualquier parte de la macro.

Restricciones

La sentencia ENVVAR no contiene ningún otro elemento.

Ejemplos

Ejemplo 1: En este ejemplo, ENVVAR define una variable, que cuando se hace referencia a la misma, devuelve el valor actual para la variable de entorno SERVER_SOFTWARE, el nombre del servidor de la Web.

```
%DEFINE SERVER_SOFTWARE = %ENVVAR
```

```
%HTML (REPORT){  
The server is $(SERVER_SOFTWARE).  
%}
```

Bloque o sentencia EXEC

Propósito

Especifica un programa externo que ha de ejecutarse cuando se hace referencia a una variable o se llama a una función.

Cuando Net.Data encuentra una variable ejecutable en una macro, busca el programa ejecutable al que se hace referencia utilizando el método siguiente:

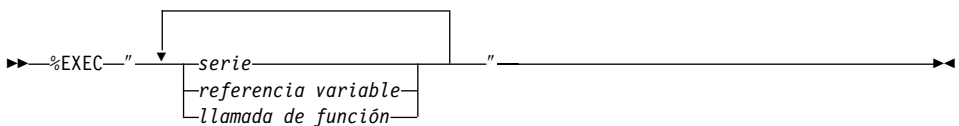
1. Busca la EXEC_PATH en el archivo de inicialización de Net.Data. Consulte el capítulo sobre configuración de la publicación *Guía de administración y programación de Net.Data* para el sistema operativo para obtener más información sobre EXEC_PATH.
2. Si Net.Data no localiza el programa, busca los directorios que define el sistema. Si localiza el programa ejecutable, Net.Data ejecuta el programa.

Sugerencia para la autorización: Asegúrese de que el ID de usuario bajo el que se ejecuta Net.Data tiene derechos de usuario para los archivos a los que hace referencia el bloque o sentencia EXEC. Consulte la sección sobre la especificación de derechos de acceso del servidor de la Web para archivos Net.Data en el capítulo de configuración de la publicación *Guía de administración y programación de Net.Data* para el sistema operativo para obtener más información.

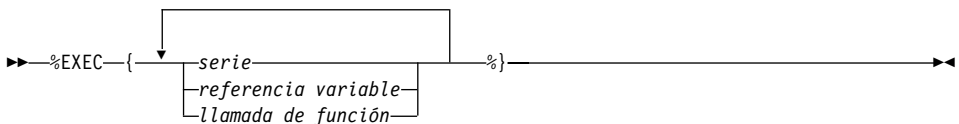
El bloque y sentencia EXEC se utilizan en dos contextos diferentes y tienen una sintaxis diferente, en función del lugar en el que se utilizan. Utilice la sentencia EXEC del bloque DEFINE y utilice el bloque EXEC en el bloque FUNCTION.

Sintaxis

La sintaxis de la sentencia EXEC cuando se utiliza en el bloque DEFINE:



La sintaxis del bloque EXEC cuando se utiliza en el bloque FUNCTION:



Valores

%EXEC

La palabra clave que especifica el nombre de un programa externo que ha de ejecutarse cuando se hace referencia a una variable o cuando se llama a una función. Cuando Net.Data encuentra una referencia variable que se define en una sentencia EXEC, procesa lo que la sentencia EXEC declara para la variable.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación. Si la serie aparece entre dobles comillas, no se permite el carácter de línea nueva.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR='abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una función incorporada de Net.Data con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

Contexto

La sentencia o bloque EXEC puede encontrarse en estos contextos:

- Bloque DEFINE
- Bloque FUNCTION

Restricciones

La sentencia o bloque EXEC puede contener estos elementos:

- Bloque Comment
- Serie
- Referencias variables
- Llamada de función

Los siguientes entornos de lenguaje proporcionados por Net.Data dan soporte a la sentencia EXEC:

- REXX
- Sistema
- Perl

Ejemplos

Ejemplo 1: Archivo ejecutable al que se hace referencia por medio de una variable

```
%DEFINE mycall = %EXEC "MYEXEC.EXE $(empno)"

%HTML (report){
  <p>Here is the report you requested:
  <hr>$(mycall)
%}
```

Este ejemplo ejecuta MYEXEC.EXE en cada una de las referencias a la variable, mycall.

Ejemplo 2: Archivo ejecutable al que se hace referencia por medio de una función

```
%FUNCTION(DTW_REXX) my_rexx_pgm(INOUT a, b, IN c, INOUT d){
  %EXEC{ mypgm.cmd this is a test %}
%}
```

Este ejemplo ejecuta mypgm.cmd cuando se llama a la función my_rexx_pgm.

Bloque FUNCTION

Propósito

Define una subrutina que Net.Data invoca desde la macro. Las sentencias ejecutables de un bloque FUNCTION pueden ser sentencias incorporadas que interprete directamente un entorno de lenguaje, o bien pueden ser una llamada a un programa externo.

Bloques EXEC en bloques de función: Si utiliza el bloque EXEC en el bloque FUNCTION, debe ser la única sentencia ejecutable del bloque FUNCTION. Antes de pasar la sentencia ejecutable al entorno de lenguaje, Net.Data agrega el nombre de archivo del programa del bloque EXEC a un nombre de vía de acceso determinado por medio de la sentencia de configuración de vía de acceso EXEC_PATH en el archivo de inicialización. La serie resultante se pasa al entorno de lenguaje que ha de ejecutarse.

El método que utiliza el entorno de lenguaje para procesar el bloque EXEC depende del entorno de lenguaje en concreto; los entornos de lenguaje REXX, Sistema y Perl que facilita Net.Data dan soporte al bloque EXEC.

Utilización de caracteres especiales en sentencias de lenguaje: Cuando se utilizan caracteres que coinciden con construcciones de lenguaje de Net.Data en la sección de sentencias de lenguaje de un bloque de función como parte del código de programa intercalado válido sintácticamente (por ejemplo, REXX o Perl), pueden interpretarse erróneamente como construcciones de lenguaje de Net.Data, ocasionando errores o un resultado imprevisible en una macro.

Por ejemplo, es posible que una función Perl utilice los caracteres de delimitador de bloque de COMMENT, `%{`. Cuando se ejecute la macro, los caracteres `%{` se interpretan como el principio de un bloque COMMENT. A continuación, Net.Data busca el final del bloque COMMENT, que piensa que ha encontrado cuando lee el final del bloque de función. Después Net.Data sigue buscando el final del bloque de función y cuando no lo encuentra emite un error.

Utilice uno de los métodos siguientes para utilizar caracteres especiales de Net.Data como parte del código de programa intercalado, sin que Net.Data los interprete como caracteres especiales:

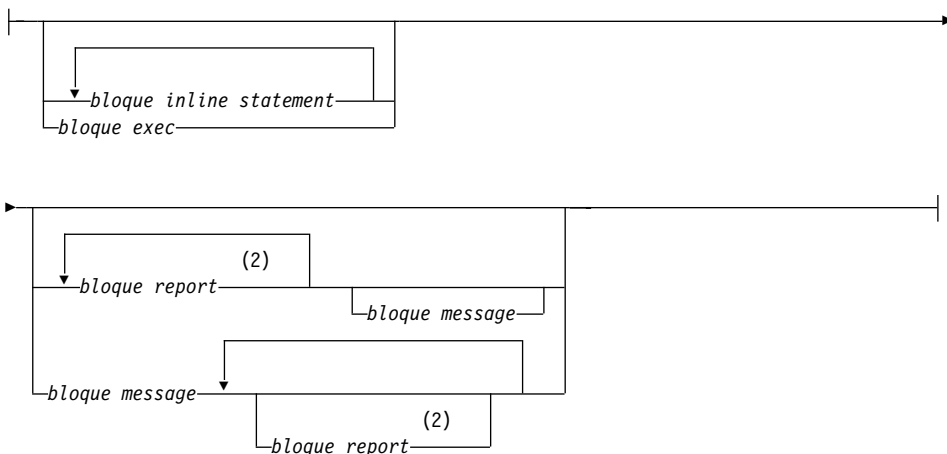
- Utilice la sentencia EXEC para llamar al código de programa, en vez de colocar el código incorporado.
- Utilice una referencia variable para especificar los caracteres especiales.

Por ejemplo, la función Perl siguiente contiene caracteres que representan a un delimitador de bloque COMMENT, `%{`, como parte de sus sentencias de lenguaje de Perl:

returns spec:



function body:



Notas:

- 1 El tipo de parámetro por omisión de IN se aplica cuando no se especifica ningún tipo de parámetro al principio de la lista de parámetros. Un parámetro sin un tipo de parámetro utiliza el tipo que se ha especificado más recientemente en la lista de parámetros, o graba IN si no se ha especificado ningún tipo. Por ejemplo, en la lista de parámetros (*parm1*, INOUT *parm2*, *parm3*, OUT *parm4*, *parm5*), los parámetros *parm1*, *parm3* y *parm5* no tienen tipos de parámetro. El parámetro *parm1* tiene un tipo de IN ya que no se ha especificado ningún tipo de parámetro inicial. El parámetro *parm3* tiene un tipo de INOUT ya que es el tipo de parámetro que se ha especificado más recientemente. De modo análogo, el parámetro *parm5* tiene un tipo de OUT ya que es el tipo de parámetro que se ha especificado más recientemente.
- 2 El bloque de informe repetido es válido para:
 - Los entornos de lenguaje de SQL y ODBC al procesar procedimientos almacenados que devuelven conjuntos de resultados múltiples para los sistemas operativos OS/390.
 - Las funciones que llaman cualquier entorno de lenguaje para los sistemas operativos OS/400, OS/2, Windows NT y UNIX.

Valores

%FUNCTION

La palabra clave que especifica una subrutina que Net.Data invoca desde la macro.

ent_leng

El entorno de lenguaje que procesa el cuerpo de la función. Consulte la publicación *Guía de administración y programación de Net.Data* para obtener más información.

nombre_función

El nombre de la función que se está definiendo puede ser una serie numérica o alfabética que comience por un carácter alfabético o un subrayado y contiene cualquier combinación de caracteres alfabéticos, numéricos o de subrayado.

parm passing spec:

IN Especifica que Net.Data pase datos de entrada al entorno de lenguaje. IN es el valor por omisión.

OUT

Especifica que el entorno de lenguaje devuelva datos de salida a Net.Data.

INOUT

Especifica que Net.Data pase datos de entrada al entorno de lenguaje y que éste devuelva datos de salida a Net.Data.

tipodatos

Especifica el tipo de datos del parámetro. Para obtener una lista de tipos de datos soportados para procedimientos almacenados, consulte el apéndice sobre el sistema operativo de la publicación *Manual de consulta de Net.Data*.

nombre

Serie numérica o alfabética que comienza por un carácter alfabético o un subrayado y contiene cualquier combinación de caracteres alfabéticos, numéricos o de subrayado.

returns spec:

RETURNS

Declara la variable que contiene el valor de función que asigna el entorno de lenguaje, una vez se complete la función.

function body:

bloque inline statement

Las sentencias sintácticamente válidas procedentes del entorno de lenguaje especificado en la definición de función, por ejemplo; REXX, SQL o Perl. Consulte en la publicación *Guía de administración y*

programación de Net.Data una descripción del entorno de lenguaje que está utilizando. Consulte la sintaxis y la utilización en el manual de consulta de programación del lenguaje de programación. La serie que representa el bloque inline statement puede contener funciones de llamada y referencias de variable de Net.Data, que se evalúan antes de ejecutar el bloque inline statement (programa).

exec block

El bloque EXEC. El nombre de un programa externo que se ejecuta cuando se llama a una función. Consulte la sintaxis y ejemplos en el apartado “Bloque o sentencia EXEC” en la página 18.

bloque report

El bloque REPORT. Instrucciones de formateo para la salida de una llamada de función. Puede utilizar información de cabecera y pie de página para el informe. Consulte la sintaxis y ejemplos en el apartado “Bloque REPORT” en la página 59.

bloque message

El bloque MESSAGE. Un conjunto de códigos de retorno, los mensajes asociados y las acciones que adopta Net.Data cuando se devuelve una llamada de función. Consulte la sintaxis y ejemplos en el apartado “Bloque MESSAGE” en la página 54.

Contexto

El bloque FUNCTION puede encontrarse en estos contextos:

- Bloque IF
- Fuera de cualquier bloque o sentencia de la parte de declaración de la macro de Net.Data.

Restricciones

- El bloque FUNCTION puede contener estos elementos:
 - Bloque Comment
 - Bloque EXEC
 - Bloque MESSAGE
 - Bloque REPORT
 - Bloques inline statement
- Las sentencias de SQL del bloque inline statement pueden tener las longitudes siguientes. Es posible que la base de datos tenga diferentes restricciones; consulte la documentación de la base de datos para determinar si la base de datos tiene una restricción más pequeña. A continuación se listan las restricciones de base de datos de IBM DB2, si son diferentes de los límites de Net.Data:
 - Para OS/2, Windows NT y UNIX: 64 KB
 - DB2 tiene las restricciones siguientes:
 - DB2 Universal Database V6 o posterior: 64 KB

- DB2 Universal Database V5.2 o anterior: 32 KB
- Para OS/390: 32 KB
- Para OS/400: 32 KB

Ejemplos

Los ejemplos siguientes son genéricos y no tratan sobre todos los entornos de lenguaje. Consulte la publicación *Net.Data Language Environment Reference* para obtener más información sobre la utilización de bloques FUNCTION con un entorno de lenguaje específico.

Ejemplo 1: Una función de subserie REXX

```
%DEFINE lstring = "longstring"
%FUNCTION(DTW_REXX) substring(IN x, y, z) RETURNS(s) {
  s = substr("$x", $(y), $(z));
}%
%DEFINE a = {@substring(lstring, "1", "4")}
%{ assigns "long" to a %}
```

Cuando se evalúa *a*, se busca la llamada de función *@substring* y se ejecuta el bloque FUNCTION de subserie. Las variables se sustituyen en las sentencias ejecutables en el bloque FUNCTION, a continuación, la serie de texto *s = substr("longstring", 1, 4)* se pasa al intérprete REXX para su ejecución. Puesto que se especifica la cláusula RETURNS, el valor de la llamada de función *@substring* en la evaluación de *a* se sustituye por "long", el valor de *s*.

Ejemplo 2: Invocación de un programa REXX externo

- Macro de Net.Data:

```
%FUNCTION(DTW_REXX)
my_rexx_pgm(INOUT a, b, IN c, OUT d) {
  %EXEC{ mypgm.cmd this is a test %}
}%
%HTML(INPUT) {
  <p> Original variable values: $(w) $(x) $(z)
  <p> @my_rexx_pgm(w, x, y, z)
  <p> Modified variable values: $(w) $(x) $(z)
}%
```

Las variables *w* y *x* se corresponden con los parámetros de INOUT *a* y *b* de la función. Sus valores y el valor de *y*, que se corresponde con el parámetro IN *c*, ya deberían haberse definido desde la entrada del formato HTML o desde una sentencia DEFINE. A las variables *a* y *b* se les asignan valores nuevos cuando los parámetros *a* y *b* devuelven valores. La variable *z* se define cuando el parámetro OUT *d* devuelve un valor.

- Programa REXX mypgm.cmd:


```

/* Sample REXX Program for Example 2 */
/* Test arguments */
num_args = arg();
say 'There are' num_args 'arguments';
do i = 1 to num_args;
    say 'arg' i 'is "'arg(i)'"';
end;
/* Set variables passed from Net.Data */
d = a || b || c; /* concatenate a, b, and c forming d */
a = '';          /* reset a to null string */
b = '';          /* reset b to null string */
return;

```

- Salida de mypgm.cmd:

```

There are 1 arguments
arg 1 is "this is a test"

```

La sentencia EXEC indica al entorno de lenguaje de REXX que indique al intérprete de REXX que ejecute el programa de REXX externo mypgm.cmd. Puesto que el entorno de lenguaje de REXX puede compartir directamente variables de Net.Data con el programa REXX, asigna a las variables de REXX *a*, *b* y *c* los valores de las variables de Net.Data *w*, *x* e *y* antes de ejecutar mypgm.cmd. mypgm.cmd puede utilizar directamente las variables *a*, *b* y *c* en sentencias REXX. Cuando finaliza el programa, las variables de REXX *a*, *b* y *d* se recuperan del programa REXX y sus valores se asignan a las variables de Net.Data *w*, *x* y *z*. Puesto que la cláusula RETURNS no se utiliza en la definición del bloque FUNCTION my_rexx_pgm, el valor de la llamada de función @my_rexx_pgm es la serie nula, "", (si el código de retorno es 0) o el valor del código de retorno del programa REXX (si el código de retorno es diferente a cero).

Ejemplo 3: Informe y consulta de SQL

```

%FUNCTION(DTW_SQL) query_1(IN x, IN y) {
    SELECT customer.num, order.num, part.num, status
    FROM customer, order, shippingpart
    WHERE customer.num = '$(x)'
        AND customer.ordernumber = order.num
        AND order.num = '$(y)'
        AND order.partnumber = part.num
    %REPORT{
        <p>Here is the status of your order:
        <p>$(NLIST)
        <ul>
        %ROW{
            <li>$(V1) $(V2) $(V3) $(V4)
        %}
        </ul>
        %}
    %}
%DEFINE customer_name="IBM"

```

```
%DEFINE customer_order="12345"
%HTML(REPORT) {
    @query_1(customer_name, customer_order)
%}
```

La llamada de función @query_1 sustituye IBM por \$(x) y 12345 por \$(y) en la sentencia SELECT. Puesto que la definición de la función de SQL query_1 no identifica una variable de tabla de salida, se utiliza la tabla por omisión (consulte el bloque de variables de TABLE para obtener más detalles). Las variables NLIST y Vi a las que se hace referencia en el bloque REPORT se definen por medio de la definición de tabla por omisión. El informe que se produce por medio del bloque REPORT se coloca en el HTML de salida en el que se invoca la función query_1.

Ejemplo 4: Una llamada de sistema para ejecutar un script de Perl

- Macro de Net.Data:

```
%FUNCTION(DTW_SYSTEM) today()
RETURNS(result) {
    %exec{ perl "today.pr1" %}
%}
%HTML(INPUT) {
    @today()
%}
```

- Programa Perl today.pr1:

```
$date = 'date';
chop $date;
open(DTW, "> $ENV{DTWPIPE}") || die "Could not open: $!";
print DTW "result = \"$date\"\n";
```

El entorno de lenguaje del Sistema interpreta las sentencias ejecutables de un bloque FUNCTION pasándolas al sistema operativo a través de la llamada de función del sistema de lenguaje C(). Este método no permite que las variables de Net.Data se recuperen o pasen directamente a las sentencias ejecutables, tal como hace el entorno de lenguaje de REXX, por lo que el entorno de lenguaje del Sistema pasa y recupera las variables tal y como se describe en este punto:

- Los parámetros de entrada se pasan como variables de entorno del sistema a través de la función putenv() y el programa de ejecución pueden recuperarlos. Diferentes lenguajes hacen referencia a las variables de modo diferente. Un script cshell de UNIX hace referencia a las variables de entorno colocando un '\$' delante del nombre de la variable de entorno, por ejemplo \$x. Un script de lenguaje Perl hace referencia a las mismas haciendo referencia a la matriz asociativa %ENV, por ejemplo %ENV{'x'}. Un archivo por lotes de DOS (.BAT) hace referencia al nombre de variable encerrado entre signos de porcentaje, por ejemplo, %x%.
- Los parámetros de salida se vuelven a pasar al entorno de lenguaje grabando en una conexión cuyo nombre se pasa en la variable de entorno DTWPIPE, excepto en la plataforma de OS/400, en la que los parámetros

de salida se pasan de nuevo al entorno de lenguaje como variables de entorno del sistema. Los datos que se graban en la conexión definida tiene el formato nombre="valor". Si un nombre de variable que se corresponde con un parámetro de salida se graba de este modo, el valor nuevo sustituye al valor actual. Si se graba un nombre de variable que no se corresponde con un parámetro de salida, éste se ignora.

Cuando se encuentra la llamada de función @today, Net.Data efectúa la sustitución de variables en las sentencias ejecutables. En este ejemplo, no hay variables de Net.Data en las sentencias ejecutables, por lo que no se efectúa ninguna sustitución de variables. Los parámetros y sentencias ejecutables se pasan al entorno de lenguaje de Sistema, que crea una conexión definida y establece la variable de entorno DTWPIPE en el nombre de la conexión.

A continuación, el programa externo se llama con la llamada de función del sistema() C. El programa externo abre la conexión como sólo de grabación y graba los valores de los parámetros de salida en la conexión como si fuera un archivo continuo estándar. El programa externo genera la salida HTML grabando en STDOUT. En este ejemplo, la salida del programa de fecha del sistema se ha asignado al resultado de la variable, que es la variable identificada en la cláusula RETURNS del bloque FUNCTION. Este valor de la variable de resultado sustituye la llamada de función @today() en el bloque HTML.

Ejemplo 5: Entorno de lenguaje Perl

```
%FUNCTION(DTW_PERL) today() RETURNS(result) {  
    $date = 'date';  
    chop $date;  
    open(DTW, "> $ENV{DTWPIPE}") || die "Could not open: $!";  
    print DTW "result = \"$date\"\n";  
}%  
%HTML(INPUT) {  
    @today()  
}%
```

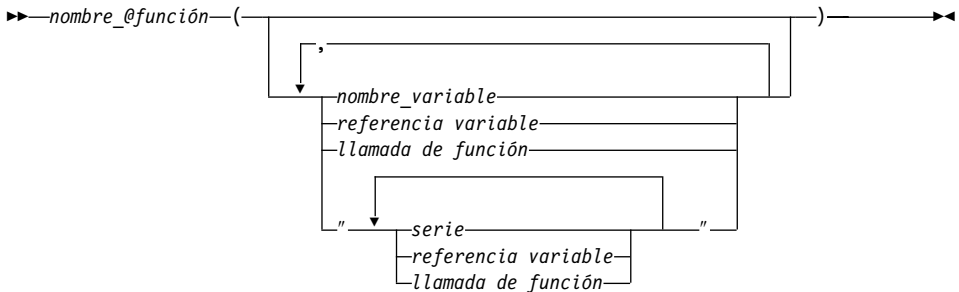
Compare este ejemplo con el Ejemplo 4 para ver el modo en que se utiliza el bloque EXEC. En el Ejemplo 4, el entorno de lenguaje del Sistema no sabe el modo en que ha de interpretar programas Perl, pero el entorno de lenguaje si sabe el modo de llamar programas externos. El bloque EXEC le indica que llame a un programa denominado perl como programa externo. El programa Perl externo interpreta las sentencias de lenguaje Perl reales. El ejemplo 5 no tiene un bloque EXEC, ya que el entorno de lenguaje Perl puede interpretar las sentencias de lenguaje Perl directamente.

Llamada de función (@)

Propósito

Invoca un bloque FUNCTION, bloque MACRO_FUNCTION o función incorporada con argumentos específicos. Si la función no es una función incorporada, debe definirla en la macro de Net.Data antes de especificar una llamada de función.

Sintaxis



Valores

nombre_@función

El nombre de cualquier función existente. Una serie numérica o alfabética que comienza por un carácter alfabético o un subrayado y contiene cualquier combinación de caracteres alfabéticos, numéricos o de subrayado.

nombre de variable

Nombre que identifica una variable. Consulte el apartado “Nombre de variable” en la página 5 para obtener información de sintaxis.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación, excepto el carácter de nueva línea.

referencia variable

Devuelve el valor de una variable y se especifica con `$` y `()`. Por ejemplo: si `VAR= 'abc'`, `$(VAR)` devuelve el valor `'abc'`. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una función incorporada de Net.Data con argumentos específicos.

Contexto

Las llamadas de función pueden encontrarse en estos contextos:

- Bloque HTML
- Bloque REPORT
- Bloque ROW
- Bloque DEFINE
- Bloque IF
- Bloque MACRO_FUNCTION
- Bloque MESSAGE
- Bloque WHILE
- Sentencia de llamada de función
- Fuera de cualquier bloque de la parte de declaración de la macro de Net.Data

Restricciones

- Las llamadas de función pueden contener estos elementos:
 - Bloque Comment
 - Series
 - Llamadas de función
 - Referencias variables
- Los valores de parámetro OUT o INOUT no pueden contener referencias variables, llamadas de función ni series literales.

Ejemplos

Ejemplo 1: Llamada a la función formQuery de SQL

```
%FUNCTION(DTW_SQL) formQuery(){
SELECT $(queryVal) from $(tableName)
%}

%HTML (input){
<p>Which columns of $(tableName) do you want to see?</p>
<form method="post" action="report">
<input name="queryval" type="checkbox" value="name" />Name
<input name="queryval" type="checkbox" value="mail" />E-mail
<input name="queryval" type="checkbox" value="fax" />FAX
<input type="submit" value="submit request" />
</form>
%}

%HTML (report){
<p>Here are the columns you selected:
<hr>@formQuery()
%}
```

Ejemplo 2: Llamada a una función REXX con parámetros de entrada y salida

```
%FUNCTION(DTW_REXX) my_rexx_pgm(INOUT a, b, IN c, OUT d) {
  %EXEC{ mypgm.cmd this is a test %}
%}
```

```
%HTML(INPUT) {
  <p> Original variable values: $(w) $(x) $(z)
  <p> @my_rexx_pgm(w, x, y, z)
  <p> Modified variable values: $(w) $(x) $(z)
%}
```

Ejemplo 3: Llamada a una función REXX con parámetros de entrada, que utiliza referencias variables y llamadas de función

```
%FUNCTION(DTW_REXX) my_rexx_pgm(IN a, b, c,
d, OUT e) {
  ...
%}
%HTML(INPUT) {
  <p> @my_rexx_pgm$(myA), @getB(), @retrieveC(), $(myD), myE)
%}
```

Ejemplo 4: Una macro que ilustra la utilización del parámetro INOUT.

```
%DEFINE a =
"initial value of a"

%FUNCTION(DTW_REXX) func1(INOUT x) {
  Say 'value at start of function:<br />'
  Say 'x =' x
  Say '<p>'
  x = "new value of a"
  %REPORT {
    <p>value at start of report block:<br />
    x = $(x)<br />
    @dtw_assign(x, "newest value of a")
    value at end of report block:<br />
    x = $(x)<br />
  %}
%}

%HTML(report) {
  initial values:<br />
  a = $(a)<br />
  @func1(a)
  value after function call:<br />
  a = $(a)<br />
%}
```

Salida resultante

```
initial
values:
a = initial value of a

value at start of function:
x = initial value of a
```

value at start of report block:
x = new value of a

value at end of report block:
x = newest value of a

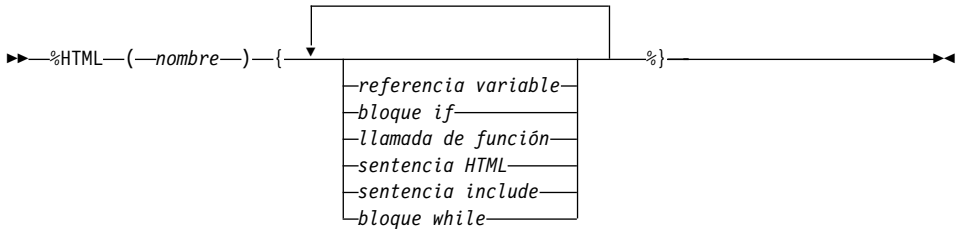
value after function call:
a = newest value of a

Bloque HTML

Propósito

Define el modo en que va a presentarse una página Web. El nombre del bloque HTML que ha de ejecutarse se especifica en el URL cuando se invoca Net.Data. El bloque HTML puede contener la mayoría de las sentencias de lenguaje de macro de Net.Data y las sentencias de presentación válidas, como por ejemplo, HTML y Javascript.

Sintaxis



Valores

%HTML

La palabra clave que especifica el bloque que contiene el texto y los códigos de HTML que han de visualizarse en el navegador del cliente.

nombre

Una serie numérica o alfabética que comienza por un carácter alfabético o un subrayado y que contiene cualquier combinación de caracteres alfabéticos, numéricos o de subrayado, incluyendo los puntos.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR= 'abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

bloque if

El bloque IF. Efectúa el proceso en serie condicional. Los valores de serie de la lista de condiciones se tratan como numéricos a efectos de comparación si son series que representan enteros y no tienen ningún espacio en blanco inicial o de cola. Pueden tener un único signo más (+) o menos (-) inicial. Consulte la sintaxis y ejemplos en el apartado “Bloque IF” en la página 37.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una

función incorporada de Net.Data con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

sentencias HTML

Incluye cualquier carácter alfabético o numérico, así como códigos HTML que han de formatearse para el navegador del cliente.

sentencia include

La sentencia INCLUDE. Lee e incorpora un archivo en la macro de Net.Data. Consulte la sintaxis y ejemplos en el apartado “Sentencia INCLUDE” en la página 44.

bloque while

El bloque WHILE. Efectúa la repetición en bucle con el proceso en serie condicional. Consulte la sintaxis y ejemplos en el apartado “Bloque WHILE” en la página 67.

Contexto

El bloque HTML puede encontrarse en estos contextos:

- Bloque IF
- Fuera de cualquier bloque de la parte de declaración de la macro de Net.Data

Restricciones

El bloque HTML puede contener estos elementos:

- Bloque Comment
- Bloque IF
- Sentencias HTML
- Sentencia INCLUDE
- Bloque WHILE
- Referencias variables
- Llamadas de función

Ejemplos

Ejemplo 1: Bloque HTML con archivos include para cabeceras y pies de página

```
%HTML(example1){  
%INCLUDE "header.html"  
<p>You can put <em>any</em> HTML in an HTML block.  
An SQL function call is made like this:  
@xmpl()  
%INCLUDE "footer.html"  
%}
```

Ejemplo 2: Bloque HTML con un nombre que contenga un punto

```
%HTML(my.report){  
%INCLUDE "header.html"  
<p>You can put <em>any</em> HTML in an HTML block.  
An SQL function call is made like this:  
@xmpl()  
%INCLUDE "footer.html"  
%}
```

Bloque IF

Propósito

Efectúa el proceso en serie condicional. El bloque IF proporciona la capacidad de probar una o más condiciones y efectuar a continuación un bloque de sentencias basadas en el resultado de la prueba de las condiciones. Puede utilizar el bloque IF de la parte de declaración de una macro de Net.Data, el bloque HTML, el bloque MACRO_FUNCTION, el bloque REPORT, el bloque WHILE y el bloque ROW, así como anidarlos dentro de otro bloque IF.

Los valores de serie de la lista de condiciones se tratan como numéricos a efectos de comparación si son series que representan enteros y no tienen ningún espacio en blanco inicial o de cola. Pueden tener un único signo más (+) o menos (-) inicial.

Restricción: Net.Data no da soporte a la comparación numérica de los números no enteros; por ejemplo, los números de coma flotante.

Bloques IF anidados: Las normas para la sintaxis del bloque IF las determina la posición del bloque en la macro. Si se anida un bloque IF dentro de un bloque IF que esté fuera de cualquier otro bloque de la parte de declaración, puede utilizar cualquier elemento que pueda utilizar el bloque externo. Si se anida un bloque IF dentro de otro bloque que esté en un bloque IF, adopta las normas de sintaxis del bloque en el que está.

En el ejemplo siguiente, el bloque IF anidado debe seguir las normas utilizadas cuando está dentro de un bloque HTML.

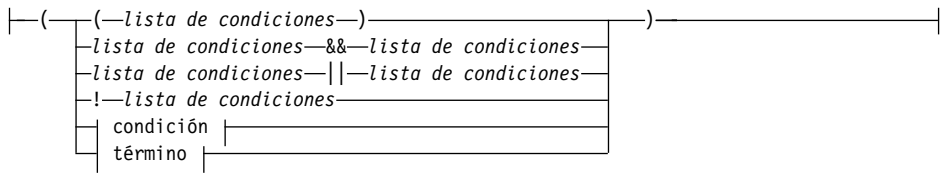
```
%IF block
...
  %HTML block
  ...
    %IF block
```

Puede anidar un máximo de 1024 bloques IF.

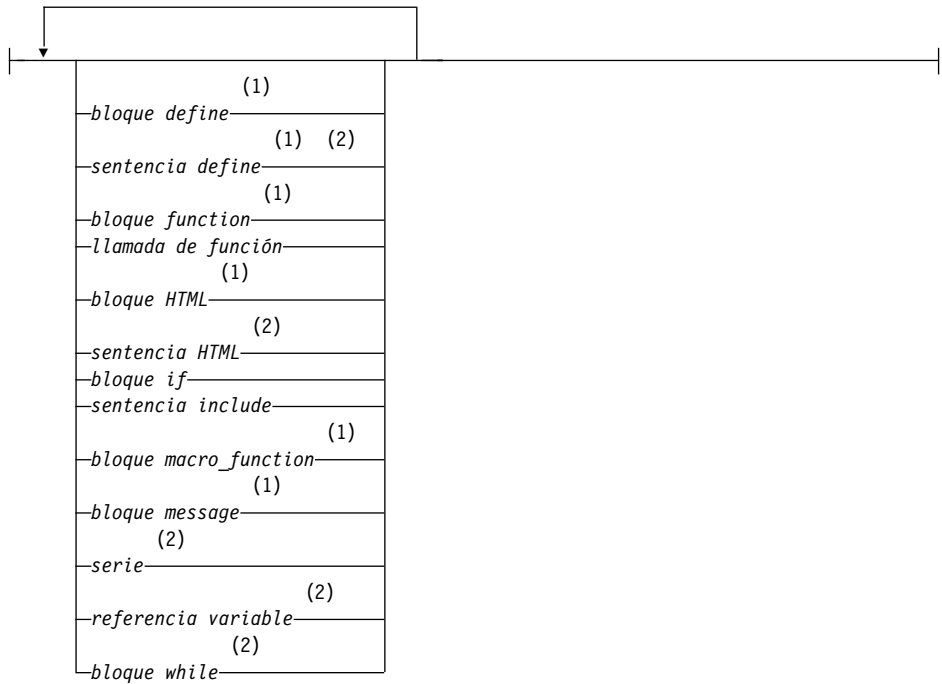
Sintaxis

```
►►—%IF—| lista de condiciones |—| bloque_sentencia |—| else_if spec |—%ENDIF—►►
```

lista de condiciones:



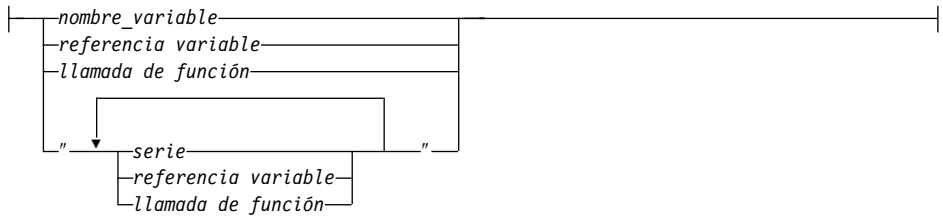
bloque_sentencia:



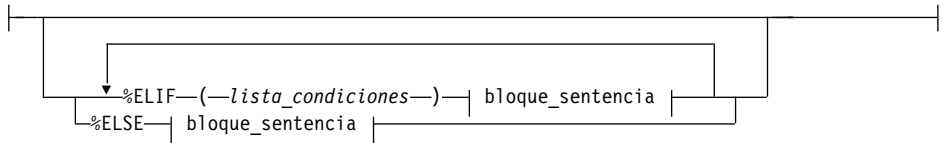
condición:



término:



else_if spec:



Notas:

- 1 Esta construcción de lenguaje es válida cuando el bloque IF está ubicado fuera de cualquier otro bloque de la parte de declaración de la macro.
- 2 Esta construcción de lenguaje es válida cuando el bloque IF está ubicado en un bloque HTML, bloque MACRO_FUNCTION, bloque REPORT, bloque ROW o bloque WHILE.

Valores

%IF

La palabra clave que especifica el proceso en serie condicional.

lista de condiciones

Compara los valores de condiciones y términos. Las listas de condiciones pueden conectarse utilizando operadores Booleanos. Una lista de condiciones puede anidarse dentro de otra lista de condiciones.

bloque_sentencia

Las construcciones de macro de Net.Data válidas siguientes. Por favor, consulte las notas de diagrama y las restricciones siguientes para determinar el contexto en el que son válidas las construcciones de macros.

sentencia define

La sentencia o bloque DEFINE. Define las variables y establece variables de configuración. Los nombres de variable deben comenzar por una letra o subrayado (_) y deben contener caracteres alfanuméricos o subrayados. Consulte la sintaxis y ejemplos en el apartado “Bloque o sentencia DEFINE” en la página 12.

bloque function

Una palabra clave que especifica una subrutina que puede invocarse desde la macro de Net.Data. Las sentencias ejecutables de un bloque FUNCTION pueden contener sentencias de lenguaje que interprete directamente un entorno de lenguaje, o bien pueden indicar una llamada a un programa externo. Consulte la sintaxis y ejemplos en el apartado “Bloque FUNCTION” en la página 21.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una función incorporada de Net.Data con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

bloque HTML

Incluye cualquier carácter alfabético o numérico, así como códigos HTML que han de formatearse para el navegador del cliente.

sentencia HTML

Incluye cualquier carácter alfabético o numérico, así como códigos HTML que han de formatearse para el navegador del cliente.

bloque if

El bloque IF. Efectúa el proceso en serie condicional. Los valores de serie de la lista de condiciones se tratan como numéricos a efectos de comparación si son series que representan enteros y no tienen ningún espacio en blanco inicial o de cola. Pueden tener un único signo más (+) o menos (-) inicial.

sentencia include

La sentencia INCLUDE. Lee e incorpora un archivo en la macro de Net.Data. Consulte la sintaxis y ejemplos en el apartado “Sentencia INCLUDE” en la página 44.

bloque macro_function

Una palabra clave que especifica una subrutina que puede invocarse desde la macro de Net.Data. Las sentencias ejecutables de un bloque MACRO_FUNCTION pueden contener sentencias fuente de lenguaje de macro de Net.Data. Consulte la sintaxis y ejemplos en el apartado “Bloque MACRO_FUNCTION” en la página 49.

bloque message

El bloque MESSAGE. Un conjunto de códigos de retorno, los mensajes asociados y las acciones que adopta Net.Data cuando se devuelve una llamada de función. Consulte la sintaxis y ejemplos en el apartado “Bloque MESSAGE” en la página 54.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación. Si la serie está en el término de la lista de condiciones,

puede contener cualquier carácter, excepto el carácter de nueva línea. Si la serie está en el bloque de código ejecutable, puede contener cualquier carácter, incluyendo el carácter de nueva línea.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR= 'abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

bloque while

El bloque WHILE. Efectúa la repetición en bucle con el proceso en serie condicional. Consulte la sintaxis y ejemplos en el apartado “Bloque WHILE” en la página 67

condición

Una comparación entre dos términos utilizando operadores de comparación. Una condición IF se trata como comparación numérica si se cumplen las siguientes condiciones:

- El operador de condiciones es uno de los operadores siguientes:
<, <=, >, >=, ==, !=
- Ambos términos son series que representan enteros válidos, siendo un entero válido una serie de dígitos, que opcionalmente van precedidos por un signo más (+) o menos (-) y ningún otro espacio en blanco.

Si no se cumple ninguna de las dos condiciones, se efectúa una comparación de serie normal.

término

Un nombre de variable, serie, referencia variable o llamada de función.

%ELIF

Palabra clave que inicia la vía de acceso de proceso alternativa y que puede contener listas de condiciones y la mayoría de las sentencias de macro de Net.Data.

%ENDIF

Palabra clave que cierra el bloque %IF.

%ELSE

Palabra clave que ejecuta sentencias asociadas si no se cumple ninguna de las demás listas de condiciones.

Contexto

El bloque IF puede encontrarse en estos contextos:

- Fuera de cualquier otro bloque de la parte de declaración de una macro de Net.Data
- Bloque HTML
- Bloque IF
- Bloque MACRO_FUNCTION

- Bloque REPORT
- Bloque ROW
- Bloque WHILE

Restricciones

El bloque IF puede contener estos elementos cuando están ubicados fuera de cualquier otro bloque de la parte de declaración de la macro de Net.Data:

- Bloque Comment
- Bloque DEFINE
- Sentencia DEFINE
- Bloque FUNCTION
- Llamada de función
- Bloque HTML
- Bloque IF
- Sentencia INCLUDE
- Bloque MACRO_FUNCTION
- Bloque MESSAGE
- Referencia variable

El bloque IF puede contener estos elementos cuando están ubicados en el bloque HTML, bloque, MACRO_FUNCTION, bloque REPORT, bloque ROW o bloque WHILE de la macro de Net.Data:

- Bloque Comment
- Llamadas de función
- Bloque IF
- Sentencia INCLUDE
- Sentencia HTML
- Serie
- Referencia variable
- Bloque WHILE

Puede anidar un máximo de 1024 bloques IF.

Ejemplos

Ejemplo 1: Un bloque IF de la parte de declaración de una macro de Net.Data.

```
%DEFINE a = "1"
%DEFINE b = "2"
...
%IF ($(DTW_HTML_TABLE) == "YES")
  %define OUT_FORMAT = "HTML"
%ELSE
  %define OUT_FORMAT = "CHARACTER"
%ENDIF
```



```
%HTML(REPORT) {
    ...
%}
```

Ejemplo 2: Un bloque IF dentro de un bloque HTML

```
%HTML(REPORT) {
@myFunctionCall()
%IF ($RETURN_CODE) == $(failure_rc))
    <p> The function call failed with failure code $(RETURN_CODE).
%ELIF ($RETURN_CODE) == $(warning_rc))
    <p> The function call succeeded with warning code $(RETURN_CODE).
%ELIF ($RETURN_CODE) == $(success_rc))
    <p>The function call was successful.
%ELSE
    P>The function call returned with unknown return code $(RETURN_CODE).
%ENDIF
%}
```

Ejemplo 3: Una comparación numérica

```
%IF (ROW_NUM < "100")
    <p>The table is not full yet...
%ELIF (ROW_NUM == "100")
    <p>The table is now full...
%ELSE
    <p>The table has overflowed...
%ENDIF
```

Se efectúa una comparación numérica debido a que la variable de tabla implícita ROW_NUM siempre devuelve un valor de entero y el valor que se compara es asimismo un entero.

Ejemplo 4: Bloques IF anidados

```
%IF (MONTH == "January")
    %IF (DATE = "1")
        HAPPY NEW YEAR!
    %ELSE
        Ho hum, just another day.
    %ENDIF
%ENDIF
```

Sentencia INCLUDE

Propósito

Lee e incorpora un archivo en la macro de Net.Data en la que se ha especificado la sentencia.

Net.Data examina los directorios especificados en la sentencia INCLUDE_PATH del archivo de inicialización para buscar el archivo include.

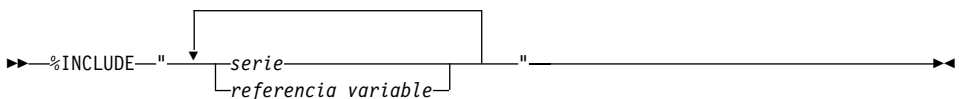
Puede utilizar archivos include del mismo modo en que puede hacerlo en la mayoría de los lenguajes de alto nivel. Pueden insertar cabeceras y pies de página comunes, definir conjuntos de variables comunes o incorporar una biblioteca de subrutina común de definiciones de bloque FUNCTION en una macro de Net.Data.

Net.Data ejecuta una sentencia INCLUDE sólo una vez al procesar la macro e inserta el contenido del archivo incluido en la ubicación de la sentencia INCLUDE de la macro. Las referencias variables en el nombre del archivo incluido se resuelven la primera vez que se ejecuta la sentencia INCLUDE y no cuando va a ejecutarse el contenido del archivo incluido.

Cuando una sentencia INCLUDE está en un bloque ROW o WHILE, Net.Data no ejecuta repetidamente la sentencia INCLUDE. Net.Data ejecuta la sentencia INCLUDE la primera vez que ejecuta el bloque ROW o WHILE, incorpora el contenido del archivo incluido en el bloque y después ejecuta repetidamente el bloque ROW o WHILE con el contenido del archivo incluido.

Sugerencia para la autorización Asegúrese de que el ID de usuario bajo el que se ejecuta Net.Data tiene derechos de usuario para los archivos a los que hace referencia la sentencia INCLUDE. Consulte la sección sobre la especificación de derechos de acceso del servidor de la Web para archivos Net.Data en el capítulo de configuración del manual *Guía de administración y programación de Net.Data* para obtener más información.

Sintaxis



Valores

%INCLUDE

La palabra clave que indica un archivo ha de leerse e incorporarse a la macro de Net.Data.

nombre

Serie numérica o alfabética que comienza por un carácter alfabético o un subrayado y contiene cualquier combinación de caracteres alfabéticos, numéricos o de subrayado.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación, excepto el carácter de nueva línea.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR='abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

Contexto

La sentencia INCLUDE puede encontrarse en estos contextos:

- Bloque DEFINE
- Bloque HTML
- Bloque REPORT
- Bloque ROW
- Bloque IF
- Bloque MESSAGE
- Bloque MACRO_FUNCTION
- Bloque WHILE
- Fuera de cualquier bloque de la parte de declaración de la macro de Net.Data

Restricciones

La sentencia INCLUDE puede contener estos elementos:

- Bloque Comment
- Series
- Referencias variables

No se permiten llamadas de función en la serie.

Puede anidar un máximo de diez bloques INCLUDE.

Ejemplos

Ejemplo 1: Una sentencia INCLUDE en un bloque HTML

```
%HTML(start){
%INCLUDE "header.hti"
...
%}
```

Ejemplo 2: Una sentencia INCLUDE en un bloque REPORT

```
%REPORT {
  %INCLUDE "report_header.txt"
  %ROW {
```

```
        %INCLUDE "row_include.txt"  
    %}  
    %INCLUDE "report_footer.txt"  
%}
```

Ejemplo 3: Referencias variables en una sentencia INCLUDE

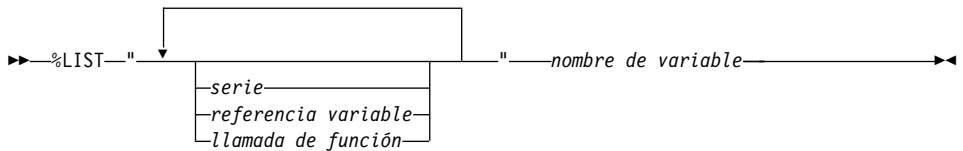
```
%define  
REMOTE_USER = %ENVVAR  
%include "${REMOTE_USER}.hti"
```

Sentencia LIST

Propósito

Crea una lista delimitada de valores. Puede utilizar la sentencia LIST al construir consultas de SQL con múltiples elementos como los que se hallan en las cláusulas WHERE o HAVING.

Sintaxis



Valores

%LIST

La palabra clave que especifica las variables que van a utilizarse para crear una lista delimitada de valores.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación, excepto el carácter de nueva línea.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR='abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una función incorporada de Net.Data con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

nombre de variable

Nombre que identifica una variable. Consulte el apartado “Nombre de variable” en la página 5 para obtener información de sintaxis.

Contexto

La sentencia LIST puede encontrarse en estos contextos:

- Sentencia DEFINE

Restricciones

La sentencia LIST puede contener estos elementos:

- Bloque Comment
- Referencias variables

- Llamadas de función
- Series

Ejemplos

Ejemplo 1: Una lista de variables

```
%DEFINE{
DATABASE="custcity"
%LIST " OR " conditions
conditions="cond1='Sao Paolo'"
conditions="cond2='Seattle'"
conditions="cond3='Shanghai'"
whereClause=conditions ? "WHERE $(conditions)" : ""
%}
```

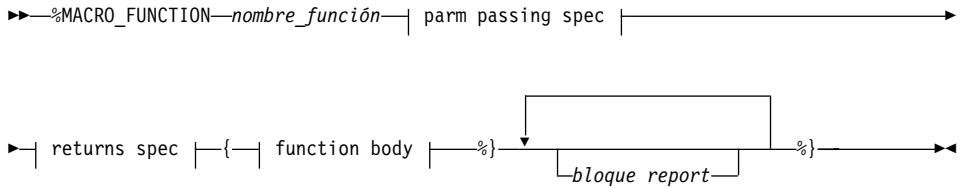
Para obtener más información sobre la utilización de sentencias LIST con variables, consulte el apartado “Variables de lista” en la página 80.

Bloque MACRO_FUNCTION

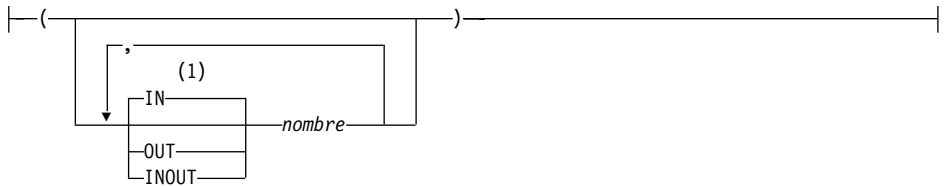
Propósito

Define una subrutina que puede invocarse desde la macro de Net.Data. Las sentencias ejecutables de un bloque MACRO_FUNCTION deben ser sentencias fuente de lenguaje de macro de Net.Data.

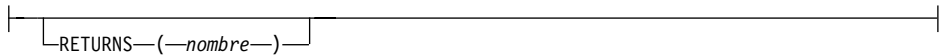
Sintaxis



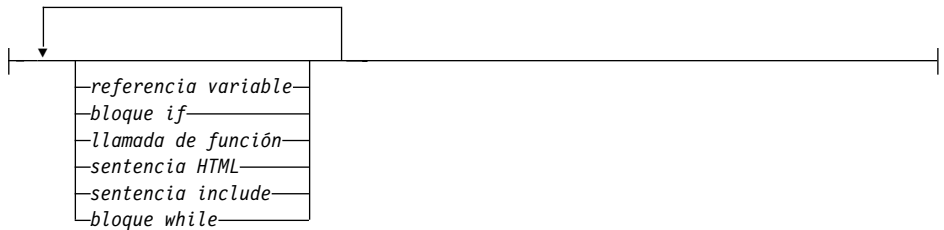
parm passing spec:



returns spec:



function body:



Notas:

- 1 El tipo de parámetro por omisión de IN se aplica cuando no se especifica ningún tipo de parámetro al principio de la lista de parámetros. Un parámetro sin un tipo de parámetro utiliza el tipo que se ha especificado más recientemente en la lista de parámetros, o graba IN si no se ha especificado ningún tipo. Por ejemplo, en la lista de parámetros (*parm1*, INOUT *parm2*, *parm3*, OUT *parm4*, *parm5*), los parámetros *parm1*, *parm3* y *parm5* no tienen tipos de parámetro. El parámetro *parm1* tiene un tipo de IN ya que no se ha especificado ningún tipo de parámetro inicial. El parámetro *parm3* tiene un tipo de INOUT ya que es el tipo de parámetro que se ha especificado más recientemente. De modo análogo, el parámetro *parm5* tiene un tipo de OUT ya que es el tipo de parámetro que se ha especificado más recientemente.

Valores

%MACRO_FUNCTION

La palabra clave que especifica una subrutina que puede invocarse desde la macro de Net.Data. Las sentencias ejecutables de un bloque MACRO_FUNCTION deben contener sentencias de lenguaje que Net.Data interprete directamente.

nombre_función

El nombre de la función que está definiéndose. Una serie numérica o alfabética que comienza por un carácter alfabético o un subrayado y contiene cualquier combinación de caracteres alfabéticos, numéricos o de subrayado.

parm passing spec:

IN Especifica que Net.Data pase datos de entrada al entorno de lenguaje. IN es el valor por omisión.

OUT

Especifica que el entorno de lenguaje devuelva datos de salida a Net.Data.

INOUT

Especifica que Net.Data pase datos de entrada al entorno de lenguaje y que éste devuelva datos de salida a Net.Data.

nombre

Serie numérica o alfabética que comienza por un carácter alfabético o un subrayado y contiene cualquier combinación de caracteres alfabéticos, numéricos o de subrayado. *nombre* puede representar una tabla de Net.Data o un conjunto de resultados.

returns spec:

RETURNS

Declara la variable que contiene el valor de función una vez se complete dicha función.

function body:

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR='abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

bloque if

El bloque IF. Efectúa el proceso en serie condicional. Los valores de serie de la lista de condiciones se tratan como numéricos a efectos de comparación si representan enteros y no tienen ningún espacio en blanco inicial o de cola. Pueden tener un signo más (+) o menos (-) inicial.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una función incorporada de Net.Data con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

sentencia HTML

Incluye cualquier carácter alfabético o numérico, así como códigos HTML que han de formatearse para el navegador del cliente.

sentencia include

La sentencia INCLUDE. Lee e incorpora un archivo en la macro de Net.Data. Consulte la sintaxis y ejemplos en el apartado “Sentencia INCLUDE” en la página 44.

bloque while

El bloque WHILE. Efectúa la repetición en bucle con el proceso en serie condicional. Consulte la sintaxis y ejemplos en el apartado “Bloque WHILE” en la página 67.

bloque report

El bloque REPORT. Instrucciones de formato para la salida de una llamada de función. Puede utilizar información de cabecera y pie de página para el informe. Consulte la sintaxis y ejemplos en el apartado “Bloque REPORT” en la página 59.

Contexto

El bloque MACRO_FUNCTION puede encontrarse en estos contextos:

- Bloque IF
- Fuera de cualquier bloque de la parte de declaración de la macro de Net.Data

Restricciones

El bloque MACRO_FUNCTION puede contener estos elementos:

- Bloque Comment
- Sentencias HTML
- Bloque IF
- Sentencia INCLUDE
- Bloque REPORT
- Bloque WHILE
- Referencias variables
- Llamadas de función

Ejemplos

Ejemplo 1: Una función de macro que especifica el manejo de mensajes

```
%MACRO_FUNCTION setMessage(IN rc, OUT message) {  
  %IF (rc == "0")  
    @dtw_assign(message, "Function call was successful.")  
  %ELIF (rc == "-1")  
    @dtw_assign(message, "Function failed, out of memory.")  
  %ELIF (rc == "-2")  
    @dtw_assign(message, "Function failed, invalid parameter.")  
  %ENDIF  
%}
```

Ejemplo 2: Una función de macro que especifica información de cabecera

```
%MACRO_FUNCTION  
setup(IN browserType) {  
  %{ call this function at the top of each HTML block in the macro %}  
  %INCLUDE "header_info.html"  
  @dtw_rdate()  
  %IF (browserType == "IBM")  
    @setupIBM()  
  %ELIF (browserType == "MS")  
    @setupMS()  
  %ELIF (browserType == "NS")  
    @setupNS()  
  %ELSE  
    @setupDefault()  
  %ENDIF  
%}
```

Ejemplo 3: Una función de macro que contiene un bloque REPORT

```
%MACRO_FUNCTION myfunc (INOUT table) {  
  %REPORT {  
    <table>  
      %ROW {  
        <tr><td>$(V1)</td><td>$(V2)</td></tr>  
      %}  
    </table>  
  %}  
%}
```

Ejemplo 4: Una función de macro que utiliza la palabra clave RETURNS

```
%MACRO_FUNCTION myfunc ()  
RETURNS(VALUE) {  
    @DTW_ASSIGN(VALUE, "Success...")  
%}
```

Bloque MESSAGE

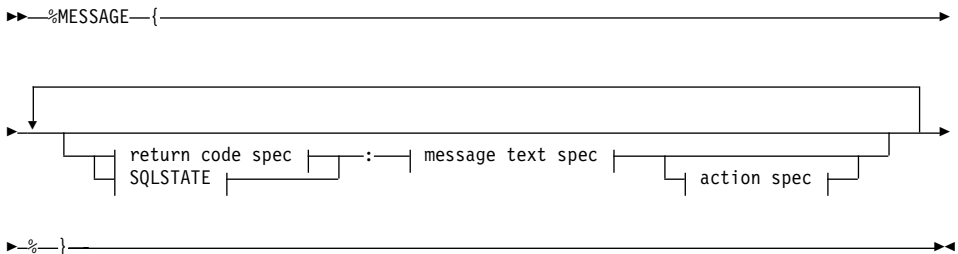
Propósito

Especifica los mensajes a visualizar y las acciones a adoptar basándose en el código de retorno de una función.

Define el conjunto de códigos de retorno, junto con sus acciones y mensajes correspondientes en el bloque MESSAGE. Cuando finaliza una llamada de función, Net.Data compara su código de retorno con los códigos de retorno definidos en el bloque MESSAGE. Si el código de retorno de la función coincide con el del bloque MESSAGE, Net.Data visualiza el mensaje y evalúa la acción para determinar si continuar el proceso o salir de la macro de Net.Data.

Un bloque MESSAGE puede tener un ámbito global, o local para un único bloque FUNCTION. Si se define el bloque MESSAGE en la capa más externa de la macro, se considera que es global. Cuando se definen múltiples bloques MESSAGE globales, sólo se considera que está activo el último bloque procesado. Si el bloque MESSAGE está definido dentro de un bloque FUNCTION, el bloque es de ámbito local para el bloque FUNCTION en el lugar en el que se ha definido. Consulte el apartado sobre el bloque MESSAGE en el manual *Guía de administración y programación de Net.Data* para conocer las normas de proceso de los códigos de retorno.

Sintaxis



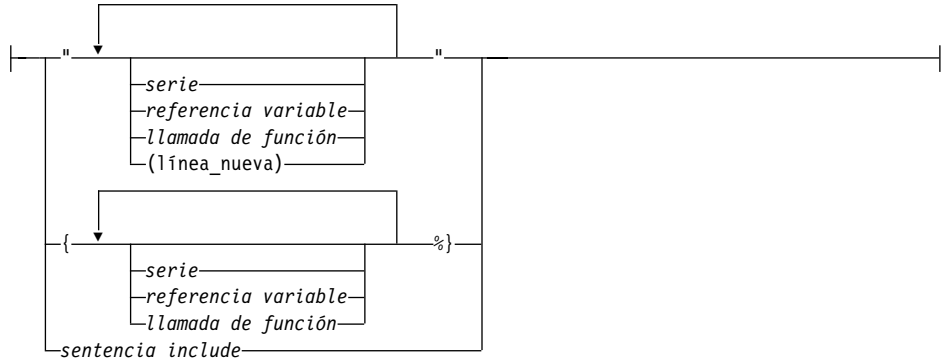
return code spec:



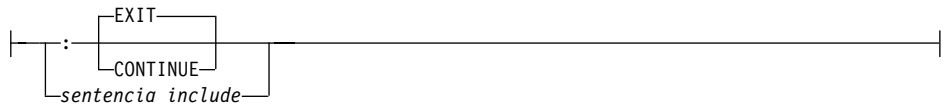
SQLSTATE:

|—SQLSTATE—:—*id_estado*—|

message text spec:



action spec:



Valores

%MESSAGE

Palabra clave para el bloque que define un conjunto de códigos de retorno, los mensajes asociados y las acciones que adopta Net.Data cuando se devuelve una llamada de función.

return code spec

Un entero positivo o negativo. Si el valor de la variable `RETURN_CODE` de Net.Data coincide con el valor de *return code spec*, el resto de la información de la sentencia de mensaje se utiliza para procesar la llamada de función. También puede especificar mensajes para los códigos de retorno que no se entran de modo específico en el bloque MESSAGE.

+DEFAULT

Palabra clave utilizada para especificar un código de mensajes positivo por omisión. Net.Data utiliza la información de esta sentencia

de mensaje para procesar la llamada de función en el caso de que RETURN_CODE sea superior a cero (0) y no se especifique una coincidencia exacta.

-DEFAULT

Una palabra clave utilizada para especificar un código de mensajes negativo por omisión. Net.Data utiliza la información de esta sentencia de mensaje para procesar la llamada de función en el caso de que RETURN_CODE sea inferior a cero (0) y no se especifique una coincidencia exacta.

DEFAULT

Una palabra clave utilizada para especificar el código de mensajes por omisión. Net.Data utiliza la información de esta sentencia de mensaje para procesar la llamada de función en el caso de que se cumplan todas las condiciones siguientes:

- Si RETURN_CODE es mayor o menor que cero, pero no es cero
- Si se ha especificado una coincidencia exacta para el código de retorno.
- Si no se especifican los valores de +DEFAULT o -DEFAULT cuando RETURN_CODE es mayor o menor que cero

código_mens

El código de mensajes que especifica los errores y avisos que pueden producirse durante el proceso. Una serie de dígitos numéricos con valores de 0 a 9.

SQLSTATE

Una palabra clave que proporciona códigos comunes a los programas de aplicación para las condiciones de error comunes. Los valores de SQLSTATE están basados en la especificación de SQLSTATE que contiene el SQL estándar y el esquema de codificación es el mismo en todas las implantaciones de SQL de IBM. Este valor se corresponde con la variable SQL_STATE de Net.Data.

id_estado

El SQLSTATE. Una serie alfanumérica de cinco caracteres (bytes) con un formato de ccsss, donde cc indica la clase y sss indica la subclase.

message text spec

Serie que se envía al navegador de la Web en el caso de que el RETURN_CODE coincida con el valor de *código_retorno* o en el caso de que la variable SQL_STATE coincida con el valor de SQLSTATE en la sentencia de mensajes actual.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación. Si la serie aparece entre dobles comillas, no se permite el carácter de línea nueva.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR='abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una función incorporada de Net.Data con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

action spec

Especifica la acción que adopta Net.Data en el caso de que la variable RETURN_CODE coincida con el valor de *código_retorno* o en el caso de que la variable SQL_STATE coincida con el valor de SQLSTATE en la sentencia de mensajes actual.

EXIT

Palabra clave que especifica salir de la macro de modo inmediato cuando se produzca el error o el aviso que se corresponda con el código de mensajes especificado. Este es el valor por omisión.

CONTINUE

Palabra clave que especifica continuar el proceso cuando se produzca el error o el aviso que se corresponda con el código de mensajes especificado.

sentencia include

La sentencia INCLUDE. Lee e incorpora un archivo en la macro de Net.Data. La sentencia INCLUDE puede aparecer en cualquier parte del MESSAGE. Consulte la sintaxis y ejemplos en el apartado “Sentencia INCLUDE” en la página 44.

Contexto

El bloque MESSAGE puede encontrarse en estos contextos:

- Bloque FUNCTION
- Bloque IF
- Fuera de todos los bloques o sentencias de la parte de declaración de la macro de Net.Data.

Restricciones

El bloque MESSAGE puede contener estos elementos:

- Bloque Comment
- Llamadas de función
- Referencias variables
- Sentencias HTML
- Series
- Sentencia INCLUDE

Para OS/390: Las funciones de SQL no pueden llamarse desde dentro de las funciones de SQL.

Ejemplos

Ejemplo 1: Un bloque MESSAGE local

```
%{ local message block inside a FUNCTION block %}
%FUNCTION(DTW_REXX) my_function() {
  %EXEC { my_command.cmd %}
  %MESSAGE{
-601: {<h3>The table has already been created, please go back and
enter your name.</h3>
<p><a href="input">Return</a>
%}
default: "<h3>Can't continue because of error
$(RETURN_CODE)</h3>"%}      : exit
%}
```

Ejemplo 2: Un bloque MESSAGE global

```
%{
global message block %}
%MESSAGE {
  -100      : "Return code -100 message"      : exit
  100      : "Return code 100 message"       : continue
  +default : {
This is a long message that spans more
than one line. You can use HTML tags, including
links and forms, in this message. %}      : continue
%}

%{ local message block inside a FUNCTION block %}
%FUNCTION(DTW_REXX) my_function() {
  %EXEC { my_command.cmd %}
  %MESSAGE {
    -100      : "Return code -100 message"      : exit
    100      : "Return code 100 message"       : continue
    -default : {
This is a long message that spans more
than one line. You can use HTML tags, including
links and forms, in this message. %}      : exit
    %}
}
```

Ejemplo 3: Un bloque MESSAGE que contiene sentencias INCLUDE

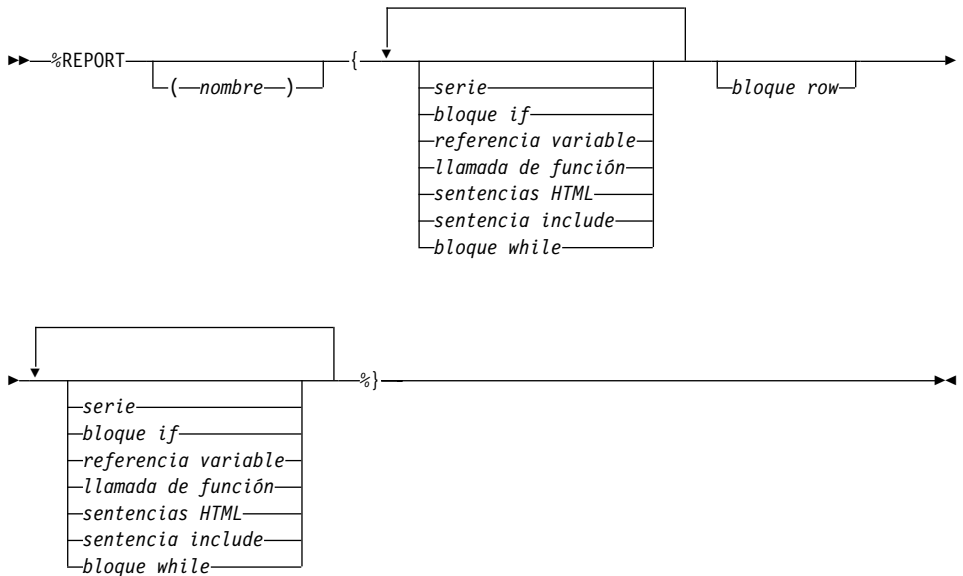
```
%message {
  %include "rc1000.msg"
  %include "rc2000.msg"
  %include "defaults.msg"
%}
```


Bloque REPORT

Propósito

Da formato a la salida desde una llamada de función. Puede entrar un parámetro de nombres de tabla para especificar que el informe va a utilizar los datos de la tabla con nombre. En caso contrario, el informe se genera con la primera tabla de salida que se encuentre en la lista de parámetros de función, o con los datos de tabla por omisión, en el caso de que no haya ningún nombre de tabla en la lista.

Sintaxis



Valores

%REPORT

La palabra clave para especificar instrucciones de formateo para la salida de una llamada de función. Puede utilizar información de cabecera y pie de página para el informe.

nombre

Este valor representa un conjunto de resultados o una tabla de `Net.Data`. Consulte el apartado sobre el bloque `Report` en el manual *Guía de administración y programación de Net.Data* para obtener más información.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación.

bloque if

El bloque IF. Efectúa el proceso en serie condicional. Los valores de serie de la lista de condiciones se tratan como numéricos a efectos de comparación si representan enteros y no tienen ningún espacio en blanco inicial o de cola. Pueden tener un signo más (+) o menos (-) inicial. Consulte la sintaxis y ejemplos en el apartado “Bloque IF” en la página 37.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR= 'abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una función incorporada de Net.Data con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

sentencias HTML

Incluye cualquier carácter alfabético o numérico, así como códigos HTML que han de formatearse para el navegador del cliente.

sentencia include

La sentencia INCLUDE. Lee e incorpora un archivo en la macro de Net.Data. Consulte la sintaxis y ejemplos en el apartado “Sentencia INCLUDE” en la página 44.

bloque row

El bloque ROW. Visualiza datos formateados de HTML una vez por cada fila de datos que se devuelve desde una llamada de función. Consulte la sintaxis y ejemplos en el apartado “Bloque ROW” en la página 62.

bloque while

El bloque WHILE. Efectúa la repetición en bucle con el proceso en serie condicional. Consulte la sintaxis y ejemplos en el apartado “Bloque WHILE” en la página 67.

Contexto

El bloque REPORT puede encontrarse en estos contextos:

- Bloque o sentencia FUNCTION
- Bloque MACRO_FUNCTION

Restricciones

El bloque REPORT puede contener estos elementos:

- Bloque Comment
- Bloque IF
- Sentencias INCLUDE
- Bloques ROW
- Bloques WHILE

- Llamadas de función

Para OS/390: Las funciones de SQL no pueden llamarse desde dentro de las funciones de SQL.

- Sentencias HTML
- Series
- Referencias variables

Para OS/390: Los bloques REPORT no se admiten en los bloques MACRO_FUNCTION.

Ejemplos

Ejemplo 1: Una tabla HTML a dos columnas que muestra una lista de nombres y ubicaciones

```
%FUNCTION(DTW_SQL) mytable() {
  %REPORT{
    <h2>Query Results</h2>
    <p>Select a name for details.
    <table border=1>
    <tr><td>Name</td><td>Location</td>
    %ROW{
    <tr>
    <td>
    <a href="/cgi-bin/db2www/name.mac/details?name=$(V1)&loc=$(V2)">$(V1)
    </a></td>
    <td>$(V2)</td>
    %}
    </table>
  %}
```

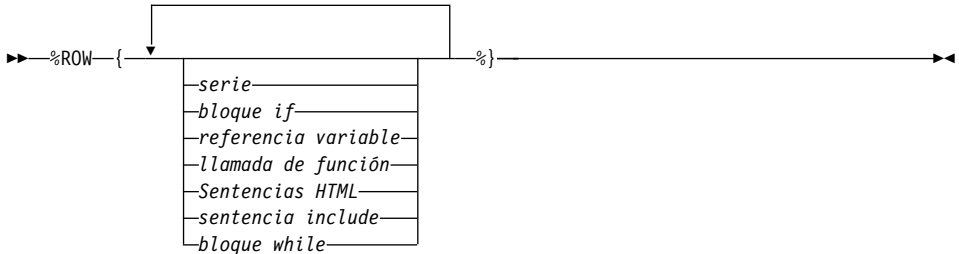
Seleccionar un nombre de la tabla llama al bloque HTML *detalles* de la macro de Net.Data *name.mac* y le envía los dos valores como parte del URL. En este ejemplo, los valores pueden utilizarse en *name.mac* para buscar detalles adicionales sobre el nombre.

Bloque ROW

Propósito

Procesa cada fila de tabla que se devuelve desde una llamada de función. Net.Data procesa las sentencias del bloque ROW una vez por cada fila.

Sintaxis



Valores

%ROW

La palabra clave que especifica que van a visualizarse los datos formateados de HTML, una vez por cada fila de datos que se devuelve desde una llamada de función.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación.

bloque if

El bloque IF. Efectúa el proceso en serie condicional. Los valores de serie de la lista de condiciones se tratan como numéricos a efectos de comparación si son series que representan enteros y no tienen ningún espacio en blanco inicial o de cola. Pueden tener un único signo más (+) o menos (-) inicial. Consulte la sintaxis y ejemplos en el apartado “Bloque IF” en la página 37.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR='abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o funciones incorporadas con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

sentencias HTML

Incluye cualquier carácter alfabético o numérico, así como códigos HTML que han de formatearse para el navegador del cliente.

sentencia include

La sentencia INCLUDE. Lee e incorpora un archivo en la macro de Net.Data. Consulte la sintaxis y ejemplos en el apartado “Sentencia INCLUDE” en la página 44.

bloque while

El bloque WHILE. Efectúa la repetición en bucle con el proceso en serie condicional. Consulte la sintaxis y ejemplos en el apartado “Bloque WHILE” en la página 67.

Contexto

El bloque ROW puede encontrarse en estos contextos:

- Bloque REPORT

Restricciones

El bloque ROW puede contener estos elementos:

- Bloque Comment
- Bloques IF
- Sentencias INCLUDE
- Bloques WHILE
- Llamadas de función

Para OS/390: Las funciones de SQL no pueden llamarse desde dentro de las funciones de SQL.

- Referencias variables
- Sentencias HTML
- Series

Ejemplos

Ejemplo 1: Una tabla HTML a dos columnas que muestra una lista de nombres y ubicaciones

```
%REPORT{
<h2>Query Results</h2>
<p>Select a name for details.
<table border=1>
<tr><td>Name</td><td>Location</td>

%ROW{
<tr>
<td>
<a
href="/cgi-bin/db2www/name.mac/details?name=$(V1)&location=$(V2)">$(V1)</a></td>
<td>$(V2)</td>
%}

</table>
%}
```

Seleccionar un nombre de la tabla llama al bloque HTML *detalles* de la macro de Net.Data *name.mac* y le envía los dos valores como parte del URL. En este

ejemplo, los valores pueden utilizarse en *name.mac* para buscar detalles adicionales sobre el nombre.

Sentencia TABLE

Propósito

Define una variable que es un conjunto de datos relacionados. La variable contiene un conjunto de filas y columnas incluyendo una fila de cabeceras de columna que describen los campos de cada una de las filas. Una sentencia de tabla sólo puede estar en un bloque o sentencia DEFINE.

Cuando se hace referencia a una variable TABLE mientras se ejecuta un bloque HTML, Net.Data visualiza el contenido de la tabla como tabla de caracteres planos o, en el caso de que la variable DTW_HTML_TABLE se haya establecido en YES, como tabla HTML. Cuando se hace referencia a una variable TABLE mientras se ejecuta un bloque XML, Net.Data devuelve la tabla como RowSet.

Sintaxis

►►%TABLE—| límite superior |—————►►

límite superior:

|
|
| (— número —)
| ALL

Valores

%TABLE

Palabra clave que especifica la definición de un conjunto de datos relacionados que contiene una matriz de registros idénticos, o filas y una matriz de nombres de columna que describen los campos de cada una de las filas.

límite superior

El número de filas que puede haber en la tabla. Si no se especifica el valor de límite superior, la tabla puede contener un número ilimitado de filas.

número

Una serie de dígitos. Un valor de 0 admite un número ilimitado de filas en la tabla.

ALL

Palabra clave que admite un número ilimitado de filas en la tabla.

Contexto

La sentencia TABLE puede encontrarse en estos contextos:

- Sentencia DEFINE

Restricciones

La sentencia TABLE puede contener estos elementos:

- Bloque Comment
- Números

Ejemplos

Ejemplo 1: Tabla de Net.Data con un límite superior de 30 filas

```
%DEFINE myTable1=%TABLE(30)
```

Ejemplo 2: Tabla de Net.Data que utiliza el valor por omisión de todas las filas

```
%DEFINE  
myTable2=%TABLE
```

Ejemplo 3: Tabla de Net.Data que especifica todas las filas

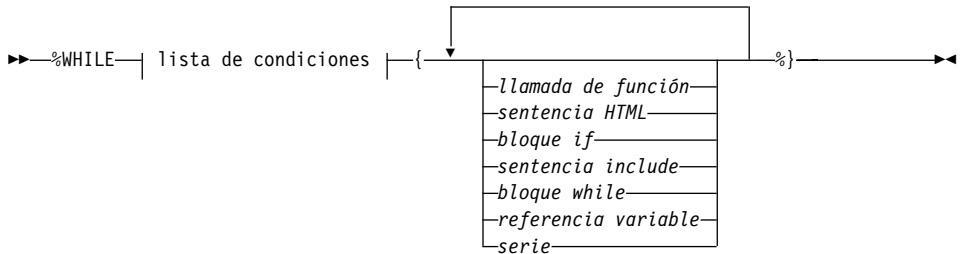
```
%DEFINE myTable3=%TABLE(ALL)
```


Bloque WHILE

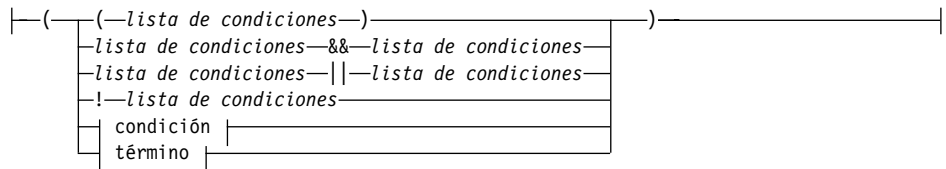
Propósito

Proporciona una construcción de repetición en bucle basada en el proceso en serie condicional. Puede utilizar el bloque WHILE del bloque HTML, el bloque REPORT, el bloque ROW, el bloque IF y el bloque MACRO_FUNCTION. Los valores de serie de la lista de condiciones se tratan como numéricos a efectos de comparación si son series que representan enteros y no tienen ningún espacio en blanco inicial o de cola. Pueden tener un único signo más (+) o menos (-) inicial.

Sintaxis



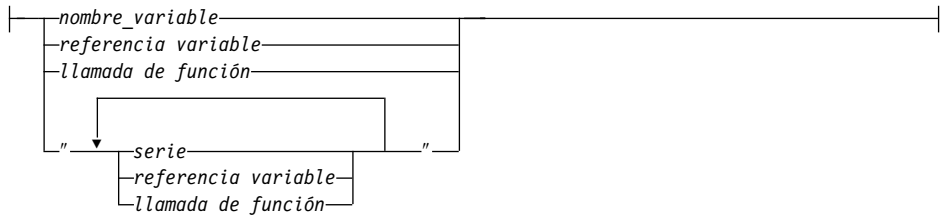
lista de condiciones:



condición:



término:



Valores

%WHILE

La palabra clave que especifica el proceso en bucle.

lista de condiciones

Compara los valores de condiciones y términos. Las listas de condiciones pueden conectarse utilizando operadores Booleanos. Una lista de condiciones puede anidarse dentro de otra lista de condiciones.

condición

Una comparación entre dos términos utilizando operadores de comparación. Una condición IF se trata como comparación numérica si se cumplen las siguientes condiciones:

- El operador de condiciones es uno de los operadores siguientes:
<, <=, >, >=, ==, !=
- Ambos términos son series que representan enteros válidos, siendo un entero válido una serie de dígitos, que opcionalmente van precedidos por un signo más (+) o menos (-) y ningún otro espacio en blanco.

Si no se cumple ninguna de las dos condiciones, se efectúa una comparación de serie normal.

término

Un nombre de variable, serie, referencia variable, para llamada de función.

llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o funciones incorporadas con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado "Llamada de función (@)" en la página 30.

sentencia HTML

Incluye cualquier carácter alfabético o numérico, así como códigos HTML que han de formatearse para el navegador del cliente.

bloque if

El bloque IF. Efectúa el proceso en serie condicional. Los valores de serie de la lista de condiciones se tratan como numéricos a efectos de comparación si representan enteros y no tienen ningún espacio en blanco.

inicial o de cola. Pueden tener un signo más (+) o menos (-) inicial. Consulte la sintaxis y ejemplos en el apartado “Bloque IF” en la página 37.

sentencia include

La sentencia INCLUDE. Lee e incorpora un archivo en la macro de Net.Data. Consulte la sintaxis y ejemplos en el apartado “Sentencia INCLUDE” en la página 44.

bloque while

El bloque WHILE. Efectúa la repetición en bucle con el proceso en serie condicional. Consulte la sintaxis y ejemplos en el apartado “Bloque WHILE” en la página 67.

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR='abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

serie

Cualquier secuencia de caracteres alfabéticos y numéricos y puntuación. Una serie en el término de la lista de condiciones puede contener cualquier carácter, excepto el carácter de nueva línea.

nombre de variable

Nombre que identifica una variable. Consulte el apartado “Nombre de variable” en la página 5 para obtener información de sintaxis.

Contexto

El bloque WHILE puede encontrarse en estos contextos:

- Bloque HTML
- Bloque REPORT
- Bloque ROW
- Bloque MACRO_FUNCTION
- Bloque IF
- Bloque WHILE

Restricciones

El bloque WHILE puede contener estos elementos:

- Bloque Comment
- Bloque IF
- Bloque WHILE
- Series
- Sentencias HTML
- Llamadas de función
- Referencias variables
- Sentencias INCLUDE

Ejemplos

Ejemplo 1: Bloque WHILE que genera filas en una tabla

```

%DEFINE loopCounter = "1"

%HTML(build_table) {
%WHILE (loopCounter <= "100") {
  %{ generate table tag and column headings %}
  %IF (loopCounter == "1")
    <table border>
    <tr>
    <th>Item #
    <th>Description
    </tr>
  %ENDIF

  %{ generate individual rows %}
  <tr>
  <td>
  <td>$(loopCounter)
  <td>@getDescription(loopCounter)
  </tr>

  %{ generate end table tag %}
  %IF (loopCounter == "100")
    </table>
  %ENDIF

  %{ increment loop counter %}
  @dtw_add(loopCounter, "1", loopCounter)
%}
%}

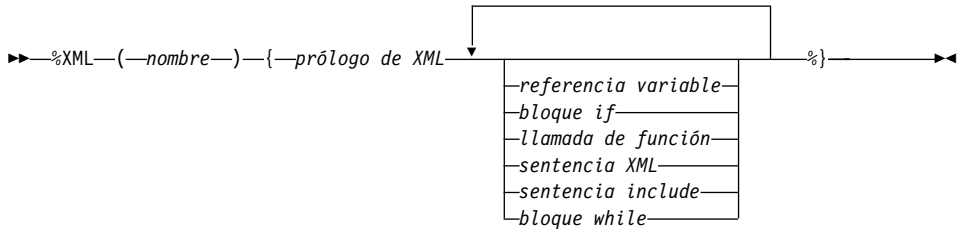
```

Bloque XML

Propósito

Define el modo en que va a presentarse una página Web en clientes Web habilitados para XML. El nombre del bloque XML que ha de ejecutarse se especifica en el URL cuando se invoca Net.Data. El bloque XML puede contener la mayoría de las sentencias de lenguaje de macro de Net.Data y cualquier contenido de XML.

Sintaxis



Valores

%HTML

La palabra clave que especifica el bloque que contiene el texto y los códigos de HTML que han de visualizarse en el navegador del cliente.

nombre

Una serie numérica o alfabética que comienza por un carácter alfabético o un subrayado y que contiene cualquier combinación de caracteres alfabéticos, numéricos o de subrayado, incluyendo los puntos.

prólogo de XML

Inserte un prólogo de XML, declarando las hojas de estilo, los DTD u otros requisitos para el documento de XML. El formato detallado de un prólogo de XML se define en <http://www.w3.org/TR/1998/REC-xml-19980210>. A continuación hay un ejemplo de un prólogo básico que utiliza una de las hojas de estilo que proporciona Net.Data:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="ndTable.xsl" ?>
```

referencia variable

Devuelve el valor de una variable y se especifica con \$ y (). Por ejemplo: si VAR='abc', \$(VAR) devuelve el valor 'abc'. Consulte el apartado “Referencia variable” en la página 5 para obtener información de sintaxis.

bloque if

El bloque IF. Efectúa el proceso en serie condicional. Los valores de serie de la lista de condiciones se tratan como numéricos a efectos de

comparación si son series que representan enteros y no tienen ningún espacio en blanco inicial o de cola. Pueden tener un único signo más (+) o menos (-) inicial. Consulte la sintaxis y ejemplos en el apartado “Bloque IF” en la página 37.

Llamada de función

Invoca uno o más bloques FUNCTION o MACRO_FUNCTION, o una función incorporada de Net.Data con argumentos específicos. Consulte la sintaxis y ejemplos en el apartado “Llamada de función (@)” en la página 30.

sentencias XML

Incluye cualquier XML que esté únicamente bien formado o cumpla asimismo con el DTD o la hoja de estilos que pudieran haberse especificado en el prólogo de XML.

sentencia include

La sentencia INCLUDE. Lee e incorpora un archivo en la macro de Net.Data. Consulte la sintaxis y ejemplos en el apartado “Sentencia INCLUDE” en la página 44.

bloque while

El bloque WHILE. Efectúa la repetición en bucle con el proceso en serie condicional. Consulte la sintaxis y ejemplos en el apartado “Bloque WHILE” en la página 67.

Contexto

El bloque XML puede encontrarse en estos contextos:

- Bloque IF
- Fuera de cualquier bloque de la parte de declaración de la macro de Net.Data

Restricciones

El bloque XML puede contener estos elementos:

- Bloque Comment
- Bloque IF
- Sentencias XML
- Sentencia INCLUDE
- Bloque WHILE
- Referencias variables
- Llamadas de función

Ejemplos

Ejemplo 1. Un bloque XML que incluye un prólogo estándar y llame a una función:

```
%XML(report3){  
%INCLUDE "style3header.xml"  
<title>Results</title>
```

```

<XMLBlock>
@xmp1()
</XMLBlock>
%}

```

Ejemplo 2. xmp1() puede definirse para devolver un pequeño conjunto de resultados a partir de una consulta de SQL:

```

%FUNCTION DTW_SQL xmp1() {
    SELECT LASTNME,EMPNO FROM EMPLOYEES
    WHERE LASTNME LIKE 'M%'
%}

%XML(report3){
<xml version="1.0" ?>
<xml-stylesheet type="text/xsl" href="ndReport.xsl" ?>
<title>Results</title>
<XMLBlock>
@xmp1()
</XMLBlock>
%}

```

Se produciría la siguiente salida:

Content Type: text/xml

```

<xml version="1.0" ?>
<xml-stylesheet type="text/xsl" href="ndReport.xsl" ?>
<title>Results</title>
<XMLBlock>
<RowSet>
  <Row number="1">
    <Column name="LASTNME">Mason</Column>
    <Column name="EMPNO">520</Column>
  </Row>
  <Row number="2">
    <Column name="LASTNME">Masse</Column>
    <Column name="EMPNO">559</Column>
  </Row>
  <Row number="3">
    <Column name="LASTNME">Mercury</Column>
    <Column name="EMPNO">312</Column>
  </Row>
</RowSet>
</XMLBlock>

```

Capítulo 2. Variables

Net.Data proporciona dos tipos de variables: las variables definidas por el usuario y las variables Net.Data.

“Variables definidas por el usuario” en la página 76

Variables que se definen para la aplicación. Puede definir las variables que efectúan las tareas siguientes:

- **“Variables condicionales” en la página 76**

Asignar un valor de variable basado en el valor de otra serie o variable.

- **“Variables de entorno” en la página 78**

Utilizar la construcción del lenguaje ENVVAR para hacer referencia a las variables de entorno.

- **“Variables ejecutables” en la página 78**

Utilizar la construcción del lenguaje EXEC para invocar otros programas desde una referencia a variable.

- **“Variables ocultas” en la página 79**

Ocultar la referencia a variable de la fuente HTML.

- **“Variables de lista” en la página 80**

Crear una serie delimitada de valores utilizando la construcción de lenguajes de LIST.

- **“Variables de tabla” en la página 82**

Pasar una matriz de valores a y desde una función. Puede utilizarse para una salida de informe.

Variables Net.Data

Variables para el proceso de varios y manipulación de archivos, proceso de tablas, formateo de informes y entornos de lenguaje.

Algunas variables tienen valores que pueden definir o modificar, otros los define Net.Data. La descripción de la variable especifica si se ha de definir o no un valor. Consulte la descripción de una variable para determinar el modo en que se define el valor.

Net.Data proporciona los tipos de variable siguientes:

- **“Variables Net.Data de proceso de tablas” en la página 83**

Net.Data las define para permitirle procesar tablas de Net.Data. Utilice estas variables para acceder a los datos desde llamadas a función y consultas SQL. Sólo se reconocen dentro de bloques REPORT o ROW, a menos que se especifique lo contrario.

- **“Variables Net.Data de informes” en la página 93**

Le ayudan a personalizar informes desde una función. Puede definir variables de informe en la sección DEFINE, o asignarlos a cualquier bloque de Net.Data.

- **“Variables Net.Data de entorno de lenguaje” en la página 103**

Le ayudan a personalizar el modo en que procesan los bloques FUNCTION, utilizando entornos de lenguaje.

- **“Variables Net.Data diversas” en la página 128**

Las define Net.Data para afectar los procesos Net.Data, averiguar el estado de una llamada de función y obtener información sobre el conjunto de resultados de una consulta a la base de datos. Algunas de las variables diversas las establece Net.Data y no pueden cambiarse.

La salida de muchas variables Net.Data depende del sistema operativo en el que se ejecutan.

Variables definidas por el usuario

Esta sección describe las variables definidas por el usuario. El usuario define estas variables en la macro.

Variables condicionales

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Una variable condicional es la que se establece basándose en el valor de otra serie o variable. También se le denomina *operación ternaria*.

La sintaxis para establecer condicionalmente una variable es:

```
condVar = testVar ? trueValue : falseValue
```

Donde:

condVar

La variable condicional que se ha de establecer.

testVar

Variable de prueba utilizada para determinar la condición. Una serie vacía se evalúa como false.

trueValue

Es el valor que se ha de utilizar si el valor de la variable de prueba es true.

falseValue

Es el valor a utilizar si el valor de la variable de prueba es false.

Ejemplo 1: Una variable condicional definida con dos valores posibles

```
varA = varB ?  
"value_1" : "value_2"
```

varB es la prueba. Por ello, a varA se le asigna o bien "value_1" o "value_2", en función de que varB exista y de que no contenga un valor NULL.

Ejemplo 2: Una variable condicional definida con una referencia a variable

```
varname = ?  
"${value_1}"
```

En este caso, *varname* es nula si *value_1* es nula, en caso contrario, varname se establece en *value_1*.

Ejemplo 3: Variable condicional utilizada con una sentencia LIST y una cláusula WHERE

```
%DEFINE{  
%list " AND " where_list  
where_list = ? "custid = $(cust_inp)"  
where_list = ? "product_name LIKE '$(prod_inp)%'"  
where_clause = ? "WHERE $(where_list)"  
%}  
  
%FUNCTION(DTW_SQL) mySelect() {  
    SELECT * FROM proddtable $(where_clause)  
%}
```

Las variables condicionales y LIST resultan más efectivas cuando se utilizan conjuntamente. El ejemplo anterior muestra el modo de configurar una cláusula WHERE en el bloque DEFINE. Las variables *cust_inp* y *prod_inp* son variables de entrada HTML que se pasan desde el navegador Web, normalmente en un formato HTML. La variable *where_list* es una variable LIST compuesta por dos sentencias condicionales, en las que cada sentencia contiene una variable procedente del navegador Web.

Si el navegador Web devuelve para ambas variables los valores *cust_inp* y *prod_inp*, por ejemplo, IBM y 755C, la variable *where_clause* es:

```
WHERE custid = IBM AND product_name  
LIKE '755C'
```

Si cualquiera de las dos variables *cust_inp* o *prod_inp* es nula o no está definida, la cláusula WHERE cambia para omitir el valor nulo. Por ejemplo, si *prod_inp* es nula, la cláusula WHERE es:

```
WHERE custid = IBM
```

Si ambos valores son nulos o no están definidos, la variable `where_clause` es nula y no aparece ninguna cláusula `WHERE` en consultas SQL que contienen `$(where_clause)`.

Variables de entorno

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Las variables de entorno le permiten utilizar la construcción del lenguaje ENVVAR de Net.Data para hacer referencia a las variables de entorno que existen en el proceso en el que se está ejecutando Net.Data.

Ejemplo 1: Se asigna a una variable el valor de una variable de entorno

```
%define SERVER_NAME=%ENVVAR
```

```
...
```

```
The server is $(SERVER_NAME)
```

La variable de entorno `SERVER_NAME` tiene el valor del nombre del servidor actual que, en este ejemplo, es `www.ibm.com`.

```
The server is www.ibm.com
```

Consulte el apartado “Sentencia ENVVAR” en la página 17 para obtener más información sobre la sentencia ENVVAR.

Variables ejecutables

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Las variables ejecutables le permiten invocar otros programas desde una referencia a variable utilizando la característica de variable ejecutable. Una variable ejecutable se define en una macro Net.Data utilizando el elemento del lenguaje EXEC. Para obtener más información sobre el elemento del lenguaje EXEC, consulte el apartado “Bloque o sentencia EXEC” en la página 18.

Cuando Net.Data encuentra una variable ejecutable en una macro, busca el programa ejecutable al que se hace referencia utilizando el método siguiente:

1. Busca la `EXEC_PATH` en el archivo de inicialización de Net.Data. Consulte el capítulo sobre configuración en el manual *Guía de administración y programación de Net.Data* para obtener más información sobre `EXEC_PATH`.
2. Si Net.Data no localiza el programa, busca los directorios definidos por el sistema. Si localiza el programa ejecutable, Net.Data ejecuta el programa.

Ejemplo 1: Una definición de variable ejecutable

```
%DEFINE runit=%exec "testProg"
```

La variable *runit* se ha definido de modo que ejecute el programa ejecutable *testProg*; *runit* se convierte en una variable ejecutable.

Net.Data ejecuta el programa ejecutable cuando se encuentra una referencia a una variable ejecutable en una macro Net.Data. Por ejemplo, el programa *testProg* se ejecutará cuando en una referencia a una variable ejecutable de una macro Net.Data se haga referencia a la variable *runit*.

Un método sencillo es el de hacer referencia a una variable ejecutable desde otra definición de variable. El ejemplo 2 muestra este método. La variable *date* se define como variable ejecutable y *dateRpt* se define a continuación como una referencia a variable, que contiene la variable ejecutable.

Ejemplo 2: Variable ejecutable como una referencia a variable

```
%DEFINE date=%exec "date"  
%DEFINE dateRpt="Today is $(date)"
```

Cuando Net.Data resuelve la referencia a variable *\$(dateRpt)*, Net.Data busca la fecha ejecutable, ejecuta el programa y devuelve:

```
Today is Tue 11-07-1995
```

Una variable ejecutable nunca se establece en el valor de la salida del programa ejecutable al que llama. Utilizando el ejemplo anterior, el valor de fecha es nulo. Si lo utiliza en una llamada a función DTW_ASSIGN para asignar su valor a otra variable, el valor de la variable nueva después de la asignación es asimismo nulo. La única finalidad de una variable ejecutable es invocar el programa que define.

También puede pasar parámetros al programa que ha de ejecutarse especificándolos con el nombre del programa en la definición de variable.

Ejemplo 3: Variables ejecutables con parámetros

```
%DEFINE mph=%exec "calcMPH $(distance) $(time)"
```

Los valores de *distance* y *time* se pasan al programa *calcMPH*.

Variables ocultas

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Con las variables ocultas, puede hacer referencia a las variables ocultando al mismo tiempo el valor real de la variable en la fuente HTML. Para utilizar variables ocultas:

1. Defina una variable para cada serie que desee ocultar.
2. En el bloque HTML en el que se hace referencia a las variables, utilice dos símbolos del dólar en vez de un único símbolo de dólar para hacer referencia a las variables. Por ejemplo, \$\$*(X)* en vez de \$(*X*).

No haga referencia a las variables ocultas con nombres de variables creados dinámicamente.

Ejemplo 1: Variables ocultas en un formato HTML

```
%HTML(INPUT) {
<form ...>
<p>Select fields to view:
<select name="field">
<option value="$$(name)"> Name
<option value="$$(addr)"> Address
.
.
.
</form>
%}

%DEFINE{
name="customer.name"
addr="customer.address"
%}

%FUNCTION(DTW_SQL) mySelect() {
  SELECT $(Field) FROM customer
%}
.
.
.
```

Cuando se visualiza el formato HTML en un navegador Web, \$\$*(name)* y \$\$*(addr)* se sustituyen por \$(*name*) y \$(*addr*) respectivamente, por lo que los nombres de columna y tabla reales nunca aparecen en el formato HTML. Cuando el cliente somete el formato, se llama al bloque HTML(REPORT). Cuando @mySelect() llama al bloque FUNCTION, \$(*Field*) se sustituye en la sentencia de SQL por customer.name o customer.addr en la consulta SQL.

Variables de lista

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Puede utilizar variables de lista para crear una serie delimitada de valores. Pueden resultar especialmente útiles para ayudarle a crear una consulta SQL con varios elementos como los que contienen algunas cláusulas WHERE o HAVING.

Los blancos son significativos. Normalmente es deseable tener un espacio en blanco a ambos lados del valor. La mayoría de las consultas utilizan operadores Booleanos o matemáticos (por ejemplo, AND, OR y >). Consulte el apartado “Sentencia LIST” en la página 47 para conocer la sintaxis y obtener más información.

Ejemplo 1: Utilización de variables condicionales, ocultas y de lista

```
%HTML(INPUT){
<form method="post" action="/cgi-bin/db2www/example2.max/report">
Select one or more cities:<br />
<input type="checkbox" name="conditions" value="$(cond1)" />Sao Paulo<br />
<input type="checkbox" name="conditions" value="$(cond2)" />Seattle<br />
<input type="checkbox" name="conditions" value="$(cond3)" />Shanghai<br />
<input type="submit" value="submit query" />
</form>
%}

%DEFINE{
DATABASE="custcity"
%LIST " OR " conditions
cond1="cond1='Sao Paolo'"
cond2="cond2='Seattle'"
cond3="cond3='Shanghai'"
whereClause= ? "WHERE $(conditions)" %}
%FUNCTION(DTW_SQL) mySelect(){ SELECT name, city FROM citylist
$(whereClause) %}
%HTML(REPORT){ @mySelect() %}
```

Si no se selecciona ningún recuadro en el formato HTML, *conditions* es nula, por lo que *whereClause* es asimismo nula en la consulta. En caso contrario, *whereClause* tiene los valores seleccionados separados mediante el operador Booleano OR. Por ejemplo, si se seleccionan las tres ciudades, la consulta de SQL es:

```
SELECT name, city FROM citylist
WHERE cond1='Sao Paolo' OR cond2='Seattle' OR cond3='Shanghai'
```

Ejemplo 2: Separadores de valores

```
%DEFINE %LIST " | " VLIST
%REPORT{
%ROW{
<em>$(ROW_NUM):</em> $(VLIST)
%}
%}
```

En este ejemplo, la variable de proceso de tabla VLIST utiliza comillas dobles y una barra OR, (|), como separador de valores. La serie de valores está separada por el valor entre comillas.

Variables de tabla

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

La variable de tabla define un conjunto de datos relacionados. Contiene un conjunto de filas y columnas incluida una fila de cabeceras de columna. Utilice las variables de tabla para pasar grupos de valores a una función. Puede hacer referencia a los elementos individuales de una tabla (las filas) en un bloque REPORT de una función o utilizando las funciones de tabla incorporadas. Las variables de tabla se utilizan a menudo para la salida de una función de SQL y la entrada a un informe, pero también puede pasarlos como parámetros IN, OUT o INOUT para cualquier función que no sea SQL. Las tablas sólo pueden pasarse a funciones de SQL como parámetros OUT. Consulte el apartado “Sentencia TABLE” en la página 65 para conocer la sintaxis y obtener más información.

Cuando se hace referencia a una variable TABLE, Net.Data visualiza el contenido de la tabla como una tabla de caracteres planos o como una tabla HTML en el caso de que la variable DTW_HTML_TABLE se haya establecido en YES. Cuando se ejecute un bloque XML, Net.Data devuelve una estructura RowSet (consulte el apartado “Bloque XML” en la página 71).

Ejemplo 1: Conjunto de resultados SQL que se pasa a un programa REXX

```
%DEFINE{
    DATABASE = "iddata"
    MyTable = %TABLE(ALL)
    DTW_DEFAULT_REPORT = "NO"
}%

%FUNCTION(DTW_SQL) Query(OUT table) {
select * from survey
}%

%FUNCTION(DTW_REXX) showTable(INOUT table) {
    Say 'Number of Rows: 'table_ROWS
    Say 'Number of Columns: 'table_COLS
    do j=1 to table_COLS
        Say "Here are all of the values for column " table_N.j ":"
        do i = 1 to table_ROWS
            Say "<b>"i"</b>: " table_V.i.j
        end
    end
}%
```



```
%HTML (report){
<html>
<pre>
@Query(MyTable)
<p>
@showTable(MyTable)
</pre>
</html>
%}
```

El bloque HTML REPORT llama a una consulta SQL, guarda el resultado en una variable de tabla y después pasa la variable a una función REXX.

Variables Net.Data de proceso de tablas

Net.Data define estas variables para su utilización en los bloques REPORT y ROW, a menos que se indique lo contrario. Utilice estas variables para hacer referencia a los valores que devuelven las consultas.

- “Nn” en la página 84
- “NLIST” en la página 85
- “NUM_COLUMNS” en la página 86
- “NUM_ROWS” en la página 87
- “ROW_NUM” en la página 88
- “TOTAL_ROWS” en la página 89
- “V_Nombrecolumna” en la página 90
- “VLIST” en la página 91
- “Vn” en la página 92

Nn

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

El nombre de la columna que devuelve una llamada a función o consulta para una columna n.

Puede hacer referencia a Nn en bloques REPORT y ROW.

Ejemplos

Ejemplo 1: Una referencia a variable para un nombre de columna

The name of column 2 is \$(N2).

Ejemplo 2: Guarda el valor de un nombre de columna para su utilización fuera de un bloque REPORT que utiliza DTW_ASSIGN

```
%define col1=""
...
%function (DTW_SQL) myfunc() {
    select * from atable
    %report {
        @dtw_assign(col1, N1)
        %row{ %}
    %}
%}

%html(report) {
@myfunc()
The column name for the first column is $(col1)
%}
```

Este ejemplo muestra el modo en que puede utilizar esta variable fuera del bloque REPORT utilizando DTW_ASSIGN. Para obtener más información, consulte el apartado “DTW_ASSIGN” en la página 208.

Ejemplo 3: Nn dentro de una tabla HTML para definir nombres de columna

```
%REPORT{
<h2>Product directory</h2>
<table border=1 cellpadding=3>
<tr><td>$(N1)</td><td>$(N2)</td><td>$(N3)</td>
%ROW{
<tr><td>$(V1)</td><td>$(V2)</td><td>$(V3)</td>
%}
</table>

%}
```

NLIST

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Contiene una lista de todos los nombres de columna a partir del resultado de una consulta o llamada de función. El separador por omisión es un espacio.

Puede hacer referencia a NLIST en los bloques REPORT y ROW.

Ejemplos

Ejemplo 1: Lista de nombres de columna con ALIGN

```
%DEFINE ALIGN="YES"
...
%FUNCTION (DTW_SQL) myfunc() {
select * from MyTable
%report{
Your query was on these columns: $(NLIST).
%row {
...
%}
%}
%}
```

La lista de nombres de columna utiliza un espacio entre nombres de columna con ALIGN establecido en YES.

Ejemplo 2: Una variable %LIST para cambiar el separador a " | "

```
%DEFINE %LIST " | " NLIST
...
%FUNCTION (DTW_SQL) myfunc() {
select * from MyTable
%report{
Your query was on these columns: $(NLIST).
%row {
...
%}
%}
%}
```

NUM_COLUMNS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

El número de columnas de tabla que Net.Data está procesando en el bloque de informe; las columnas las devuelve una consulta o llamada a función.

Puede hacer referencia a NUM_COLUMNS en bloques REPORT y ROW.

Ejemplos

Ejemplo 1: NUM_COLUMNS se utiliza como una referencia a variable con NLIST

```
%REPORT{  
Your query result has $(NUM_COLUMNS) columns: $(NLIST).  
...  
%}
```

NUM_ROWS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

El número de filas de la tabla que Net.Data está procesando en el bloque REPORT. El número de filas resulta afectado por el valor del parámetro *límite superior* definido por la tabla de Net.Data que posee los datos. Por ejemplo, si *límite superior* se establece en 30, pero la sentencia SELECT devuelve 1000 filas, el valor de NUM_ROWS es 30. Adicionalmente, si *límite superior* se establece en 30 y la sentencia SELECT devuelve 20 filas, NUM_ROWS es igual a 20. Consulte el apartado “Sentencia TABLE” en la página 65 para obtener más información sobre la sentencia TABLE y el parámetro *límite superior*.

NUM_ROWS no resulta afectado por el valor de START_ROW_NUM en tanto START_ROW_NUM no se pase al entorno de lenguaje. Por ejemplo, si START_ROW_NUM se establece en 5 (especificando que la tabla que se visualiza en la página Web debería poblarse comenzando a partir de la fila 5) y la sentencia SELECT devuelve 25 filas, NUM_ROWS se establece en 25 y no en 21. Las primeras cuatro filas se descartan de la tabla, pero se incluyen en el valor de NUM_ROWS. Sin embargo, si START_ROW_NUM se pasa al entorno de lenguaje, NUM_ROWS sólo contendrá el número de filas a partir de la fila que especifica START_ROW_NUM. En el ejemplo anterior, NUM_ROWS se establecerá en 21.

Puede hacer referencia a NUM_ROWS en bloques REPORT y ROW.

Ejemplos

Ejemplo 1: Visualiza el número de nombres que se procesan en el bloque REPORT

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"

%REPORT{
<h2>E-mail directory</h2>
<ul>
%ROW{
<li>Name: <a href="mailto:$(V1)">$(V2)</a><br />
Location: $(V3)
%}
</ul>
Names displayed: $(NUM_ROWS)<br />
Names found: $(TOTAL_ROWS)
%}
```

ROW_NUM

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Una variable de tabla cuyo valor Net.Data aumenta cada vez que se procesa una fila de una tabla de Net.Data. La variable actúa como contador y su valor es el número de la fila actual que se está procesando.

RPT_MAX_ROWS puede afectar al valor de ROW_NUM. Por ejemplo, si hay 100 filas en una tabla y ha establecido RPT_MAX_ROWS en 20, el valor final de ROW_NUM es 20, ya que la fila 20 fue la última fila procesada.

Sólo puede hacer referencia a ROW_NUM desde un bloque ROW.

Ejemplos

Ejemplo 1: Rellena una columna en la salida de HTML utilizando ROW_NUM para etiquetar cada una de las filas de la tabla

```
%REPORT{
<table border=1>
<tr><td> Row Number </td> <td> Customer </td>
%ROW{
<tr><td> $(ROW_NUM) </td> <td> $(V_custname) </td>
%}
</table>
%}
```

El bloque REPORT genera una tabla como la que se muestra a continuación.

Número de fila	Cliente
1	Jane Smith
2	Jon Chiu
3	Frank Nguyen
4	Mary Nichols

TOTAL_ROWS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

El número total de filas que devuelve una consulta, sin tener en cuenta el valor de parámetro *limite_superior* para la construcción de lenguaje TABLE. Por ejemplo, si RPT_MAX_ROWS se establece de modo que se visualice un máximo de 20 filas, pero la consulta devuelve 100 filas, esta variable se establece en 100 después del proceso de ROW.

Diferencias entre los sistemas operativos:

- En el sistema operativo OS/400, se puede hacer referencia a esta variable en cualquier parte de un bloque REPORT o ROW.
- En los sistemas operativos OS/390, OS/2, Windows NT y UNIX, sólo se puede hacer referencia a esta variable en el pie de página REPORT.

Restricción en entornos de lenguajes: Utilice esta variable únicamente con los entornos de lenguaje de base de datos siguientes:

- SQL
- ODBC
- Oracle

Obligatorio: Debe establecer DTW_SET_TOTAL_ROWS en YES para utilizar esta variable. Consulte el apartado “DTW_SET_TOTAL_ROWS” en la página 115 para obtener más información.

Ejemplos

Ejemplo 1: Visualiza el número total de nombres que se ha encontrado

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"

%REPORT{
<h2>E-mail directory</h2>
<ul>
%ROW{
<li>Name: <a href="mailto:$(V1)">$(V2)</a><br />
Location: $(V3)
%}
</ul>
Names found: $(TOTAL_ROWS)
%}
```

V_Nombrecolumna

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

El valor del nombre de columna especificado para la fila actual. La variable no se establece para los nombres de columna no definidos. Una consulta que contenga dos nombres de columna iguales proporcionará resultados imprevisibles. Puede utilizar una cláusula AS en SQL para red denominar los nombres de columna duplicados.

Sólo puede hacer referencia a V_Nombrecolumna en un bloque ROW.

Net.Data asigna la variable a cada columna de la tabla; utilice la variable en una referencia a variable, especificando el nombre de la columna a la que desea hacer referencia. Para utilizar esta variable fuera del bloque, asigne el valor de V_Nombrecolumna a una variable global definida con anterioridad o a una variable de parámetro de función OUT o INOUT.

Valores

V_Nombrecolumna

Tabla 1. Valores de V_Nombrecolumna

Valores	Descripción
Nombrecolumna	El nombre de columna de la fila actual de la tabla de la base de datos.

Ejemplos

Ejemplo 1: Utilización de V_Nombrecolumna como una referencia a variable

```
%FUNCTION(DTW_SQL) myQuery() {  
  SELECT NAME, ADDRESS from $(qtable)  
  %REPORT{  
  
    %ROW{  
  
      Value of NAME column in row $(ROW_NUM) is $(V_NAME).<br />  
    %}  
  %}  
}
```


VLIST

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Lista de todos los valores de campo para la fila actual que se está procesando en un bloque ROW.

Sólo puede hacer referencia a VLIST en un bloque ROW. El separador por omisión es un espacio.

Net.Data asigna la variable a cada una de las filas de la tabla; haga referencia a la variable para obtener los valores de todos los campos de la fila actual. Para utilizar esta variable fuera del bloque, asigne el valor de VLIST a una variable global definida con anterioridad o a una variable de parámetro de función OUT o INOUT.

Ejemplos

Ejemplo 1: Utilización de códigos de lista para visualizar los resultados de la consulta

```
%DEFINE ALIGN="YES"

%REPORT{
Here are the results of your query:
<ol>
%ROW{
<li>$(VLIST)
%}
</ol>
%}
```

Ejemplo 2: Utilización de una variable de lista para cambiar el separador a <p>

```
%DEFINE %LIST "<p>" VLIST

%REPORT{
Here are the results of your query:
%ROW{
<hr>$(VLIST)
%}
%}
```

V_n

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

El valor del número *n* de columnas especificado para la fila actual.

Sólo puede hacer referencia a *V_n* en un bloque ROW.

Net.Data asigna la variable a cada campo de la tabla; utilice la variable en una referencia a variable, especificando el número del campo al que desea hacer referencia. Para utilizar esta variable fuera del bloque, asigne el valor de *V_n* a una variable global definida con anterioridad o a una variable de parámetro de función OUT o INOUT.

Ejemplos

Ejemplo 1: Informe que muestra una tabla HTML

```
%REPORT{
<h2>E-mail directory</h2>
<table border=1 cellpadding=3>
<tr><td>Name</td><td>E-mail
address</td><td>Location</td>
%ROW{
<tr><td>$(V1)</td>
<td><a href="mailto:$(V2)">$(V2)</a></td>
<td>$(V3)</td>
%}
</table>
%}
```

La segunda columna muestra la dirección de correo electrónico (e-mail). Puede enviar un mensaje a la persona pulsando en el enlace.

Variables Net.Data de informes

Estas variables le ayudan a personalizar los informes. Cada variable tiene un valor por omisión. Puede alterar temporalmente el valor por omisión asignando un valor nuevo a la variable.

- “ALIGN” en la página 94
- “DTW_DEFAULT_REPORT” en la página 95
- “DTW_HTML_TABLE” en la página 96
- “RPT_MAX_ROWS” en la página 97
- “START_ROW_NUM” en la página 99

ALIGN

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Controla los espacios iniciales y de cola utilizados con las variables de proceso de tabla NLIST y VLIST.

Sugerencia sobre rendimiento Sólo ha de utilizarse ALIGN cuando sea necesario ya que necesita que Net.Data determine la longitud de columna máxima para todas las columnas de la tabla a fin de calcular los requisitos de relleno. Este proceso puede afectar al rendimiento.

Cuando se establece en YES, ALIGN proporciona relleno para alinear las variables de proceso de tablas para su visualización. Si desea intercalar resultados de consulta en enlaces HTML o en acciones de formato, utilice el valor por omisión NO para impedir que Net.Data intercale entre las variables de informe espacios iniciales y de cola.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

ALIGN="YES" | "NO"

Tabla 2. Valores de ALIGN

Valores	Descripción
YES	Net.Data añade espacios iniciales y de cola a las variables de informes con espacios, a fin de alinearlas para su visualización.
NO	Net.Data no añade espacios iniciales y de cola. NO es el valor por omisión.

Ejemplos

Ejemplo 1: Utilización de la variable ALIGN para separar cada una de las columnas por medio de un espacio

```
%DEFINE ALIGN="YES"
```

```
<p>Your query was on these columns: $(NLIST)
```

DTW_DEFAULT_REPORT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Determina si Net.Data genera un informe por omisión para las funciones que no tengan un bloque REPORT. Cuando esta variable se establece en YES, Net.Data genera el informe por omisión. Cuando se establece en NO, Net.Data suprime la generación del informe por omisión. La supresión del informe por omisión es útil, por ejemplo, si recibe el resultado de una llamada función de una variable de tabla y desea pasar el resultado a una función diferente para su proceso.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

DTW_DEFAULT_REPORT="YES"|"NO"|"MULTIPLE"

Tabla 3. Valores de DTW_DEFAULT_REPORT

Valores	Descripción
YES	Net.Data genera el informe por omisión para las funciones sin bloques REPORT y visualiza el resultado en el navegador. YES es el valor por omisión. Para OS/390: Net.Data genera informes por omisión para cada tabla de salida o conjunto de resultados que no esté asignado a un bloque REPORT.
NO	Net.Data descarta el informe por omisión para las funciones sin bloques REPORT.
MULTIPLE	Net.Data genera informes por omisión para tablas de salida o conjuntos de resultados que no estén asignados a un bloque REPORT, en las funciones con múltiples bloques REPORT. Para OS/390: MULTIPLE no está soportado

Ejemplos

Ejemplo 1: Alteración temporal del informe por omisión generado por Net.Data

```
%DEFINE  
DTW_DEFAULT_REPORT="NO"
```

DTW_HTML_TABLE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Visualiza los resultados en una tabla HTML en vez de visualizar la tabla en un formato de tipo texto (es decir, utilizando los códigos TABLE en vez de los códigos de PRE). Esta variable no tiene efecto en la salida de un bloque XML.

El código TABLE generado incluye una especificación de margen y de relleno de celdas.

```
<table border cellpadding=2>
```

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

```
DTW_HTML_TABLE="YES"|"NO"
```

Tabla 4. Valores de DTW_HTML_TABLE

Valores	Descripción
YES	Visualiza datos de tabla utilizando códigos de tabla HTML.
NO	Visualiza datos de tabla en formato de texto, utilizando códigos PRE. NO es el valor por omisión.

Ejemplos

Ejemplo 1: Visualiza los resultados de una función de SQL con códigos HTML

```
%DEFINE DTW_HTML_TABLE="YES"

%FUNCTION(DTW_SQL){
SELECT NAME, ADDRESS FROM $(qTable)
%}
```

RPT_MAX_ROWS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Especifica el número de filas de una tabla que se procesa en un bloque REPORT de función o durante la generación de un informe por omisión en el caso de que no se especifique un bloque REPORT.

Los entornos de lenguaje de base de datos utilizan esta variable para limitar el número de filas devuelto, que puede aumentar sustancialmente el rendimiento para grandes conjuntos de resultados. Utilice este variable con START_ROW_NUM para dividir consultas con grandes conjuntos de resultados en tablas más pequeñas, cada una con su propia página HTML.

Especifique el valor de esta variable utilizando una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

RPT_MAX_ROWS="ALL" | "0" | "número"

Tabla 5. Valores de RPT_MAX_ROWS

Valores	Descripción
ALL	Indica que no hay un límite en el número de filas a visualizar en una tabla generada por medio de una llamada a función. Se visualizarán todas las filas.
0	Especifica que se visualizarán todas las filas de la tabla. Este valor es igual que especificar ALL.
número	Un entero positivo que indica el número máximo de filas a visualizar en una tabla generada por medio de una llamada a función. Si el bloque FUNCTION contiene un bloque REPORT y ROW, este número especifica el número de veces que se ejecuta el bloque ROW.

Ejemplos

Ejemplo 1: Define RPT_MAX_ROWS en una sentencia DEFINE

```
%DEFINE RPT_MAX_ROWS="20"
```

El método anterior limita a 20 filas el número de filas que devuelve cualquier función.

Ejemplo 2: Utiliza la entrada HTML para definir la variable con un formato HTML

```
Maximum rows to return (0  
for no limit):  
<input type="text" name="rpt_max_rows" size=3 />
```

Las líneas del ejemplo anterior pueden colocarse en un código FORM para permitir a los usuarios de la aplicación establecer el número de filas que desean que devuelva una consulta.

START_ROW_NUM

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Especifica el número de la fila inicial de una tabla que se procesará en un bloque REPORT de función o durante la generación de un informe por omisión en el caso de que no se especifique un bloque REPORT.

Los entornos de lenguaje de base de datos utilizan esta variable para determinar la fila inicial del conjunto de resultados para comenzar el proceso. Para mejorar sustancialmente el rendimiento para grandes conjuntos de resultados, utilice esta variable con RPT_MAX_ROWS para dividir las consultas con conjuntos de resultados grandes en tablas más pequeñas.

Usuarios de OS/400, Windows NT, OS/2 y UNIX: Para pasar esta variable al entorno de lenguaje, inclúyala como un parámetro IN en la sentencia ENVIRONMENT del entorno de lenguaje de base de datos del archivo de inicialización de Net.Data. Para obtener más información sobre la sentencia de entorno de lenguaje de base de datos, consulte el capítulo sobre configuración de la publicación *Guía de administración y programación de Net.Data* para el sistema operativo.

Usuarios de OS/390: START_ROW_NUM se pasa implícitamente a los entornos de lenguaje de base de datos cuando está definida en la macro.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

START_ROW_NUM="número"

Tabla 6. Valores de START_ROW_NUM

Valores	Descripción
número	<p>Un entero positivo que indica el número de fila con la que comenzar a visualizar un informe. El valor por omisión es 1.</p> <p>Si START_ROW_NUM se especifica en una sentencia de entorno del entorno de lenguaje de base de datos del archivo de inicialización, este número especifica el número de fila del conjunto de resultados que procesa el entorno de lenguaje de base de datos.</p> <p>Si START_ROW_NUM no se pasa al entorno de lenguaje, este número especifica el número de fila de la tabla de Net.Data utilizada para visualizar un informe.</p>

Ejemplos

Ejemplo 1: Desplazamiento con los botones Siguiente y Anterior del formato HTML

```
%define {
    DTW_HTML_TABLE      = "YES"
    START_ROW_NUM        = "1"
    RPT_MAX_ROWS         = "10"
    totalSize            = ""
    includeNext           = "YES"
    includePrev          = "YES"
    includeLast          = "YES"
    includeFirst         = "YES"
}%}

%function(DTW_SQL) myQuery(){
    select * from NETDATADEV.CUSTOMER
}%}

%function(DTW_SQL) count(OUT size){
    select count(*) from NETDATADEV.CUSTOMER
    %report{
        %row{
            @DTW_ASSIGN(size,V1)
        }
    }
}%}

%html(report) {
    %{ get the total number of records if we haven't already %}
    %if (totalSize == "")
        @count(totalSize)
    %endif

    %{ set START_ROW_NUM based on the button user clicked %}
    %if (totalSize <= RPT_MAX_ROWS)
        %{ there's only one page of data %}
    %endif
}
```

```

@DTW_ASSIGN(START_ROW_NUM, "1")
@DTW_ASSIGN(includeFirst, "NO")
@DTW_ASSIGN(includeLast, "NO")
@DTW_ASSIGN(includeNext, "NO")
@DTW_ASSIGN(includePrev, "NO")
%elif (submit == "First Page" || submit == "")
%{ first time through or user selected "First Page" button %}
@DTW_ASSIGN(START_ROW_NUM, "1")
@DTW_ASSIGN(includePrev, "NO")
@DTW_ASSIGN(includeFirst, "NO")
%elif (submit == "Last Page")
%{ user selected "Last Page" button %}
@DTW_SUBTRACT(totalSize, RPT_MAX_ROWS, START_ROW_NUM)
@DTW_ADD(START_ROW_NUM, "1", START_ROW_NUM)
@DTW_ASSIGN(includeLast, "NO")
@DTW_ASSIGN(includeNext, "NO")
%elif (submit == "Next")
%{ user selected "Next" button %}
@DTW_ADD(START_ROW_NUM, RPT_MAX_ROWS, START_ROW_NUM)
%if (@DTW_rADD(START_ROW_NUM, RPT_MAX_ROWS) > totalSize)
@DTW_ASSIGN(includeNext, "NO")
@DTW_ASSIGN(includeLast, "NO")
%endif
%elif (submit == "Previous")
%{ user selected "Previous" button %}
@DTW_SUBTRACT(START_ROW_NUM, RPT_MAX_ROWS, START_ROW_NUM)
%if (START_ROW_NUM <= "1" )
@DTW_ASSIGN(START_ROW_NUM, "1")
@DTW_ASSIGN(includePrev, "NO")
@DTW_ASSIGN(includeFirst, "NO")
%endif
%endif
%endif

%{ run the query to get the data %}
@myQuery()

%{ output the correct buttons at the bottom of the report %}
<center>
<form method="POST" action="report">
  <input name="START_ROW_NUM" type="hidden" value="$(START_ROW_NUM)" />
  <input name="totalSize" type="hidden" value="$(totalSize)" />
  %if (includeFirst == "YES" )
  <input name="submit" type="submit" value="First Page" />
  %endif
  %if (includePrev == "YES" )
  <input name="submit" type="submit" value="Previous" />
  %endif
  %if (includeNext == "YES" )
  <input name="submit" type="submit" value="Next" />
  %endif
  %if (includeLast == "YES" )
  <input name="submit" type="submit" value="Last Page" />

```

```
%endif  
</form>  
</center>  
%}
```

Variables Net.Data de entorno de lenguaje

Utilice estas variables con funciones para ayudarle a personalizar el modo en que los entornos de lenguaje procesan los bloques FUNCTION por medio de entornos de lenguaje. Cada variable tiene un valor por omisión. Puede alterar temporalmente el valor por omisión asignando un valor nuevo a la variable.

- “DATABASE” en la página 104
- “DB_CASE” en la página 106
- “DB2PLAN” en la página 107
- “DB2SSID” en la página 108
- “DTW_APPLET_ALTTEXT” en la página 109
- “DTW_EDIT_CODES” en la página 110
- “DTW_PAD_PGM_PARMS” en la página 112
- “DTW_SAVE_TABLE_IN” en la página 114
- “DTW_SET_TOTAL_ROWS” en la página 115
- “LOCATION” en la página 119
- “LOGIN” en la página 120
- “NULL_RPT_FIELD” en la página 121
- “PASSWORD” en la página 122
- “SHOWSQL” en la página 123
- “SQL_STATE” en la página 125
- “TRANSACTION_SCOPE” en la página 126

DATABASE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X		X	X	X	X

Propósito

Especifica la base de datos o fuente de datos ODBC a la que se accede al llamar a una función de base de datos. Esta variable puede cambiarse múltiples veces en una macro para acceder a diferentes bases de datos o fuentes de datos ODBC.

Sistema operativo OS/400: Este parámetro es opcional. Por omisión, Net.Data, especifica DATABASE="*LOCAL"; el entorno de lenguaje DTW_SQL utiliza la entrada de directorio de base de datos relacional local.

Sistemas operativos Windows NT, OS/2 y UNIX: Defina esta variable antes de llamar a cualquier función de base de datos, excepto al utilizar el entorno de lenguaje DTW_ORA (Oracle). Adicionalmente, debe utilizar Live Connection (Conexión en directo) al acceder a diferentes bases de datos desde el mismo bloque HTML y a través del mismo entorno de lenguaje.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

DATABASE="*nombredb*"

Tabla 7. Valores de DATABASE

Valores	Descripción
<i>nombredb</i>	El nombre de la base de datos a la que se conecta Net.Data.

Ejemplos

Ejemplo 1: Especifica la conexión a la base de datos CELDIAL para cualquier operación de SQL

```
%DEFINE DATABASE="CELDIAL"

%FUNCTION (DTW_SQL) getRpt() {
SELECT * FROM customer
%}

%HTML (report) {
%INCLUDE "rpthead.htm"
@getRpt()
%INCLUDE "rptfoot.htm"
%}
```

Se accede a la base de datos CELDIAL cuando se llama a la función getRpt.

Ejemplo 2: Altera temporalmente las definiciones de DATABASE anteriores con DTW_ASSIGN

```
%DEFINE  
DATABASE="DB2C1"  
...  
%HTML(monthRpt){  
@DTW_ASSIGN(DATABASE, "DB2D1")  
%INCLUDE "rpthead.htm"  
@getRpt()  
%INCLUDE "rptfoot.htm"  
%}
```

El bloque HTML consulta la base de datos DB2D1, independientemente del valor anterior de DATABASE.

DB_CASE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Especifica si se han de utilizar mayúsculas o minúsculas en los en los mandatos SQL y convierte todos los caracteres a mayúsculas o minúsculas. Si no se define esta variable, la acción por omisión no convierte los caracteres de mandatos SQL.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

DB_CASE="UPPER" | "LOWER"

Tabla 8. Valores de DB_CASE

Valores	Descripción
UPPER	Convierte en mayúsculas todos los caracteres de mandatos SQL.
LOWER	Convierte en minúsculas todos los caracteres de mandatos SQL.

Ejemplos

Ejemplo 1: Especifica que todos los mandatos SQL estén en mayúsculas

```
%DEFINE DB_CASE="UPPER"
```


DB2PLAN

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
				X				

Propósito

Asigna un plan para establecer una conexión con un subsistema DB2 local. La variable especifica el nombre de un plan del entorno de lenguaje SQL de Net.Data en el subsistema DB2 local al que accederá Net.Data.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Requisito: Para que surta efecto en la macro el valor de esta variable, debe listarse en la sentencia ENVIRONMENT para el entorno de lenguaje SQL. Consulte la publicación *Administration and Programming Guide* del sistema operativo para obtener más información.

Valores

DB2PLAN="*nombre_plan*"

Tabla 9. Valores de DB2PLAN

Valores	Descripción
<i>nombre_plan</i>	El nombre del plan de DB2. El nombre puede tener ocho caracteres o menos.

Ejemplos

Ejemplo 1: Especifica el plan de la sentencia DEFINE

```
%DEFINE DB2PLAN="DTWGAV22"
```

DB2SSID

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
				X				

Propósito

Establece una conexión con un subsistema DB2 local. La variable especifica el ID de subsistema del subsistema DB2 local al que accederá Net.Data. Sólo se admite una conexión de base de datos local para cada macro.

Requisito: Para que surta efecto el valor de esta variable en la macro, debe listarse en la sentencia ENVIRONMENT del entorno de lenguaje SQL. Consulte la publicación *Administration and Programming Guide* del sistema operativo para obtener más información.

Valores

DB2PLAN="*id_subsistema*"

Tabla 10. Valores de DB2SSID

Valores	Descripción
<i>id_subsistema</i>	El nombre del subsistema DB2. El nombre puede tener ocho caracteres o menos.

Ejemplos

Ejemplo 1: Especifica un ID de subsistema en la sentencia DEFINE

```
%DEFINE DB2SSID="DBNC"
```

DTW_APPLET_ALTTEXT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Visualiza texto y códigos HTML para los navegadores que no reconocen el código APPLET y se utiliza con el entorno de lenguaje de Applet.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

DTW_APPLET_ALTTEXT="*texto_y_markup*"

Tabla 11. Valores de DTW_APPLET_ALTTEXT

Valores	Descripción
<i>texto_y_markup</i>	Texto y markup que debería visualizarse para los navegadores que no reconocen el código APPLET.

Ejemplos

Ejemplo 1: Texto alternativo que indica una restricción de navegador Web

```
%DEFINE
```

```
DTW_APPLET_ALTTEXT="

Sorry, your browser is not  
java-enabled.</p>"


```

DTW_EDIT_CODES

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito
Convierte los tipos de datos NUMERIC, DECIMAL, INTEGER y SMALLINT que se devuelven como resultado de una operación de SQL para el entorno de lenguaje DTW_SQL. La variable DTW_EDIT_CODES es una serie de caracteres que se corresponde con las columnas resultantes de la tabla que creará DTW_SQL LE; por ejemplo, el quinto carácter de DTW_EDIT_CODES se aplicará a la quinta columna del conjunto de resultados en el caso de que dicha columna sea de uno de los tipos soportados. Este único carácter puede ser cualquiera de los códigos de edición proporcionados por el sistema soportado que se definen en la publicación *Data Description Specification Reference*.

Por ejemplo, un campo DECIMAL(6,0) se representaría normalmente como la serie de caracteres '112698'. Especificando un código de edición de 'Y' para dicha columna en la variable DTW_EDIT_CODES, se visualiza la columna correspondiente de la tabla resultante como una serie de caracteres que representa la fecha '11/26/98'.

Sugerencia: Aplicar un código de edición proporcionado por el usuario a una columna que dé como resultado una serie de caracteres con caracteres no numéricos (por ejemplo, comas o símbolos monetarios) puede ocasionar errores de sintaxis en el caso de que la serie de caracteres se devuelva al servidor para su proceso posterior en una macro Net.Data. Por ejemplo, el valor de columna no numérico puede utilizarse para las comparaciones numéricas en las llamadas a funciones DTW_SQL posteriores, ocasionando errores de sintaxis.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores
DTW_EDIT_CODES="código_edición"

Tabla 12. Valores de DTW_EDIT_CODES

Valores	Descripción
código_edición	Especifica una serie de caracteres que se corresponde con las columnas resultantes de la tabla que crea el entorno de lenguaje SQL.

Ejemplos

Ejemplo 1:

```
@DTW_ASSIGN(DTW_EDIT_CODES "JJLJJ*****Y")
```

DTW_PAD_PGM_PARMS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Indica a un entorno de lenguaje si los parámetros de caracteres (tipo de datos CHAR o CHARACTER) se han de rellenar o no con blancos cuando se pasan a un programa o procedimiento almacenado.

Para los parámetros IN o INOUT, si la longitud del valor del parámetro es inferior a la precisión especificada, los blancos se insertan a la derecha del valor de parámetro hasta que la longitud del valor del parámetro sea la misma que la precisión.

Para los parámetros OUT, el valor del parámetro se establece en blancos de *precisión*.

Después de la llamada al programa o procedimiento almacenado, se eliminan todos los blancos de cola de los valores de los parámetros OUT e INOUT.

Establezca esta variable en el archivo de inicialización de Net.Data para especificar un valor para todas las macros. Puede alterar temporalmente el valor definiéndolo en la macro. Si DTW_PAD_PGM_PARMS no está definida en la macro, utiliza el valor del archivo de inicialización de Net.Data.

Los entornos de lenguaje de Direct Call y SQL dan soporte a DTW_PAD_PGM_PARMS.

Valores

DTW_PAD_PGM_PARMS="YES" | "NO"

Tabla 13. Valores de DTW_PAD_PGM_PARMS

Valores	Descripción
YES	Todos los valores de los parámetros de caracteres IN e INOUT se justifican a la izquierda y se rellenan con blancos según la precisión definida del parámetro, antes de que los parámetros se pasen a un programa o procedimiento almacenado. Los blancos de cola se eliminan después de la llamada a un programa o procedimiento almacenado.
NO	No se añade ningún relleno a los valores de los parámetros de caracteres (los valores finalizan en NULL) al pasar parámetros a programas o procedimientos almacenados. Los blancos de cola no se eliminan después de una llamada a un programa o procedimiento almacenado.

Ejemplos

Ejemplo 1: Rellena los parámetros con blancos

```
DTW_PAD_PGM_PARMS="YES"
```

DTW_SAVE_TABLE_IN

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Identifica una variable de tabla que utiliza el entorno de lenguaje SQL para almacenar datos de tabla de una consulta. Esta tabla puede utilizarse posteriormente, por ejemplo, en un programa REXX que analice los datos de tabla.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

DTW_SAVE_TABLE_IN="var_nombre_tabla"

Tabla 14. Valores de DTW_SAVE_TABLE_IN

Valores	Descripción
var_nombre_tabla	El nombre de una tabla que utiliza el entorno de lenguaje SQL para almacenar datos de tabla de una consulta.

Ejemplos

Ejemplo 1: Variable de tabla definida con anterioridad que se utiliza en una llamada REXX

```
%DEFINE theTable
= %TABLE(2)
%DEFINE DTW_SAVE_TABLE_IN = "theTable"

%FUNCTION(DTW_SQL) doQuery() {
SELECT MODNO, COST, DESCRIP FROM EQPTABLE
WHERE TYPE='MONITOR'
%}

%FUNCTION(DTW_REXX) analyze_table(myTable) {
  %EXEC{ anzTbl.cmd %}
%}

%HTML(doTable) {
@doQuery()
@analyze_table(theTable)
%}
```

Un bloque REXX FUNCTION llama al programa REXX anzTbl.cmd, que utiliza la variable de tabla theTable para analizar los datos de la tabla. La variable theTable se ha devuelto desde una llamada a función de SQL anterior.

DTW_SET_TOTAL_ROWS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Especifica para un entorno de lenguaje de base de datos que el número total de filas del conjunto de resultados para una consulta debería asignarse a TOTAL_ROWS.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Usuarios de OS/400, OS/2, Windows NT y UNIX: Para pasar esta variable al entorno de lenguaje, inclúyala como variable IN en la sentencia ENVIRONMENT del entorno de lenguaje de base de datos del archivo de inicialización de Net.Data. Para obtener más información sobre la sentencia de entorno de lenguaje de base de datos, consulte el capítulo sobre configuración en el manual *Guía de administración y programación de Net.Data*.

Usuarios de OS/390: DTW_SET_TOTAL_ROWS se pasa implícitamente a los entornos de lenguaje de base de datos cuando se define en la macro.

Sugerencia sobre rendimiento: Establecer DTW_SET_TOTAL_ROWS en YES afecta al rendimiento debido a que, para determinar el número total de filas, el entorno de lenguaje de base de datos requiere que se recuperen todas las filas.

Valores

DTW_SET_TOTAL_ROWS="YES"|"NO"

Tabla 15. Valores de DTW_SET_TOTAL_ROWS

Valores	Descripción
YES	Asigna el valor del número total de filas a la variable TOTAL_ROWS. Importante: Debe definir este valor si desea hacer referencia a la variable TOTAL_ROWS para determinar el número de filas que devuelve una consulta.
NO	Net.Data no establece la variable TOTAL_ROWS y no puede hacerse referencia a TOTAL_ROWS en una macro. NO es el valor por omisión.

Ejemplos

Ejemplo 1: Define DTW_SET_TOTAL_ROWS para utilizar TOTAL_ROWS

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"
```

```
...
```

```
%FUNCTION (DTW_SQL) myfunc() {  
select * from MyTable  
%report {
```

```
...
```

```
%row
```

```
...
```

```
%}
```

```
<p>Your query is limited to $(TOTAL_ROWS) rows. The query  
returned too many rows.
```

```
%}
```

```
%}
```

DTW_USE_DB2_PREPARE_CACHE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X			X	X	X

Propósito

Especifica que Net.Data debe sacar partido de la característica de antememoria de preparación de DB2. Net.Data lo efectúa sustituyendo las variables Net.Data en una sentencia de SQL por marcadores de parámetros y vinculando a continuación la sentencia antes de pasarla a DB2 para su proceso. Si DB2 se configura para la antememoria de preparación y si anteriormente se ha ejecutado la sentencia, utilizando los mismos valores de las variables Net.Data, DB2 puede beneficiarse de la información almacenada en la antememoria. Este característica aumentará el rendimiento en consultas SQL que se llamen de modo reiterado y que tengan muchos valores de consulta de SQL que permanezcan sin variar.

Restricciones:

- Net.Data sustituye cada una de las variables de la sentencia de SQL por un marcador de parámetro; por tanto la sentencia puede hacer referencia a las variables Net.Data únicamente en la cláusula where de la consulta. Por ejemplo, con esta característica no se admite una sentencia como `SELECT $(col) FROM TAB1`.
- Debe tenerse un cuidado muy especial y manejar los apóstrofes simples correctamente en las variables a las que se haga referencia. Por ejemplo, si la columna devuelve una serie que contenga apóstrofes, como "O'Brien," utilice la función `DTW_ADDQUOTE()` para evitar los apóstrofes de la serie.

Valores

`DTW_USE_DB2_PREPARE_CACHE="YES"|"NO"`

Tabla 16. Valores de `DTW_USE_DB2_PREPARE_CACHE`

Valores	Descripción
YES	Especifica que las referencias a variables y los parámetros de DB2 deben sustituirse por marcadores de parámetros.
NO	Especifica que no se utilizarán marcadores de parámetros; la sentencia no se cambiará. Este es el valor por omisión.

Ejemplos

Ejemplo 1: Marcador de parámetro en la cláusula where.

```
%DEFINE DTW_USE_DB2_PREPARE_CACHE="YES"
```

...

```
%FUNCTION (DTW_SQL) myfunc() {  
    select id,projname,due from projects  
    where month in '$(m)' and year in '$(y)'  
    %}
```

La sentencia resultante en DB2 sería:

```
select id,projname,due from projects  
where month in ? and year in ?
```

LOCATION

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
				X				

Propósito

Establece una conexión con un servidor de base de datos remoto. La variable especifica el nombre por el que el subsistema DB2 local conoce al servidor remoto. El valor de LOCATION debe definirse en la tabla SYSIBM.SYSLOCATIONS de la Base de datos de comunicaciones (CDB). Si no se define esta variable en una macro, las peticiones de SQL que efectúa la macro se ejecutan en el subsistema DB2 local.

Requisito: Para que surta efecto el valor de esta variable, debe listarse en la sentencia ENVIRONMENT del entorno de lenguaje SQL. Consulte la publicación *Administration and Programming Guide* del sistema operativo para obtener más información.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

LOCATION="nombre_based_remoto"

Tabla 17. Valores de LOCATION

Valores	Descripción
nombre_based_remoto	El nombre del servidor de base de datos remoto válido que se define en la tabla SYSIBM.SYSLOCATIONS de la CDB. El nombre puede tener ocho caracteres o menos.

Ejemplos

Ejemplo 1: Define la ubicación remota de la base de datos en la sentencia DEFINE

```
%DEFINE LOCATION="QMFDJ00"
```

LOGIN

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X		X	X	X	X

Propósito

Proporciona acceso a los datos protegidos pasando un ID de usuario al entorno de lenguaje de base de datos. Utilice esta variable con PASSWORD para incorporar los algoritmos de seguridad de DB2.

Usuarios de OS/400: OS/400 ignora tanto LOGIN como PASSWORD en el caso de que la variable de DATABASE no esté definida o de que se haya establecido en un valor de `"*LOCAL"`. El acceso a la base de datos se direcciona a través del perfil de usuario bajo el que se está ejecutando Net.Data.

Sugerencia sobre seguridad: Aunque puede codificar este valor en la macro Net.Data, es preferible que sea el usuario de la aplicación quién entre los ID de usuario en formato HTML. Adicionalmente, la utilización del valor por omisión del ID del servidor Web proporciona un nivel de acceso que es posible que no reúna sus necesidades de seguridad.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

LOGIN="*id_usuario_basedatos*"

Tabla 18. Valores de LOGIN

Valores	Descripción
<i>id_usuario_basedatos</i>	Un ID de usuario de base de datos válido. El valor por omisión es utilizar el ID de usuario que ha iniciado el servidor de la Web.

Ejemplos

Ejemplo 1: Limitar el acceso al ID de usuario, DB2USER

```
%DEFINE LOGIN="DB2USER"
```

Ejemplo 2: Utilización de una línea de entrada en formato HTML

```
USERID&#58; <input type="text"  
name="login" size=6 />
```

Este ejemplo muestra una línea que puede incluirse como parte de un formato HTML para que los usuarios de la aplicación entren sus ID de usuario.

NULL_RPT_FIELD

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Especifica una serie que el usuario puede facilitar al entorno de lenguaje DTW_SQL para representar valores NULL que se devuelven en un conjunto de resultados SQL.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

NULL_RPT_FIELD="car_nulo"

Tabla 19. Valores de NULL_RPT_FIELD

Valores	Descripción
car_nulo	Especifica una serie para representar valores NULL que se devuelven en un conjunto de resultados SQL. El valor por omisión es una serie vacía.

Ejemplos

Ejemplo 1: Especifica una serie que representa valores NULL en el entorno de lenguaje SQL

```
%DEFINE NULL_RPT_FIELD = "++++"
```

PASSWORD

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X		X	X	X	X

Propósito

Proporciona acceso a los datos protegidos pasando una contraseña al entorno de lenguaje de base de datos. Utilice esta variable con LOGIN para incorporar los algoritmos de seguridad de DB2.

Usuarios de OS/400: OS/400 ignora tanto LOGIN como PASSWORD en el caso de que la variable DATABASE no esté definida o de que se haya establecido en un valor de *"*LOCAL"*. El acceso a la base de datos se direcciona a través del perfil de usuario bajo el que se está ejecutando Net.Data.

Sugerencia sobre seguridad: Aunque puede codificar este valor en la macro Net.Data, es preferible que sean los usuarios de la aplicación quienes entren las contraseñas en formato HTML.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

PASSWORD=*"contraseña"*

Tabla 20. Valores de PASSWORD

Valores	Descripción
<i>contraseña</i>	Especifica una contraseña válida para facilitar el acceso automático al entorno de lenguaje de base de datos.

Ejemplos

Ejemplo 1: Limitar el acceso a usuarios de aplicaciones con la contraseña NETDATA

```
%DEFINE  
PASSWORD="NETDATA"
```

Ejemplo 2: Línea de entrada en formato HTML

```
PASSWORD&#58; <input type="password" name="password" size=8 />
```

Este ejemplo muestra una línea que puede incluirse como parte de un formato HTML para que los usuarios de la aplicación entren sus propias contraseñas.

SHOWSQL

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Oculta o visualiza el SQL de la consulta utilizada en el navegador Web. Visualizar el SQL durante la prueba resulta especialmente útil cuando se depuran macros Net.Data. SHOWSQL sólo puede utilizarse si DTW_SHOWSQL se establece en YES en el archivo de configuración de Net.Data. Para obtener más información sobre la variable de configuración DTW_SHOWSQL, consulte el capítulo sobre configuración en la publicación *Guía de administración y programación de Net.Data* para el sistema operativo.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

SHOWSQL="YES" | "NO"

Tabla 21. Valores de SHOW_SQL

Valores	Descripción
YES	Visualiza el SQL de la consulta que se envía a la database.
NO	Oculta el SQL de la consulta que se envía a la database. NO es el valor por omisión.

Restricción: SHOWSQL genera markup de HTML o XML. Cuando se utiliza dentro de otras sentencias de presentación, por ejemplo, JavaScript, pueden producirse errores de sintaxis.

Ejemplos

Ejemplo 1: Visualiza todas las consultas SQL

En el archivo de configuración:

```
DTW_SHOWSQL YES
```

En la macro:

```
%DEFINE SHOWSQL="YES"
```

Ejemplo 2: Especificar si se ha de visualizar SQL utilizando la entrada en formato HTML.

En el archivo de configuración:

DTW_SHOWSQL YES

En la macro:

SHOWSQL: <input type="radio" name="showsql" value="yes" /> Yes
 <input type="radio" name="showsql" value="" checked /> No

SQL_STATE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Accede o visualiza el valor del estado de SQL que devuelve la base de datos.

Esta variable es una variable predefinida y su valor no se puede modificar.
Utilice la variable como referencia variable.

Ejemplos

Ejemplo 1: Visualiza el estado de SQL en el bloque REPORT

```
%FUNCTION (DTW_SQL) val1() {  
  select * from customer  
%REPORT {  
  ...  
%ROW {  
  ...  
%}  
  SQLSTATE=$(SQL_STATE)  
%}
```

TRANSACTION_SCOPE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Especifica el ámbito de transacciones de los mandatos SQL, determinando si Net.Data emite COMMIT (confirmación) después de cada mandato SQL o después de que todos los mandatos SQL de un bloque HTML hayan finalizado satisfactoriamente. Cuando se especifica que todos los mandatos SQL deben finalizar satisfactoriamente antes de una confirmación, un mandato SQL no satisfactorio hace que se todos los mandatos SQL ejecutados con anterioridad en *la misma base de datos* de dicho bloque se restituyan.

Para que surta efecto la variable TRANSACTION_SCOPE, inclúyala en la sentencia ENVIRONMENT del archivo de configuración de Net.Data. A continuación, puede especificar el valor de esta variable utilizando una sentencia DEFINE o con la función @DTW_ASSIGN().

Consideraciones sobre coherencia: En sistemas operativos que no sean OS/400 y OS/390, es posible que las actualizaciones para la base de datos que está recibiendo respuestas no satisfactorias se restituyan y al mismo tiempo es posible que las actualizaciones para las demás bases de datos a las que se accede en el mismo bloque HTML se confirmen cuando se cumplan todas las condiciones siguientes:

- Se ha especificado TRANSACTION_SCOPE = "MULTIPLE"
- Se accede a múltiples bases de datos en un bloque HTML (lo cual es posible cuando se utiliza Live Connection (Conexión en directo))
- Una petición SQL devuelve una respuesta no satisfactoria

Si accede a múltiples bases de datos desde Net.Data en OS/400 o utilizando DataJoiner de IBM, las actualizaciones de múltiples bases de datos serán coordinadas y coherentes al efectuar la actualización desde Net.Data.

En OS/400 y OS/390, TRANSACTION_SCOPE = "MULTIPLE" hace que todas las actualizaciones de bases de datos IBM procedentes de un único bloque HTML se confirmen o restituyan conjuntamente.

En los sistemas operativos que no sean OS/400, los entornos de lenguaje REXX, Perl y Java se ejecutan en los diferentes procesos de sus propios sistemas operativos. De este modo, las actualizaciones de base de datos que se emiten desde dichos entornos de lenguaje se confirmarán o restituirán

independientemente de las actualizaciones de base de datos emitidas desde una macro Net.Data, sin tener en cuenta el valor de la variable TRANSACTION_SCOPE de Net.Data.

Valores

TRANSACTION_SCOPE="SINGLE" | "MULTIPLE"

Tabla 22. Valores de TRANSACTION_SCOPE

Valores	Descripción
SINGLE	Net.Data emite una COMMIT (confirmación) después de que finalice satisfactoriamente cada uno de los mandatos SQL de un bloque HTML.
MULTIPLE	Especifica que Net.Data emita una COMMIT (confirmación) únicamente después de que todos los mandatos SQL de un bloque HTML finalicen satisfactoriamente. MULTIPLE es el valor por omisión.

Ejemplos

Ejemplo 1: Especifica la emisión de una COMMIT (confirmación) después de cada una de las transacciones

```
%DEFINE TRANSACTION_SCOPE="SINGLE"
```

Variables Net.Data diversas

Estas variables son variables definidas por Net.Data que pueden utilizarse para afectar al proceso de Net.Data, averiguar el estado de una llamada a función y obtener información sobre el conjunto de resultados de una consulta de la base de datos, así como para determinar información sobre fechas y ubicaciones de archivos. Es posible que encuentre útiles estas variables en las funciones que escriba o puede utilizarlas para probar las macros Net.Data.

- “DTW_CURRENT_FILENAME” en la página 129
- “DTW_CURRENT_LAST_MODIFIED” en la página 130
- “DTW_DEFAULT_MESSAGE” en la página 131
- “DTW_LOG_LEVEL” en la página 132
- “DTW_MACRO_FILENAME” en la página 133
- “DTW_MACRO_LAST_MODIFIED” en la página 134
- “DTW_MBMODE” en la página 135
- “DTW_MP_PATH” en la página 137
- “DTW_MP_VERSION” en la página 138
- “DTW_PRINT_HEADER” en la página 139
- “DTW_REMOVE_WS” en la página 141
- “RETURN_CODE” en la página 142

DTW_CURRENT_FILENAME

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

El nombre y extensión del archivo de entrada actual. El archivo de entrada es una macro Net.Data o un archivo especificado en una sentencia INCLUDE.

Esta variable es una variable predefinida y su valor no se puede modificar. Utilice la variable como referencia variable.

Ejemplos

<p>This file is <i>\$(DTW_CURRENT_FILENAME)</i>,
and was updated on \$(DTW_CURRENT_LAST_MODIFIED).

DTW_CURRENT_LAST_MODIFIED

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

La fecha y la hora en que se modificó por última vez el archivo actual. El archivo actual puede ser una macro Net.Data o un archivo especificado en una sentencia INCLUDE. El formato de salida lo determina el sistema en el que se ejecuta Net.Data.

Esta variable es una variable predefinida y su valor no se puede modificar. Utilice la variable como referencia variable.

Ejemplos

```
<p>This file is <i>$(DTW_CURRENT_FILENAME)</i>,  
and was updated on $(DTW_CURRENT_LAST_MODIFIED).
```


DTW_DEFAULT_MESSAGE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Contiene el texto del mensaje que devuelve una llamada a una función incorporada o al entorno de lenguaje cuando se produce un error.

Puede utilizar la variable DTW_DEFAULT_MESSAGE en cualquier parte de la macro Net.Data.

Esta variable es una variable predefinida, no se recomienda modificar su valor. Utilice la variable como una referencia a variable.

Ejemplos

Ejemplo 1: Mensaje que indica si la función ha finalizado satisfactoriamente

```
@function1()  
%IF ("$(RETURN_CODE)" == "0")  
  The function completed successfully.  
%ELSE  
  The function failed with the return code $(RETURN_CODE). The error message  
  returned is "$(DTW_DEFAULT_MESSAGE)".  
%ENDIF
```

Ejemplo 2: El texto por omisión para el momento en que una función devuelve un código de retorno diferente a cero

```
%MESSAGE{  
default: {<h2>Net.Data received return code: $(RETURN_CODE).  
Error message is $(DTW_DEFAULT_MESSAGE)</h2> %} : continue  
%}
```

El usuario ve el mensaje de error por omisión, en el caso de que una función devuelva un código de retorno que no sea un 0.

DTW_LOG_LEVEL

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X			X	X	X

Propósito

El nivel de mensajes que Net.Data graba en el archivo de registro cronológico.

Puede especificar el valor de esta variable utilizando una sentencia DEFINE o con la función @DTW_ASSIGN().

Requisito: Defina DTW_LOG_DIR en el archivo de inicialización de Net.Data para iniciar el registro cronológico; en caso contrario Net.Data no registra cronológicamente mensajes cuando se especifique la variable DTW_LOG_LEVEL en la macro.

Valores

DTW_LOG_LEVEL="OFF|ERROR|WARNING"

Tabla 23. Valores de DTW_LOG_LEVEL

Valores	Descripción
OFF	Net.Data no registra cronológicamente errores. OFF es el valor por omisión.
ERROR	Net.Data registra cronológicamente mensajes de error.
WARNING	Net.Data registra cronológicamente avisos, así como mensajes de error.

Ejemplos

```
%DEFINE DTW_LOG_LEVEL="ERROR"
```

DTW_MACRO_FILENAME

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

El nombre y extensión de la macro Net.Data actual.

Esta variable es una variable predefinida y su valor no se puede modificar.
Utilice la variable como referencia variable.

Ejemplos

```
<p>This Net.Data macro is  
<i>$(DTW_MACRO_FILENAME)</i>,  
and was updated on $(DTW_MACRO_LAST_MODIFIED).
```

DTW_MACRO_LAST_MODIFIED

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

La fecha y la hora en que se modificó por última vez la macro Net.Data. El formato de salida depende del sistema en el que se ejecuta Net.Data.

Esta variable es una variable predefinida y su valor no se puede modificar. Utilice la variable como referencia variable.

Ejemplos

```
<p>This Net.Data macro is  
<i>$(DTW_MACRO_FILENAME)</i>,  
and was updated on $(DTW_MACRO_LAST_MODIFIED).
```

DTW_MBMODE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X		X	X	X

Propósito

Proporciona soporte al conjunto de caracteres de múltiples bytes (MBCS) para funciones de texto y series incorporadas de Net.Data. Puede establecer esta variable en el archivo de inicialización de Net.Data, pero puede utilizarla en la macro para establecer o alterar temporalmente el valor actual.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Usuarios de OS/400: Net.Data para OS/400 habilita automáticamente las funciones para el soporte de MBCS y no necesita esta variable. Net.Data para OS/400 ignora esta variable en las macros que se migran al sistema operativo OS/400.

Esta variable de configuración funciona con la variable de configuración de DTW_UNICODE. Si DTW_UNICODE utiliza el valor por omisión de NO, se utiliza el valor de DTW_MBMODE. Si DTW_UNICODE se establece en un valor que no sea NO, se utiliza su valor. La Tabla 24 ilustra el modo en que los valores de estas dos variables determinan el modo en que las funciones incorporadas procesan las series:

Tabla 24. Relación entre los valores de DTW_UNICODE y DTW_MBMODE

Si DTW_UNICODE se establece en ...	Si DTW_MBMODE=YES	Si DTW_MBMODE=NO
NO	Da soporte a DBCS combinado con SBCS	Sólo da soporte a SBCS
UTF8	Da soporte a UTF-8	Da soporte a UTF-8

Valores

DTW_MBMODE="YES" | "NO"

Tabla 25. Valores de DTW_MBMODE

Valores	Descripción
YES	Especifica soporte de MBCS para funciones de texto y de serie.
NO	Especifica que las funciones de texto y de series no dispongan de soporte de MBCS. NO es el valor por omisión.

Ejemplos

Ejemplo 1: Altera temporalmente el valor en el archivo INI

Archivo INI:

```
DTW_MBMODE NO
```

Macro:

```
%DEFINE DTW_MBMODE = "YES"
```

DTW_MP_PATH

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

La vía de acceso y el nombre del archivo ejecutable de Net.Data. En función del sistema, la salida será similar al nombre y vía de acceso de ejemplo siguientes:

```
/usr/lpp/internet/server_root/cgi-bin/db2www
```

Esta variable es una variable predefinida y su valor no se puede modificar. Utilice la variable como referencia variable.

Ejemplos

The Net.Data executable file is `$(DTW_MP_PATH)`.

DTW_MP_VERSION

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

La versión y número de release de Net.Data que se ejecuta en el servidor.

Esta variable es una variable predefinida y su valor no se puede modificar.
Utilice la variable como referencia variable.

Ejemplos

This Web application uses \$(DTW_MP_VERSION).

DTW_PRINT_HEADER

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Especifica el texto para la cabecera HTTP.

Debe haber establecido esta variable antes de que Net.Data procese alguno de los textos enviados al navegador Web, ya que Net.Data lee una vez esta variable antes de visualizar el texto y no vuelve a consultarla. Los cambios en la variable DTW_PRINT_HEADER se ignoran después de que Net.Data haya enviado texto al navegador.

Si está utilizando DTW_PRINT_HEADER para generar sus propias cabeceras (DTW_PRINT_HEADER = "NO"), debe asegurarse de que DTW_REMOVE_WS se ha establecido en "NO" o utilizar la función incorporada DTW_rHEXTOCHAR() para generar una nueva línea detrás de las cabeceras HTTP.

Especifique el valor de esta variable mediante una sentencia DEFINE o mediante la función @DTW_ASSIGN().

Valores

DTW_PRINT_HEADER="YES" | "NO"

Tabla 26. Valores de DTW_PRINT_HEADER

Valores	Descripción
YES	Net.Data imprime el texto Content-type: text/html o Content-type: text/xml para la cabecera HTTP. YES es el valor por omisión.
NO	Net.Data no imprime una cabecera HTTP. Puede generar información de cabecera HTTP personalizada.

Ejemplos

Ejemplo 1: Establecer DTW_PRINT_HEADER en NO para personalizar su propia cabecera.

```
%define DTW_REMOVE_WS="YES"
%define DTW_PRINT_HEADER="NO"
@DTW_ASSIGN(CRLF, "@DTW_rHEXTOCHAR("0D25")")
%HTML(report) {
Expires: Thu, 31 Jul 2000 16:00:00 GMT$(CRLF)
Content-type: text/html$(CRLF)$(CRLF)
```

```
The Time is @DTW_rtime()
%}
```

DTW_REMOVE_WS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Si establece el valor de esta variable en "YES" en el bloque DEFINE, Net.Data eliminará el espacio en blanco adicional en las páginas web resultantes.

Utilización de códigos <pre></pre>: Definir esta variable en YES afecta a la cantidad y tipo de espacio en blanco que se imprime. Si la variable se establece en YES, es posible que haya partes de páginas HTML que utilicen códigos <pre></pre> que no se visualicen tal y como se pretendía.

Valores

DTW_REMOVE_WS="YES"|"NO"

Tabla 27. Valores de DTW_REMOVE_WS

Valores	Descripción
YES	Net.Data comprime una secuencia de dos o más espacios en blanco en un carácter de nueva línea, generando páginas de resultados de HTML más cortas.
NO	Net.Data no comprime espacios en blanco. NO es el valor por omisión.

Ejemplos

Ejemplo 1: Eliminación de espacios en blanco adicionales

DTW_REMOVE_WS="YES"

RETURN_CODE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

El código de retorno que devuelve una llamada a una función incorporada o una llamada a un entorno de lenguaje. Net.Data utiliza este valor para procesar bloques MESSAGE. Puede utilizar esta variable para determinar si una llamada a función ha fallado o ha resultado satisfactoria. Un valor de cero indica que una llamada a función ha finalizado satisfactoriamente.

Puede hacer referencia a la variable RETURN_CODE en cualquier parte de la macro Net.Data.

Este valor está predefinido; no es recomendable modificar el valor. Utilícelo como una referencia a variable.

Ejemplos

Ejemplo 1: Mensaje que indica si la función ha finalizado satisfactoriamente

```
@function1()  
%IF ("$(RETURN_CODE)" == "0")  
  The function completed successfully.  
%ELSE  
  The function failed with the return code $(RETURN_CODE).  
%ENDIF
```

Ejemplo 2: Mensaje por omisión cuando el código de retorno no es 0

```
%MESSAGE{  
  default: "<h2>Net.Data received return code:  
$(RETURN_CODE)</h2>" : continue  
%}
```

En el caso de que una función devuelva un código de retorno que no sea un 0, se visualiza el mensaje por omisión.

Capítulo 3. Funciones incorporadas de Net.Data

Net.Data proporciona una amplia variedad de funciones que pueden utilizarse sin crear sus propios bloques FUNCTION. Las funciones incorporadas de Net.Data se dividen en las categorías siguientes:

- Las **funciones de uso general** le ayudan a desarrollar páginas Web con Net.Data y no se adaptan a las demás categorías. Consulte el apartado “Funciones generales” en la página 145.
- Las **funciones matemáticas** efectúan operaciones matemáticas. Consulte el apartado “Funciones matemáticas” en la página 188.
- Las **funciones de manipulación de serie** modifican series y caracteres. Consulte el apartado “Funciones de serie” en la página 207.
- Las **funciones de manipulación de texto** modifican palabras o conjuntos de palabras. Consulte el apartado “Funciones de texto” en la página 240.
- Las **funciones de manipulación de tablas** le ayudan a generar formularios e informes a partir de los datos de tabla. Consulte el apartado “Funciones de tabla” en la página 255.
- Las **funciones de interfaz de archivo plano** efectúan la salida y entrada de archivos. Consulte el apartado “Funciones de interfaz de archivo plano” en la página 309.
- Las **funciones de registro Web** efectúan operaciones en un registro Web. Consulte el apartado “Funciones de registro Web” en la página 350.
- Las **funciones de macros permanentes** dan soporte al proceso de transacciones en Net.Data. Consulte el apartado “Funciones de macro permanentes” en la página 373.

Aunque algunos parámetros de función se describen como poseedoras del tipo *entero* o *flotante*, se utilizan los términos para indicar una serie que representa un valor flotante o entero, respectivamente.

Nombres de función

Las funciones incorporadas de Net.Data comienzan por DTW, el cual es un prefijo reservado. Las funciones definidas por el usuario no deberían utilizar este prefijo.

La utilización del prefijo DTW para las funciones que no sean funciones incorporadas de Net.Data puede dar como resultado un comportamiento imprevisible.

Los nombres de función incorporada no son sensibles a mayúsculas y minúsculas.

Parámetros de entrada y salida

Las funciones pueden tener especificaciones que pasan parámetros que determinan si Net.Data utiliza el parámetro para la entrada, salida o tanto para la entrada como para la salida. Estas especificaciones que pasan parámetros las especifican las siguientes palabras clave:

- IN** Especifica que el parámetro pase datos de entrada al entorno de lenguaje desde Net.Data.
- OUT** Especifica que el parámetro devuelva datos de salida desde el entorno de lenguaje a Net.Data.
- INOUT** Especifica que el parámetro pase datos de entrada al entorno de lenguaje y devuelva datos de salida desde el entorno de lenguaje a Net.Data.

Cómo dar formato a los resultados de función

Muchas funciones tienen uno o más de los siguientes formatos:

- Las funciones comienzan por DTW_r y DTWR_r devuelven sus resultados a la llamada de función, por lo que no tienen un parámetro de salida. Este ejemplo muestra el tiempo de servidor.

La hora local actual es @DTW_rTIME().

- Las funciones que comienzan por DTW_m efectúan la función en múltiples parámetros. Cada parámetro se comporta como parámetro de entrada y de salida. La función se efectúa sobre el parámetro y los resultados se devuelven en el parámetro. Este ejemplo convierte los tres parámetros de entrada en letras mayúsculas para una apariencia coherente en la pantalla:

@DTW_mUPPERCASE(model, style, shipNo)

Shipment \$(shipNo) contains \$(quantity) of model \$(model) \$(style).

- Otras funciones que comienzan por DTW_, DTWF_ y DTWR_ devuelven sus resultados en un parámetro de salida. Debe especificar el parámetro de salida. Este ejemplo muestra el tiempo de servidor.

@DTW_TIME(nowTime)

Current local time is \$(nowTime).

Normas de parámetros de función

Coloque los parámetros de función en el orden correcto. Debe especificar todos los parámetros de *entrada* antes de que pueda especificarse el último parámetro de entrada, o especifique un nulo (“”) para aceptar el valor por omisión. Por ejemplo, puede llamar a DTW_TB_INPUT_TEXT como en el ejemplo siguiente:

```
@DTW_TB_INPUT_TEXT(myTable, "1", "2", "", "", "32")
```

En el ejemplo anterior los parámetros cuarto y quinto utilizan valores por omisión. Inclúyalos como nulos para indicar que “32” es el valor para MAXLENGTH en el HTML generado. El parámetro final no se especifica, por lo que se utiliza el valor por omisión. Si opta por aceptar el valor por omisión para MAXLENGTH y los dos parámetros anteriores, omítalos, tal y como se muestra a continuación:

```
@DTW_TB_INPUT_TEXT(myTable, "1", "2")
```

Debe especificar valores nulos intermedios en las listas de parámetros para parámetros de entrada cuando existan parámetros de *entrada* no nulos subsiguientes. No ha de especificar parámetros de entrada nulos intermedios antes de especificar el parámetro de *salida* final.

Funciones generales

Las funciones generales le ayudan a desarrollar páginas Web con Net.Data y no se adaptan a las demás categorías. Las funciones siguientes son funciones de uso general:

- “DTW_ADDQUOTE” en la página 146
- “DTW_CACHE_PAGE” en la página 148
- “DTW_DATE” en la página 153
- “DTW_EXIT” en la página 155
- “DTW_GETCOOKIE” en la página 157
- “DTW_GETENV” en la página 160
- “DTW_GETINIDATA” en la página 162
- “DTW_HTMLENCODE” en la página 164
- “DTW_QHTMLENCODE” en la página 167
- “DTW_SENDMAIL” en la página 169
- “DTW_SETCOOKIE” en la página 177
- “DTW_SETENV” en la página 181
- “DTW_TIME” en la página 183
- “DTW_URLESCSEQ” en la página 185

DTW_ADDQUOTE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Sustituye comillas simples en una serie de entrada por dos comillas simples.

Formato

@DTW_ADDQUOTE(*serieEntrada*, *serieSalida*)

@DTW_rADDQUOTE(*serieEntrada*)

@DTW_mADDQUOTE(*serieMult*, *serieMult2*, ..., *serieMultn*)

Parámetros

Tabla 28. Parámetros de DTW_ADDQUOTE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal. DTW_mADDQUOTE puede tener múltiples series de entrada.
serie	<i>serieSalida</i>	OUT	Variable que contiene el formato modificado de <i>serieEntrada</i> .
serie	<i>serieMult</i>	INOUT	<ul style="list-style-type: none">En la entrada: Variable que contiene una serie.En la salida: Variable que contiene la serie de entrada en la que cada carácter de comilla simple (') se ha sustituido por dos caracteres de comilla simple.

Códigos de Retorno

Tabla 29. Códigos de retorno de DTW_ADDQUOTE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.

Tabla 29. Códigos de retorno de DTW_ADDQUOTE (continuación)

Código de retorno	Explicación
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Notas de utilización

Tenga en cuenta la utilización de esta función para todas las sentencias de SQL INPUT en las que la entrada se obtenga de un navegador Web. Por ejemplo, si escribe O'Brien como apellido, como en el siguiente ejemplo, la comilla simple podría ocasionar un error:

```
INSERT INTO USER1.CUSTABLE (LNAME, FNAME)
VALUES ('O'Brien', 'Patrick')
```

La utilización de la función DTW_ADDQUOTE cambia la sentencia de SQL e impide el error:

```
INSERT
INTO USER1.CUSTABLE (LNAME, FNAME)
VALUES ('O''Brien', 'Patrick')
```

Ejemplos

Ejemplo 1: Añade una comilla simple adicional al parámetro OUT

```
@DTW_ADDQUOTE(string1,string2)
```

- La entrada: string1="John's Web page"
- Devuelve: string2="John''s Web page"

Ejemplo 2: Añade una comilla simple adicional al valor devuelto de la llamada de función

```
@DTW_rADDQUOTE("The title of the article is 'Once upon a time'")
```

- Devuelve: "The title of the article is ''Once upon a time''"

Ejemplo 3: Añade comillas simples adicionales a cada uno de los parámetros INOUT de la llamada de función

```
@DTW_mADDQUOTE(string1,string2)
```

- La entrada: string1="Joe's bag", string2="'to be or not to be'"
- Devuelve string1="Joe''s bag", string2="''to be or not to be''"

Ejemplo 4: Inserta comillas simples adicionales a los datos que se insertan en una tabla DB2

```
%FUNCTION(DTW_SQL) insertName(){
INSERT INTO USER1.CUSTABLE (LNAME,FNAME)
VALUES ('@DTW_rADDQUOTE(lastname)', '@DTW_rADDQUOTE(firstname)')
%}
```

- La entrada: lastname="O'Brien", firstname="Patrick"
- Devuelve: "O''Brien", "Patrick"

DTW_CACHE_PAGE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X								X

Propósito

Pone en antememoria páginas Web parciales o completas que se generan como consecuencia del proceso de las macros.

Formato

@DTW_CACHE_PAGE(IDantememoria, IDpágina, edad, estado)

Parámetros

Tabla 30. Parámetros de DTW_CACHE_PAGE

Parámetro	Utilización	Descripción
<i>IDantememoria</i>	IN	Variable de serie que identifica la antememoria en la que se colocará la página.
<i>IDpágina</i>	IN	Variable de serie que contiene un identificador utilizado para localizar la página puesta en antememoria en una petición de antememoria DTW_CACHE_PAGE subsiguiente. La serie puede ser un URL.
<i>edad</i>	IN	<p>Variable de serie que contiene un espacio de tiempo en segundos. Este parámetro determina si una página ha caducado. Si la página es más antigua que el valor de <i>edad</i>, la página no se envía al navegador.</p> <p>Si <i>edad</i> se especifica como -1 y la página existe en la antememoria, Net.Data lo envía al navegador de la Web directamente desde la antememoria sin tener en cuenta su antigüedad. Net.Data no sustituye la página en la antememoria.</p>

Tabla 30. Parámetros de DTW_CACHE_PAGE (continuación)

Parámetro	Utilización	Descripción
<i>estado</i>	OUT	<p>Variable de serie que indica el estado de la página en antememoria. Los valores posibles están en minúsculas:</p> <ul style="list-style-type: none"> • ok: La página de salida se pondrá en antememoria cuando finalice la ejecución de la macro. • new: La página no está en la antememoria. • renew: La página está en la antememoria, pero ha caducado. • no_cache: El identificador de antememoria especificado no existe. Debe estar definido en los archivos de configuración de antememoria. La macro puede seguir ejecutándose sin poner las páginas en la antememoria. • inactive: La antememoria que ha especificado ha sido marcada como inactiva. La macro puede seguir ejecutándose sin poner las páginas en la antememoria. • busy: La macro ha emitido la función incorporada de DTW_CACHE_PAGE antes de esta ejecución. La macro puede seguir ejecutándose. • error: Se ha producido un error al intentar comunicarse con la antememoria.

Códigos de Retorno

Tabla 31. Códigos de retorno de DTW_CACHE_PAGE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Tabla 31. Códigos de retorno de DTW_CACHE_PAGE (continuación)

Código de retorno	Explicación
1007	Un parámetro contiene un valor que no es válido.

Notas de utilización

1. Cuando se invoca, DTW_CACHE_PAGE() intenta recuperar la página especificada de la antememoria y enviarla al navegador de la Web como si fuera la página de salida generada desde la macro. Si se encuentra la página y no ha caducado, Net.Data deja de procesar la macro, sale de la misma y envía la página en antememoria al navegador de la Web.

Si la página solicitada no está en la antememoria o la página en antememoria existente es más antigua que el valor de *edad*, Net.Data genera una nueva página de salida. Cuando la macro se completa satisfactoriamente, Net.Data envía la página nueva al navegador y pone la página en antememoria.

2. Para la mayoría de las aplicaciones de antememoria, especifique DTW_CACHE_PAGE() en la parte superior de la macro para poner en antememoria la totalidad de la página Web que se genera cuando se ejecuta la macro. Esta técnica facilita el mantenimiento de la macro cuando se actualiza la misma. Por ejemplo, cuando la función esté en medio de la macro, es posible que no se observara en el momento en que una sección de informe de HTML se hubiera añadido con anterioridad en la macro. Net.Data no colocaría en antememoria la nueva salida de informe. Adicionalmente, este método mejora el rendimiento a medida que Net.Data detiene todo proceso subsiguiente cuando determina que la página está en la antememoria.

Para las aplicaciones en antememoria avanzadas, puede colocar la función en ubicaciones específicas de la macro cuando necesite tomar la decisión de poner en antememoria en un punto específico durante el proceso, en vez de al principio de la macro. Por ejemplo, es posible que necesite tomar la decisión de poner en antememoria basándose en el número de filas que se devuelve desde una consulta o desde una llamada de función.

Ejemplos

Ejemplo 1: Coloca la función DTW_CACHE_PAGE() al principio de la macro para capturar toda la salida de HTML

```
%IF (customer_status == "Classic")
@DTW_CACHE_PAGE("mymacro.mac", "http://www.mypage.org", "-1", status)
%ENDIF
% DEFINE { ...%}

...

%HTML(OUTPUT) {
<title>This is the page title
```

```

</head>
<body>
<center>
This is the Main Heading
<p>It is $(time). Have a nice day!
</body>
</html>

```

```
%}
```

Ejemplo 2: Coloca la función en el bloque de HTML debido a que la decisión de poner en antememoria depende del tamaño de salida de HTML que se espera

```
%DEFINE { ...%}
```

```
...
```

```

%FUNCTION(DTW_SQL) count_rows(){
    select count(*) from customer
%REPORT{
    %ROW{
        @DTW_ASSIGN(ALL_ROWS, V1)
    %}
    %}
    %}

```

```

%FUNCTION(DTW_SQL) all_customers(){
    select * from customer
%}

```

```

%HTML(OUTPUT) {
<html>
<head>
<title>This is the customer list
</head>
<body>

```

```
@count_rows()
```

```

    %IF ($(ALL_ROWS) > "100")
    @DTW_CACHE_PAGE("mymacro.mac", "http://www.mypage.org", "-1", status)
%ENDIF

```

```
@all_customers()
```

```

</body>
</html>
%}

```

En este ejemplo, la página se pone en antememoria o se recupera en función del tamaño de la salida de HTML que se espera. Sólo se considera que vale la pena poner en antememoria las páginas de salida de HTML cuando la tabla de bases de datos contiene más de 100 filas. Net.Data siempre envía el texto

en el bloque OUTPUT, This is the customer list, al navegador después de ejecutar la macro; el texto nunca se pone en antememoria. Las líneas que siguen a la llamada de función, @count_rows(), se ponen en antememoria o se recuperan cuando se cumplen todas las condiciones del bloque IF. Juntas, ambas partes forman una página de salida de Net.Data completa.

Ejemplo 3: Recupera dinámicamente el ID de antememoria y el ID de página en antememoria

```
%HTML(OUTPUT) {  
  %IF (customer == "Joe Smith")  
  
    @DTW_CACHE_PAGE(@DTW_rGETENV("DTW_MACRO_FILENAME"),  
    @DTW_rGETENV("URL"), "-1", status)  
  
  %ENDIF  
  
  ...  
  
  <html>  
    <head>  
      <title>This is the page title</title>  
    </head>  
    <body>  
      <center>  
        <h3>This is the Main Heading</h3>  
        <p>It is @DTW_rDATE(). Have a nice day!</p>  
      </center>  
    </body>  
  </html>  
  
  %}
```

DTW_DATE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve la fecha actual del sistema en el formato especificado.

Formato

@DTW_DATE(formato, serieSalida)

@DTW_DATE(serieSalida)

@DTW_rDATE(formato)

@DTW_rDATE()

Parámetros

Tabla 32. Parámetros de DTW_DATE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>formato</i>	IN	Variable o serie literal que especifica el formato de datos. Los formatos válidos incluyen: D - Día del año (001-366) E - Formato de fecha europea (dd/mm/aa) N - Formato de fecha normal (dd mes aaaa) O - Formato de fecha ordenado (aa/mm/dd) S - Formato de fecha estándar (aaaammdd) U - Formato de fecha de Estados Unidos (mm/dd/aa) El valor por omisión es N.
serie	<i>serieSalida</i>	OUT	Variable que contiene la fecha en el formato especificado.

Códigos de Retorno

Tabla 33. Códigos de retorno de DTW_DATE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.

Tabla 33. Códigos de retorno de DTW_DATE (continuación)

Código de retorno	Explicación
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1: Formato de fecha normal

@DTW_DATE(results)

- Devuelve: results = "25 Abr 1997"

Ejemplo 2: Formato de fecha europeo

@DTW_DATE("E", results)

- Devuelve: results="25/04/97"

Ejemplo 3: Formato de fecha de estados Unidos

```
%HTML(report){
```

```
<p>This report created on @DTW_rDATE("U").</p>
```

- Devuelve: 04/25/97

DTW_EXIT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Especifica salir de la macro de modo inmediato. Net.Data envía las páginas Web generadas antes de que se llame DTW_EXIT() para el navegador de la Web.

Formato

@DTW_EXIT()

Códigos de Retorno

Tabla 34. Códigos de retorno de DTW_EXIT

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Notas de utilización

1. Utilice DTW_EXIT() para detener de modo inmediato el proceso de una macro. La utilización de esta técnica ahorra el tiempo que Net.Data utilizaría para procesar todo el archivo.
2. Asegúrese de que toda la macro sea sintácticamente correcta antes de añadir la función DTW_EXIT(). La utilización de DTW_EXIT() hace que Net.Data deje de procesar la macro cuando encuentre la llamada a esta función, lo cual puede evitarle los errores que se producen después de que se haya procesado la función DTW_EXIT().

Ejemplos

Ejemplo 1: Cómo salir de una macro

```
%HTML(cache_example) {  
  
<html>  
  <head>  
    <title>This is the page title</title>  
  </head>  
  <body>  
    <center>  
      <h3>This is the Main Heading</h3>  
      <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>  
      <! Joe Smith sees a very short page                               !>  
      <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>  
      %IF (customer == "Joe Smith")
```

```
@DTW_EXIT()
```

```
%ENDIF
```

```
...
```

```
</body>  
</html>  
%}
```

DTW_GETCOOKIE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve el valor del cookie especificado.

Formato

@DTW_GETCOOKIE(nombre_cookie IN, valor_cookie OUT)

@DTW_rGETCOOKIE(nombre_cookie IN)

Parámetros

Tabla 35. Parámetros de DTW_GETCOOKIE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombre_cookie</i>	IN	Variable o serie literal que especifica el nombre del cookie.
serie	<i>valor_cookie</i>	OUT	<p>Variable que contiene el valor del cookie que recupera la función, por ejemplo información de estado de usuario.</p> <p>Usuarios de OS/400 y OS/390: Si el valor de cookie tiene codificaciones de estilo de URL (por ejemplo "%20"), el valor de cookie se decodifica antes de que se devuelva el valor.</p> <p>Usuarios de estación de trabajo: Si el valor de cookie tiene codificaciones de estilo de URL (por ejemplo "%20"), el valor <i>no</i> se decodifica antes de que se devuelva el valor.</p>

Códigos de Retorno

Tabla 36. Códigos de retorno de DTW_GETCOOKIE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.

Tabla 36. Códigos de retorno de DTW_GETCOOKIE (continuación)

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
8000	No se puede encontrar el cookie.

Notas de utilización

Defina y recupere un cookie en dos peticiones HTTP independientes. Puesto que un cookie sólo resulta visible después de haberlo enviado al cliente, en el caso de que una macro intente obtener un cookie que se haya definido en la misma petición de HTTP, puede que reciba resultados inesperados.

Ejemplos

Ejemplo 1: Recupera cookies que contengan información de contraseña e ID de usuario

```
@DTW_GETCOOKIE("mycookie_name_for_userID", userID)
@DTW_GETCOOKIE("mycookie_name_for_password", password)
```

Ejemplo 2: Determina si existe un cookie para un usuario antes de recopilar la información de usuario

```
%MESSAGE {
    8000 : "" : continue
}%

%HTML(welcome) {
    <html>
    <body>
    <h1>Net.Data Club</h1>
    @DTW_GETCOOKIE("NDC_name", name)
    %IF ($(RETURN_CODE) == "8000") %{ The cookie is not found. %}
    <form method="post" action="remember">
    <p>Welcome to the club. Please enter your name.<br />
    <input name="name" />
    <input type="submit" value="submit" /><br />
    </form>
    %ELSE
    <p>Hi, $(name). Welcome back.</p>
    %ENDIF
    </body>
    </html>
    %}
```

La sección de bienvenida de HTML comprueba si existe el cookie NDC_name. Si existe el cookie, el navegador visualiza una salutación personalizada. Si no existe el cookie, el formulario solicita el nombre de usuario y lo remite a la sección "remember", la cual define el nombre de usuario en el NDC_name del cookie tal y como se muestra a continuación:

```
%HTML(remember) {  
    <html>  
    <body>  
    <h1>Net.Data Club</h1>  
    @DTW_SETCOOKIE("NDC_name",  
                    name,  
                    "expires=Wednesday, 01-Dec-2010 00:00:00;path=/")  
    <p>Thank you.</p>  
    <p><a href="welcome">Come back</a></p>  
    </body>  
    </html>  
    %}
```

DTW_GETENV

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve el valor de la variable de entorno especificada.

Formato

@DTW_GETENV(NombreVarent, ValorVarent)

@DTW_rGETENV(NombreVarent)

Parámetros

Tabla 37. Parámetros de DTW_GETENV

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>NombreVarent</i>	IN	Variable o serie literal.
serie	<i>ValorVarent</i>	OUT	El valor de la variable de entorno especificada en <i>NombreVarent</i> . Se devuelve una serie nula si no se encuentra el valor.

Códigos de Retorno

Tabla 38. Códigos de retorno de DTW_GETENV

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Notas de utilización

También puede utilizar la sentencia ENVVAR para hacer referencia a los valores de las variables de entorno. Para obtener más información, consulte el apartado “Sentencia ENVVAR” en la página 17.

Ejemplos

Ejemplo 1: Devuelve el valor para la sentencia PATH en el parámetro OUT

```
@DTW_GETENV(myEnvVarName, myEnvVarValue)
```

- La entrada: myEnvVarName = "PATH"
- Devuelve: myEnvVarValue = "/usr/bin"

Ejemplo 2: Devuelve el valor para la sentencia PATH

```
@DTW_rGETENV(myPath)
```

- La entrada: myPath = "PATH"
- Devuelve: "/usr/bin"

Ejemplo 3: Devuelve el valor para el protocolo del servidor

```
<p>The server is @DTW_rGETENV("SERVER_PROTOCOL").</p>
```

Devuelve:

El servidor es "HTTP/1.0".

DTW_GETINIDATA

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve el valor de la variable de configuración especificada.

Formato

@DTW_GETINIDATA(NombreVarini, ValorVarini)

@DTW_rGETINIDATA(NombreVarini)

Parámetros

Tabla 39. Parámetros de DTW_GETINIDATA

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>NombreVarini</i>	IN	Variable o serie literal.
serie	<i>ValorVarini</i>	OUT	El valor de la variable de configuración especificado en <i>NombreVarini</i> .

Códigos de Retorno

Tabla 40. Códigos de retorno de DTW_GETINIDATA

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Notas de utilización

1. Si se especifica una variable de configuración que no es el archivo de configuración, Net.Data devuelve una serie vacía.
2. **Para usuarios OS/390, OS/2, Windows NT y UNIX:** con esta llamada no se pueden recuperar ni las variables de vía de acceso de configuración (MACRO_PATH, EXEC_PATH y INCLUDE_PATH), ni las sentencias ENVIRONMENT.

3. **Para usuarios OS/400:** con esta llamada no se pueden recuperar sentencias ENVIRONMENT.

Ejemplos

Ejemplo 1: Devuelve el valor de la variable de vía de acceso de Net.Data.

```
myEnvVarName = "FFI_PATH"  
@DTW_GETINIDATA(myEnvVarName, myEnvVarValue)
```

Da como resultado: myEnvVarValue = "D:\FFI"

DTW_HTMLENCODE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Codifica los caracteres seleccionados utilizando códigos de escape de caracteres HTML.

Formato

@DTW_HTMLENCODE(serieEntrada, serieSalida)

@DTW_rHTMLENCODE(serieEntrada)

Parámetros

Tabla 41. Parámetros de DTW_HTMLENCODE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
serie	<i>serieSalida</i>	OUT	Variable que contiene la serie de entrada modificada en la que determinados caracteres han sido sustituidos por los códigos de escape de caracteres HTML.

Códigos de Retorno

Tabla 42. Códigos de retorno de DTW_HTMLENCODE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Notas de utilización

1. Utilice esta función para codificar los datos de tipo carácter que no desea que el navegador de la Web interprete como HTML. Por ejemplo, utilizando el código de escape apropiado, puede visualizar caracteres

como por ejemplo menor que (<) y mayor que (>) en una página Web, que, de otro modo el navegador interpretaría como componentes de los códigos de HTML.

2. La Tabla 43 muestra los caracteres que se codifican por medio de la función DTW_HTMLENCODE.

Tabla 43. Códigos de escape de caracteres para HTML

Carácter	Nombre	Código
SPACE	Espacio	
"	Comillas dobles	"
#	Símbolo de número	#
%	Porcentaje	%
&	Ampersand	&
[Corchete izquierdo	(
]	Corchete derecho)
+	Signo más	+
\	Barra inclinada	\
:	Dos puntos	:
;	Punto y coma	;
<	Menor que	<
=	Igual a	=
>	Mayor que	>
?	Signo de interrogación	?
@	Signo de arrobas	@
/	Barra inclinada invertida	/
^	Acento circunflejo	^
{	Llave izquierda	{
	Barra vertical	|
}	Llave derecha	}
~	Tilde	~

Ejemplos

Ejemplo 1: Codifica el carácter de espacio

```
@DTW_HTMLENCODE(string1,string2)
```

- La entrada: string1 = "Jim's dog"
- Devuelve: string2 = "Jim's dog"

Ejemplo 2: Codifica espacios, el signo de menor que y el signo igual

@DTW_rHTMLENCODE("X <= 10")

- Devuelve: "X <= 10"

DTW_QHTMLENCODE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Efectúa la misma función que @DTW_HTMLENCODE pero codifica asimismo el carácter de comilla simple (') como '. Los códigos de escape de carácter de HTML que utiliza DTW_QHTMLENCODE se muestran en la Tabla 43 en la página 165.

Formato

@DTW_QHTMLENCODE(*serieEntrada*, *serieSalida*)

@DTW_rQHTMLENCODE(*serieEntrada*)

Parámetros

Tabla 44. Parámetros de DTW_QHTMLENCODE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
serie	<i>serieSalida</i>	OUT	Variable que contiene el formato modificado de <i>serieEntrada</i> en el que determinados caracteres se sustituyen por medio de códigos de escape de caracteres HTML.

Códigos de Retorno

Tabla 45. Códigos de retorno de DTW_QHTMLENCODE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1: Codifica un apóstrofo y un espacio

@DTW_QHTMLENCODE(string1,string2)

- La entrada: string1 = "Jim's dog"
- Devuelve: string2 = "Jim's dog"

Ejemplo 2: Codifica apóstrofos, espacios y un ampersand

@DTW_rQHTMLENCODE("John's & Jane's")

- Devuelve: "John's & Jane's"

DTW_SENDMAIL

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Transmite y crea dinámicamente mensajes de correo electrónico (e-mail).

Formato

Nota para clientes Beta

La implantación de esta característica sigue revisándose y pudiera haber cambiado cuando el producto se entregue finalmente al público en general.

@DTW_SENDMAIL(Remitente IN, Destinatario IN, Mensaje IN, Tema IN, Copia IN, Con copia oculta IN, Responder a IN, Organización IN, Conexiones IN)

@DTW_SENDMAIL((Remitente IN, Destinatario IN, Mensaje IN, Tema IN, Copia IN, Con copia oculta IN, Responder a IN, Organización IN)

@DTW_SENDMAIL((Remitente IN, Destinatario IN, Mensaje IN, Tema IN, Copia IN, Con copia oculta IN, Responder a IN)

@DTW_SENDMAIL((Remitente IN, Destinatario IN, Mensaje IN, Tema IN, Copia IN, Con copia oculta IN)

@DTW_SENDMAIL((Remitente IN, Destinatario IN, Mensaje IN, Tema IN, Copia IN)

@DTW_SENDMAIL((Remitente IN, Destinatario IN, Mensaje IN, Tema IN)

@DTW_SENDMAIL((Remitente IN, Destinatario IN, Mensaje IN)

Parámetros

Tabla 46. Parámetros de DTW_SENDMAIL

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>remitente</i>	IN	Variable o serie literal que especifica la dirección del autor. Este parámetro es obligatorio. Los formatos válidos son: <ul style="list-style-type: none">• Nombre <usuario@dominio>• <usuario@dominio>• usuario@dominio

Tabla 46. Parámetros de DTW_SENDMAIL (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>destinatario</i>	IN	Variable o serie literal que especifica las direcciones de correo electrónico (e-mail) al que se enviará este mensaje. Este valor puede contener varios destinatarios, separados por una coma (.). Este parámetro es obligatorio. Los formatos de <i>destinatario</i> válidos son: <ul style="list-style-type: none"> • Nombre <usuario@dominio> • <usuario@dominio> • usuario@dominio
serie	<i>mensaje</i>	IN	Variable o serie literal que contiene el texto del mensaje de correo electrónico (e-mail). Este parámetro es obligatorio.
serie	<i>tema</i>	IN	Variable o serie literal que contiene el texto de la línea del tema. Este es un parámetro opcional. Debe especificar una serie nula ("") para especificar parámetros adicionales.
serie	<i>Copia</i>	IN	Variable o serie literal que contiene las direcciones de correo electrónico (e-mail) o los nombres y direcciones de correo electrónico (e-mail) de destinatarios adicionales. Este valor puede contener varios destinatarios adicionales separados por una coma (.). Consulte el parámetro <i>Destinatario</i> para ver cuales son los formatos de destinatario válidos. Este es un parámetro opcional. Debe especificar una serie nula ("") para especificar parámetros adicionales.

Tabla 46. Parámetros de DTW_SENDMAIL (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>Concopiaoculta</i>	IN	Variable o serie literal que contiene las direcciones de correo electrónico (e-mail) o los nombres y direcciones de correo electrónico (e-mail) de destinatarios adicionales, aunque los destinatarios no aparezcan en la cabecera del correo electrónico (e-mail). Este valor puede contener varios destinatarios adicionales separados por una coma (.). Consulte el parámetro <i>Destinatario</i> para ver cuales son los formatos de destinatario válidos. Este es un parámetro opcional. Debe especificar una serie nula ("") para especificar parámetros adicionales.
serie	<i>ResponderA</i>	IN	Variable o serie literal que contiene las direcciones de correo electrónico (e-mail) a las que deberían enviarse las respuestas a este mensaje. Este es un parámetro opcional. Debe especificar una serie nula ("") para especificar parámetros adicionales. Los formatos <i>ResponderA</i> válidos son: <ul style="list-style-type: none"> • Nombre <usuario@dominio> • <usuario@dominio> • usuario@dominio
serie	<i>Organización</i>	IN	Variable o serie literal que contiene el nombre de organización del <i>remitente</i> . Este es un parámetro opcional.

Tabla 46. Parámetros de DTW_SENDMAIL (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
serie	Conexiones	IN	<p>Variable o serie literal que especifica las vías de acceso relativas a los archivos que han de enviarse. Este valor puede contener varios archivos, separados por una coma (,), pero no puede contener la serie "..". Estos archivos se examinarán en el orden de los directorios listados en la variable de configuración ATTACHMENT_PATH.</p> <p>Pueden enviarse los siguientes tipos de contenido como conexiones:</p> <ul style="list-style-type: none"> • image/jpeg • image/gif • audio/basic • application/octet-stream (general)

Códigos de Retorno

Tabla 47. Códigos de retorno de DTW_SENDMAIL

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
7000	Net.Data no puede conectarse al servidor SMTP especificado.
7001	Se ha producido un error SMTP mientras Net.Data intentaba transmitir el mensaje de correo electrónico al servidor SMTP especificado.
7002	El servidor SMTP especificado no da soporte a ESMTP (protocolo simple de transferencia de correo ampliado).

Notas de utilización

1. Puede utilizar la variable de configuración opcional, DTW_SMTP_SERVER, para especificar el servidor de SMTP a utilizar para transmitir mensajes de correo electrónico (e-mail). El valor de este parámetro puede ser un nombre de sistema principal o una dirección de IP. Cuando esta variable no esté definida, Net.Data utiliza el sistema principal local como servidor de SMTP. Consulte el capítulo sobre configuración de la *Guía de administración y programación de Net.Data* para el sistema operativo para obtener más información sobre esta variable. DTW_SMTP_SERVER se ignora en OS/390.
2. **Usuarios de OS/2, Windows NT y UNIX:** Los servidores del Protocolo simple de transferencia de correo estándar (SMTP) sólo aceptan datos de 7 bits, por ejemplo los caracteres ASCII de inglés americano. Si el mensaje tiene caracteres de 8 bits, es recomendable especificar un servidor de Protocolo simple de transferencia de correo ampliado (ESMTP); los servidores de ESMTP aceptan caracteres de 8 bits. Net.Data no codifica los datos de 8 bits en datos de 7 bits. Si no tiene acceso a un servidor de ESMTP, elimine todos los caracteres de 8 bits del mensaje de correo electrónico (e-mail).

Los usuarios de Net.Data para OS/390 no necesitan modificar mensajes de correo electrónico (e-mail) para servidores de SMTP.

3. Soporte del conjunto de caracteres:
 - **Usuarios de OS/400:** Puede utilizar la variable de configuración opcional, DTW_SMTP_CHARSET para especificar el carácter de ASCII que ha de utilizarse al convertir el mensaje de EBCDIC a ASCII. Si no se especifica DTW_SMTP_CHARSET, el conjunto de caracteres por omisión es iso-8859-1. Consulte el capítulo sobre configuración del manual *Net.Data Administration and Programming Guide for OS/400* para obtener más información sobre esta variable y los conjuntos de caracteres soportados.
 - **Usuarios de OS/2, Windows NT y UNIX:** La Tabla 48 describe los conjuntos de caracteres soportados:

Tabla 48. Conjuntos de caracteres a los que da soporte Net.Data

Entorno nacional	Conjunto de caracteres	Página de códigos OS/2 o UNIX	Página de códigos Windows NT
Estados Unidos, Europa occidental	"iso-8859-1"	819	1252
Japón	"x-sjis"	943	932
Chino (simplificado)	"gb2312"	1381	936
Corea	"euc-kr"	970	949
Chino (tradicional)	"big5"	950	950

4. La lista siguiente describe las condiciones bajo las que Net.Data no envía un mensaje de correo electrónico (e-mail):
- No se puede acceder al servidor SMTP especificado.
 - El servidor SMTP especificado no da soporte al 8BITMIMEProtocolo simple de transferencia de correo ampliado (ESMTP), pero el mensaje de correo electrónico contiene caracteres ASCII no de inglés americano.

Ejemplos

Ejemplo 1: La llamada de función que crea y envía un mensaje de correo electrónico (e-mail) simple.

```
@DTW_SENDMAIL("<ibmuser1@ibm.com>",  
"<ibmuser2@ibm.com>","There is a  
meeting at 9:30.", "Status meeting")
```

La función DTW_SENDMAIL envía un mensaje de correo electrónico (e-mail) con la información siguiente:

```
Date: Mon, 3 Apr 1998 09:54:33 PST  
To: <ibmuser2@ibm.com>  
From: <ibmuser1@ibm.com>  
Subject: Status meeting
```

There is a meeting at 9:30.

La información para Fecha se interpreta utilizando las funciones de fecha y hora del sistema y se formatea en un formato de datos específico de SMTP.

Ejemplo 2: Llamada de función que crea y envía un mensaje de correo electrónico (e-mail) con varios destinatarios, copia y destinatarios con copia oculta y el nombre de la empresa.

```
@DTW_SENDMAIL("IBM User 1 <ibmuser1@ibm.com>","  
"IBM User 2 <ibmuser2@ibm.com>,"  
IBM User 3 <ibmuser3@ibm.com>, IBM User 4 <ibmuser4@ibm.com>","  
"There is a  
meeting at 9:30.", "Status meeting", "IBM User 5  
<ibmuser5@ibm.com>","  
"IBM User 6 <ibmuser6@ibm.com>, "meeting@ibm.com", "IBM")
```

La función DTW_SENDMAIL envía un mensaje de correo electrónico (e-mail) con la información siguiente:

```
Date: Mon, 3 Apr 1998 09:54:33 PST  
To: IBM User 2 <ibmuser2@ibm.com>, IBM User 3  
<ibmuser3@ibm.com>,"  
IBM User 4 <ibmuser4@ibm.com>  
CC: IBM User 5 <ibmuser5@ibm.com>  
BCC: IBM User 6 <ibmuser6@ibm.com>  
From: IBM User 1 <ibmuser1@ibm.com>  
ReplyTo: meeting@ibm.com
```

Organization: IBM
Subject: Status meeting

There is a meeting at 9:30.

Ejemplo 3: Macro que crea y envía un correo electrónico (e-mail) a través de una interfaz de formulario de Web

```
%HTML(start) {  
  <html>  
  <body>  
  <h1>Net.Data E-Mail Example</h1>  
  <form method="post" action="sendemail">  
  <p>To:<br /><input name="recipient" /></p>  
  <p>Subject:<br /><input name="subject" /></p>  
  <p>Message:<br /><textarea name="message" rows=20 cols=40>  
  </textarea></p>  
  <p><input type="submit" value="Send E-mail"></p>  
  </form>  
  </body>  
  </html>  
  %}  
  
%HTML(sendemail) {  
  <html>  
  <body>  
  <h1>Net.Data E-Mail Example</h1>  
  @DTW_SENDMAIL("Net.Data E-mail Service <netdata@us.ibm.com>",  
    recipient, message, subject)  
  <p>E-mail has been sent out.</p>  
  </body>  
  </html>  
  %}
```

Esta macro envía un correo electrónico (e-mail) a través de una interfaz de formulario de Web. La sección de inicio de HTML visualiza un formulario en el que puede escribirse un mensaje, un tema y la dirección de correo electrónico (e-mail) del destinatario. Cuando el usuario pulsa el botón de **Enviar correo electrónico** (e-mail), el mensaje se envía a los destinatarios especificados en la sección HTML(sendemail). Esta sección llama a DTW_SENDMAIL y utiliza los parámetros que se han obtenido del formulario de la Web para determinar el contenido del mensaje de correo electrónico (e-mail), así como el remitente y los destinatarios. Una vez se han enviado los mensajes de correo electrónico (e-mail), se visualiza un aviso de confirmación.

Ejemplo 4: Macro que utiliza una consulta de SQL para determinar la lista de destinatarios

```
%Function(DTW_SQL) mailing_list(IN message) {  
  SELECT EMAIL_ADDRESS FROM CUSTOMERS WHERE ZIPCODE='CA'  
  %REPORT {  
    Sending product information to our customers in California...<p>  
  %ROW {
```

```

@DTW_SENDMAIL("John Doe Corp. <john.doe@doe.com>", V1, message,
               "New Product Release")
E-mail sent out to customer $(V1).<br />
%}
%}
%}

```

Esta macro envía un mensaje de correo electrónico (e-mail) automatizado a un grupo de clientes especificado que determina el resultado de una consulta de SQL a partir de la base de datos del cliente. La consulta de SQL también recupera las direcciones de correo electrónico (e-mail) de los clientes. El contenido del correo electrónico (e-mail) lo determina el valor del *mensaje* y puede ser estático o dinámico (por ejemplo, puede utilizar otra consulta de SQL para especificar dinámicamente el número de versión del producto o los precios de diversas ofertas).

DTW_SETCOOKIE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Genera código de JavaScript que define un cookie en el sistema cliente.

Formato

@DTW_SETCOOKIE(nombre_cookie IN, valor_cookie IN, opcs_avan IN)

@DTW_SETCOOKIE(nombre_cookie IN, valor_cookie IN)

Parámetros

Tabla 49. Parámetros de DTW_SETCOOKIE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombre_cookie</i>	IN	Variable o serie literal que especifica el nombre del cookie
serie	<i>valor_cookie</i>	IN	Variable o serie literal que especifica el valor del cookie. Evite la utilización de puntos y comas, comas y espacios como parte del <i>valor_cookie</i> . Cuando sea necesario, utilice la función DTW_rURLESCSEQ de Net.Data para procesar la serie que contiene los caracteres especiales antes de pasarla a DTW_SETCOOKIE. Por ejemplo, @DTW_SETCOOKIE("my_cookie_name", @DTW_rURLESCSEQ("my cookie value"))
serie	<i>opcs_avan</i>	IN	Serie que contiene atributos opcionales, separados por puntos y comas, que se utilizan para definir el cookie.*

Tabla 49. Parámetros de DTW_SETCOOKIE (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
*Los atributos opcionales pueden ser:			
	expires = <i>fecha</i>		<p>Especifica una serie de fecha que define el tiempo de vida válido del cookie. Después de que caduque la fecha, el cookie ya no se almacena ni se recupera.</p> <p>Sintaxis:</p> <p><i>día laborable</i>, <i>DD-mes-AAAA</i> <i>HH:MM:SSGMT</i></p> <p>Donde:</p> <p><i>día laborable</i> Especifica el nombre completo del día laborable.</p> <p><i>DD</i> Especifica la fecha numérica del mes.</p> <p><i>mes</i> Especifica la abreviatura de tres caracteres del mes.</p> <p><i>AAAA</i> Especifica el número del año con cuatro caracteres.</p> <p><i>HH:MM:SS</i> Especifica la indicación de la hora con horas, minutos y segundos.</p>
	domain = <i>nombre_dominio</i>		<p>Especifica los atributos de dominio del cookie, para su utilización en la comparación del atributo de dominio.</p>
	path = <i>vía de acceso</i>		<p>Especifica el subconjunto de los URL en un dominio para el que el cookie es válido.</p>
	secure		<p>Especifica que el cookie se transmita sólo a través de canales protegidos hacia servidores HTTPS.</p> <p>Cuando no se especifica la opción secure, el cookie puede enviarse a través de canales no protegidos. La opción protegida (secure) no necesita que el navegador cifre el cookie, ni asegurarse de que la página que contiene la sentencia DTW_SETCOOKIE se transmita a través de SSL.</p>
Para obtener información adicional sobre todas las opciones avanzadas, consulte la especificación de cookie de Netscape en http://home.netscape.com			

Códigos de Retorno

Tabla 50. Códigos de retorno de DTW_SETCOOKIE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.

Notas de utilización

1. Si el navegador de la Web cliente no da soporte a Java Script, el navegador no define el cookie.
2. Puesto que DTW_SETCOOKIE genera el código de Java Script, no llama a DTW_SETCOOKIE dentro de un elemento HTML <script> o <noscript>.
3. Para recuperar un cookie, utilice la función DTW_GETCOOKIE(). Consulte "DTW_GETCOOKIE" en la página 157 para obtener información sobre el modo de definir un cookie.
4. Defina y recupere un cookie en dos peticiones HTTP independientes. Puesto que un cookie sólo resulta visible después de haberlo enviado al cliente, en el caso de que una macro intente obtener un cookie que se haya definido en la misma petición de HTTP, es posible que reciba resultados inesperados.

Ejemplos

Ejemplo 1: Defina los cookies que contengan información de contraseña e ID de usuario con la opción avanzada Secure (Proteger)

```
@DTW_SETCOOKIE("mycookie_name_for_userID", "User1")
@DTW_SETCOOKIE("mycookie_name_for_password", "sd3dT", "secure")
```

Ejemplo 2: Defina cookies que contienen la opción avanzada de fecha de caducidad

```
@DTW_SETCOOKIE("mycookie_name_for_userID", "User1",
    "expires=Wednesday 01-Dec-2010 00:00:00")
@DTW_SETCOOKIE("mycookie_name_for_password", "sd3dT",
    "expires=Wednesday, 01-Dec-2010 00:00:00;secure")
```

Las llamadas de función deben estar en una línea; las líneas se dividen en este ejemplo al objeto de darle formato.

Ejemplo 3: Determina si existe un cookie para un usuario antes de recopilar información de usuario

```
%HTML(welcome) {  
  <html>  
  <body>  
  <h1>Net.Data Club</h1>  
  @DTW_GETCOOKIE("NDC_name", name)  
  %IF ($(RETURN_CODE) == "8000") %{ The cookie is not found. %}  
  <form method="post" action="remember">  
  <p>Welcome to the club. Please enter your name.<br />  
  <input name="name">  
  <input type="submit" value="submit"><br />  
  </form>  
  %ELSE  
  <p>Hi, $(name). Welcome back.  
  %ENDIF  
  </body>  
  </html>  
  %}
```

La sección de HTML(bienvenida) comprueba si existe el cookie NDC_name. Si existe el cookie, el navegador visualiza una salutación personalizada. Si no existe el cookie, el navegador solicita el nombre de usuario y lo remite a la sección HTML(remember). Esta sección registra el nombre de usuario en el cookie NDC_name tal y como se muestra a continuación:

```
%HTML(remember) {  
  <html>  
  <body>  
  <h1>Net.Data Club</h1>  
  @DTW_SETCOKIE("NDC_name", name,  
    "expires=Wednesday, 01-Dec-2010 00:00:00;path=/")  
  <p>Thank you.</p>  
  <p><a href="welcome">Come back</a></p>  
  </body>  
  </html>  
  %}
```

DTW_SETENV

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Asigna una variable de entorno con un valor específico y devuelve el valor anterior.

Formato

@DTW_SETENV(NombreVarent, ValorVarent, Valorant)

@DTW_rSETENV(NombreVarent, ValorVarent)

Parámetros

Tabla 51. Parámetros de DTW_SETENV

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>NombreVarent</i>	IN	Variable o serie literal que representa la variable de entorno.
serie	<i>ValorVarent</i>	IN	Variable o serie literal con el valor al que se asigna la variable de entorno.
serie	<i>Valorant</i>	OUT	Variable que contiene el valor anterior de la variable de entorno.

Códigos de Retorno

Tabla 52. Códigos de retorno de DTW_SETENV

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Notas de utilización

Si no se encuentra un valor anterior para la variable de entorno, se devuelve una serie vacía.

Ejemplos

Ejemplo 1: Devuelve el valor para la vía de acceso anterior

```
@DTW_SETENV("PATH", "myPath", prevValue)
```

- La entrada: envVarName = "PATH", envVarValue = "myPath"
- Devuelve: prevValue = "myPreviousPath"

Ejemplo 2: Devuelve el valor para la vía de acceso anterior y asigna el valor para el valor de PATH

```
@DTW_rSETENV("PATH", "myPath")
```

- La entrada: envVarName = "PATH", envVarValue = "myPath"
- Devuelve: "myPreviousPath", PATH = "myPath"

DTW_TIME

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve la hora actual del sistema en el formato especificado.

Formato

@DTW_TIME(*serieEntrada*, *serieSalida*)

@DTW_TIME(*serieSalida*)

@DTW_rTIME(*serieEntrada*)

@DTW_rTIME()

Parámetros

Tabla 53. Parámetros de DTW_TIME

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal que especifica el formato de fecha. Los formatos válidos son: C - Hora civil (hh:mmAM/PM utilizando un reloj de 12 horas) L - Hora local (hh:mm:ss) N - Hora normal (hh:mm:ss utilizando un reloj de 24 horas); valor por omisión X - Hora ampliada (hh:mm:ss.ccc, utilizando un reloj de 24 horas y en el que ccc es el número de milisegundos) H - Número de horas desde la medianoche M - Número de minutos desde la medianoche S - Número de segundos desde la medianoche
serie	<i>serieSalida</i>	OUT	Variable que contiene la hora en el formato especificado.

Códigos de Retorno

Tabla 54. Códigos de retorno de DTW_TIME

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1: Formato de reloj de 24 horas

@DTW_TIME(results)

- Devuelve: results = "10:30:53"

Ejemplo 2: Formato de hora civil

@DTW_TIME("C", results)

- Devuelve: results = "10:30AM"

Ejemplo 3: Devuelve el número de minutos desde la medianoche con la llamada de función

@DTW_rTIME("M")

- Devuelve: "630"

Ejemplo 4: Devuelve los formatos de datos y hora por omisión con la llamada de función

%REPORT{

<p>This report was created at @DTW_rTIME(), @DTW_rDATE().</p>

%}

- Devuelve: This report was created 15:04:39, 01 May 1997.

DTW_URLESCSEQ

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Sustituye los caracteres seleccionados que no se admiten en un URL con sus valores de escape, conocidos como códigos codificados de URL.

Formato

@DTW_URLESCSEQ(*serieEntrada*, *serieSalida*)

@DTW_rURLESCSEQ(*serieEntrada*)

Parámetros

Tabla 55. Parámetros de DTW_URLESCSEQ

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
serie	<i>serieSalida</i>	OUT	Variable que contiene la serie de entrada con caracteres que no se admiten en los URL que se sustituyen por sus valores de escape hexadecimal.

Códigos de Retorno

Tabla 56. Códigos de retorno de DTW_URLESCSEQ

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Notas de utilización

Utilice esta función para pasar cualquiera de los caracteres listados en la Tabla 57 en la página 186 a otra macro o bloque HTML.

Tabla 57. Valores de escape de caracteres para los URL

Carácter	Nombre	Código
ESPACIO	Espacio	%20
"	Comillas dobles	%22
#	Símbolo de número	%23
%	Porcentaje	%25
&	Ampersand	%26
+	Signo más	%2B
\	Barra inclinada invertida	%2F
:	Dos puntos	%3A
;	Punto y coma	%3B
<	Menor que	%3C
=	Igual a	%3D
>	Mayor que	%3E
?	Signo de interrogación	%3F
@	Signo de arrobas	%40
[Corchete izquierdo	%5B
/	Barra inclinada	%5C
]	Corchete derecho	%5D
^	Acento circunflejo	%5E
{	Llave izquierda	%7B
	Barra vertical	%7C
}	Llave derecha	%7D
~	Tilde	%7E

Ejemplos

Ejemplo 1: Sustituye los caracteres de espacio y de ampersand de *string1* por sus valores de escape y asigna el resultado a *string2*

```
@DTW_URLESCSEQ(string1,string2)
```

- La entrada: `string1 = "Guys & Dolls"`
- Devuelve: `string2 = "Guys%20%26%20Dolls"`

Ejemplo 2: Sustituye los caracteres de espacio de ampersand por sus códigos de escape.

```
@DTW_rURLESCSEQ("Guys & Dolls")
```

- Devuelve: `"Guys%20%26%20Dolls"`

Ejemplo 3: Utiliza DTW_rURLESCSEQ en un bloque ROW y sustituye los caracteres de espacio y 'at' (@) por sus códigos de escape.

```
%ROW{  
<p><a href="fullRpt.mac/input  
?name=@DTW_rURLESCSEQ(V1)&email=@DTW_rURLESCSEQ(V2)">  
$(V1)</a>  
%}
```

- La entrada: V1="Patrick O'Brien", V2="obrien@ibm.com"
- Devuelve:

```
<p><a href="fullrpt.mac/input?name=Patrick%20O'Brien  
&email="obrien%40ibm.com">Patrick O'Brien</a></p>
```

Cuando el usuario de la aplicación pulsa en el nombre "Patrick O'Brien," los valores que se han especificado para el nombre y dirección de correo electrónico (e-mail) fluyen dentro de la serie de consulta del URL que hace que Net.Data ejecute la sección de entrada de la macro fullrpt.mac.

Funciones matemáticas

Estas funciones le permiten efectuar cálculos matemáticos.

Consideraciones de NLS para funciones matemáticas: Net.Data visualiza comas decimales en los valores numéricos basándose en los valores regionales especificados en el servidor de la Web bajo el que se esté ejecutando Net.Data. Por ejemplo, si la coma decimal se especifica como coma (.) en el servidor de la Web, Net.Data utiliza la coma para dar formato a los datos decimales. Net.Data utiliza los valores siguientes para determinar el carácter que se utiliza para especificar una coma decimal:

Para los sistemas operativos OS/390, Windows NT, OS/2 y UNIX:

El LOCALE (entorno nacional) bajo el que se ejecuta el servidor de la Web

Para el sistema operativo OS/400:

- V4R2 o releases sucesivos: especificado por el perfil de usuario bajo el que se está ejecutando el proceso.
- V4R1 o releases anteriores: recuperado desde el valor del sistema QDECFMT.

Las funciones siguientes están disponibles para el cálculo matemático:

- “DTW_ADD” en la página 189
- “DTW_DIVIDE” en la página 191
- “DTW_DIVREM” en la página 193
- “DTW_FORMAT” en la página 195
- “DTW_INTDIV” en la página 199
- “DTW_MULTIPLY” en la página 201
- “DTW_POWER” en la página 203
- “DTW_SUBTRACT” en la página 205

DTW_ADD

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Añade dos números.

Formato

@DTW_ADD(número1, número2, precisión, resultado)

@DTW_ADD(número1, número2, resultado)

@DTW_rADD(número1, número2, precisión)

@DTW_rADD(número1, número2)

Parámetros

Tabla 58. Parámetros de DTW_ADD

Tipo de datos	Parámetro	Utilización	Descripción
flotante	<i>número1</i>	IN	Variable o serie literal que representa un número.
flotante	<i>número2</i>	IN	Variable o serie literal que representa un número.
entero	<i>precisión</i>	IN	Variable o serie literal que representa un número entero positivo que especifica la precisión del resultado. El valor por omisión es 9.
flotante	<i>resultado</i>	OUT	Variable que contiene la suma de <i>número1</i> y <i>número2</i> .

Códigos de Retorno

Tabla 59. Códigos de retorno de DTW_ADD

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Tabla 59. Códigos de retorno de DTW_ADD (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
4000	Un parámetro contiene un valor de número entero no válido.
4001	Un parámetro contiene un valor de número no válido.
4002	El resultado de una operación aritmética tenía un exponente que estaba fuera del rango soportado, comprendido entre -999,999,999 y +999,999,999.

Ejemplos

Ejemplo 1:

```
@DTW_ADD(NUM1, NUM2, "2",
result)
```

- La entrada: NUM1 = "105", NUM2 = "3"
- Devuelve: result = "1.1E+2"

Ejemplo 2:

```
@DTW_rADD("12", NUM2, "5")
```

- La entrada: NUM2 = "7.00"
- Devuelve: "19.00"

DTW_DIVIDE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Divide un número por otro.

Formato

@DTW_DIVIDE(número1, número2, precisión, resultado)

@DTW_DIVIDE(número1, número2, resultado)

@DTW_rDIVIDE(número1, número2, precisión)

@DTW_rDIVIDE(número1, número2)

Parámetros

Tabla 60. Parámetros de DTW_DIVIDE

Tipo de datos	Parámetro	Utilización	Descripción
flotante	<i>número1</i>	IN	Variable o serie literal que representa el número que ha de dividirse.
flotante	<i>número2</i>	IN	Variable o serie literal que representa un número.
entero	<i>precisión</i>	IN	Variable o serie literal que representa un número entero positivo que especifica la precisión del resultado. El valor por omisión es 9.
flotante	<i>resultado</i>	OUT	Variable que contiene el resultado de <i>número1</i> dividido por <i>número2</i> .

Códigos de Retorno

Tabla 61. Códigos de retorno de DTW_DIVIDE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Tabla 61. Códigos de retorno de DTW_DIVIDE (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
4000	Un parámetro contiene un valor de número entero no válido.
4001	Un parámetro contiene un valor de número no válido.
4002	El resultado de una operación aritmética tenía un exponente que estaba fuera del rango soportado, comprendido entre -999,999,999 y +999,999,999.

Ejemplos

Ejemplo 1:

@DTW_DIVIDE("8.0", NUM2, result)

- La entrada: NUM2 = "2"
- Devuelve: result = "4"

Ejemplo 2:

@DTW_rDIVIDE("1", NUM2, "5")

- La entrada: "1", NUM2 = "3"
- Devuelve: "0.33333"

Ejemplo 3:

@DTW_rDIVIDE(NUM1, "2", "5")

- La entrada: NUM1 = "5"
- Devuelve: "2.5"

DTW_DIVREM

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Divide un número por otro y devuelve el resto.

Formato

@DTW_DIVREM(número1, número2, precisión, resultado)

@DTW_DIVREM(número1, número2, resultado)

@DTW_rDIVREM(número1, número2, precisión)

@DTW_rDIVREM(número1, número2)

Parámetros

Tabla 62. Parámetros de DTW_DIVREM

Tipo de datos	Parámetro	Utilización	Descripción
flotante	<i>número1</i>	IN	Variable o serie literal que representa el número que ha de dividirse.
flotante	<i>número2</i>	IN	Variable o serie literal que representa un número.
entero	<i>precisión</i>	IN	Variable o serie literal que representa un número entero positivo que especifica la precisión del resultado. El valor por omisión es 9.
flotante	<i>resultado</i>	OUT	Variable que contiene el resto de <i>número1</i> dividido por <i>número2</i> .

Códigos de Retorno

Tabla 63. Códigos de retorno de DTW_DIVIDEREM

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Tabla 63. Códigos de retorno de DTW_DIVIDEREM (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
4000	Un parámetro contiene un valor de número entero no válido.
4001	Un parámetro contiene un valor de número no válido.
4002	El resultado de una operación aritmética tenía un exponente que estaba fuera del rango soportado, comprendido entre -999,999,999 y +999,999,999.

Notas de utilización

El signo del resto, si es diferente a cero, es el mismo que el del primer parámetro.

Ejemplos

Ejemplo 1:

```
@DTW_DIVREM(NUM1, NUM2,
result)
```

- La entrada: NUM1 = "2.1", NUM2 = "3"
- Devuelve: result = "2.1"

Ejemplo 2:

```
@DTW_rDIVREM("10", NUM2)
```

- La entrada: NUM2 = "0.3"
- Devuelve: "0.1"

Ejemplo 3:

```
@DTW_rDIVREM("3.6", "1.3")
```

- Devuelve: "1.0"

Ejemplo 4:

```
@DTW_rDIVREM("-10", "3")
```

- Devuelve: "-1"

DTW_FORMAT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Personaliza el formato de un número.

Formato

@DTW_FORMAT(número, antes, después, expp, expt, precisión, resultado)

@DTW_FORMAT(número, antes, después, expp, expt, resultado)

@DTW_FORMAT(número, antes, después, expp, resultado)

@DTW_FORMAT(número, antes, después, resultado)

@DTW_FORMAT(número, antes, resultado)

@DTW_FORMAT(número, resultado)

@DTW_rFORMAT(número, antes, después, expp, expt, precisión)

@DTW_rFORMAT(número, antes, después, expp, expt)

@DTW_rFORMAT(número, antes, después, expp)

@DTW_rFORMAT(número, antes, después)

@DTW_rFORMAT(número, antes)

@DTW_rFORMAT(número)

Parámetros

Tabla 64. Parámetros de DTW_FORMAT

Tipo de datos	Parámetro	Utilización	Descripción
flotante	<i>número</i>	IN	Variable o serie literal que representa un número.
entero	<i>antes</i>	IN	Variable o serie literal que representa un número entero positivo. Este es un parámetro opcional. Debe escribir una serie nula ("") para disponer de parámetros adicionales.
entero	<i>después</i>	IN	Variable o serie literal que representa un número entero positivo. Este es un parámetro opcional. Debe escribir una serie nula ("") para especificar parámetros adicionales.
entero	<i>expp</i>	IN	Variable o serie literal que representa un número entero positivo. Debe especificar una serie nula ("") para especificar parámetros adicionales.

Tabla 64. Parámetros de DTW_FORMAT (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
entero	<i>expt</i>	IN	Variable o serie literal que representa un número entero positivo. Debe escribir una serie nula ("") para especificar parámetros adicionales.
entero	<i>precisión</i>	IN	Variable o serie literal que representa un número entero positivo que especifica la precisión del resultado. El valor por omisión es 9.
flotante	<i>resultado</i>	OUT	Variable que contiene el número con el formato y redondeo especificados.

Códigos de Retorno

Tabla 65. Códigos de retorno de DTW_FORMAT

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
4000	Un parámetro contiene un valor de número entero no válido.
4001	Un parámetro contiene un valor de número no válido.

Notas de utilización

1. Si *número* es el único parámetro especificado, se da formato al resultado como si se hubiera ejecutado @DTW_rADD(number,"0").
2. Los parámetros *antes* y *después* describen el número de caracteres que se utilizan para las partes enteros y decimales del parámetro *resultado*, respectivamente. Si omite uno de estos parámetros o los dos, se utilizan tantos caracteres para esa parte como se necesiten.
3. Si el parámetro *antes* no es lo suficientemente grande como para que contenga la parte entera del número (más el signo de un número

negativo), se produce un error. Si el parámetro *antes* es más grande que el que se necesita para dicha parte, el valor del parámetro *número* se rellena con blancos a la izquierda. Si el parámetro *después* no tiene el mismo tamaño que la parte decimal del parámetro *número*, el número se redondea (o se amplía con ceros) para que quepa. Especificar 0 hace que el número se redondea a un entero.

4. Los parámetros *expp* y *expt* controlan la parte exponencial del resultado. El parámetro *expp* establece el número de espacios para la parte exponencial; el valor por omisión es utilizar tantos como sea necesario (que puede ser cero). El parámetro *expt* establece el punto desencadenante de la utilización de la notación exponencial. El valor por omisión es el valor por omisión del parámetro de precisión.
5. Si *expp* es 0, no se proporciona ningún exponente y el número se expresa en un formato simple añadiendo los ceros que sean necesarios. Si *expp* no es lo suficientemente grande para que contenga el exponente, se produce un error.
6. Si el número de espacios necesarios para la parte decimal o entera supera *expt* o dos veces *expt*, respectivamente, utilice la notación exponencial. Si *expt* es 0, siempre se utiliza la notación exponencial a menos que el exponente sea 0. (Si *expp* es 0, este valor altera temporalmente un valor de 0 de *expt*.) Si el exponente es 0 cuando se especifica un valor de *expp* diferente a cero, se proporcionan *expp*+2 blancos para la parte exponencial del resultado. Si el exponente es 0 y no se especifica *expp*, se utiliza el formato simple.

Ejemplos

Ejemplo 1:

```
@DTW_FORMAT(NUM, BEFORE, result)
```

- La entrada: NUM = "3", BEFORE = "4"
- Devuelve: result = " 3"

Ejemplo 2:

```
@DTW_FORMAT("1.73", "4", "0",  
result)
```

- Devuelve: result = " 2"

Ejemplo 3:

```
@DTW_FORMAT("1.73", "4",  
"3", result)
```

- Devuelve: result = " 1.730"

Ejemplo 4:

```
@DTW_FORMAT(" - 12.73", "",  
"4", result)
```

- Devuelve: result = "-12.7300"

Ejemplo 5:

```
@DTW_FORMAT("12345.73", "",  
"", "2", "2", result)
```

- Devuelve: result = "1.234573E+04"

Ejemplo 6:

```
@DTW_FORMAT("1.234573", "",  
"3", "", "0", result)
```

- Devuelve: result = "1.235"

Ejemplo 7:

```
@DTW_rFORMAT(" - 12.73")
```

- Devuelve: " - 12.73"

Ejemplo 8:

```
@DTW_rFORMAT("0.000")
```

- Devuelve: "0"

Ejemplo 9:

```
@DTW_rFORMAT("12345.73", "",  
"", "3", "6")
```

- Devuelve: "12345.73"

Ejemplo 10:

```
@DTW_rFORMAT("1234567e5",  
"", "3", "0")
```

- Devuelve: "123456700000.000"

Ejemplo 11:

```
@DTW_rFORMAT("12345.73",  
"", "3", "", "0")
```

- Devuelve: "1.235E+4"

DTW_INTDIV

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Divide un número por otro y devuelve la parte entera del resultado.

Formato

@DTW_INTDIV(número1, número2, precisión, resultado)

@DTW_INTDIV(número1, número2, resultado)

@DTW_rINTDIV(número1, número2, precisión)

@DTW_rINTDIV(número1, número2)

Parámetros

Tabla 66. Parámetros de DTW_INTDIV

Tipo de datos	Parámetro	Utilización	Descripción
flotante	<i>número1</i>	IN	Variable o serie literal que representa el número que ha de dividirse.
flotante	<i>número2</i>	IN	Variable o serie literal que representa un número.
entero	<i>precisión</i>	IN	Variable o serie literal que representa un número entero positivo que especifica la precisión del resultado. El valor por omisión es 9.
flotante	<i>resultado</i>	OUT	Variable que contiene la parte de entera de <i>número1</i> dividido por <i>número2</i> .

Códigos de Retorno

Tabla 67. Códigos de retorno de DTW_INTDIV

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.

Tabla 67. Códigos de retorno de DTW_INTDIV (continuación)

Código de retorno	Explicación
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
4000	Un parámetro contiene un valor de número entero no válido.
4001	Un parámetro contiene un valor de número no válido.
4002	El resultado de una operación aritmética tenía un exponente que estaba fuera del rango soportado, comprendido entre -999,999,999 y +999,999,999.

Ejemplos

Ejemplo 1:

```
@DTW_INTDIV(NUM1, NUM2,
result)
```

- La entrada: NUM1 = "10", NUM2 = "3"
- Devuelve: result= " 3"

Ejemplo 2:

```
@DTW_rINTDIV("2", NUM2)
```

- La entrada: NUM2 = "3"
- Devuelve: "0"

DTW_MULTIPLY

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Multiplica dos números.

Formato

@DTW_MULTIPLY(número1, número2, precisión, resultado)

@DTW_MULTIPLY(número1, número2, resultado)

@DTW_rMULTIPLY(número1, número2, precisión)

@DTW_rMULTIPLY(número1, número2)

Parámetros

Tabla 68. Parámetros de DTW_MULTIPLY

Tipo de datos	Parámetro	Utilización	Descripción
flotante	<i>número1</i>	IN	Variable o serie literal que representa un número.
flotante	<i>número2</i>	IN	Variable o serie literal que representa un número.
entero	<i>precisión</i>	IN	Variable o serie literal que representa un número entero positivo que especifica la precisión del resultado. El valor por omisión es 9.
flotante	<i>resultado</i>	OUT	Variable que contiene el producto de <i>número1</i> y <i>número2</i> .

Códigos de Retorno

Tabla 69. Códigos de retorno de DTW_MULTIPLY

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Tabla 69. Códigos de retorno de DTW_MULTIPLY (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
4000	Un parámetro contiene un valor de número entero no válido.
4001	Un parámetro contiene un valor de número no válido.
4002	El resultado de una operación aritmética tenía un exponente que estaba fuera del rango soportado, comprendido entre -999,999,999 y +999,999,999.

Ejemplos

Ejemplo 1:

```
@DTW_MULTIPLY(NUM1, NUM2,
result)
```

- La entrada: NUM1 = "4", NUM2 = "5"
- Devuelve: result = "20"

Ejemplo 2:

```
@DTW_rMULTIPLY("0.9", NUM2)
```

- La entrada: NUM2 = "0.8"
- Devuelve: "0.72"

DTW_POWER

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Eleva un número entero a una potencia entera.

Formato

@DTW_POWER(número1, número2, precisión, resultado)

@DTW_POWER(número1, número2, resultado)

@DTW_rPOWER(número1, número2, precisión)

@DTW_rPOWER(número1, número2)

Parámetros

Tabla 70. Parámetros de DTW_POWER

Tipo de datos	Parámetro	Utilización	Descripción
flotante	<i>número1</i>	IN	Variable o serie literal que representa el número que ha de elevarse a una potencia.
flotante	<i>número2</i>	IN	Variable o serie literal que representa un número.
entero	<i>precisión</i>	IN	Variable o serie literal que representa un número entero positivo que especifica la precisión del resultado. El valor por omisión es 9.
flotante	<i>resultado</i>	OUT	Variable que contiene el resultado de <i>número1</i> elevado a la potencia de <i>número2</i> .

Códigos de Retorno

Tabla 71. Códigos de retorno de DTW_POWER

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Tabla 71. Códigos de retorno de DTW_POWER (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
4000	Un parámetro contiene un valor de número entero no válido.
4001	Un parámetro contiene un valor de número no válido.
4002	El resultado de una operación aritmética tenía un exponente que estaba fuera del rango soportado, comprendido entre -999,999,999 y +999,999,999.

Ejemplos

Ejemplo 1:

```
@DTW_POWER(NUM1, NUM2,
result)
```

- La entrada: NUM1 = "2", NUM2 = "-3"
- Devuelve: result = "0.125"

Ejemplo 2:

```
@DTW_rPOWER("1.7", NUM2,
precision)
```

- La entrada: NUM2 = "8", precision = "5"
- Devuelve: "69.758"

DTW_SUBTRACT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Resta un número del otro número.

Formato

@DTW_SUBTRACT(número1, número2, precisión, resultado)

@DTW_SUBTRACT(número1, número2, resultado)

@DTW_rSUBTRACT(número1, número2, precisión)

@DTW_rSUBTRACT(número1, número2)

Parámetros

Tabla 72. Parámetros de DTW_SUBTRACT

Tipo de datos	Parámetro	Utilización	Descripción
flotante	<i>número1</i>	IN	Variable o serie literal que representa el número del que ha de restarse otro número.
flotante	<i>número2</i>	IN	Variable o serie literal que representa un número.
entero	<i>precisión</i>	IN	Variable o serie literal que representa un número entero positivo que especifica la precisión del resultado. El valor por omisión es 9.
flotante	<i>resultado</i>	OUT	Variable que contiene la diferencia de <i>número1</i> y <i>número2</i> .

Códigos de Retorno

Tabla 73. Códigos de retorno de DTW_SUBTRACT

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Tabla 73. Códigos de retorno de DTW_SUBTRACT (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
4000	Un parámetro contiene un valor de número entero no válido.
4001	Un parámetro contiene un valor de número no válido.
4002	El resultado de una operación aritmética tenía un exponente que estaba fuera del rango soportado, comprendido entre -999,999,999 y +999,999,999.

Ejemplos

Ejemplo 1:

```
@DTW_SUBTRACT(NUM1, NUM2, comp)
%IF(comp > "0")
<p>$(NUM1) is larger than $(NUM2).
%ENDIF
```

- La entrada: NUM2 = "2.07"
- Devuelve: "-0.77"

Este ejemplo muestra el modo de comparar valores numéricos, que son series en Net.Data.

Ejemplo 2:

```
@DTW_SUBTRACT(NUM1, NUM2, result)
```

- La entrada: NUM1 = "1.3, NUM2 = "1.07"
- Devuelve: result = "0.23"

Ejemplo 3:

```
@DTW_rSUBTRACT("1.3", NUM2)
```

- La entrada: NUM2 = "2.07"
- Devuelve: "-0.77"

Funciones de serie

Las funciones siguientes son el conjunto de funciones de serie estándar a las que da soporte Net.Data:

- “DTW_ASSIGN” en la página 208
- “DTW_CHARTOHEX” en la página 209
- “DTW_CONCAT” en la página 211
- “DTW_DELSTR” en la página 213
- “DTW_HEXTOCHAR” en la página 215
- “DTW_INSERT” en la página 217
- “DTW_LASTPOS” en la página 220
- “DTW_LENGTH” en la página 222
- “DTW_LOWERCASE” en la página 224
- “DTW_POS” en la página 226
- “DTW_REPLACE” en la página 228
- “DTW_REVERSE” en la página 230
- “DTW_STRIP” en la página 232
- “DTW_SUBSTR” en la página 234
- “DTW_TRANSLATE” en la página 236
- “DTW_UPPERCASE” en la página 238

Soporte de MBCS para OS/390, OS/2, Windows NT y UNIX: Puede especificar el soporte del conjunto de caracteres de múltiples bytes (MBCS) para funciones de texto y serie con el valor de configuración de DTW_MBMODE. Especifique este valor en el archivo de inicialización de Net.Data; el valor por omisión es que no haya soporte. Puede alterar temporalmente el valor en el archivo de inicialización definiendo la variable DTW_MBMODE en una macro de Net.Data. Consulte la sección sobre la variable de configuración en el manual *Guía de administración y programación de Net.Data* y “DTW_MBMODE” en la página 135 para obtener más información.

Soporte de MBCS para OS/400: El soporte de DBCS se facilita automáticamente y no necesita esta variable.

Soporte de Unicode para OS/2, Windows NT y UNIX: Puede especificar soporte de Unicode UTF-8 para funciones de texto y serie con el valor de configuración de DTW_UNICODE. Especifique este valor en el archivo de inicialización de Net.Data; el valor por omisión es que no haya soporte. Consulte la sección sobre la variable de configuración en el manual *Guía de administración y programación de Net.Data* para obtener más información.

DTW_ASSIGN

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Asigna un valor a una variable.

Formato

@DTW_ASSIGN(*serieSalida*, *serieEntrada*)

Parámetros

Tabla 74. Parámetros de DTW_ASSIGN

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieSalida</i>	OUT	Variable que contiene la serie literal idéntica a <i>serieEntrada</i> .
serie	<i>serieEntrada</i>	IN	Variable o serie literal.

Códigos de Retorno

Tabla 75. Códigos de retorno de DTW_ASSIGN

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1:

```
@DTW_ASSIGN(RC, "0")
```

- Establece RC en "0".

Ejemplo 2:

```
@DTW_ASSIGN(string1, string2)
```

- Establece *string1* en el valor de *string2*.

DTW_CHARTOHEX

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Convierte cada carácter de una serie en dos caracteres hexadecimales.

Formato

@DTW_CHARTOHEX(*serieEntrada*, *serieSalida*)

@DTW_rCHARTOHEX(*serieEntrada*)

Parámetros

Tabla 76. Parámetros de DTW_CHARTOHEX

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal que va a convertirse.
serie	<i>serieSalida</i>	OUT	Variable que contiene <i>serieEntrada</i> representada en formato hexadecimal.

Códigos de Retorno

Tabla 77. Códigos de retorno de DTW_CHARTOHEX

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Notas de utilización

Cada carácter hexadecimal representa 4 bits del carácter de entrada (un carácter se representa por medio de 8 bits).

Ejemplos

Ejemplo 1: Sistemas operativos EBCDIC

@DTW_rCHARTOHEX("12345")

- Devuelve: "F1F2F3F4F5"

Ejemplo 2: Sistemas operativos ASCII

@DTW_rCHARTOHEX("12345")

- Devuelve: "3132333435"

DTW_CONCAT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Concatena dos series.

Formato

@DTW_CONCAT(*serieEntrada1*, *serieEntrada2*, *serieSalida*)

@DTW_rCONCAT(*serieEntrada1*, *serieEntrada2*)

Parámetros

Tabla 78. Parámetros de DTW_CONCAT

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada1</i>	IN	Variable o serie literal.
serie	<i>serieEntrada2</i>	IN	Variable o serie literal.
serie	<i>serieSalida</i>	OUT	Variable que contiene la serie ' <i>serieEntrada1serieEntrada2</i> ', donde <i>serieEntrada1</i> se concatena con <i>serieEntrada2</i> .

Códigos de Retorno

Tabla 79. Códigos de retorno de DTW_CONCAT

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1:

```
@DTW_CONCAT("This", " is a  
test.", result)
```

- Devuelve: result = "This is a test."

Ejemplo 2:

```
@DTW_CONCAT(string1,  
"1-2-3", result)
```

- La entrada: string1 = "Testing "
- Devuelve: result = "Testing 1-2-3"

Ejemplo 3:

```
@DTW_rCONCAT("This", " is a  
test.")
```

- Devuelve "This is a test."

DTW_DELSTR

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Suprime una subserie de una serie a partir del carácter *n* para *longitud* caracteres.

Formato

@DTW_DELSTR(*serieEntrada*, *n* , *longitud*, *serieSalida*)

@DTW_DELSTR(*serieEntrada*, *n* , *serieSalida*)

@DTW_rDELSTR(*serieEntrada*, *n*, *longitud*)

@DTW_rDELSTR(*serieEntrada*, *n*)

Parámetros

Tabla 80. Parámetros de DTW_DELSTR

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
entero	<i>n</i>	IN	La posición del carácter en el que comienza la supresión de la subserie. Si <i>n</i> supera la longitud de <i>serieEntrada</i> , <i>serieSalida</i> se establece en el valor de <i>serieEntrada</i> .
entero	<i>longitud</i>	IN	La longitud de la subserie a suprimir. El valor por omisión es suprimir todos los caracteres hasta el final de <i>serieEntrada</i> .
serie	<i>serieSalida</i>	OUT	Variable que contiene el formato modificado de <i>serieEntrada</i> .

Códigos de Retorno

Tabla 81. Códigos de retorno de DTW_DELSTR

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.

Tabla 81. Códigos de retorno de DTW_DELSTR (continuación)

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_DELSTR("abcde", "3", "2", result)
```

- Devuelve: result = "abe"

Ejemplo 2:

```
@DTW_rDELSTR("abcde", "4", "1")
```

- Devuelve: "abce"

DTW_HEXTOCHAR

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Convierte cada carácter hexadecimal de una serie en un valor de carácter.

Formato

@DTW_HEXTOCHAR(*serieEntrada*, *serieSalida*)

@DTW_rHEXTOCHAR(*serieEntrada*)

Parámetros

Tabla 82. Parámetros de DTW_HEXTOCHAR

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal que va a convertirse.
serie	<i>serieSalida</i>	OUT	Variable que contiene <i>serieEntrada</i> representada en formato de carácter.

Códigos de Retorno

Tabla 83. Códigos de retorno de DTW_HEXTOCHAR

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Notas de utilización

Cada carácter hexadecimal de la serie de entrada representa 4 bits en la serie de caracteres resultantes (un carácter está representado por medio de 8 bits). La serie de entrada debe contener un número par de caracteres hexadecimales y puede contener los caracteres siguientes: 0-9, A-F y a-f.

Ejemplos

Ejemplo 1: Sistemas operativos EBCDIC

@DTW_rHEXTOCHAR("F1F2F3")

- Devuelve: "123"

Ejemplo 2: Sistemas operativos ASCII

@DTW_rHEXTOCHAR("313233")

- Devuelve: "123"

DTW_INSERT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Inserta una serie en otra serie que comience después del carácter *n*.

Formato

@DTW_INSERT(*serieEntrada1*, *serieEntrada2*, *n*, *longitud*, *relleno*, *serieSalida*)

@DTW_INSERT(*serieEntrada1*, *serieEntrada2*, *n*, *longitud*, *serieSalida*)

@DTW_INSERT(*serieEntrada1*, *serieEntrada2*, *n*, *serieSalida*)

@DTW_INSERT(*serieEntrada1*, *serieEntrada2*, *serieSalida*)

@DTW_rINSERT(*serieEntrada1*, *serieEntrada2*, *n*, *longitud*, *relleno*)

@DTW_rINSERT(*serieEntrada1*, *serieEntrada2*, *n*, *longitud*)

@DTW_rINSERT(*serieEntrada1*, *serieEntrada2*, *n*)

@DTW_rINSERT(*serieEntrada1*, *serieEntrada2*)

Parámetros

Tabla 84. Parámetros de DTW_INSERT

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada1</i>	IN	Variable o serie literal que ha de insertarse en <i>serieEntrada2</i> .
serie	<i>serieEntrada2</i>	IN	Variable o serie literal.
entero	<i>n</i>	IN	La posición de carácter de <i>serieEntrada2</i> después de la cual se inserta <i>serieEntrada1</i> . Si <i>n</i> supera la longitud de <i>serieEntrada2</i> , se rellena con el carácter de relleno, <i>relleno</i> , hasta que tenga caracteres suficientes. El valor por omisión es insertar al principio de <i>serieEntrada2</i> .
entero	<i>longitud</i>	IN	El número de caracteres de <i>serieEntrada1</i> a insertar. La serie se rellena con el carácter de relleno, <i>relleno</i> , si este parámetro supera la longitud de <i>serieEntrada1</i> . El valor por omisión es la longitud de <i>serieEntrada1</i> .
entero	<i>relleno</i>	IN	El carácter de relleno, tal y como se describe para <i>n</i> y <i>longitud</i> . El valor de relleno por omisión es un blanco.

Tabla 84. Parámetros de DTW_INSERT (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieSalida</i>	OUT	Variable que contiene <i>serieEntrada2</i> que se modifica insertando <i>serieEntrada1</i> o parte de la misma.

Códigos de Retorno

Tabla 85. Códigos de retorno de DTW_INSERT

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_INSERT("123", "abc",
result)
```

- Devuelve: result = "123abc"

Ejemplo 2:

```
@DTW_INSERT("123", "abc",
"5", result)
```

- Devuelve: result = "abc 123"

Ejemplo 3:

```
@DTW_INSERT("123", "abc",
"5", "6", result)
```

- Devuelve: result = "abc 123"

Ejemplo 4:

```
@DTW_INSERT("123", "abc",
"5", "6", "/", result)
```

- Devuelve: result = "abc//123///"

Ejemplo 5:

```
@DTW_rINSERT("123", "abc",  
"5", "6", "+")
```

- Devuelve: "abc++123++"

DTW_LASTPOS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve la posición de la última ocurrencia de una serie en otra serie, comenzando a partir del carácter *n* y funciona en sentido inverso (de derecha a izquierda).

Formato

@DTW_LASTPOS(*serieEntrada1*, *serieEntrada2*, *n*, posición)

@DTW_LASTPOS(*serieEntrada1*, *serieEntrada2*, posición)

@DTW_rLASTPOS(*serieEntrada1*, *serieEntrada2*, *n*)

@DTW_rLASTPOS(*serieEntrada1*, *serieEntrada2*)

Parámetros

Tabla 86. Parámetros de DTW_LASTPOS

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada1</i>	IN	Variable o serie literal en que se busca <i>serieEntrada2</i> .
serie	<i>serieEntrada2</i>	IN	Variable o serie literal.
entero	<i>n</i>	IN	La posición de carácter de <i>serieEntrada2</i> en la que ha de comenzar a buscarse <i>serieEntrada1</i> . El valor por omisión es el de comenzar a buscar el último carácter y efectuar la exploración en sentido inverso (de derecha a izquierda).
entero	<i>posición</i>	OUT	La posición de la última ocurrencia de <i>serieEntrada1</i> en <i>serieEntrada2</i> . Si no se encuentra ninguna ocurrencia, se devuelve un 0.

Códigos de Retorno

Tabla 87. Códigos de retorno de DTW_LASTPOS

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.

Tabla 87. Códigos de retorno de DTW_LASTPOS (continuación)

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_LASTPOS(" ", "abc def  
ghi", result)
```

- Devuelve: result = "8"

Ejemplo 2:

```
@DTW_LASTPOS(" ", "abc def  
ghi", "10", result)
```

- Devuelve: result = "8"

Ejemplo 3:

```
@DTW_rLASTPOS(" ", "abc def  
ghi", "7")
```

- Devuelve: "4"

DTW_LENGTH

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve la longitud de una serie.

Formato

@DTW_LENGTH(*serieEntrada*, longitud)

@DTW_rLENGTH(*serieEntrada*)

Parámetros

Tabla 88. Parámetros de DTW_LENGTH

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
entero	<i>longitud</i>	OUT	Símbolo que contiene el número de caracteres de <i>serieEntrada</i> .

Códigos de Retorno

Tabla 89. Códigos de retorno de DTW_LENGTH

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1:

```
@DTW_LENGTH("abcdefgh",  
result)
```

- Devuelve: result = "8"

Ejemplo 2:

```
@DTW_rLENGTH("")
```

- Devuelve: "0"

DTW_LOWERCASE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve una serie todo en minúsculas.

Formato

@DTW_LOWERCASE(*serieEntrada*, *serieSalida*)

@DTW_rLOWERCASE(*serieEntrada*)

@DTW_mLOWERCASE(*serieMult1*, *serieMult2*, ..., *serieMultn*)

Parámetros

Tabla 90. Parámetros de DTW_LOWERCASE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal con caracteres en mayúsculas y minúsculas.
serie	<i>serieSalida</i>	OUT	Variable que contiene <i>serieEntrada</i> con todos los caracteres en minúsculas.
serie	<i>serieMult</i>	INOUT	<ul style="list-style-type: none">En la entrada: Variable que contiene una serie.En la salida: Variable que contiene la serie de entrada convertida en minúsculas.

Códigos de Retorno

Tabla 91. Códigos de retorno de DTW_LOWERCASE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1:

```
@DTW_LOWERCASE("This",  
stringOut)
```

- Devuelve: stringOut = "this"

Ejemplo 2:

```
@DTW_rLOWERCASE(string1)
```

- La entrada: string1 = "Hello"
- Devuelve: "hello"

Ejemplo 3:

```
@DTW_mLOWERCASE(string1,  
string2, string3)
```

- La entrada: string1 = "THIS", string2 = "IS", string3 = "LOWERCASE"
- Devuelve: string1 = "this", string2 = "is", string3 = "lowercase"

DTW_POS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve la posición de la primera ocurrencia de una serie en otra serie, utilizando una patrón de búsqueda hacia adelante.

Formato

@DTW_POS(*serieEntrada1*, *serieEntrada2*, *n*, *nSalida*)

@DTW_POS(*serieEntrada1*, *serieEntrada2*, *nSalida*)

@DTW_rPOS(*serieEntrada1*, *serieEntrada2*, *n*)

@DTW_rPOS(*serieEntrada1*, *serieEntrada2*)

Parámetros

Tabla 92. Parámetros de DTW_POS

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada1</i>	IN	Variable o serie literal que ha de buscarse.
serie	<i>serieEntrada2</i>	IN	Variable o serie literal a buscar.
entero	<i>n</i>	IN	La posición de carácter de <i>serieEntrada2</i> en la que ha de comenzar la búsqueda. El valor por omisión es el de comenzar la búsqueda en el primer carácter de <i>serieEntrada2</i> .
entero	<i>nSalida</i>	OUT	Variable que contiene la posición de la primera ocurrencia de <i>serieEntrada1</i> en <i>serieEntrada2</i> . Si no se encuentra ninguna ocurrencia, se devuelve un 0.

Códigos de Retorno

Tabla 93. Códigos de retorno de DTW_POS

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.

Tabla 93. Códigos de retorno de DTW_POS (continuación)

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_POS("day", "Saturday",
result)
```

- Devuelve: result = "6"

Ejemplo 2:

```
@DTW_POS("a", "Saturday",
"3", result)
```

- Devuelve: result = "7"

Ejemplo 3:

```
@DTW_rPOS(" ", "abc def
ghi", "5")
```

- Devuelve: "8"

DTW_REPLACE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Sustituye caracteres en una serie.

Formato

@DTW_REPLACE(*serieEntrada*, *serieDesde*, *serieA*, *n*, *opción*, *serieSalida*)

@DTW_REPLACE(*serieEntrada*, *serieDesde*, *serieA*, *n*, *serieSalida*)

@DTW_REPLACE(*serieEntrada*, *serieDesde*, *serieA*, *serieSalida*)

@DTW_rREPLACE(*serieEntrada*, *serieDesde*, *serieA*, *n*, *opción*)

@DTW_rREPLACE(*serieEntrada*, *serieDesde*, *serieA*, *n*)

@DTW_rREPLACE(*serieEntrada*, *serieDesde*, *serieA*)

Parámetros

Tabla 94. Parámetros de DTW_REPLACE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal que va a buscarse.
serie	<i>serieDesde</i>	IN	Variable o serie literal que ha de sustituirse.
serie	<i>serieA</i>	IN	Variable o serie literal que sustituye las ocurrencias de <i>serieDesde</i> .
entero	<i>n</i>	IN	La posición del carácter en el que ha de comenzar la búsqueda.
serie	<i>opción</i>	IN	<p>Especifica si sustituir todas las ocurrencias o únicamente la primera ocurrencia y puede tener uno de los valores siguientes:</p> <p>A o a Sustituye todas las ocurrencias. El valor por omisión es A.</p> <p>F o f Sustituye sólo la primera ocurrencia.</p>
serie	<i>serieSalida</i>	OUT	Variable que contiene <i>serieEntrada</i> con ocurrencias de <i>serieDesde</i> sustituidas por <i>serieA</i> .

Códigos de Retorno

Tabla 95. Códigos de retorno de DTW_REPLACE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1:

```
@DTW_rREPLACE("ABCABCABC",  
"AB", "1234")
```

- Devuelve: "1234C1234C1234C"

DTW_REVERSE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Invierte una serie de modo que el último carácter sea el primero, el penúltimo sea el segundo, hasta que se haya invertido toda la serie.

Formato

@DTW_REVERSE(*serieEntrada*, *serieSalida*)

@DTW_rREVERSE(*serieEntrada*)

Parámetros

Tabla 96. Parámetros de DTW_REVERSE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal a invertir.
serie	<i>serieSalida</i>	OUT	Variable que contiene el formato invertido de <i>serieEntrada</i> .

Códigos de Retorno

Tabla 97. Códigos de retorno de DTW_REVERSE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1:

```
@DTW_REVERSE("This is it.",  
result)
```

- Devuelve: result = ".ti si sihT"

Ejemplo 2:

```
@DTW_rREVERSE(string1)
```

- La entrada: `string1 = "reversed"`
- Devuelve: `"desrever"`

DTW_STRIP

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Elimina los blancos iniciales, los blancos de cola o ambos de una serie.

Formato

@DTW_STRIP(*serieEntrada*, opción, *serieSalida*)

@DTW_STRIP(*serieEntrada*, *serieSalida*)

@DTW_rSTRIP(*serieEntrada*, opción)

@DTW_rSTRIP(*serieEntrada*)

Parámetros

Tabla 98. Parámetros de DTW_STRIP

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
serie	<i>opción</i>	IN	Especifica los blancos a eliminar de la <i>serieEntrada</i> . El valor por omisión es B. B o b - eliminar los blancos iniciales y los blancos de cola L o l - eliminar únicamente blancos iniciales T o t - eliminar únicamente blancos de cola
serie	<i>serieSalida</i>	OUT	Variable que contiene <i>serieEntrada</i> con los blancos eliminados tal y como ha especificado la opción.

Códigos de Retorno

Tabla 99. Códigos de retorno de DTW_STRIP

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Tabla 99. Códigos de retorno de DTW_STRIP (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_STRIP(" day ", result)
```

- Devuelve: result = "day"

Ejemplo 2:

```
@DTW_STRIP(" day ", "T",  
result)
```

- Devuelve: result = "day"

Ejemplo 3:

```
@DTW_rSTRIP(" a day ", "L")
```

- Devuelve: "a day "

DTW_SUBSTR

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve una subserie de una serie, con caracteres de relleno opcionales.

Formato

@DTW_SUBSTR(*serieEntrada1*, *n*, *longitud*, *relleno*, *serieSalida*)

@DTW_SUBSTR(*serieEntrada1*, *n*, *longitud*, *serieSalida*)

@DTW_SUBSTR(*serieEntrada1*, *n*, *serieSalida*)

@DTW_rSUBSTR(*serieEntrada1*, *n*, *longitud*, *relleno*)

@DTW_rSUBSTR(*serieEntrada1*, *n*, *longitud*)

@DTW_rSUBSTR(*serieEntrada1*, *n*)

Parámetros

Tabla 100. Parámetros de DTW_SUBSTR

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal que ha de buscarse.
entero	<i>n</i>	IN	La primera posición de carácter de la subserie. El valor por omisión es comenzar al principio de <i>serieEntrada</i> .
entero	<i>longitud</i>	IN	El número de caracteres de la subserie. El valor por omisión es el resto de la serie.
serie	<i>relleno</i>	IN	El carácter de relleno utilizado si <i>n</i> supera la longitud de <i>serieEntrada</i> o si <i>longitud</i> es mayor que <i>serieEntrada</i> . El valor por omisión es un blanco.
serie	<i>serieSalida</i>	OUT	Variable que contiene una subserie de <i>serieEntrada</i> .

Códigos de Retorno

Tabla 101. Códigos de retorno de DTW_SUBSTR

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.

Tabla 101. Códigos de retorno de DTW_SUBSTR (continuación)

Código de retorno	Explicación
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_SUBSTR("abc", "2",
result)
```

- Devuelve: result = "bc"

Ejemplo 2:

```
@DTW_SUBSTR("abc", "2", "4",
result)
```

- Devuelve: result = "bc"

Ejemplo 3:

```
@DTW_SUBSTR("abc", "2", "4",
".", result )
```

- Devuelve: result = "bc.."

Ejemplo 4:

```
@DTW_rSUBSTR("abc", "2", "6",
".")
```

- Devuelve: "bc...."

DTW_TRANSLATE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve una serie en la que cada uno de los caracteres se ha convertido en otro carácter o sin modificar.

Formato

@DTW_TRANSLATE(*serieEntrada*, *tablaO*, *tablaI*, *valor por omisión*, *serieSalida*)

@DTW_TRANSLATE(*serieEntrada*, *tablaO*, *tablaI*, *serieSalida*)

@DTW_TRANSLATE(*serieEntrada*, *tablaO*, *serieSalida*)

@DTW_TRANSLATE(*serieEntrada*, *serieSalida*)

@DTW_rTRANSLATE(*serieEntrada*, *tablaO*, *tablaI*, *valor por omisión*)

@DTW_rTRANSLATE(*serieEntrada*, *tablaO*, *tablaI*)

@DTW_rTRANSLATE(*serieEntrada*, *tablaO*)

@DTW_rTRANSLATE(*serieEntrada*)

Parámetros

Tabla 102. Parámetros de DTW_TRANSLATE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
serie	<i>tablaO</i>	IN	Variable o serie literal utilizada como tabla de conversión. Utilice nulo ("") para especificar <i>tablaI</i> o <i>valor por omisión</i> ; en caso contrario este parámetro es opcional.
serie	<i>tablaI</i>	IN	Variable o serie literal en que se busca <i>serieEntrada</i> . Utilice nulo ("") para especificar <i>valor por omisión</i> ; en caso contrario este parámetro es opcional.
serie	<i>valor por omisión</i>	IN	El carácter por omisión a utilizar. El valor por omisión es un blanco.
serie	<i>serieSalida</i>	OUT	Variable que contiene el resultado convertido de <i>serieEntrada</i> .

Códigos de Retorno

Tabla 103. Códigos de retorno de DTW_TRANSLATE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Notas de utilización

1. Si *tablaI*, *tablaO* y el carácter *valor por omisión* no están en la lista de parámetros, el parámetro *serieEntrada* se convierte a mayúsculas.
2. Si *tablaI* y *tablaO* están en la lista, se busca *tablaI* en cada uno de los caracteres de la serie de entrada y se convierte en *tablaO*. Si un carácter de *tablaI* no tiene un carácter correspondiente en *tablaO*, en su lugar se utiliza el carácter *valor por omisión*.

Ejemplos

Ejemplo 1:

```
@DTW_TRANSLATE("abbc",  
result)
```

- Devuelve: result = "ABBC"

Ejemplo 2:

```
@DTW_TRANSLATE("abbc", "R",  
"bc", result)
```

- Devuelve: result = "aRR "

Ejemplo 3:

```
@DTW_rTRANSLATE("abcdef",  
"12", "abcd", ".")
```

- Devuelve: "12..ef"

Ejemplo 4:

```
@DTW_rTRANSLATE("abbc", "",  
"", "")
```

- Devuelve: "abbc"

DTW_UPPERCASE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve una serie en mayúsculas.

Formato

@DTW_UPPERCASE(*serieEntrada*, *serieSalida*)

@DTW_rUPPERCASE(*serieEntrada*)

@DTW_mUPPERCASE(*serieMult1*, *serieMult2*, ..., *serieMultn*)

Parámetros

Tabla 104. Parámetros de DTW_UPPERCASE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal con caracteres en mayúsculas y minúsculas.
serie	<i>serieSalida</i>	OUT	Variable que contiene <i>serieEntrada</i> con todos los caracteres en mayúsculas.
serie	<i>serieMult</i>	INOUT	<ul style="list-style-type: none">En la entrada: Variable que contiene una serie.En la salida: Variable que contiene la serie de entrada convertida en mayúsculas.

Códigos de Retorno

Tabla 105. Códigos de retorno de DTW_UPPERCASE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1:

```
@DTW_UPPERCASE("Test",  
result)
```

- Devuelve: result = "TEST"

Ejemplo 2:

```
@DTW_rUPPERCASE(string1)
```

- La entrada: string1 = "Web pages"
- Devuelve: "WEB PAGES"

Ejemplo 3:

```
@DTW_mUPPERCASE(string1,  
string2, string3)
```

- La entrada: string1 = "This", string2 = "is", string3 = "uppercase"
- Devuelve: string1 = "THIS", string2 = "IS", string3 = "UPPERCASE"

Funciones de texto

Estas funciones suplementan las funciones de serie modificando palabras o conjuntos de palabras. Net.Data interpreta una palabra como serie de espacio delimitado, o una serie con espacios a ambos lados. He aquí algunos ejemplos:

Valor de serie	Número de palabras
una dos tres	3
una , dos , tres	5
Parte 2: Crecimiento de las ventas en Internet	5

Soporte de MBCS para OS/390, OS/2, Windows NT y UNIX: Puede especificar el soporte del conjunto de caracteres de múltiples bytes (MBCS) para funciones de texto y serie con el valor de configuración de DTW_MBMODE. Especifique este valor en el archivo de inicialización de Net.Data; el valor por omisión es que no haya soporte. Puede alterar temporalmente el valor en el archivo de inicialización definiendo la variable DTW_MBMODE en una macro Net.Data. Consulte la sección sobre la variable de configuración en el manual *Guía de administración y programación de Net.Data* y “DTW_MBMODE” en la página 135 para obtener más información.

Soporte de MBCS para OS/400: El soporte de DBCS se facilita automáticamente y no necesita esta variable.

Soporte de Unicode para OS/2, Windows NT y UNIX: Puede especificar soporte de Unicode UTF-8 para funciones de texto y serie con el valor de configuración de DTW_UNICODE. Especifique este valor en el archivo de inicialización de Net.Data; el valor por omisión es que no haya soporte. Consulte la sección sobre la variable de configuración en el manual *Guía de administración y programación de Net.Data* para obtener más información.

Las funciones siguientes son funciones de texto a las que da soporte Net.Data:

- “DTW_DELWORD” en la página 241
- “DTW_SUBWORD” en la página 243
- “DTW_WORD” en la página 245
- “DTW_WORDINDEX” en la página 247
- “DTW_WORDLENGTH” en la página 249
- “DTW_WORDPOS” en la página 251
- “DTW_WORDS” en la página 253

DTW_DELWORD

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Suprime texto en una serie, comenzando a partir de la palabra *n* en el número de palabras que se ha especificado por medio del valor *longitud*.

Formato

@DTW_DELWORD(*serieEntrada*, *n*, *longitud*, *serieSalida*)

@DTW_DELWORD(*serieEntrada*, *n*, *serieSalida*)

@DTW_rDELWORD(*serieEntrada*, *n*, *longitud*)

@DTW_rDELWORD(*serieEntrada*, *n*)

Parámetros

Tabla 106. Parámetros de DTW_DELWORD

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
entero	<i>n</i>	IN	La posición de texto de la primera palabra que ha de suprimirse.
entero	<i>longitud</i>	IN	El número de palabras a suprimir. El valor por omisión es suprimir todas las palabras desde <i>n</i> hasta el final de <i>serieEntrada</i> . Es un parámetro opcional.
serie	<i>serieSalida</i>	OUT	Variable que contiene el formato modificado de <i>serieEntrada</i> .

Códigos de Retorno

Tabla 107. Códigos de retorno de DTW_DELWORD

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.

Tabla 107. Códigos de retorno de DTW_DELWORD (continuación)

Código de retorno	Explicación
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1

```
@DTW_DELWORD("Now is the  
time", "5", result)
```

- Devuelve: result = "Now is the time"

Ejemplo 2:

```
@DTW_DELWORD("Now is the  
time", "2", result)
```

- Devuelve: result = "Now"

Ejemplo 3:

```
@DTW_DELWORD("Now is the  
time", "2", "2", result)
```

- Devuelve: result = "Now time"

Ejemplo 4:

```
@DTW_rDELWORD("Now is the  
time.", "3")
```

- Devuelve: "Now is"

DTW_SUBWORD

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve una subserie de una serie, comenzando en la palabra *n* para el número de palabras que se ha especificado por medio del valor *longitud*.

Formato

@DTW_SUBWORD(*serieEntrada*, *n*, *longitud*, *serieSalida*)

@DTW_SUBWORD(*serieEntrada*, *n*, *serieSalida*)

@DTW_rSUBWORD(*serieEntrada*, *n*, *longitud*)

@DTW_rSUBWORD(*serieEntrada*, *n*)

Parámetros

Tabla 108. Parámetros de DTW_SUBWORD

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
entero	<i>n</i>	IN	La posición de texto de la primera palabra de la subserie. Se devuelve un nulo si este valor es mayor que el número de palabras de <i>serieEntrada</i> .
entero	<i>longitud</i>	IN	El número de palabras de la subserie. Si este valor es mayor que el número de palabras desde <i>n</i> al final de la <i>serieEntrada</i> , se devuelven todas las palabras que hay hasta el final de <i>serieEntrada</i> . El valor por omisión es devolver todas las palabras desde <i>n</i> hasta el final de <i>serieEntrada</i> .
serie	<i>serieSalida</i>	OUT	Variable que contiene una subserie de <i>serieEntrada</i> especificada por medio del valor de <i>n</i> y <i>longitud</i> .

Códigos de Retorno

Tabla 109. Códigos de retorno de DTW_SUBWORD

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.

Tabla 109. Códigos de retorno de DTW_SUBWORD (continuación)

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_SUBWORD("Now is the
time", "5", result)
• Devuelve: result = ""
```

Ejemplo 2:

```
@DTW_SUBWORD("Now is the
time", "2", result)
• Devuelve: result = "is the   time"
```

Ejemplo 3:

```
@DTW_SUBWORD(Now is the
time", "2", "2", result)
• Devuelve: result = "is the"
```

Ejemplo 4:

```
@DTW_rSUBWORD("Now is the
time", "3")
• Devuelve: "the   time"
```

DTW_WORD

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve la palabra *n* de una serie.

Formato

@DTW_WORD(*serieEntrada*, *n*, *serieSalida*)

@DTW_rWORD(*serieEntrada*, *n*)

Parámetros

Tabla 110. Parámetros de DTW_WORD

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
entero	<i>n</i>	IN	La posición de texto de la palabra a devolver. Si este valor es mayor que el número de palabras de <i>serieEntrada</i> , se devuelve un nulo.
serie	<i>serieSalida</i>	OUT	Variable que contiene el texto en la posición de la palabra <i>n</i> .

Códigos de Retorno

Tabla 111. Códigos de retorno de DTW_WORD

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_WORD("Now is the time",  
"3", result)
```

- Devuelve: result = "the"

Ejemplo 2:

```
@DTW_WORD("Now is the time",  
"5", result)
```

- Devuelve: result = ""

Ejemplo 3:

```
@DTW_rWORD("Now is the  
time", "4")
```

- Devuelve: "time"

DTW_WORDINDEX

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve la posición de carácter del primer carácter de la palabra *n* de una serie.

Formato

@DTW_WORDINDEX(*serieEntrada*, *n*, *serieSalida*)

@DTW_rWORDINDEX(*serieEntrada*, *n*)

Parámetros

Tabla 112. Parámetros de DTW_WORDINDEX

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
entero	<i>n</i>	IN	La posición de texto de la palabra a indexar. Si este valor es mayor que el número de palabras de la serie de entrada, se devuelve un 0.
serie	<i>serieSalida</i>	OUT	Variable que contiene la posición de carácter de la palabra <i>n</i> de <i>serieEntrada</i> .

Códigos de Retorno

Tabla 113. Códigos de retorno de DTW_WORDINDEX

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_WORDINDEX("Now is the  
time", "3", result)
```

- Devuelve: result = "8"

Ejemplo 2:

```
@DTW_WORDINDEX("Now is the  
time", "6", result)
```

- Devuelve: result = "0"

Ejemplo 3:

```
@DTW_rWORDINDEX("Now is the  
time", "2")
```

- Devuelve: "5"

DTW_WORDLENGTH

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve la longitud de la palabra *n* de una serie.

Formato

@DTW_WORDLENGTH(*serieEntrada*, *n*, *serieSalida*)

@DTW_rWORDLENGTH(*serieEntrada*, *n*)

Parámetros

Tabla 114. Parámetros de DTW_WORDLENGTH

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
entero	<i>n</i>	IN	La posición de texto de la palabra cuya longitud se desea conocer. Si este valor es mayor que el número de palabras de la serie de entrada, se devuelve un 0.
serie	<i>serieSalida</i>	OUT	Variable que contiene la longitud de la palabra <i>n</i> de <i>serieEntrada</i> .

Códigos de Retorno

Tabla 115. Códigos de retorno de DTW_WORDLENGTH

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Ejemplos

Ejemplo 1:

```
@DTW_WORDLENGTH("Now is the  
time", "1", result)
```

- Devuelve: result= " 3"

Ejemplo 2:

```
@DTW_rWORDLENGTH("Now is the  
time", "6")
```

- Devuelve: "0"

DTW_WORDPOS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve el número de palabra de la primera ocurrencia de una serie dentro de otra.

Formato

@DTW_WORDPOS(*serieEntrada1*, *serieEntrada2*, *n*, *serieSalida*)

@DTW_WORDPOS(*serieEntrada1*, *serieEntrada2*, *serieSalida*)

@DTW_rWORDPOS(*serieEntrada1*, *serieEntrada2*, *n*)

@DTW_rWORDPOS(*serieEntrada1*, *serieEntrada2*)

Parámetros

Tabla 116. Parámetros de DTW_WORDPOS

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada1</i>	IN	Variable o serie literal.
serie	<i>serieEntrada2</i>	IN	Variable o serie literal a buscar.
entero	<i>n</i>	IN	La posición de texto de <i>serieEntrada2</i> en que ha de comenzarse la búsqueda. Si este valor es mayor que el número de palabras de <i>serieEntrada2</i> , se devuelve un 0. El valor por omisión es buscar desde el principio de <i>serieEntrada2</i> .
serie	<i>serieSalida</i>	OUT	La posición de texto de <i>serieEntrada1</i> en <i>serieEntrada2</i> .

Códigos de Retorno

Tabla 117. Códigos de retorno de DTW_WORDPOS

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.

Tabla 117. Códigos de retorno de DTW_WORDPOS (continuación)

Código de retorno	Explicación
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Notas de utilización

Para la comparación, los blancos múltiples se tratan como blancos simples.

Ejemplos

Ejemplo 1:

```
@DTW_WORDPOS("the", "Now is
the time", result)
```

- Devuelve: result= " 3"

Ejemplo 2:

```
@DTW_WORDPOS("The", "Now is
the time", result)
```

- Devuelve: result = "0"

Ejemplo 3:

```
@DTW_WORDPOS("The", "Now is
the time", "5", result)
```

- Devuelve: result = "0"

Ejemplo 4:

```
@DTW_WORDPOS("is the", "Now
is the time", result)
```

- Devuelve: result = " 2"

Ejemplo 5:

```
@DTW_rWORDPOS("be", "To be or
not to be", "3")
```

- Devuelve: "6"

DTW_WORDS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve el número de palabras de una serie.

Formato

@DTW_WORDS(*serieEntrada*, *serieSalida*)

@DTW_rWORDS(*serieEntrada*)

Parámetros

Tabla 118. Parámetros de DTW_WORDS

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>serieEntrada</i>	IN	Variable o serie literal.
serie	<i>serieSalida</i>	OUT	Variable que contiene el número de palabras de <i>serieEntrada</i> .

Códigos de Retorno

Tabla 119. Códigos de retorno de DTW_WORDS

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1:

```
@DTW_WORDS("Now is the  
time", result)
```

- Devuelve:
result = "4"

Ejemplo 2:

```
@DTW_rWORDS(" ")
```

- Devuelve: "0"

Funciones de tabla

Estas funciones simplifican el trabajo con tablas de Net.Data y son más eficaces que escribir sus propias funciones utilizando REXX, C o Perl.

- “DTW_TB_APPENDROW” en la página 256
- “DTW_TB_COLS” en la página 258
- “DTW_TB_DELETEROW” en la página 262
- “DTW_TB_DELETECOL” en la página 260
- “DTW_TB_DLIST” en la página 264
- “DTW_TB_DUMPH” en la página 267
- “DTW_TB_DUMPV” en la página 269
- “DTW_TB_GETN” en la página 271
- “DTW_TB_GETV” en la página 273
- “DTW_TB_HTMLENCODER” en la página 275
- “DTW_TB_INPUT_CHECKBOX” en la página 277
- “DTW_TB_INPUT_RADIO” en la página 279
- “DTW_TB_INPUT_TEXT” en la página 281
- “DTW_TB_INSERTCOL” en la página 284
- “DTW_TB_INSERTROW” en la página 286
- “DTW_TB_LIST” en la página 288
- “DTW_TB_QUERYCOLNONJ” en la página 291
- “DTW_TB_ROWS” en la página 293
- “DTW_TB_SELECT” en la página 295
- “DTW_TB_SETCOLS” en la página 298
- “DTW_TB_SETN” en la página 300
- “DTW_TB_SETV” en la página 302
- “DTW_TB_TABLE” en la página 304
- “DTW_TB_TEXTAREA” en la página 307

DTW_TB_APPENDROW

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X		X	X	X	X	X	X

Propósito

Añade una o más filas al final de una tabla de Net.Data.

Formato

@DTW_TB_APPENDROW(tabla, filas)

Parámetros

Tabla 120. Parámetros de DTW_TB_APPENDROW

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	INOUT	La variable de tabla de la macro a la que se agregan filas.
entero	<i>filas</i>	IN	El número de filas a agregar a <i>tabla</i> .

Códigos de Retorno

Tabla 121. Códigos de retorno de DTW_TB_APPENDROW

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.
1010	Se han grabado datos en la tabla hasta que se ha llenado y se han desechado los datos restantes.

Notas de utilización

1. El número de columnas de la tabla debe definirse antes de llamar a DTW_TB_APPENDROW(). Puede establecer el número de columnas con las funciones DTW_TB_SETCOLS() o DTW_TB_INSERTCOL(), o pasando la tabla a un entorno de lenguaje a establecer.
2. Puede asignar valores a las filas nuevas con la función DTW_TB_SETV() una vez se agreguen filas a la tabla, o pasar la tabla a un entorno de lenguaje para el proceso.
3. Si hay un límite en el número total de filas que se admite en la tabla y el número de filas a agregar puede ocasionar que se supere el límite, se devuelve un error a quién emite la llamada.

Ejemplos

Ejemplo 1: Agrega diez filas a la tabla

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_APPENDROW(myTable, "10")
```

DTW_TB_COLS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve el número de columnas de una tabla de Net.Data.

Formato

@DTW_TB_COLS(tabla, cols)

@DTW_TB_rCOLS(tabla)

Parámetros

Tabla 122. Parámetros de DTW_TB_COLS

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla de la macro para la que se devuelven columnas.
entero	<i>cols</i>	OUT	Variable que contiene el número de columnas de <i>tabla</i> .

Códigos de Retorno

Tabla 123. Códigos de retorno de DTW_TB_COLS

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1: Recupera el número de columnas y asigna el valor a *cols*


```
%DEFINE myTable = %TABLE
%DEFINE cols = ""
...
@FillTable(myTable)
...
@DTW_TB_COLS(myTable, cols)
```

Ejemplo 2: Recupera y visualiza el valor para el número actual de columnas de la tabla

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<p>My table contains @DTW_TB_rCOLS(myTable) columns.</p>
```

DTW_TB_DELETECOL

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X		X	X	X	X	X	X

Propósito

Suprime una o más columnas de una tabla de Net.Data.

Formato

@DTW_TB_DELETECOL(tabla, col_después, cols)

Parámetros

Tabla 124. Parámetros de DTW_TB_DELETECOL

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	INOUT	La variable de tabla de la macro de la que han de suprimirse las columnas.
entero	<i>col_después</i>	IN	El número de la columna a partir de la cual van a suprimirse las columnas posteriores. Para suprimir la primera columna, especifique 0.
entero	<i>cols</i>	IN	El número de columnas a suprimir de <i>tabla</i> .

Códigos de Retorno

Tabla 125. Códigos de retorno de DTW_TB_DELETECOL

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Suprime la tercera y cuarta columnas de la tabla

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_DELETECOL(myTable, "3", "2")
```

Ejemplo 2: Suprime la primera columna de la tabla

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_DELETECOL(myTable, "0", "1")
```

DTW_TB_DELETEROW

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X		X	X	X	X	X	X

Propósito

Suprime una o más filas de una tabla de Net.Data.

Formato

@DTW_TB_DELETEROW(tabla, fila_inicio, filas)

Parámetros

Tabla 126. Parámetros de DTW_TB_DELETEROW

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	INOUT	La variable de tabla de la macro de la que han de suprimirse las filas.
entero	<i>fila_inicio</i>	IN	El número de fila de la primera fila de <i>tabla</i> que ha de suprimirse.
entero	<i>filas</i>	IN	El número de columnas a suprimir de <i>tabla</i> .

Códigos de Retorno

Tabla 127. Códigos de retorno de DTW_TB_DELETEROW

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Notas de utilización

El número de columnas de la tabla debe definirse antes de llamar a DTW_TB_DELETE_ROW(). Puede establecer el número de columnas con las funciones DTW_TB_SETCOLS() o DTW_TB_INSERTCOL(), o pasando la tabla a un entorno de lenguaje a establecer.

Ejemplos

Ejemplo 1: Suprime cinco filas a partir de la fila 10 de una tabla

```
%DEFINE myTable = %TABLE  
  
@DTW_TB_DELETE_ROW(myTable, "10", "5")
```

Ejemplo 2: Suprime todas las filas de una tabla

```
%DEFINE myTable = %TABLE  
  
@DTW_TB_DELETE_ROW(myTable, "1", @DTW_TB_nROWS(myTable))
```

DTW_TB_DLIST

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Genera una lista de definición de HTML a partir de una tabla de Net.Data.

Formato

@DTW_TB_DLIST(tabla, término, def, estilotérmino, estilodef, enlace, u_enlace, imagen, u_imagen)

@DTW_TB_DLIST(tabla, término, def, estilotérmino, estilodef, enlace, u_enlace, imagen)

@DTW_TB_DLIST(tabla, término, def, estilotérmino, estilodef, enlace, u_enlace)

@DTW_TB_DLIST(tabla, término, def, estilotérmino, estilodef, enlace)

@DTW_TB_DLIST(tabla, término, def, estilotérmino, estilodef)

@DTW_TB_DLIST(tabla, término, def, estilotérmino)

@DTW_TB_DLIST(tabla, término, def)

@DTW_TB_DLIST(tabla, término)

@DTW_TB_DLIST(tabla)

Parámetros

Tabla 128. Parámetros de DTW_TB_DLIST

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	Símbolo que especifica la variable de tabla de la macro que ha de visualizarse como lista de definición de HTML.
entero	<i>término</i>	IN	El número de columna de <i>tabla</i> que contiene valores de nombre de <i>término</i> (el texto que va detrás del código <dt>). El valor por omisión es utilizar la primera columna.
entero	<i>def</i>	IN	El número de columna de <i>tabla</i> que contiene valores de definición de término (el texto que va detrás del código <dd>). El valor por omisión es utilizar la segunda columna.

Tabla 128. Parámetros de DTW_TB_DLIST (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>estilotérmino</i>	IN	Variable o serie literal que contiene una lista de elementos de HTML para los valores de nombre de <i>término</i> . El valor por omisión es no utilizar códigos de estilo.
serie	<i>estilodef</i>	IN	Variable o serie literal que contiene una lista de elementos de HTML para los valores de definición de <i>término</i> . El valor por omisión es no utilizar códigos de estilo.
serie	<i>enlace</i>	IN	Especifica los elementos de HTML para los que se genera un enlace HTML. Los valores válidos son DT y DD. El valor por omisión es no generar enlaces de HTML.
entero	<i>u_enlace</i>	IN	El número de columna de <i>tabla</i> que contiene los URL para las referencias de HTML. El valor por omisión es no generar enlaces de HTML.
serie	<i>imagen</i>	IN	Especifica los elementos de HTML para los que se genera una imagen incorporada. Los valores válidos son DT y DD. El valor por omisión es no generar imágenes incorporadas (DT).
entero	<i>u_imagen</i>	IN	El número de columna de <i>tabla</i> que contiene los URL para las imágenes incorporadas. El valor por omisión es no generar imágenes incorporadas.

Códigos de Retorno

Tabla 129. Códigos de retorno de DTW_TB_DLIST

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.

Tabla 129. Códigos de retorno de DTW_TB_DLIST (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Crea una lista de definición que produce el HTML que se muestra a continuación, en función de los datos de tabla

```
@DTW_TB_DLIST(Mytable,"3","4","b
i","strong","DD","2","DT","1")
```

Resultado:

```
<dl>
<dt>
<b><i>image1text</i></b></dt>
<dd>
<a
href="http://www.mycompany.com/link1.html"><strong>link1text</strong></a></dd>
<dt>
<b><i>image2text</i></b></dt>
<dd>
<a
href="http://www.mycompany.com/link2.html"><strong>link2text</strong></a></dd>
<dt>
<b><i>image3text</i></b></dt>
<dd>
<a
href="http://www.mycompany.com/link3.html"><strong>link3text</strong></a></dd>
<dt>
<b><i>image4text</i></b></dt>
<dd>
<a
href="http://www.mycompany.com/link4.html"><strong>link4text</strong></a></dd>
</dl>
```


DTW_TB_DUMPH

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Imprime el contenido de una tabla de Net.Data que utiliza el código <pre> de HTML, en el que cada fila de la tabla se visualiza en una línea.

Formato

@DTW_TB_DUMPH(tabla)

Parámetros

Tabla 130. Parámetros de DTW_TB_DUMPH

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	Símbolo que especifica la variable de tabla de la macro que ha de visualizarse.

Códigos de Retorno

Tabla 131. Códigos de retorno de DTW_DB_DUMPH

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.

Notas de utilización

Si la tabla de Net.Data está vacía, se devuelve un error.

Ejemplos

Ejemplo 1:

@DTW_TB_DUMPH(Mytable)

El HTML generado por medio de este ejemplo se parece al siguiente:

```

Name      Department      Position
Jack Smith Internet Technologies Software Engineer
Helen Williams Database      Development Manager
Alex Jones Manufacturing Industrial Engineer
Tom Baker Procurement    Sales Rep

```

DTW_TB_DUMPV

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Imprime el contenido de la tabla de Net.Data que utiliza el código <pre> de HTML, en el que cada campo de la tabla está en una línea.

Formato

@DTW_TB_DUMPV(tabla)

Parámetros

Tabla 132. Parámetros de DTW_TB_DUMPV

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	Símbolo que especifica la variable de tabla de la macro que ha de visualizarse.

Códigos de Retorno

Tabla 133. Códigos de retorno de DTW_TB_DUMPV

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.

Notas de utilización

Si la tabla de Net.Data está vacía, se devuelve un error

Ejemplos

Ejemplo 1:

@DTW_TB_DUMPV(Mytable)

El HTML generado para este ejemplo se parece al siguiente:

```
<pre>
http://www.mycompany.com/images/image1.gif
http://www.mycompany.com/link1.html
image1text
link1text
http://www.mycompany.com/images/image2.gif
http://www.mycompany.com/link2.html
image2text
link2text
http://www.mycompany.com/images/image3.gif
http://www.mycompany.com/link3.html
image3text
link3text
http://www.mycompany.com/images/image4.gif
http://www.mycompany.com/link4.html
image4text
link4text
</pre>
```

DTW_TB_GETN

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve una cabecera de columna a partir de una tabla de Net.Data.

Formato

@DTW_TB_GETN(tabla, col, nombre)

@DTW_TB_rGETN(tabla, col)

Parámetros

Tabla 134. Parámetros de DTW_TB_GETN

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla de la macro desde la que se devuelve un nombre de columna.
entero	<i>col</i>	IN	El número de columna de la columna cuyo nombre ha de devolverse.
serie	<i>nombre</i>	OUT	Variable que contiene el nombre de la columna especificada en <i>col</i> .

Códigos de Retorno

Tabla 135. Códigos de retorno de DTW_TB_GETN

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Tabla 135. Códigos de retorno de DTW_TB_GETN (continuación)

Código de retorno	Explicación
1008	Un parámetro está fuera de los límites de la tabla.

Notas de utilización

Antes de llamar a DTW_TB_GETN(), defina el número de columnas en la tabla. Puede establecer el número de columnas con las funciones DTW_TB_SETCOLS() o DTW_TB_INSERTCOL(), o pasando la tabla a un entorno de lenguaje a establecer.

Ejemplos

Ejemplo 1: Recupera el nombre de columna de la columna 4

```
%DEFINE myTable = %TABLE
%DEFINE name = ""
...
@FillTable(myTable)
...
@DTW_TB_GETN(myTable, "4", name)
```

Ejemplo 2: Recupera el nombre de columna de la última columna de la tabla

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<p>The column name of the last column is @DTW_TB_rGETN(myTable,
@DTW_TB_rCOLS(myTable))</p>
```

DTW_TB_GETV

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve el valor en una columna y fila determinada de una tabla de Net.Data.

Formato

@DTW_TB_GETV(tabla, fila, col, valor)

@DTW_TB_rGETV(tabla, fila, col)

Parámetros

Tabla 136. Parámetros de DTW_TB_GETV

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla de la macro para la que se devuelve un valor de tabla.
entero	<i>fila</i>	IN	El número de fila del valor que ha de devolverse.
entero	<i>col</i>	IN	El número de columna del valor que ha de devolverse.
serie	<i>valor</i>	OUT	Variable que contiene el valor de la fila y columna especificados en <i>fila</i> y <i>col</i> .

Códigos de Retorno

Tabla 137. Códigos de retorno de DTW_TB_GETV

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.

Tabla 137. Códigos de retorno de DTW_TB_GETV (continuación)

Código de retorno	Explicación
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Notas de utilización

Antes de llamar a DTW_TB_GETV(), defina el número de columnas de la tabla. Puede establecer el número de columnas con las funciones DTW_TB_SETCOLS() o DTW_TB_INSERTCOL(), o pasando la tabla a un entorno de lenguaje a establecer.

Ejemplos

Ejemplo 1: Recupera el valor de tabla en la fila 6, columna 3

```
%DEFINE myTable = %TABLE
%DEFINE value = ""
...
@FillTable(myTable)
...
@DTW_TB_GETV(myTable, "6", "3", value)
```

Ejemplo 2: Recupera el valor de tabla en la fila 1, columna 1

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<p>The table value of row 1, column 1 is @DTW_TB_rGETV(myTable,
"1", "1").</p>
```


DTW_TB_HTMLENCODE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Sustituye determinados caracteres en los datos ubicados en una tabla de Net.Data por sus códigos de escape de carácter de HTML correspondientes.

Formato

@DTW_TB_HTMLENCODE(tabla, collist)

@DTW_TB_HTMLENCODE(tabla)

Parámetros

Tabla 138. Parámetros de DTW_TB_HTMLENCODE

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	INOUT	La variable de tabla de la macro que ha de modificarse.
serie	<i>collist</i>	IN	Los números de columna de <i>tabla</i> a codificar. El valor por omisión es codificar todas las columnas.

Códigos de Retorno

Tabla 139. Códigos de retorno de DTW_TB_HTMLENCODE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Notas de utilización

Los caracteres que se sustituyen se indican en la siguiente tabla.

Nombre	Carácter	Código
Ampersand	&	&
Comillas dobles	"	"
Mayor que	>	>
Menor que	<	<

Ejemplos

Ejemplo 1:

```
@DTW_TB_HTMLENCODE(Mytable, "3 4")
```

Los caracteres especiales de las columnas 3 y 4 de la tabla especificada se sustituyen por sus formatos codificados.

DTW_TB_INPUT_CHECKBOX

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Genera uno o más códigos de entrada de recuadro de selección de HTML a partir de una tabla de Net.Data.

Formato

@DTW_TB_INPUT_CHECKBOX(tabla, indicadormandatos, nombcol, valorcol, filas, filascomprobadas)

@DTW_TB_INPUT_CHECKBOX(tabla, indicadormandatos, nombcol, valorcol, filas)

@DTW_TB_INPUT_CHECKBOX(tabla, indicadormandatos, nombcol, valorcol)

@DTW_TB_INPUT_CHECKBOX(tabla, indicadormandatos, nombcol)

Parámetros

Tabla 140. Parámetros de DTW_TB_INPUT_CHECKBOX

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla de la macro que ha de visualizarse como códigos de entrada de recuadro de selección.
serie	<i>indicadormandatos</i>	IN	El número de columna de <i>tabla</i> o serie que contiene el texto a visualizar junto al recuadro de selección. Este parámetro es obligatorio pero puede tener un valor de nulo (""). Cuando <i>indicadormandatos</i> es nulo, el valor utilizado es el valor que se define para <i>nombcol</i> .
serie	<i>nombcol</i>	IN	El número de columna de <i>tabla</i> o serie que contiene los nombres de campo de entrada.
entero	<i>valorcol</i>	IN	El número de columna de <i>tabla</i> que contiene los valores de campo de entrada. El valor por omisión es 1.
entero	<i>filas</i>	IN	La lista de filas de <i>tabla</i> desde la que generar los campos de entrada. El valor por omisión es utilizar todas las filas.

Tabla 140. Parámetros de DTW_TB_INPUT_CHECKBOX (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
entero	<i>filascomprobadas</i>	IN	La lista de filas que especifica las <i>filas</i> de <i>tabla</i> que han de comprobarse. El valor por omisión es no comprobar los campos.

Códigos de Retorno

Tabla 141. Códigos de retorno de DTW_TB_INPUT_CHECKBOX

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Genera HTML para tres códigos de entrada de recuadro de selección

```
@DTW_TB_INPUT_CHECKBOX(Mytable,"3","4","", "2 3 4","1 3 4")
```

Resultado:

```
<input type="checkbox" name="link2text"
value="1" />image2text<br />
<input type="checkbox" name="link3text" value="1" checked
/>image3text<br />
<input type="checkbox" name="link4text" value="1" checked
/>image4text<br />
```

DTW_TB_INPUT_RADIO

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Genera códigos de entrada de botón de selección a partir de una tabla de Net.Data.

Formato

@DTW_TB_INPUT_RADIO(*tabla*, *indicadormandatos*, *nombcol*, *valorcol*, *filas*, *filascomprobadas*)

@DTW_TB_INPUT_RADIO(*tabla*, *indicadormandatos*, *nombcol*, *valorcol*, *filas*)

@DTW_TB_INPUT_RADIO(*tabla*, *indicadormandatos*, *nombcol*, *valorcol*)

Parámetros

Tabla 142. Parámetros de DTW_TB_INPUT_RADIO

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla de la macro que ha de visualizarse como códigos de entrada de botón de selección.
serie	<i>indicadormandatos</i>	IN	El número de columna de <i>tabla</i> o una serie que contiene el texto a visualizar junto al botón de selección. Este parámetro es obligatorio pero puede contener un valor de nulo (""). Cuando <i>indicadormandatos</i> es un nulo, utiliza un valor de <i>valorcol</i> .
serie	<i>nombcol</i>	IN	El número de columna de <i>tabla</i> o serie que contiene los nombres de campo de entrada.
entero	<i>valorcol</i>	IN	El número de columna de <i>tabla</i> que contiene los valores de campo de entrada.
serie	<i>filas</i>	IN	La lista de filas de <i>tabla</i> desde la que generar los campos de entrada. El valor por omisión es utilizar todas las filas.
entero	<i>filascomprobadas</i>	IN	Número de fila de <i>tabla</i> para visualizar el botón de selección correspondiente cuando se compruebe. Sólo se admite un valor.

Códigos de Retorno

Tabla 143. Códigos de retorno de DTW_TB_INPUT_RADIO

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Genera HTML para tres códigos de entrada de botón de selección

```
@DTW_TB_INPUT_RADIO(Mytable,"3","Radio4","4","2 3 4","4")
```

Resultado:

```
<input type="radio" name="radio4"
value="link2text" />image2text<br />
<input type="radio" name="radio4" value="link3text"
/>image3text<br />
<input type="radio" name="radio4" value="link4text" checked
/>image4text<br />
```

DTW_TB_INPUT_TEXT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Genera códigos de HTML `<input />` para las filas que se especifiquen en una tabla de Net.Data.

Formato

@DTW_TB_INPUT_TEXT(*tabla*, *indicadormandatos*, *nombcol*, *valorcol*, *tamaño*, *longmáx*, *filas*)

@DTW_TB_INPUT_TEXT(*tabla*, *indicadormandatos*, *nombcol*, *valorcol*, *tamaño*, *longmáx*)

@DTW_TB_INPUT_TEXT(*tabla*, *indicadormandatos*, *nombcol*, *valorcol*, *tamaño*)

@DTW_TB_INPUT_TEXT(*tabla*, *indicadormandatos*, *nombcol*, *valorcol*)

@DTW_TB_INPUT_TEXT(*tabla*, *indicadormandatos*, *nombcol*)

Parámetros

Tabla 144. Parámetros de DTW_TB_INPUT_TEXT

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla de la macro que ha de visualizarse como códigos de entrada de texto.
serie	<i>indicadormandatos</i>	IN	El número de columna de <i>tabla</i> o serie que contiene el texto a visualizar junto al campo de entrada. Si <i>indicadormandatos</i> es un nulo, no se visualiza ningún texto.
serie	<i>nombcol</i>	IN	El número de columna de <i>tabla</i> que contiene los nombres de campo de entrada.
entero	<i>valorcol</i>	IN	El número de columna de <i>tabla</i> que contiene los valores de campo de entrada por omisión, que se especifica para el atributo VALUE en el código de INPUT. El valor por omisión es no generar el valor de atributo de VALUE.

Tabla 144. Parámetros de DTW_TB_INPUT_TEXT (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
entero	<i>tamaño</i>	IN	El número de caracteres del campo de entrada, que se especifica para el atributo SIZE en el código INPUT. El valor por omisión es la longitud del valor de entrada por omisión más largo, o 10 si no hay una entrada por omisión.
entero	<i>longmáx</i>	IN	La longitud máxima de una serie de entrada, que se especifica para el atributo MAXLENGTH del código INPUT. El valor por omisión es no generar el valor de atributo de MAXLENGTH.
entero	<i>filas</i>	IN	La lista de filas de <i>tabla</i> desde la que generar los campos de entrada. El valor por omisión es utilizar todas las filas.

Códigos de Retorno

Tabla 145. Códigos de retorno de DTW_TB_INPUT_TEXT

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Devuelve tres códigos de HTML <input />

```
@DTW_TB_INPUT_TEXT(Mytable,"3","3","4","35","40","1 2 3")
```

Resultado:

```
<p>image1text  
<input type="text" name="image1text" value="link1text" size="35"  
maxlength="40" /></p>  
<p>image2text  
<input type="text" name="image2text" value="link2text" size="35"  
maxlength="40" /></p>  
<p>image3text  
<input type="text" name="image3text" value="link3text" size="35"  
maxlength="40" /></p>
```

DTW_TB_INSERTCOL

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X		X	X	X		X	X

Propósito

Inserta una o más columnas en una tabla de Net.Data.

Formato

@DTW_TB_INSERTCOL(tabla, col_después, cols)

Parámetros

Tabla 146. Parámetros de DTW_TB_INSERTCOL

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	INOUT	La variable de tabla de la macro en la que van a insertarse las columnas.
entero	<i>col_después</i>	IN	El número de columna de la columna a partir de la cual van a insertarse las nuevas columnas. Para insertar columnas al principio de la tabla, especifique 0.
entero	<i>cols</i>	IN	El número de columnas a insertar en la <i>tabla</i> .

Códigos de Retorno

Tabla 147. Códigos de retorno de DTW_TB_INSERTCOL

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Inserta cinco columnas al final de una tabla

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_INSERTCOL(myTable, @DTW_TB_rCOLS(myTable), "5")
```

Ejemplo 2: Inserta una columna al principio de una tabla

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_INSERTCOL(myTable, "0", "1")
```

DTW_TB_INSERTROW

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X		X	X	X		X	X

Propósito

Inserta una o más filas en una tabla de Net.Data.

Formato

@DTW_TB_INSERTROW(tabla, fila_después, filas)

Parámetros

Tabla 148. Parámetros de DTW_TB_INSERTROW

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	INOUT	La variable de tabla de la macro en la que van a insertarse las filas.
entero	<i>fila_después</i>	IN	El número de fila a partir de la cual van a insertarse nuevas filas. Para insertar filas al principio de la tabla, especifique 0.
entero	<i>filas</i>	IN	El número de filas a insertar en la <i>tabla</i> .

Códigos de Retorno

Tabla 149. Códigos de retorno de DTW_TB_INSERTROW

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Notas de utilización

Antes de llamar a DTW_TB_INSERTROW(), defina el número de columnas en la tabla. Puede establecer el número de columnas con las funciones DTW_TB_SETCOLS() o DTW_TB_INSERTCOL(), o pasando la tabla a un entorno de lenguaje a establecer.

Ejemplos

Ejemplo 1: Inserta una fila detrás de la fila cinco de una tabla

```
%DEFINE myTable = %TABLE  
  
@DTW_TB_INSERTROW(myTable, "5", "1")
```

Ejemplo 2: Inserta tres filas al principio de una tabla

```
%DEFINE myTable = %TABLE  
  
@DTW_TB_INSERTROW(myTable, "0", "3")
```

DTW_TB_LIST

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Genera una lista de HTML a partir de una tabla de Net.Data.

Formato

@DTW_TB_LIST(tabla, tipolista, elementolista, estiloelemento, u_enlace, u_imagen)

@DTW_TB_LIST(tabla, tipolista, elementolista, estiloelemento, u_enlace)

@DTW_TB_LIST(tabla, tipolista, elementolista, estiloelemento)

@DTW_TB_LIST(tabla, tipolista, elementolista)

@DTW_TB_LIST(tabla, tipolista)

Parámetros

Tabla 150. Parámetros de DTW_TB_LIST

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	Símbolo que especifica la variable de tabla de la macro que ha de visualizarse como lista de HTML.
serie	<i>tipolista</i>	IN	El tipo de lista a generar. Los valores aceptables incluyen: DIR MENU OL UL
entero	<i>elementolista</i>	IN	El número de columna de <i>tabla</i> que contiene los valores de lista (el texto que ha de ir detrás del código). El valor por omisión es utilizar la primera columna.
serie	<i>estiloelemento</i>	IN	Variable o serie literal que contiene una lista de elementos de HTML para los valores de nombre de término. El valor por omisión es no utilizar códigos de estilo.
entero	<i>u_enlace</i>	IN	El número de columna de <i>tabla</i> que contiene los URL para los enlaces de HTML. Si no se especifica este valor, no se genera ningún enlace de HTML.

Tabla 150. Parámetros de DTW_TB_LIST (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
entero	<i>u_imagen</i>	IN	El número de columna de <i>tabla</i> que contiene los URL para las imágenes incorporadas. Si no se especifica este valor, no se genera ninguna imagen incorporada.

Códigos de Retorno

Tabla 151. Códigos de retorno de DTW_TB_LIST

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Genera códigos de HTML para una lista ordenada

@DTW_TB_LIST(Mytable,"0L","4","TT U","2","1")

Resultado:

```
<tt><u>
<ol>
<li><a href="http://www.mycompany.com/link1.html">
link1text</a></li>
<li><a href="http://www.mycompany.com/link2.html">
link2text</a></li>
<li><a href="http://www.mycompany.com/link3.html">
<IMG SRC="http://www.mycompany.com/images/image3.gif"
```

```
ALT="">link3text</a></li>
<li><a href="http://www.mycompany.com/link4.html">
link4txt</a></li>
</ol>
</u></tt>
```


DTW_TB_QUERYCOLNONJ

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X		X	X	X		X	X

Propósito

Devuelve el número de columna asociado con una cabecera de columna de una tabla de Net.Data.

Formato

@DTW_TB_QUERYCOLNONJ(tabla, nombre, col)

@DTW_TB_rQUERYCOLNONJ(tabla, nombre)

Parámetros

Tabla 152. Parámetros de DTW_TB_QUERYCOLNONJ

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla de la macro desde la que se devuelve un número de columna.
serie	<i>nombre</i>	IN	El nombre de la cabecera de columna para el que se devuelve el número de columna. Si la cabecera de columna no existe en la tabla, se devuelve un 0.
entero	<i>col</i>	OUT	Variable que contiene el número de columna de la columna cuyo nombre se especifica en <i>nombre</i> .

Códigos de Retorno

Tabla 153. Códigos de retorno de DTW_TB_QUERYCOLNONJ

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.

Tabla 153. Códigos de retorno de DTW_TB_QUERYCOLNONJ (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Notas de utilización

1. Antes de llamar a DTW_TB_QUERYCOLNONJ(), defina el número de columnas de la tabla. Puede establecer el número de columnas con las funciones DTW_TB_SETCOLS() o DTW_TB_INSERTCOL(), o pasando la tabla a un entorno de lenguaje a establecer.
2. Si la cabecera de columna no existe en la tabla, se devuelve un 0.

Ejemplos

Ejemplo 1: Recupera el número de columna para la columna cuyo nombre sea SERIAL_NUMBER

```
%DEFINE myTable = %TABLE
%DEFINE col = ""
```

```
@DTW_TB_QUERYCOLNONJ(myTable, "SERIAL_NUMBER", col)
```

Ejemplo 2: Recupera el número de columna para la columna cuyo nombre sea SERIAL_NUMBER

```
%DEFINE myTable = %TABLE
<p>The "SERIAL_NUMBER" column is column number
@DTW_TB_rQUERYCOLNONJ(myTable, "SERIAL_NUMBER")</p>
```

DTW_TB_ROWS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Devuelve el número de filas de una tabla de Net.Data.

Formato

@DTW_TB_ROWS(tabla, filas)

@DTW_TB_rROWS(tabla)

Parámetros

Tabla 154. Parámetros de DTW_TB_ROWS

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla de la macro para la que se devuelve el número de filas actual.
entero	<i>filas</i>	OUT	Variable que contiene el número actual de filas de <i>tabla</i> .

Códigos de Retorno

Tabla 155. Códigos de retorno de DTW_TB_ROWS

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Ejemplos

Ejemplo 1: Recupera el número actual de filas de la tabla y asigna el valor a *filas*

```
%DEFINE myTable = %TABLE
%DEFINE rows = ""

...
@FillTable(myTable)
...
@DTW_TB_ROWS(myTable, rows)
```

DTW_TB_SELECT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Genera una selección de HTML a partir de una tabla de Net.Data.

Formato

@DTW_TB_SELECT(tabla, nombre, opcióncol, tamaño, múltiple, filas, filasseleccionadas, valorcol)

@DTW_TB_SELECT(tabla, nombre, opcióncol, tamaño, múltiple, filas, filasseleccionadas)

@DTW_TB_SELECT(tabla, nombre, opcióncol, tamaño, múltiple, filas)

@DTW_TB_SELECT(tabla, nombre, opcióncol, tamaño, múltiple)

@DTW_TB_SELECT(tabla, nombre, opcióncol, tamaño)

@DTW_TB_SELECT(tabla, nombre, opcióncol)

@DTW_TB_SELECT(tabla, nombre)

Parámetros

Tabla 156. Parámetros de DTW_TB_SELECT

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla de la macro que ha de visualizarse como campo SELECT.
serie	<i>nombre</i>	IN	El valor del atributo NAME del campo SELECT.
entero	<i>opcióncol</i>	IN	El número de columna de <i>tabla</i> con los valores a utilizar en los códigos OPTION del campo SELECT. El valor por omisión es utilizar la primera columna.
entero	<i>tamaño</i>	IN	El número de filas de <i>tabla</i> a utilizar para los códigos OPTION en el campo SELECT. El valor por omisión es utilizar todas las filas.
serie	<i>múltiple</i>	IN	Especifica si se admiten selecciones múltiples. El valor por omisión es N, el cual no admite selecciones múltiples.

Tabla 156. Parámetros de DTW_TB_SELECT (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>filas</i>	IN	Los números de fila de <i>tabla</i> a utilizar en el campo SELECT. El valor por omisión es utilizar todas las filas.
serie	<i>filasseleccionadas</i>	IN	La lista de filas de la tabla cuyos códigos de OPTION se comprueban. Para especificar más de una fila, debe de establecer el parámetro múltiple en Y. El valor por omisión es seleccionar el primer elemento.
serie	valorcol	IN	El número de columna de <i>tabla</i> a utilizar para el atributo VALUE de los códigos OPTION. El valor por omisión es 1. Este parámetro es opcional.

Códigos de Retorno

Tabla 157. Códigos de retorno de DTW_TB_SELECT

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Genera un menú SELECT de HTML con selecciones múltiples

```
@DTW_TB_SELECT(Mytable,"URL6","4","","y","1 2 4","1 4")
```

Resultado:

```

<select name="url6" size="3" multiple>
<option selected>image1text
<option>image2text
<option selected>image4text
</select>

```

Ejemplo 2: Utiliza el parámetro *valorcol* para generar un menú SELECT de HTML que utilice un número de columna desde el que obtener los valores.

```
@DTW_TB_SELECT(Mytable,"URL6","4","","y","1 2 4","1 4", "2")
```

Resultado:

```

<select name="url6" size="3" multiple>
<option value="text_string1" selected>image1text</option>
<option value="text_string2">image2text</option>
<option value="text_string4" selected>image4text</option>
</select>

```

DTW_TB_SETCOLS

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X		X	X	X		X	X

Propósito

Establece el número de columnas de una tabla de Net.Data.

Formato

@DTW_TB_SETCOLS(tabla, cols)

Parámetros

Tabla 158. Parámetros de DTW_TB_SETCOLS

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	INOUT	La variable de tabla de la macro para la que se define el número de columnas.
entero	<i>cols</i>	IN	El número inicial de columnas a asignar en <i>tabla</i> .

Códigos de Retorno

Tabla 159. Códigos de retorno de DTW_TB_SETCOLS

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.

Notas de utilización

1. La función DTW_TB_SETCOLS() sólo puede utilizarse una vez por tabla. A continuación, utilice las funciones DTW_TB_DELETECOL() o DTW_TB_INSERTCOL() para cambiar el número de columnas de la tabla.

2. Especifique las cabeceras de columna utilizando la función DTW_TB_SETN().

Ejemplos

Ejemplo 1: Asigna tres columnas para la tabla y asigna los nombres a las columnas

```
%DEFINE myTable = %TABLE

@DTW_TB_SETCOLS(myTable, "3")
@DTW_TB_SETN(myTable, "Name", "1")
@DTW_TB_SETN(myTable, "Address", "2")
@DTW_TB_SETN(myTable, "Phone", "3")
```

DTW_TB_SETN

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X		X	X	X		X	X

Propósito

Asigna un nombre a una cabecera de columna de una Net.Data.

Formato

@DTW_TB_SETN(tabla, nombre, col)

Parámetros

Tabla 160. Parámetros de DTW_TB_SETN

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	INOUT	La variable de tabla de la macro en la que se establecerá un nombre de columna.
serie	<i>nombre</i>	IN	Serie de carácter que se asigna a la cabecera de columna de la columna especificada en <i>col</i> .
entero	<i>col</i>	IN	El número de columna de la columna cuya cabecera esté definiéndose.

Códigos de Retorno

Tabla 161. Códigos de retorno de DTW_TB_SETN

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Notas de utilización

1. Antes de llamar a DTW_TB_SETN(), defina el número de columnas en la tabla. Puede establecer el número de columnas con las funciones DTW_TB_SETCOLS() o DTW_TB_INSERTCOL(), o pasando la tabla a un entorno de lenguaje a establecer.
2. Para suprimir una cabecera de columna, asigne el valor de cabecera de columna a NULL.

Ejemplos

Ejemplo 1: Asigna un nombre a las cabeceras de columna de la 1 a la 3

```
%DEFINE myTable = %TABLE  
  
@DTW_TB_SETCOLS(myTable, "3")  
@DTW_TB_SETN(myTable, "Name", "1")  
@DTW_TB_SETN(myTable, "Address", "2")  
@DTW_TB_SETN(myTable, "Phone", "3")
```

Ejemplo 2: Suprimir la cabecera de columna para la columna 2. Esto se hace pasando una variable en la llamada de función que no se ha definido. Por omisión, esta variable tendrá un valor de NULL

```
%DEFINE myTable = %TABLE  
  
@DTW_TB_SETN(myTable, nullVar, "2")
```

DTW_TB_SETV

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X		X	X	X		X	X

Propósito

Asigna un valor a una columna y fila determinada de una tabla de Net.Data.

Formato

@DTW_TB_SETV(tabla, valor, fila, col)

Parámetros

Tabla 162. Parámetros de DTW_TB_SETV

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	INOUT	La variable de tabla de la macro en la que se establecerá un valor de tabla.
serie	<i>valor</i>	IN	Serie de carácter que se asigna al valor de tabla de la fila y columna especificadas en <i>fila</i> y <i>col</i> .
entero	<i>fila</i>	IN	El número de fila del valor que ha de establecerse.
entero	<i>col</i>	IN	El número de columna del valor que ha de establecerse.

Códigos de Retorno

Tabla 163. Códigos de retorno de DTW_TB_SETV

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.

Tabla 163. Códigos de retorno de DTW_TB_SETV (continuación)

Código de retorno	Explicación
1008	Un parámetro está fuera de los límites de la tabla.

Notas de utilización

1. Antes de llamar a DTW_TB_SETV(), defina el número de columnas en la tabla. Puede establecer el número de columnas con las funciones DTW_TB_SETCOLS() o DTW_TB_INSERTCOL(), o pasando la tabla a un entorno de lenguaje a establecer.
2. Para suprimir un valor de tabla, asigne el valor para NULL.

Ejemplos

Ejemplo 1: Asigna un valor a la fila 3 columna 3

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_SETV(myTable, "value3.3", "3", "3")
```

Ejemplo 2: Suprimir el valor de tabla en la fila 4, columna 2. Esto se hace pasando una variable en la llamada de función que no se ha definido. Por omisión, esta variable tendrá un valor de NULL.

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_SETV(myTable, nullVar, "4", "2")
```

DTW_TB_TABLE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Genera una tabla de HTML a partir de una tabla de Net.Data.

Formato

@DTW_TB_TABLE(tabla, opciones, collist, estilocelda, u_enlace, u_imagen, texto_url, estilo_url)

@DTW_TB_TABLE(tabla, opciones, collist, estilocelda, u_enlace, u_imagen, texto_url)

@DTW_TB_TABLE(tabla, opciones, collist, estilocelda, u_enlace, u_imagen)

@DTW_TB_TABLE(tabla, opciones, collist, estilocelda, u_enlace)

@DTW_TB_TABLE(tabla, opciones, collist, estilocelda)

@DTW_TB_TABLE(tabla, opciones, collist)

@DTW_TB_TABLE(tabla, opciones)

@DTW_TB_TABLE(tabla)

Parámetros

Tabla 164. Parámetros de DTW_TB_TABLE

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	Variable de tabla de la macro que ha de visualizarse como tabla HTML.
serie	<i>opciones</i>	IN	Los atributos de tabla que hay dentro del código TABLE. El valor por omisión es no utilizar atributos. Los valores válidos incluyen: <ul style="list-style-type: none">• BORDER• CELSPACING• WIDTH
serie	<i>collist</i>	IN	Los números de columna de <i>tabla</i> a utilizar en la tabla HTML. El valor por omisión es utilizar todas las columnas,
serie	<i>estilocelda</i>	IN	Lista de elementos de estilo de HTML, por ejemplo, B e I, para acompañar al texto en cada código de TD. El valor por omisión es no utilizar códigos de estilo.

Tabla 164. Parámetros de DTW_TB_TABLE (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
entero	<i>u_enlace</i>	IN	El número de columna de <i>tabla</i> que contiene los URL utilizados para crear enlaces de HTML. Debe especificar asimismo la columna de <i>collist</i> . El valor por omisión es no generar enlaces de HTML.
entero	<i>u_imagen</i>	IN	El número de columna de <i>tabla</i> que contiene los URL utilizados para crear imágenes incorporadas. Debe especificar asimismo la columna de <i>collist</i> . El valor por omisión es no generar códigos de imagen.
entero	<i>texto_url</i>	IN	El número de columna de <i>tabla</i> que contiene el texto a visualizar para los enlaces de URL o imágenes incorporadas. El valor por omisión es utilizar el propio URL.
serie	<i>estilo_url</i>	IN	Lista de elementos de estilo de HTML para el texto especificado en <i>texto_url</i> . El valor por omisión es no generar códigos de estilo.

Códigos de Retorno

Tabla 165. Códigos de retorno de DTW_TB_TABLE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Genera códigos de HTML para una tabla con un límite y utilizar códigos B (negrita) e I (cursiva)

@DTW_TB_TABLE(Mytable,"BORDER","4 2 1","i","2","1","4","b")

Resultado:

```
<table border>
<tr>
<th>TITLE</th>
<th>LINKURL</th>
<th>IMAGEURL</th>
<tr>
<td><i>link1text</i></td>
<td><a
href="http://www.mycompany.com/link1.html"><b>link1text</b></a></td>
<td><b>link1text</b></td>
</tr><tr>
<td><i>link2text</i></td>
<td><a
href="http://www.mycompany.com/link2.html"><b>link2text</b></a></td>
<td><b>link2text</b></td>
</tr><tr>
<td><i>link3text</i></td>
<td><a
href="http://www.mycompany.com/link3.html"><b>link3text</b></a></td>
<td><b>link3text</b></td>
</tr></table>
```


DTW_TB_TEXTAREA

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Genera un área de texto de HTML a partir de una tabla de Net.Data.

Formato

@DTW_TB_TEXTAREA(tabla, nombre, númfilas, númcols, valorcol, filas)

@DTW_TB_TEXTAREA(tabla, nombre, númfilas, númcols, valorcol)

@DTW_TB_TEXTAREA(tabla, nombre, númfilas, númcols)

@DTW_TB_TEXTAREA(tabla, nombre, númfilas)

@DTW_TB_TEXTAREA(tabla, nombre)

Parámetros

Tabla 166. Parámetros de DTW_TB_TEXTAREA

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	Variable de tabla de la macro que ha de mostrarse como código de TEXTAREA.
serie	<i>nombre</i>	IN	El nombre del área de texto.
entero	<i>númfilas</i>	IN	La altura del área de texto, especificada en filas. El valor por omisión es el número de filas de <i>tabla</i> .
entero	<i>númcols</i>	IN	La anchura del área de texto, especificada en columnas. El valor por omisión es la longitud de la fila más larga de <i>tabla</i> .
entero	<i>valorcol</i>	IN	El número de columna de <i>tabla</i> cuyos valores se muestran en el área de texto. El valor por omisión es la primera columna.
serie	<i>filas</i>	IN	Lista de filas de <i>tabla</i> utilizada para generar el código de TEXTAREA. El valor por omisión es utilizar todas las filas.

Códigos de Retorno

Tabla 167. Códigos de retorno de DTW_TB_TEXTAREA

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
1008	Un parámetro está fuera de los límites de la tabla.

Ejemplos

Ejemplo 1: Genera códigos de HTML TEXTAREA y especifica las filas a incluir

```
@DTW_TB_TEXTAREA(Mytable,"textarea5","3","70","4","1 3 4")
```

Resultado:

```
<textarea name="textarea5" rows="3"
cols="70">
link1text
link3text
link4text
</textarea>
```

Funciones de interfaz de archivo plano

La interfaz de archivo plano (FFI) le permite abrir, leer y manipular datos procedentes de fuentes de archivos planos (archivos de texto), así como datos de la tienda en archivos planos. Están disponibles las funciones incorporadas de interfaz de archivo plano siguientes:

- “DTWF_APPEND” en la página 314
- “DTWF_CLOSE” en la página 318
- “DTWF_DELETE” en la página 320
- “DTWF_OPEN” en la página 326
- “DTWF_READ” en la página 329
- “DTWF_READFILE” en la página 333
- “DTWF_REMOVE” en la página 336
- “DTWF_SEARCH” en la página 338
- “DTWF_UPDATE” en la página 342
- “DTWF_WRITE” en la página 346

Las secciones siguientes tratan sobre el modo de utilizar las funciones incorporadas de FFI y de acceder a las fuentes de archivos planos:

- “Cómo acceder a las fuentes de datos de archivos planos”
- “Delimitadores de la interfaz de archivo plano” en la página 312
- “Bloqueo de archivos” en la página 313

Cómo acceder a las fuentes de datos de archivos planos

La sentencia de configuración de vía de acceso de FFI_PATH se utiliza en el archivo de inicialización de Net.Data para listar los directorios y subdirectorios que se pueden especificar al utilizar las funciones de FFI y para proporcionar seguridad para los archivos que no están en los directorios incluidos en la sentencia de vía de acceso. El archivo de inicialización de Net.Data se envía sin FFI_PATH. Consulte el manual *Guía de administración y programación de Net.Data* para averiguar el modo de configurar la vía de acceso.

FFI_PATH utiliza la sintaxis siguiente:

```
FFI_PATH  
/path1;/path2;/path3...
```

Cuando se llama un entorno de lenguaje de FFI en una función macro, se especifica la vía de acceso al archivo plano con el que está trabajando la función FFI, utilizando el parámetro *nombarchivo* de la función FFI. Por ejemplo:

```
%DEFINE myfile = "/macros/myfile.txt" @DTWF_READ(myfile, ...)
```

Las secciones siguientes tratan:

- “El modo en que Net.Data determina la ubicación del archivo plano”
- “Normas de configuración de archivo plano” en la página 311
- “Recomendaciones de seguridad” en la página 311
- “Requisito de autorización” en la página 312

El modo en que Net.Data determina la ubicación del archivo plano

Net.Data utiliza la información del parámetro *nombarchivo* para que las funciones FFI busquen la sentencia FFI_PATH en el archivo de inicialización de Net.Data y determinen si se ha de utilizar un directorio especificado o el directorio actual.

Cuando se especifica un nombre de archivo en una función FFI, Net.Data intenta localizar el archivo buscando cada una de las vías de acceso listadas en FFI_PATH, comenzando a partir de la primera vía de acceso que se especifique. Net.Data utiliza la primera copia que encuentra. Si no se encuentra el archivo, Net.Data intenta encontrar el archivo en el directorio de trabajo actual del paso o proceso en el que está ejecutándose Net.Data.

Ejemplo: Net.Data utiliza la sentencia de configuración FFI_PATH para localizar un archivo

FFI_PATH contiene los directorios siguientes:

```
FFI_PATH
/macros;/macros/org1;/macros/org2
```

Y el archivo está ubicado tanto en el directorio actual como en /macros/org1. Si la llamada de función es:

```
DTWF_READ("myfile.txt")
```

Net.Data will use /macros/org1/myfile.txt.

Si la función DTWF_READ se está utilizando para leer un archivo existente y se especifica un nombre de archivo de myfile.txt, Net.Data busca los directorios /macros, /macros/org1 y /macros/org2 para el archivo, asumiendo que FFI_PATH contiene la lista de vías de acceso especificadas anteriormente.

Determinación del directorio actual:

El directorio actual para Net.Data depende de la configuración del servidor de la Web:

- Si está utilizando CGI, el directorio actual es el directorio desde el que está ejecutándose Net.Data.

- Si está utilizando la API del servidor de la Web, es posible que varíe el directorio actual. Si se cambia la correlación de recursos o el direccionamiento de peticiones por omisión del servidor, es posible que también se cambie el directorio actual.

Recomendaciones para especificar el acceso de archivo plano:

Utilice las recomendaciones siguientes para asegurarse de que Net.Data pueda acceder a fuentes de datos de archivo plano.

- Al utilizar la función DTWF_OPEN para crear archivos planos, asegúrese de especificar una vía de acceso de directorio que esté en FFI_PATH o de que sabe cuál es el directorio actual. Si no especifica un directorio, Net.Data intenta crear el archivo en el directorio de trabajo actual.
- Si incluye directorios en el parámetro *nombarchivo*, especifique la vía de acceso completa que coincide con una de las vías de acceso de FFI_PATH ya que Net.Data no busca subdirectorios en los directorios especificados en FFI_PATH.
- Utilice vías de acceso absolutas para el parámetro *nombarchivo*, especialmente si está utilizando la API del servidor de la Web.

Normas de configuración de archivo plano

Utilice las normas siguientes al añadir o actualizar FFI_PATH en el archivo de inicialización de Net.Data:

- Las sentencias de vía de acceso FFI_PATH deben contener caracteres imprimibles válidos. FFI no permite las vías de acceso que incluyan un signo de interrogación (?) o dobles comillas (“”).
- Todos los directorios y subdirectorios que se utilizan con el parámetro *nombarchivo* de la macro deben especificarse en FFI_PATH. Los subdirectorios de las vías de acceso listadas en *nombarchivo* no se examinan a menos que se especifiquen explícitamente en FFI_PATH.
- Utilice vías de acceso absolutas para la sentencia FFI_PATH.

Recomendaciones de seguridad

Puede especificar los archivos a los que pueden acceder las funciones FFI con la sentencia FFI_PATH en el archivo de inicialización de Net.Data. FFI sólo examina las vías de acceso listadas en la sentencia, por lo que los archivos de otros directorios están protegidos frente al acceso no autorizado.

Por ejemplo, puede especificar una FFI_PATH similar a la que aparece a continuación, diseñando directorios para los ID de usuario públicos o subordinados.

```
FFI_PATH      C:\public;E:\WWW;E:\guest;A:
```

La lista siguiente facilita recomendaciones para proteger los archivos planos:

- Elija los directorios que resultan apropiados para su utilización en las operaciones de archivo plano. Estos directorios han de añadirse a la FFI_PATH para limitar la búsqueda a dichos directorios.
- Asegúrese de permitir que se efectúe DTWF_REMOVE u otras operaciones de exportación en la macro para evitar que se eliminen o modifiquen archivos con las extensiones .dll y .cmd que se pudieran tener en el directorio actual.
- Adopte los pasos apropiados para salvaguardar los archivos en el sistema utilizando un control razonable sobre las macros que se añaden al sistema.
- No especifique una vía de acceso en FFI_PATH que permita a usuarios de FTP anónimos grabar en la vía de acceso. Si lo hace, alguien podría colocar una macro de Net.Data en el sistema que permita acciones que no se permitían con anterioridad.
- No añada la vía de acceso del archivo de inicialización de Net.Data a la FFI_PATH.

Requisito de autorización

Asegúrese de que el ID de usuario bajo el que se ejecuta Net.Data tiene derechos de acceso a los archivos que utilizan las funciones incorporadas de FFI. Consulte la sección sobre la especificación de derechos de acceso del servidor de la Web para archivos Net.Data en el capítulo de configuración del manual *Guía de administración y programación de Net.Data* para obtener más información.

Delimitadores de la interfaz de archivo plano

Para mejorar el rendimiento, puede conservar la salida tabular de Net.Data de una serie de peticiones de SQL en un archivo plano. Puede recuperar el archivo plano en peticiones sucesivas, en vez de volver a emitir las peticiones de SQL.

Los archivos planos de Net.Data pueden crearse desde las tablas de Net.Data y pueden crearse tablas de Net.Data a partir de archivos planos. Para efectuar las transformaciones entre las tablas y archivos planos, debe definir la correlación entre columnas de una tabla y los registros de un archivo plano. Un *delimitador* es un distintivo o separador que utiliza FFI al dividir el archivo en partes (por ejemplo las columnas de una fila) con arreglo a la transformación solicitada. Los delimitadores facilitan un método para definir el modo en que las partes de registros de un archivo plano pueden separarse y correlacionarse a las columnas de una tabla y el modo en que las columnas de una tabla pueden correlacionarse con los registros de un archivo plano.

Hay dos tipos de delimitadores:

Carácter de nueva línea (ASCII TEXT)

Utilice esta transformación cuando la tabla esté compuesta por una columna. Net.Data correlaciona cada una de las líneas con el archivo

plano correspondiente en una única fila de la tabla. En este caso, el carácter de línea nueva del archivo plano es el único delimitador utilizado.

Serie de delimitador y carácter de nueva línea (DELIMITED)

Utilice esta transformación cuando la tabla esté compuesta por varias columnas. Cuando Net.Data graba datos de fila en una línea de un archivo plano, coloca la serie del delimitador como separador entre las entradas de columna. Cuando Net.Data vuelve a crear una tabla a partir de un archivo plano, utiliza la serie del delimitador para determinar la parte de cada línea que ha de colocarse en una columna de la tabla. En este caso, el carácter de nueva línea regular separa las líneas en el archivo plano que se corresponde con las filas de la tabla y la serie de delimitador separa los elementos dentro de una única línea.

Para las operaciones de lectura, el delimitador separa el contenido del archivo en filas y columnas de una tabla. Para las operaciones de grabación, el delimitador indica el final de un valor en una columna y fila de tabla. Net.Data pasa el delimitador al FFI como serie de la macro de Net.Data y no incluye un carácter nulo al final de los caracteres a menos que se liste explícitamente en el parámetro DELIMITER.

Para utilizar el carácter nulo en el delimitador, especifique el parámetro DELIMITER como barra inclinada y un cero entre dobles comillas, “\0”, en vez de una serie vacía por medio de la utilización de dos dobles comillas, “””. Si especifica la transformación de ASCIITEXT, Net.Data utiliza el carácter de nueva línea como delimitador e ignora cualquier delimitador especificado.

En el caso de que utilice un delimitador diferente para las operaciones de grabación que para las operaciones de lectura es posible que se produzcan cambios no deseados. Net.Data graba el archivo con el nuevo delimitador.

La longitud máxima de un delimitador es de 256 caracteres.

Bloqueo de archivos

Puede bloquear archivos planos utilizando las funciones DTWF_OPEN y DTWF_CLOSE. Con estas funciones, Net.Data reserva un archivo plano para que ninguna otra aplicación pueda leer o actualizar el archivo.

Para bloquear un archivo, utilice la función DTWF_OPEN. Esta función asegura que el archivo no está disponible para otras aplicaciones e impide que cambie el archivo entre el momento en que se lee y el momento en que se actualiza.

Para liberar el archivo, utilice la función DTWF_CLOSE. Esta función libera el archivo para que otras aplicaciones puedan leer o actualizar el archivo.

DTWF_APPEND

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Graba el contenido de una tabla de Net.Data al final de un archivo de texto.

Formato

@DTWF_APPEND(nombarchivo, transform, delimitador, tabla, reintento, filas)

@DTWF_APPEND(nombarchivo, transform, delimitador, tabla, reintento)

@DTWF_APPEND(nombarchivo, transform, delimitador, tabla)

Parámetros

Tabla 168. Parámetros de DTWF_APPEND

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo al que se está añadiendo el contenido de la variable. Cuando la llamada se completa satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
serie	<i>transform</i>	IN	El formato del archivo: <ul style="list-style-type: none">• ASCIITEXT - graba la tabla para el archivo con un carácter de nueva línea entre valores de columna e ignora el parámetro <i>delimitador</i>.• DELIMITED - graba la tabla para el archivo con el delimitador especificado en el parámetro <i>delimitador</i>. Un carácter de nueva línea en un archivo indica el final de una fila de una tabla de macro de Net.Data para transformaciones de ASCIITEXT y DELIMITED.
serie	<i>delimitador</i>	IN	Serie de caracteres para indicar los extremos de los valores. Este parámetro es sensible a mayúsculas y minúsculas. Se ignora si <i>transform</i> es ASCIITEXT.

Tabla 168. Parámetros de DTWF_APPEND (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla desde la que se leen los registros. Para usuarios no OS/400: La longitud máxima de una fila en una tabla FFI es de 16383 caracteres. Este límite incluye un carácter nulo para cada columna de la tabla de macro de Net.Data.
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que el archivo no pueda agregarse de modo inmediato. El valor por omisión es no efectuar ningún reintento.
entero	<i>filas</i>	IN	El número máximo de filas de la <i>tabla</i> a agregar. El valor por omisión es agregar todas las filas. Si se especifica un 0 se agregan todas las filas.

Códigos de Retorno

Tabla 169. Códigos de retorno de DTWF_APPEND

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.

Tabla 169. Códigos de retorno de DTWF_APPEND (continuación)

Código de retorno	Explicación
2003	Una función incorporada de la interfaz de archivo plano no pudo leer una fila de datos en una variable de tabla porque el número de bytes de la fila excedía del número máximo de bytes soportados.
2004	Una función incorporada de la interfaz de archivo plano estaba intentando encontrar un archivo, pero ha encontrado una vía de acceso en la variable del archivo de configuración FFI_PATH cuya longitud era superior al número máximo de bytes soportados, que es 4095.
2005	Se ha producido una anomalía en una llamada a una función del sistema.
2006	Una función incorporada de la interfaz de archivo plano no ha podido acceder al archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Notas de utilización

El contenido actual de un archivo afecta al resultado de utilizar DTWF_APPEND, especialmente el contenido de la última columna de la última fila. Si un carácter de nueva línea va detrás del último valor de columna de la última fila del archivo, los datos agregados se colocan en una fila nueva. En caso contrario, los datos agregados se convierten en parte de la última fila del archivo. Si no existe el archivo a agregar, se crea un archivo.

Ejemplos

Ejemplo 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "DELIMITED", " ;", myTable)
```

Ejemplo 2:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "ASCIITEXT", " ;", myTable)
```

Ejemplo 3:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
}%  
@DTWF_APPEND(myFile, "ASCIIITEXT", " ;", myTable, "0", "10")
```

DTWF_CLOSE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Cierra un archivo abierto por medio de DTWF_OPEN.

Formato

@DTWF_CLOSE(nombarchivo, reintento)

@DTWF_CLOSE(nombarchivo)

Parámetros

Tabla 170. Parámetros de DTWF_CLOSE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo que ha de cerrarse. Cuando la llamada se completa satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que el archivo no pueda cerrarse de modo inmediato. El valor por omisión es no efectuar ningún reintento.

Códigos de Retorno

Tabla 171. Códigos de retorno de DTWF_CLOSE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.

Tabla 171. Códigos de retorno de DTWF_CLOSE (continuación)

Código de retorno	Explicación
2002	Una función incorporada de la interfaz de archivo plano no ha podido cerrar el archivo especificado porque esta invocación de macro no lo había abierto.
2004	Una función incorporada de la interfaz de archivo plano estaba intentando encontrar un archivo, pero ha encontrado una vía de acceso en la variable del archivo de configuración FFI_PATH cuya longitud era superior al número máximo de bytes soportados, que es 4095.

Ejemplos

Ejemplo 1:

```
@DTWF_CLOSE(myFile,
"5")
```

DTWF_DELETE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Suprime líneas de un archivo de texto.

Formato

@DTWF_DELETE(nombarchivo, transform, delimitador, reintento, filas,
líneainicio)

@DTWF_DELETE(nombarchivo, transform, delimitador, reintento, filas)

@DTWF_DELETE(nombarchivo, transform, delimitador, reintento)

@DTWF_DELETE(nombarchivo, transform, delimitador)

Parámetros

Tabla 172. Parámetros de DTW_DELETE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo cuyos registros van a suprimirse. Cuando la llamada se completa satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
serie	<i>transform</i>	IN	El formato del archivo: <ul style="list-style-type: none">• ASCIITEXT - graba la tabla para el archivo con un carácter de nueva línea entre valores de columna e ignora el parámetro <i>delimitador</i>.• DELIMITED - graba la tabla para el archivo con el delimitador especificado en el parámetro <i>delimitador</i>. Un carácter de nueva línea en un archivo indica el final de una fila de una tabla de macro de Net.Data para transformaciones de ASCIITEXT y DELIMITED.
serie	<i>delimitador</i>	IN	Serie de caracteres para indicar los extremos de los valores. Este parámetro es sensible a mayúsculas y minúsculas. Se ignora si <i>transform</i> es ASCIITEXT.

Tabla 172. Parámetros de DTW_DELETE (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que los registros no puedan suprimirse de modo inmediato. El valor por omisión es no efectuar ningún reintento.
entero	<i>filas</i>	IN	El número máximo de filas a suprimir. El valor por omisión es suprimir todas las filas. Si se especifica un 0 se suprimen todas las filas.
entero	<i>líneainicio</i>	INOUT	El número de línea desde el que comenzar la supresión. Un valor de 1 significa comenzar la supresión en la primera línea. Si este valor es mayor que el número de líneas del archivo, se devuelve un valor y el valor de este parámetro se cambia por el número de líneas del archivo. El valor por omisión es 1.

Códigos de Retorno

Tabla 173. Códigos de retorno de DTWF_DELETE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.

Tabla 173. Códigos de retorno de DTWF_DELETE (continuación)

Código de retorno	Explicación
2003	Una función incorporada de la interfaz de archivo plano no pudo leer una fila de datos en una variable de tabla porque el número de bytes de la fila excedía del número máximo de bytes soportados.
2004	Una función incorporada de la interfaz de archivo plano estaba intentando encontrar un archivo, pero ha encontrado una vía de acceso en la variable del archivo de configuración FFI_PATH cuya longitud era superior al número máximo de bytes soportados, que es 4095.
2005	Se ha producido una anomalía en una llamada a una función del sistema.
2006	Una función incorporada de la interfaz de archivo plano no ha podido acceder al archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Ejemplos

Ejemplo 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "5000"
    myRows = "2"
}%
@DTWF_DELETE(myFile, "Delimited", "|", myWait, myRows)
```

Ejemplo 2:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myStart = "1"
    myRows = "2"
}%
@DTWF_DELETE(myFile, "Asciitext", "|", "0", myRows, myStart)
```


DTWF_INSERT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Inserta líneas en un archivo de texto existente.

Formato

@DTWF_INSERT(nombarchivo, transform, delimitador, tabla, reintento, filas, líneainicio)

@DTWF_INSERT(nombarchivo, transform, delimitador, tabla, reintento, filas)

@DTWF_INSERT(nombarchivo, transform, delimitador, tabla, reintento)

@DTWF_INSERT(nombarchivo, transform, delimitador, tabla)

Parámetros

Tabla 174. Parámetros de DTWF_INSERT

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo en el que se insertan los registros. Cuando la llamada se completa satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
serie	<i>transform</i>	IN	El formato del archivo: <ul style="list-style-type: none">• ASCIITEXT - graba la tabla para el archivo con un carácter de nueva línea entre valores de columna e ignora el parámetro <i>delimitador</i>.• DELIMITED - graba la tabla para el archivo con el delimitador especificado en el parámetro <i>delimitador</i>. Un carácter de nueva línea en un archivo indica el final de una fila de una tabla de macro de Net.Data para transformaciones ASCIITEXT y DELIMITED.
serie	<i>delimitador</i>	IN	Serie de caracteres para indicar los extremos de los valores. Este parámetro es sensible a mayúsculas y minúsculas. Se ignora si <i>transform</i> es ASCIITEXT.

Tabla 174. Parámetros de DTWF_INSERT (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	Variable de tabla desde la que se insertan líneas en el archivo. Para usuarios no OS/400: La longitud máxima de una fila en una tabla FFI es de 16383 caracteres. Este límite incluye un carácter nulo para cada columna de la tabla de macro de Net.Data.
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que no pueda grabarse en el archivo de modo inmediato. El valor por omisión es no efectuar ningún reintento.
entero	<i>filas</i>	IN	El número máximo de filas a insertar desde la <i>tabla</i> . El valor por omisión es insertar todas las filas. Un valor de 0 inserta todas las filas.
entero	<i>líneainicio</i>	INOUT	El número de línea desde el que comenzar la inserción. Si este valor es mayor que el número de líneas del archivo, se devuelve un valor y el valor de este parámetro se cambia al número de líneas del archivo. Especificar un 0 significa comenzar la inserción al principio del archivo. El valor por omisión es 0.

Códigos de Retorno

Tabla 175. Códigos de retorno de DTWF_INSERT

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.

Tabla 175. Códigos de retorno de DTWF_INSERT (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.
2003	Una función incorporada de la interfaz de archivo plano no pudo leer una fila de datos en una variable de tabla porque el número de bytes de la fila excedía del número máximo de bytes soportados.
2004	Una función incorporada de la interfaz de archivo plano estaba intentando encontrar un archivo, pero ha encontrado una vía de acceso en la variable del archivo de configuración FFI_PATH cuya longitud era superior al número máximo de bytes soportados, que es 4095.
2005	Se ha producido una anomalía en una llamada a una función del sistema.
2006	Una función incorporada de la interfaz de archivo plano no ha podido acceder al archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Ejemplos

Ejemplo 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "3000"
}%
@DTWF_INSERT(myFile, "Delimited", "|", myTable, myWait)
```

Ejemplo 2:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myStart = "1"
    myRows = "2"
}%
@DTWF_INSERT(myFile, "Asciitext", "|", myTable, "0", myRows, myStart)
```

DTWF_OPEN

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Abre un archivo de texto.

Formato

@DTWF_OPEN(nombarchivo, modalidad, reintento)

@DTWF_OPEN(nombarchivo, modalidad)

Parámetros

Tabla 176. Parámetros de DTWF_OPEN

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo que ha de abrirse. Cuando la llamada se completa satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
serie	<i>modalidad</i>	IN	El tipo de acceso solicitado: <ul style="list-style-type: none">• r - abre un archivo existente para su lectura.• w - crea un archivo para su grabación. (Destruye un archivo existente con el mismo nombre, en el caso de que exista).• a - abre un archivo para su agregación. Net.Data crea el archivo en el caso de que éste no se encuentre.• r+ - abre un archivo existente para su lectura y grabación.• w+ - crea un archivo para su lectura y grabación. (Destruye un archivo existente con el mismo nombre, en el caso de que exista).• a+ - abre un archivo en modalidad de agregación para su lectura o agregación. Net.Data crea el archivo en el caso de que éste no se encuentre.

Tabla 176. Parámetros de DTWF_OPEN (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que el archivo no pueda abrirse de modo inmediato. El valor por omisión es no efectuar ningún reintento.

Códigos de Retorno

Tabla 177. Códigos de retorno de DTWF_OPEN

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.
2001	Una función incorporada de la interfaz de archivo plano no ha podido abrir el archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Notas de utilización

1. Cuando el archivo no existe, debería especificarse una vía de acceso absoluta para el nombre de archivo y el directorio en el que va a crearse el archivo debe coincidir con un directorio especificado en FFI_PATH. Si no se utiliza una vía de acceso absoluta, el archivo se abrirá en el directorio de trabajo actual.
2. DTWF_OPEN mantiene abierto el archivo, en caso contrario, el archivo se cierra después de cada operación de archivo plano.

3. Utilice DTWF_OPEN para reducir el número de veces que se abre un archivo. Si no se utiliza DTWF_OPEN, el archivo se cierra después de cada operación de archivo plano. El archivo se deja abierto hasta que se cierra utilizando DTWF_CLOSE o finaliza el proceso de la macro.

Ejemplos

Ejemplo 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myMode = "r+"  
%}  
@DTWF_OPEN(myFile, myMode, "1000")
```

DTWF_READ

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Lee líneas de un archivo de texto en una tabla de Net.Data.

Formato

@DTWF_READ(nombarchivo, transform, delimitador, tabla, reintento, líneas, líneainicio, columnas)

@DTWF_READ(nombarchivo, transform, delimitador, tabla, reintento, líneas, líneainicio)

@DTWF_READ(nombarchivo, transform, delimitador, tabla, reintento, líneas)

@DTWF_READ(nombarchivo, transform, delimitador, tabla, reintento)

@DTWF_READ(nombarchivo, transform, delimitador, tabla)

Parámetros

Tabla 178. Parámetros de DTWF_READ

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo cuyos registros se leen en una variable de tabla. Cuando la llamada se completa satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
serie	<i>transform</i>	IN	<p>El formato del archivo:</p> <ul style="list-style-type: none">• ASCII TEXT - graba la tabla para el archivo con un carácter de nueva línea entre valores de columna e ignora el parámetro <i>delimitador</i>.• DELIMITED - graba la tabla para el archivo con el delimitador especificado en el parámetro <i>delimitador</i>. <p>Un carácter de nueva línea en un archivo indica el final de una fila de una tabla de macro de Net.Data para transformaciones ASCII TEXT y DELIMITED.</p>

Tabla 178. Parámetros de DTWF_READ (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>delimitador</i>	IN	Serie de caracteres para indicar los extremos de los valores. Este parámetro es sensible a mayúsculas y minúsculas. Se ignora si <i>transform</i> es ASCII TEXT.
tabla	<i>tabla</i>	OUT	La variable de tabla en la que se leen los registros de archivo. Para usuarios no OS/400: La longitud máxima de una fila en una tabla FFI es de 16383 caracteres. Este límite incluye un carácter nulo para cada columna de la tabla de macro de Net.Data.
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que el archivo no pueda leerse de modo inmediato. El valor por omisión es no efectuar ningún reintento.
entero	<i>líneas</i>	INOUT	El número de líneas del archivo que han de leerse en la tabla. Un valor de 0 significa leer hasta el final del archivo o hasta que la tabla esté completa; este es el valor por omisión. Cuando esta llamada de función se completa de modo satisfactorio, se devuelve el número de filas de la tabla resultante.
entero	<i>líneainicio</i>	IN	La línea del archivo desde la que se ha de iniciar la lectura. El valor por omisión es comenzar la lectura en la primera línea.
entero	<i>columnas</i>	OUT	Devuelve el número de columnas de la tabla.

Códigos de Retorno

Tabla 179. Códigos de retorno de DTWF_READ

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.

Tabla 179. Códigos de retorno de DTWF_READ (continuación)

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
1010	Se han grabado datos en la tabla hasta que se ha llenado y se han desechado los datos restantes.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.
2003	Una función incorporada de la interfaz de archivo plano no pudo leer una fila de datos en una variable de tabla porque el número de bytes de la fila excedía del número máximo de bytes soportados.
2004	Una función incorporada de la interfaz de archivo plano estaba intentando encontrar un archivo, pero ha encontrado una vía de acceso en la variable del archivo de configuración FFI_PATH cuya longitud era superior al número máximo de bytes soportados, que es 4095.
2005	Se ha producido una anomalía en una llamada a una función del sistema.
2006	Una función incorporada de la interfaz de archivo plano no ha podido acceder al archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Ejemplos

Ejemplo 1:

```
%DEFINE {
  myFile = "c:/private/myfile"
  myTable = %TABLE
  myWait = "1000"
}%
@DTWF_READ(myFile, "DELIMITED", ";", myTable, myWait)
```

Ejemplo 2:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "0"  
    myRows = "0"  
    myStartrow = "1"  
    myColumns = ""  
}%  
@DTWF_READ(myFile, "DELIMITED", ";", myTable, myWait, myRows,  
            myStartrow, myColumns)
```

Ejemplo 3:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
}%  
@DTWF_READ(myFile, "ASCIITEXT", ";", myTable)  
@DTW_TB_TABLE(myTable, "BORDER", "")
```

DTWF_READFILE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X			X	X	X

Propósito

Utilice esta función para leer un archivo y asignarlo a una variable. La función busca en FFI_PATH el archivo especificado.

Esta función puede utilizarse para pasar una gran cantidad de datos a otra función o procedimiento almacenado. Si los datos de archivo contienen caracteres NULL, pueden truncarse cuando se hace referencia a la variable en la macro.

Formato

@DTWF_READFILE(archivoEntrada,varSalida)

Parámetros

Tabla 180. Parámetros de DTWF_READFILE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>archivoEntrada</i>	IN	El nombre del archivo que ha de leerse.
serie	<i>varSalida</i>	OUT	La variable a la que se asigna el contenido del archivo.

Códigos de Retorno

Tabla 181. Códigos de retorno de DTWF_READFILE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.

Tabla 181. Códigos de retorno de DTWF_READFILE (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
1010	Se han grabado datos en la tabla hasta que se ha llenado y se han desechado los datos restantes.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.
2001	Una función incorporada de la interfaz de archivo plano no ha podido abrir el archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.
2003	Una función incorporada de la interfaz de archivo plano no pudo leer una fila de datos en una variable de tabla porque el número de bytes de la fila excedía del número máximo de bytes soportados.
2004	Una función incorporada de la interfaz de archivo plano estaba intentando encontrar un archivo, pero ha encontrado una vía de acceso en la variable del archivo de configuración FFI_PATH cuya longitud era superior al número máximo de bytes soportados, que es 4095.
2005	Se ha producido una anomalía en una llamada a una función del sistema.
2006	Una función incorporada de la interfaz de archivo plano no ha podido acceder al archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Ejemplos

Ejemplo 1: Un archivo se lee y visualiza completamente en mayúsculas.

```
%HTML(report){
@DTWF_READFILE("sample.html",sampfile)
@DTW_UPPERCASE(sampfile)
%}
```

Ejemplo 2: En este ejemplo, se carga un archivo en el servidor, después se lee utilizando DTWF_READFILE y se pasa a un procedimiento almacenado para su proceso y visualización en un navegador XML.

En el archivo de configuración:

```
DTW_UPLOAD_DIR /usr/lpp/internet/server_root/tmplobs
FFI_PATH       /usr/lpp/internet/server_root/tmplobs
```

En la macro:

```
%DEFINE{
    DATABASE="SALES_DB"
%}

%FUNCTION(DTW_SQL) dxx_GenXML(IN CLOB dad,          IN CHAR(31) result_tab,
                              IN INTEGER ovType, IN VARCHAR(1024) over,
                              IN INTEGER MaxRow, OUT INTEGER NumRow,
                              OUT INTEGER dxxRC, OUT VARCHAR(99) dxxMSG) {

    CALL db2xml!dxxGenXML
    %REPORT{
        %ROW{
            @DTW_rHTMLENCODE(V1)
        %}
    %}
    %MESSAGE{
        default: "" : continue
    %}
%}

%HTML(input){
<form enctype="multipart/form-data" method="post" action="report">
    <p>File to upload:</p>
    <p><input type="file" name="dad_file" /></p>
    <p><input type="submit" /></p>
</form>
%}

%XML(report){
    @DTWF_READFILE(DAD_file, DAD)
    <xml version="1.0" ?>
    <xml-stylesheet type="text/xsl" href="ndTable.xsl" ?>
    <XMLBlock>
        @dxx_GenXML( DAD, "result_tab", "0", "" , "1", numRows, dxxRC, dxxMSG )
    </XMLBlock>
%}
```

DTWF_REMOVE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Suprime un archivo completo.

Formato

@DTWF_REMOVE(nombarchivo, reintento)

@DTWF_REMOVE(nombarchivo)

Parámetros

Tabla 182. Parámetros de DTWF_REMOVE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo que ha de suprimirse. Cuando la llamada se completa satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que el archivo no pueda suprimirse de modo inmediato. El valor por omisión es no efectuar ningún reintento.

Códigos de Retorno

Tabla 183. Códigos de retorno de DTWF_REMOVE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.

Tabla 183. Códigos de retorno de DTWF_REMOVE (continuación)

Código de retorno	Explicación
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.
2006	Una función incorporada de la interfaz de archivo plano no ha podido acceder al archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Ejemplos

Ejemplo 1:

```
%DEFINE myFile =
"c:/private/myfile"
@DTWF_REMOVE(myFile)
```

Ejemplo 2:

```
%DEFINE {
  myFile = "c:/private/myfile"
  myWait = "2000"
}%
@DTWF_REMOVE(myFile, myWait)
```

DTWF_SEARCH

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Busca una serie en un archivo de texto, devolviendo el resultado a una tabla de Net.Data.

Formato

@DTWF_SEARCH(nombarchivo, transform, delimitador, tabla, seriebúsqueda, reintento, líneasbúsqueda, líneainicio)

@DTWF_SEARCH(nombarchivo, transform, delimitador, tabla, seriebúsqueda, reintento, líneasbúsqueda)

@DTWF_SEARCH(nombarchivo, transform, delimitador, tabla, seriebúsqueda, reintento)

@DTWF_SEARCH(nombarchivo, transform, delimitador, tabla, seriebúsqueda)

Parámetros

Tabla 184. Parámetros de DTWF_SEARCH

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo que ha de buscarse. Cuando la llamada se completa satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
serie	<i>transform</i>	IN	<p>El formato del archivo:</p> <ul style="list-style-type: none">• ASCIITEXT - graba la tabla para el archivo con un carácter de nueva línea entre valores de columna e ignora el parámetro <i>delimitador</i>.• DELIMITED - graba la tabla para el archivo con el delimitador especificado en el parámetro <i>delimitador</i>. <p>Un carácter de nueva línea en un archivo indica el final de una fila de una tabla de macro de Net.Data para transformaciones ASCIITEXT y DELIMITED.</p>

Tabla 184. Parámetros de DTWF_SEARCH (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>delimitador</i>	IN	Serie de caracteres para indicar los extremos de los valores. Este parámetro es sensible a mayúsculas y minúsculas. Se ignora si <i>transform</i> es ASCIITEXT.
tabla	<i>tabla</i>	OUT	<p>La variable de tabla en la que se coloca el resultado de la búsqueda. Se devuelven tres columnas:</p> <ul style="list-style-type: none"> • La línea en la que se encontró la correspondencia • El campo en el que se ha encontrado la correspondencia (para una transformación de ASCIITEXT, siempre es un 1) • El valor del campo que contenía la serie de búsqueda <p>Para usuarios no OS/400: La longitud máxima de una fila en una tabla FFI es de 16383 caracteres. Este límite incluye un carácter nulo para cada columna de la tabla de macro de Net.Data.</p>
serie	<i>seriebúsqueda</i>	IN	La serie de caracteres que ha de buscarse.
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que el archivo no pueda examinarse de modo inmediato. El valor por omisión es no efectuar ningún reintento.
entero	<i>líneasbusqueda</i>	INOUT	El número de líneas del archivo que han de examinarse. Un valor de 0 significa que se examinan todas las líneas del archivo o hasta que la tabla esté completa; este es el valor por omisión. Cuando se complete de modo satisfactorio, este parámetro devuelve el número de filas de la tabla resultante.
entero	<i>líneainicio</i>	IN	La línea del archivo desde la que se ha de iniciar la búsqueda. El valor por omisión es 1, que hace que la búsqueda comience en el primer registro.

Códigos de Retorno

Tabla 185. Códigos de retorno de DTWF_SEARCH

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
1010	Se han grabado datos en la tabla hasta que se ha llenado y se han desechado los datos restantes.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.
2003	Una función incorporada de la interfaz de archivo plano no pudo leer una fila de datos en una variable de tabla porque el número de bytes de la fila excedía del número máximo de bytes soportados.
2004	Una función incorporada de la interfaz de archivo plano estaba intentando encontrar un archivo, pero ha encontrado una vía de acceso en la variable del archivo de configuración FFI_PATH cuya longitud era superior al número máximo de bytes soportados, que es 4095.
2005	Se ha producido una anomalía en una llamada a una función del sistema.
2006	Una función incorporada de la interfaz de archivo plano no ha podido acceder al archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Notas de utilización

1. La tabla que se devuelve para DTWF_SEARCH tiene tres columnas. Las primeras dos columnas contienen el número de fila y de columna en el que se encuentra la correspondencia; la última columna contiene el valor de columna que contiene los caracteres que se han especificado en el parámetro *Seriebúsqueda*. Por ejemplo, si la cuarta fila del archivo contiene caracteres coincidentes en la columna tres, la tabla devuelta tiene una fila con el número 4 en la primera columna para indicar la fila del archivo del que procedía; tiene un número 3 en la segunda columna para indicar la columna del archivo que contiene una coincidencia y tiene el valor de columna completo en la columna tercera.
2. El parámetro *Seriebúsqueda* no puede incluir el contenido del parámetro *delimitador*.

Ejemplos

Ejemplo 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "1000"  
    mySearch = "0123456789abcdef"  
@DTWF_SEARCH(myFile, "DELIMITED", ";",  
    myTable, mySearch, myWait)
```

Ejemplo 2:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    mySearch = "answer:"  
    myWait = "0"  
    myRows = "0"  
    myStartrow = "1"  
%}  
@DTWF_SEARCH(myFile, "DELIMITED", ";", myTable,  
    mySearch, myWait, myRows, myStartrow)
```

DTWF_UPDATE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Actualiza líneas de un archivo de texto con datos procedentes de una tabla de Net.Data.

Formato

@DTWF_UPDATE(nombarchivo, transform, delimitador, tabla, reintento, filas, lineainicio)

@DTWF_UPDATE(nombarchivo, transform, delimitador, tabla, reintento, filas)

@DTWF_UPDATE(nombarchivo, transform, delimitador, tabla, reintento)

@DTWF_UPDATE(nombarchivo, transform, delimitador, tabla)

Parámetros

Tabla 186. Parámetros de DTWF_UPDATE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo cuyos registros se actualizan desde una variable de tabla. Cuando la llamada se completa satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
serie	<i>transform</i>	IN	<p>El formato del archivo:</p> <ul style="list-style-type: none">• ASCII TEXT - graba la tabla para el archivo con un carácter de nueva línea entre valores de columna e ignora el parámetro <i>delimitador</i>.• DELIMITED - graba la tabla para el archivo con el delimitador especificado en el parámetro <i>delimitador</i>. <p>Un carácter de nueva línea en un archivo indica el final de una fila de una tabla de macro de Net.Data para transformaciones ASCII TEXT y DELIMITED.</p>

Tabla 186. Parámetros de DTWF_UPDATE (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>delimitador</i>	IN	Serie de caracteres para indicar los extremos de los valores. Este parámetro es sensible a mayúsculas y minúsculas. Se ignora si <i>transform</i> es ASCIITEXT.
tabla	<i>tabla</i>	IN	La variable de tabla desde la que se actualizan los registros de archivo. Para usuarios no OS/400: La longitud máxima de una fila en una tabla FFI es de 16383 caracteres. Este límite incluye un carácter nulo para cada columna de la tabla de macro de Net.Data.
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que no pueda grabarse en el archivo de modo inmediato. El valor por omisión es no efectuar ningún reintento.
entero	<i>filas</i>	IN	El número de filas de la tabla a utilizar al actualizar el archivo. Un valor de 0 significa utilizar todas las filas al actualizar el archivo; este es el valor por omisión. Tenga en cuenta que sólo se actualizan las líneas existentes del archivo, no se añade ninguna línea.
entero	<i>líneainicio</i>	INOUT	El primer registro de archivo a actualizar. El valor por omisión es 1, que significa comenzar a actualizar el principio del archivo. Si el valor es mayor que el número de líneas del archivo, se cambia el valor para indicar el número de la última línea del archivo y se devuelve un error.

Códigos de Retorno

Tabla 187. Códigos de retorno de DTWF_UPDATE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.

Tabla 187. Códigos de retorno de DTWF_UPDATE (continuación)

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.
2001	Una función incorporada de la interfaz de archivo plano no ha podido abrir el archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.
2003	Una función incorporada de la interfaz de archivo plano no pudo leer una fila de datos en una variable de tabla porque el número de bytes de la fila excedía del número máximo de bytes soportados.
2004	Una función incorporada de la interfaz de archivo plano estaba intentando encontrar un archivo, pero ha encontrado una vía de acceso en la variable del archivo de configuración FFI_PATH cuya longitud era superior al número máximo de bytes soportados, que es 4095.
2005	Se ha producido una anomalía en una llamada a una función del sistema.
2006	Una función incorporada de la interfaz de archivo plano no ha podido acceder al archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Notas de utilización

Si el archivo no existe, debería especificarse una vía de acceso absoluta para el nombre de archivo y el directorio en el que va a crearse el archivo debe coincidir con un directorio especificado en FFI_PATH. Si no se utiliza una vía de acceso absoluta, el archivo se abrirá en el directorio de trabajo actual.

Ejemplos

Ejemplo 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "1500"  
    myRows = "2"  
%}  
@DTWF_UPDATE(myFile, "Delimited", "|", myTable, myWait, myRows)
```

Ejemplo 2:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myStart = "1"  
    myRows = "2"  
%}  
@DTWF_UPDATE(myFile, "Asciitext", "|", myTable, "0", myRows, myStart)
```

DTWF_WRITE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X	X	X	X	X	X	X	X	X

Propósito

Graba el contenido de una tabla de Net.Data en un archivo de texto.

Formato

@DTWF_WRITE(nombarchivo, transform, delimitador, tabla, reintento, filas, lineainicio)

@DTWF_WRITE(nombarchivo, transform, delimitador, tabla, reintento, filas)

@DTWF_WRITE(nombarchivo, transform, delimitador, tabla, reintento)

@DTWF_WRITE(nombarchivo, transform, delimitador, tabla)

Parámetros

Tabla 188. Parámetros de DTWF_WRITE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>nombarchivo</i>	INOUT	El nombre del archivo en el que se graban los registros de la variable de tabla. Cuando la llamada se complete satisfactoriamente, este parámetro devuelve el nombre de archivo completamente calificado.
serie	<i>transform</i>	IN	El formato del archivo: <ul style="list-style-type: none">• ASCIITEXT - graba la tabla para el archivo con un carácter de nueva línea entre valores de columna e ignora el parámetro <i>delimitador</i>.• DELIMITED - graba la tabla para el archivo con el delimitador especificado en el parámetro <i>delimitador</i>. Un carácter de nueva línea en un archivo indica el final de una fila de una tabla de macro de Net.Data para transformaciones ASCIITEXT y DELIMITED.
serie	<i>delimitador</i>	IN	Serie de caracteres para indicar los extremos de los valores. Este parámetro es sensible a mayúsculas y minúsculas. Se ignora si <i>transform</i> es ASCIITEXT.

Tabla 188. Parámetros de DTWF_WRITE (continuación)

Tipo de datos	Parámetro	Utilización	Descripción
tabla	<i>tabla</i>	IN	La variable de tabla utilizada para exportar filas al archivo. Para usuarios no OS/400: La longitud máxima de una fila en una tabla FFI es de 16383 caracteres. Este límite incluye un carácter nulo para cada columna de la tabla de macro de Net.Data.
entero	<i>reintento</i>	IN	El número de veces que ha de efectuarse un reintento en el caso de que no pueda grabarse en el archivo de modo inmediato. El valor por omisión es no efectuar ningún reintento.
entero	<i>filas</i>	IN	El número de filas de la tabla a grabar en el archivo. Un valor de 0 significa que se graban todas las filas de la tabla en el archivo; este es el valor por omisión.
entero	<i>líneainicio</i>	INOUT	El número de línea del archivo desde el que se ha de iniciar la grabación. Un valor de 1 significa comenzar en la primera línea del archivo; este es el valor por omisión. Si se especifica un valor posterior al final del archivo, se devuelve un valor y este parámetro se establece en el número de líneas del archivo.

Códigos de Retorno

Tabla 189. Códigos de retorno de DTWF_WRITE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.

Tabla 189. Códigos de retorno de DTWF_WRITE (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
2000	Una función incorporada de la interfaz de archivo plano no ha podido encontrar el archivo especificado.
2003	Una función incorporada de la interfaz de archivo plano no pudo leer una fila de datos en una variable de tabla porque el número de bytes de la fila excedía del número máximo de bytes soportados.
2004	Una función incorporada de la interfaz de archivo plano estaba intentando encontrar un archivo, pero ha encontrado una vía de acceso en la variable del archivo de configuración FFI_PATH cuya longitud era superior al número máximo de bytes soportados, que es 4095.
2005	Se ha producido una anomalía en una llamada a una función del sistema.
2006	Una función incorporada de la interfaz de archivo plano no ha podido acceder al archivo especificado porque lo estaba utilizando éste u otro proceso y no se podía compartir en la modalidad especificada.

Notas de utilización

- Si el archivo no existe, debería especificarse una vía de acceso absoluta para el nombre de archivo y el directorio en el que va a crearse el archivo debe coincidir con un directorio especificado en FFI_PATH.
- Si no se utiliza una vía de acceso absoluta, el archivo se abrirá en el directorio de trabajo actual.
- Si no se ha abierto anteriormente un archivo, DTWF_WRITE() borraré todo el contenido del archivo antes de grabar los datos procedentes de la tabla que se ha pasado al archivo.

Ejemplos

Ejemplo 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_WRITE(myFile, "DELIMITED", ";", myTable)
```

Ejemplo 2:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
%}  
@DTWF_WRITE(myFile, "ASCII TEXT", ";", myTable, "5000")
```

Ejemplo 3:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
%}  
@DTWF_WRITE(myFile, "ASCII TEXT", ";", myTable, "5000", "10", "50")
```

Funciones de registro Web

Un registro Web es un archivo con una clave que mantiene Net.Data para permitirle añadir, recuperar y suprimir entradas con facilidad. Puede crear múltiples registros Web de Net.Data en un único sistema. Cada registro tiene un nombre y puede contener múltiples entradas. Net.Data proporciona funciones para mantener los registros y las entradas que contienen.

- “DTWR_ADDENTRY” en la página 351
- “DTWR_CLEARREG” en la página 353
- “DTWR_CLOSEREG” en la página 355
- “DTWR_CREATEREG” en la página 357
- “DTWR_DELENTY” en la página 359
- “DTWR_DELREG” en la página 361
- “DTWR_LISTREG” en la página 363
- “DTWR_LISTSUB” en la página 365
- “DTWR_OPENREG” en la página 367
- “DTWR_RTVENTRY” en la página 369
- “DTWR_UPDATEENTRY” en la página 371

Restricciones:

- No utilice asteriscos (*) para los parámetros *registro*, *Variableregistro* y *Datosregistro* al utilizar OS/2.
- Cada parámetro que se pasa a una función de Registro Web se limita a un máximo de 2048 caracteres.

DTWR_ADDENTRY

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X			X		X			X

Propósito

Añade una entrada a un registro Web.

Formato

@DTWR_ADDENTRY(registro, Variableregistro, Datosregistro, índice)

@DTWR_ADDENTRY(registro, Variableregistro, Datosregistro)

Parámetros

Tabla 190. Parámetros de DTWR_ADDENTRY

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro para el que se añade la entrada.
serie	<i>Variableregistro</i>	IN	El valor de la parte de serie <i>Variableregistro</i> de la entrada de registro a añadir.
serie	<i>Datosregistro</i>	IN	El valor de la parte de serie <i>Datosregistro</i> de la entrada de registro a añadir.
serie	<i>índice</i>	IN	El valor de la parte de índice de la serie <i>Variableregistro</i> de una entrada indexada a añadir. Este parámetro es opcional. Si se especifica, se añade una entrada indexada al registro especificado.

Códigos de Retorno

Tabla 191. Códigos de retorno de DTWR_ADDENTRY

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Tabla 191. Códigos de retorno de DTWR_ADDENTRY (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
3003	Una función de registro Web no ha podido añadir una entrada al registro especificado porque la entrada especificada ya existe.
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.
3006	Una función incorporada de registro Web no ha podido crear el registro especificado porque no existe una vía de acceso del nombre de registro.
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el petionario no tiene la autorización adecuada para el registro especificado.

Ejemplos

Ejemplo 1:

```
@DTWR_ADDENTRY("Myregistry",
"Jones", "http://Advantis.com/~Jones/webproj")
```

Ejemplo 2:

```
@DTWR_ADDENTRY("URLLIST",
"SMITH", "http://www.ibm.com/software/",
"WORK_URL",")
```

DTWR_CLEARREG

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X			X		X			X

Propósito

Borra entradas de un registro Web.

Formato

@DTWR_CLEARREG(registro)

Parámetros

Tabla 192. Parámetros de DTWR_CLEARREG

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro que ha de borrarse.

Códigos de Retorno

Tabla 193. Códigos de retorno de DTWR_CLEARREG

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.
3006	Una función incorporada de registro Web no ha podido crear el registro especificado porque no existe una vía de acceso del nombre de registro.
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el peticionario no tiene la autorización adecuada para el registro especificado.

Ejemplos

Ejemplo 1:

```
@DTWR_CLEARREG("Myregistry")
```


DTWR_CLOSEREG

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Cierra un registro Web

Formato

@DTWR_CLOSEREG(registro)

Parámetros

Tabla 194. Parámetros de DTWR_CLOSEREG

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro que ha de cerrarse. Restricción: No utilice caracteres especiales como por ejemplo el asterisco (*) o la barra inclinada invertida (\) en nombres de registro Web.

Códigos de Retorno

Tabla 195. Códigos de retorno de DTWR_CLOSEREG

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.

Ejemplos

Ejemplo 1: Cerrar un registro

```
@DTWR_CLOSEREG("/qsys.lib/mylib.lib/myreg.file")
```

DTWR_CREATEREG

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X			X		X			X

Propósito

Crea un registro Web nuevo.

Formato

@DTWR_CREATEREG(registro, seguridad)

@DTWR_CREATEREG(registro)

Parámetros

Tabla 196. Parámetros de DTWR_CREATEREG

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro que ha de crearse. Restricción: No utilice caracteres especiales como por ejemplo el asterisco (*) o la barra inclinada invertida (\) en nombres de registro Web.
serie	<i>seguridad</i>	IN	El tipo de seguridad con la que crear el <i>registro</i> . En sistemas operativos UNIX, la seguridad por omisión es la misma que la del directorio en la que se crea el registro. Especifique la seguridad para los tres grupos de seguridad: usuario, grupo y público. R facilita el permiso de lectura, W proporciona permiso de grabación y X proporciona permiso de ejecución. Por ejemplo, para dar a los tres grupos una autorización completa, especifique *RWX, *RWX, *RWX para este parámetro. Este parámetro es opcional.

Códigos de Retorno

Tabla 197. Códigos de retorno de DTWR_CREATEREG

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.

Tabla 197. Códigos de retorno de DTWR_CREATEREG (continuación)

Código de retorno	Explicación
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
3002	Una función incorporada de registro Web no ha podido suprimir el registro especificado.
3006	Una función incorporada de registro Web no ha podido crear el registro especificado porque no existe una vía de acceso del nombre de registro.
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el peticionario no tiene la autorización adecuada para el registro especificado.
3008	Una función incorporada de registro Web no ha podido crear el registro especificado por razones desconocidas.

Ejemplos

Ejemplo 1:

```
@DTWR_CREATEREG("myRegistry")
```

Ejemplo 2:

```
@DTWR_CREATEREG("URLLIST",  
" *RWX, *RWX, *R")
```

DTWR_DELENTY

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X			X		X			X

Propósito

Suprime una entrada de un registro Web.

Formato

@DTWR_DELENTY(registro, Variableregistro, índice)

@DTWR_DELENTY(registro, Variableregistro)

Parámetros

Tabla 198. Parámetros de DTWR_DELENTY

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro del que se elimina la entrada.
serie	<i>Variableregistro</i>	IN	El valor de la parte de serie <i>Variableregistro</i> de la entrada a eliminar.
serie	<i>índice</i>	IN	El valor de la parte de índice de la serie <i>Variableregistro</i> de una entrada indexada. Este es un parámetro opcional. Si se especifica, la entrada indexada se elimina del registro.

Códigos de Retorno

Tabla 199. Códigos de retorno de DTWF_DELENTY

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.

Tabla 199. Códigos de retorno de DTWF_DELENTY (continuación)

Código de retorno	Explicación
3004	Una función incorporada de registro Web no ha podido eliminar o recuperar una entrada del registro especificado porque la entrada especificada no existe.
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el peticionario no tiene la autorización adecuada para el registro especificado.

Ejemplos

Ejemplo 1:

```
@DTWR_DELENTY("Myregistry",
"Jones")
```

Ejemplo 2:

```
@DTWR_DELENTY("URLLIST",
"SMITH", "WORK_URL")
```

DTWR_DELREG

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X			X		X			X

Propósito

Suprime un registro Web

Formato

@DTWR_DELREG(registro)

Parámetros

Tabla 200. Parámetros de DTWR_DELREG

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro que ha de suprimirse.

Códigos de Retorno

Tabla 201. Códigos de retorno de DTWR_DELREG

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el peticionario no tiene la autorización adecuada para el registro especificado.

Ejemplos

Ejemplo 1:

```
@DTWR_DELREG("Myregistry")
```


DTWR_LISTREG

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X			X		X			X

Propósito

Lista el contenido de un registro Web.

Formato

@DTWR_LISTREG(registro, Tablaregistro)

Parámetros

Tabla 202. Parámetros de DTWR_LISTREG

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro que ha de listarse.
tabla	<i>Tablaregistro</i>	OUT	El nombre de la variable de tabla en la que van a colocarse las entradas de registro.

Códigos de Retorno

Tabla 203. Códigos de retorno de DTWR_LISTREG

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1004	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de tabla de macro Net.Data, pero era un tipo de variable diferente.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.

Tabla 203. Códigos de retorno de DTWR_LISTREG (continuación)

Código de retorno	Explicación
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el peticionario no tiene la autorización adecuada para el registro especificado.

Notas de utilización

DTWR_LISTREG devuelve información sobre las entradas de registro en una variable de tabla OUT que ha pasado el usuario. La variable de tabla se ha definido en la macro de usuario antes de pasarse como parámetro al bloque FUNCTION para la operación de registro LISTREG.

Si el usuario ha definido la variable de tabla utilizando la opción ALL para el número máximo de filas para la tabla, esta operación lista todas las entradas de registro disponibles en la tabla, una por cada fila de tabla. Por otro lado, si el usuario ha especificado un valor X para el número máximo de filas de tabla, si a continuación hay más entradas X en el registro especificado, sólo se listan las primeras entradas X y se envía de nuevo un código de error para indicar que sólo puede hacerse un listado parcial ya que no hay suficientes filas de tabla disponibles para listar entradas adicionales. Se listan todas las entradas de registro en el caso de que el valor X supere el número de entradas disponibles en el registro especificado.

Siempre hay 2 columnas en la tabla. Las cabeceras de columna para la tabla se establecen en REGISTRY_VARIABLE y REGISTRY_DATA.

Ejemplos

Ejemplo 1:

```
%DEFINE RegistryTable =
%TABLE(ALL)

@DTWR_LISTREG("URLLIST", RegistryTable)
```

DTWR_LISTSUB

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X			X					X

Propósito

Lista las entradas de subclave inmediatas de un registro Web.

Formato

@DTWR_LISTSUB(registro, Tablaregistro)

Parámetros

Tabla 204. Parámetro de DTWR_LISTSUB

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro que ha de listarse.
tabla	<i>Tablaregistro</i>	OUT	El nombre de la variable de tabla en la que van a colocarse las entradas de registro.

Códigos de Retorno

Tabla 205. Códigos de retorno de DTWR_LISTSUB

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.

Tabla 205. Códigos de retorno de DTWR_LISTSUB (continuación)

Código de retorno	Explicación
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el peticionario no tiene la autorización adecuada para el registro especificado.

Notas de utilización

1. DTWR_LISTSUB devuelve información sobre las entradas de registro de un parámetro de tabla OUT que ha pasado el usuario. La variable de tabla se ha definido en la macro antes de pasarse como parámetro a la operación de registro LISTSUB.

Si el usuario ha definido la variable de tabla utilizando la opción ALL para el número máximo de filas para la tabla, esta operación lista todas las entradas de registro disponibles en la tabla, una por cada fila de tabla. Por otro lado, si el usuario ha especificado un valor X para el número máximo de filas de tabla, en el caso de que haya más entradas X en el registro especificado, sólo se listan las primeras entradas X y se envía de nuevo un código de error para indicar que sólo puede hacerse un listado parcial ya que no hay suficientes filas de tabla disponibles para listar entradas adicionales. Se listan todas las entradas de registro en el caso de que el valor X supere el número de entradas disponibles en el registro especificado. El número de columnas de la tabla siempre es uno.

La cabecera de columna para la tabla se establece en "REGISTRY_SUBKEY".

2. Esta función sólo es válida en los sistemas operativos que son compatibles con Registros del sistema Windows 95.

Ejemplos

Ejemplo 1:

```
%DEFINE RegistryTable =
%TABLE(ALL)

@DTWR_LISTSUB("URLLIST", RegistryTable)
```

DTWR_OPENREG

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Abre un registro Web.

Formato

@DTWR_OPENREG(registro, confirmar)

@DTWR_OPENREG(registro)

Parámetros

Tabla 206. Parámetros de DTWR_OPENREG

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro que ha de abrirse.
serie	<i>confirmar</i>	IN	Un único símbolo o serie literal que especifica si se abre el registro bajo control de confirmación o no. Los valores posibles son: Y Abrir el registro bajo control de confirmación. N No abrir el registro bajo control de confirmación. El valor por omisión es N

Códigos de Retorno

Tabla 207. Códigos de retorno de DTWR_OPENREG

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.

Tabla 207. Códigos de retorno de DTWR_OPENREG (continuación)

Código de retorno	Explicación
1007	Un parámetro contiene un valor que no es válido.
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el peticionario no tiene la autorización adecuada para el registro especificado.

Ejemplos

Ejemplo 1: Abrir el registro bajo control de confirmación

```
@DTWR_OPENREG("/qsys.lib/mylib.lib/myreg.file", "Y")
```

DTWR_RTENTRY

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X			X		X			X

Propósito

Recupera un valor de serie de registro de un registro Web.

Formato

@DTWR_RTENTRY(registro, Variableregistro, Datosregistro, índice)

@DTWR_RTENTRY(registro, Variableregistro, Datosregistro)

@DTWR_rRTENTRY(registro, Variableregistro, índice)

@DTWR_rRTENTRY(registro, Variableregistro)

Parámetros

Tabla 208. Parámetros de DTWR_RTENTRY

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro con entradas que han de recuperarse.
serie	<i>Variableregistro</i>	IN	El valor de la parte de serie de <i>Variableregistro</i> de la entrada de registro cuya serie de <i>Datosregistro</i> se recupera.
serie	<i>Datosregistro</i>	OUT	Devuelve el valor de la parte de serie de <i>Datosregistro</i> de la entrada de registro que coincida con la <i>Variableregistro</i> .
serie	<i>índice</i>	IN	El valor de la parte de índice de <i>Variableregistro</i> en una entrada indexada cuya serie de <i>Datosregistro</i> se devuelva. Este es un parámetro opcional. Si se especifica, se devuelve la serie de <i>Datosregistro</i> de la entrada indexada.

Códigos de Retorno

Tabla 209. Códigos de retorno de DTWR_RTENTRY

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.

Tabla 209. Códigos de retorno de DTWR_RTVENTRY (continuación)

Código de retorno	Explicación
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.
1007	Un parámetro contiene un valor que no es válido.
3004	Una función incorporada de registro Web no ha podido eliminar o recuperar una entrada del registro especificado porque la entrada especificada no existe.
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el peticionario no tiene la autorización adecuada para el registro especificado.

Ejemplos

Ejemplo 1:

```
%DEFINE RegistryData = ""
@DTWR_RTVENTRY("Myregistry", "Jones", RegistryData)
```

Ejemplo 2:

```
@DTWR_RTVENTRY("URLLIST",
"SMITH", RegistryData, "WORK_URL")
```

Ejemplo 3:

```
@DTWR_rRTVENTRY("Myregistry", "Jones")
```

Ejemplo 4:

```
@DTWR_rRTVENTRY("URLLIST",
"SMITH", "WORK_URL")
```


DTWR_UPDATEENTRY

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
X			X		X			X

Propósito

Actualiza un valor de serie de registro en el registro Web.

Formato

@DTWR_UPDATEENTRY(registro, Variableregistro, Datosnuevos, índice)

@DTWR_UPDATEENTRY(registro, Variableregistro, Datosnuevos)

Parámetros

Tabla 210. Parámetros de DTWR_UPDATEENTRY

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>registro</i>	IN	El nombre del registro con la entrada a actualizar.
serie	<i>Variableregistro</i>	IN	El valor de la parte de serie <i>Variableregistro</i> de la entrada de registro a actualizar.
serie	<i>Datosnuevos</i>	IN	El valor nuevo para la parte de serie <i>Datosregistro</i> de la entrada de registro a actualizar.
serie	<i>índice</i>	IN	El valor de la parte de índice de la serie <i>Variableregistro</i> de una entrada indexada a actualizar. Este es un parámetro opcional. Si se especifica, la entrada indexada se actualiza.

Códigos de Retorno

Tabla 211. Códigos de retorno de DTWR_UPDATEENTRY

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1002	Un parámetro de entrada que contenía un valor de serie que consistía en el carácter de terminación nulo.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Tabla 211. Códigos de retorno de DTWR_UPDATEENTRY (continuación)

Código de retorno	Explicación
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
3003	Una función de registro Web no ha podido añadir una entrada al registro especificado porque la entrada especificada ya existe.
3004	Una función incorporada de registro Web no ha podido eliminar o recuperar una entrada del registro especificado porque la entrada especificada no existe.
3005	Una función incorporada de registro Web no ha podido utilizar el registro especificado porque no se puede encontrar.
3007	Una función incorporada de registro Web no ha podido completar la operación especificada porque el peticionario no tiene la autorización adecuada para el registro especificado.

Notas de utilización

El nombre de entrada de registro que se corresponde con el valor no puede cambiarse.

Ejemplos

Ejemplo 1:

```
@DTWR_UPDATEENTRY("Myregistry", "Jones",
"http://advantis.com/~Jones/personal")
```

Ejemplo 2:

```
@DTWR_UPDATEENTRY("URLLIST",
"SMITH", "http://www.ibm.com/software/personal",
"WORK_URL")
```

Funciones de macro permanentes

Las funciones de macros permanentes dan soporte al proceso de transacciones en Net.Data ayudándole a definir los bloques de macro que son permanentes dentro de una única transacción. Utilice estas funciones para definir el principio y el final de una transacción, los bloques de HTML que son permanentes a través de dicha transacción, el ámbito de las variables dentro de la transacción y si se han de confirmar o retrotraer los cambios dentro de la misma.

- “DTW_ACCEPT” en la página 374
- “DTW_COMMIT” en la página 376
- “DTW_ROLLBACK” en la página 377
- “DTW_RTVHANDLE” en la página 378
- “DTW_STATIC” en la página 380
- “DTW_TERMINATE” en la página 382

DTW_ACCEPT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Define el manejador de transacciones utilizado para invocar una macro permanente.

Formato

@DTW_ACCEPT(manejador, tiempoesperaexcedido)

@DTW_ACCEPT(manejador)

Parámetros

Tabla 212. Parámetros de DTW_ACCEPT

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>manejador</i>	IN	Variable o serie literal que especifica un manejador de transacciones a utilizar en los URL para las invocaciones de macro sucesivas en esta transacción permanente.
entero	<i>tiempoesperaexcedido</i>	IN	Variable o serie literal que especifica la cantidad de tiempo en segundos que ha de esperar una respuesta el trabajo que da servicio a este puerto. Este valor altera temporalmente cualquier valor de tiempo de espera excedido especificado en la función DTW_STATIC().

Códigos de Retorno

Tabla 213. Códigos de retorno de DTW_ACCEPT

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.

Tabla 213. Códigos de retorno de DTW_ACCEPT (continuación)

Código de retorno	Explicación
1007	Un parámetro contiene un valor que no es válido.
8200	La permanencia de macro no está habilitada.
8201	Se ha llamado a una función incorporada permanente fuera de secuencia.

Notas de utilización

1. Net.Data requiere que el manejador de transacciones se incluya en el URL que invoca la macro como respuesta procedente del navegador de la Web. Cuando llega una petición al servidor de la Web, el servidor utiliza el manejador de transacciones para dirigir la petición al proceso de CGI que está procesando la transacción.
El manejador de transacciones debe llamarse al principio de cada bloque HTML de macro hasta el último bloque lógico, que contiene una llamada a DTW_TERMINATE(). Si no se encuentra la llamada a DTW_ACCEPT() o a DTW_TERMINATE() antes de que se produzca la salida de texto al navegador, se produce un error de Net.Data.
2. Puede especificar un valor de tiempo de espera excedido para esta página que altere temporalmente el valor de tiempo de espera excedido especificado en la función @DTW_STATIC(). El servidor de la Web espera el tiempo especificado (en segundos) a que el usuario responda a esta petición.
3. Si esta función se llama cuando la macro no está en estado permanente, se produce un error de Net.Data.
4. Los URL que contienen el manejador de transacciones pueden codificarse como acciones en pulsadores de formato o como enlaces de hipertexto en la página que se presenta al navegador.

Ejemplos

Ejemplo 1:

```
%DEFINE handle = ""
@DTW_RTVHANLDE(handle)

%HTML(REPORT){
@DTW_ACCEPT(handle)
...
%}
```

DTW_COMMIT

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Convierte en permanentes los cambios pendientes efectuados en los recursos bajo control de confirmación desde el último límite de confirmación y establece un nuevo límite de confirmación.

Formato

@DTW_COMMIT()

Parámetros

Ninguno.

Códigos de Retorno

Tabla 214. Códigos de retorno de DTW_COMMIT

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Ejemplos

Ejemplo 1: Especifica una confirmación

```
@DTW_COMMIT()  
%HTML(report){  
%}
```

DTW_ROLLBACK

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Vuelve a establecer el último límite de confirmación como límite de confirmación actual. Se restituyen todos los cambios en los recursos bajo control de confirmación para el proceso bajo el que se está ejecutando Net.Data efectuados desde el último límite de confirmación.

Formato

@DTW_ROLLBACK()

Parámetros

Ninguno.

Códigos de Retorno

Tabla 215. Códigos de retorno de DTW_ROLLBACK

Código de retorno	Explicación
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.

Ejemplos

Ejemplo 1: Especifica una retrotracción

```
@DTW_ROLLBACK()  
%HTML(report){  
%}
```

DTW_RTVHANDLE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Genera y devuelve un manejador de transacciones que es exclusivo para esta macro entre diferentes invocaciones y que se calcula basándose en una combinación de información de paso, indicación de la hora y usuario actual.

Formato

@DTW_RTVHANDLE(manejador)

Parámetros

Tabla 216. Parámetros de DTW_RTVHANDLE

Tipo de datos	Parámetro	Utilización	Descripción
serie	<i>manejador</i>	OUT	Variable que contiene un manejador de transacciones para la macro permanente actual.

Códigos de Retorno

Tabla 217. Códigos de retorno de DTW_RTVHANDLE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1006	Se ha pasado una serie literal en una llamada de función para un parámetro que debería haber sido un parámetro de salida.

Notas de utilización

El manejador de transacciones puede utilizarse para asegurarse de que los URL especificados como parte de una transacción permanente son exclusivos para el servidor HTTP y pueden identificarse de forma segura como peticiones válidas.

Ejemplos

Ejemplo 1: Define la variable `handle` utilizada para recuperar el manejador de transacciones

```
%DEFINE  
handle = ""  
@DTW_RTVHANLDE(handle)
```

DTW_STATIC

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Indica que toda la macro es permanente.

Formato

@DTW_STATIC(tiempoesperaexcedido)

@DTW_STATIC()

Parámetros

Tabla 218. Parámetros de DTW_STATIC

Tipo de datos	Parámetro	Utilización	Descripción
entero	<i>tiempoesperaexcedido</i>	IN	Variable o serie literal que especifica la cantidad de tiempo, en segundos, que ha de esperar una respuesta el proceso que maneja esta transacción.

Códigos de Retorno

Tabla 219. Códigos de retorno de DTW_STATIC

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1001	Un parámetro de entrada contenía un valor NULL.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
1005	Se ha pasado un parámetro en una llamada de función, que debería haber sido una variable de serie, pero era un tipo de variable diferente.
1007	Un parámetro contiene un valor que no es válido.
8202	No se ha podido habilitar la permanencia.

Notas de utilización

1. DTW_STATIC debe ser la primera sentencia de la macro. Todas las variables definidas en la macro después de esta llamada de función serán permanentes entre múltiples invocaciones de macro a menos que se especifique lo contrario y hasta que se llame a DTW_TERMINATE() o finalice el proceso.

2. Puede especificarse un valor de tiempo de espera excedido, en segundos, en la llamada de función para indicar la cantidad de tiempo que espera una respuesta del navegador el proceso bajo el que se está ejecutando Net.Data. Si expira el valor de tiempo de espera excedido, el proceso finaliza y se retrotraen todos los cambios en los recursos bajo control de confirmación desde el último límite de confirmación.
3. Si se especifica un valor de tiempo de espera excedido en una llamada de @DTW_ACCEPT() posterior, Net.Data sobregraba este valor con el valor de la llamada posterior. Si no se especifica un valor de tiempo de espera excedido en esta llamada o en una llamada de @DTW_ACCEPT() posterior, se utiliza el valor de tiempo de espera excedido por omisión del servidor de la Web.

Ejemplos

Ejemplo 1: Llamada a DTW_STATIC() que especifica un valor de tiempo de espera excedido de 60 segundos.

```
@DTW_STATIC("60")
```

DTW_TERMINATE

AIX	HP-UX	Linux	OS/2	OS/390	OS/400	PTX	SUN	Win NT
					X			

Propósito

Finaliza una transacción permanente. Se hacen permanentes todos los cambios en los recursos bajo control de confirmación desde el último límite de confirmación.

Formato

@DTW_TERMINATE()

Parámetros

Ninguno

Códigos de Retorno

Tabla 220. Códigos de retorno de DTW_TERMINATE

Código de retorno	Explicación
-1001	El servidor no ha podido procesar una petición de Net.Data para asignar memoria.
1003	Se ha pasado un número incorrecto de parámetros en una llamada de función.
8200	La permanencia de macro no está habilitada.
8201	Se ha llamado a una función incorporada permanente fuera de secuencia.

Notas de utilización

1. La función DTW_TERMINATE se llama al principio del último bloque HTML lógico de la transacción permanente antes de que se envíe texto alguno como salida al navegador. Si, dentro del bloque, aparece alguna salida de texto antes de la función, se producirá un error de Net.Data. Tenga en cuenta que es posible que haya más de un último bloque HTML lógico en función del modo en que se ha grabado la aplicación.
2. Si esta función se llama cuando la macro no está en estado permanente, se producirá un error de Net.Data.

Ejemplos

Ejemplo 1: Finaliza la transacción permanente

```
%HTML(QUIT){  
@DTW_TERMINATE()  
...  
%}
```

Apéndice A. Biblioteca técnica de Net.Data

La biblioteca técnica de Net.Data está disponible en el sitio Web de Net.Data Web en

<http://www.ibm.com/software/data/net.data/library.html>

Documento	Descripción
<ul style="list-style-type: none">• <i>Net.Data Administration and Programming Guide for OS/390</i>• <i>Guía de administración y programación de Net.Data para OS/2, Windows NT y UNIX</i>• <i>Net.Data Administration and Programming Guide for OS/400</i>	Contiene información de conceptos y tareas sobre cómo instalar, configurar e invocar Net.Data. También describe cómo escribir macros Net.Data, utilizar técnicas de rendimiento de Net.Data, utilizar entornos de lenguaje de Net.Data, gestionar conexiones y utilizar registros cronológicos y rastreos de Net.Data para resolver los problemas y ajustar el rendimiento.
<i>Manual de consulta de Net.Data</i>	Describe el lenguaje de macros de Net.Data, las variables y funciones incorporadas.
<i>Net.Data Language Environment Interface Reference</i>	Describe la interfaz de entorno de lenguaje de Net.Data.
<i>Consulta de códigos y mensajes de Net.Data</i>	Lista mensajes de error y códigos de retorno de Net.Data.

Apéndice B. Características desaprobadas

Las características siguientes siguen estando soportadas, pero no son recomendables. Si tiene macros que contienen estas construcciones de lenguaje, es recomendable que se utilicen las alternativas sugeridas.

EXEC_SQL

Una construcción de lenguaje de DB2 WWW Connection que llama a un bloque de SQL. Le recomendamos que en su lugar llame sentencias de SQL como funciones. Consulte el apartado “Bloque FUNCTION” en la página 21 para obtener más información.

HTML_INPUT

Una construcción de lenguaje de DB2 WWW Connection que es igual que un bloque de HTML denominado INPUT. Consulte el apartado “Bloque HTML” en la página 34 para obtener más información.

HTML_REPORT

Una construcción de lenguaje de DB2 WWW Connection que es igual que un bloque de HTML denominado REPORT. Consulte el apartado “Bloque HTML” en la página 34 para obtener más información.

INCLUDE_URL

Sentencia utilizada para leer e incorporar otro archivo en la salida generada por Net.Data. En el momento de la ejecución, el contenido completo del archivo incluido sustituirá la sentencia INCLUDE_URL. El archivo especificado puede existir en un servidor local o remoto.

Aunque no hay una alternativa recomendada a la inclusión del contenido de un URL remoto, puede utilizarse la sentencia INCLUDE para incluir archivos locales, incluyendo macros. No es recomendable incluir URL remotos ya que no hay forma de interrumpir ninguno de los numerosos errores que pueden producirse al acceder a un sitio remoto.

SQL

Una construcción de lenguaje de DB2 WWW Connection que equivale a una función que se llama con FUNCTION(DTW_SQL) en Net.Data.

Puede contener sentencias de SQL_REPORT y SQL_MESSAGE, que también proceden de DB2 WWW Connection. DB2 WWW Connection no da soporte a los bloques %SQL con nombre.

Ejemplos:

Ejemplo 1: Una macro de DB2 WWW Connection

```
%SQL{
UPDATE $(dbtbl) SET URL='$(URL)' WHERE ID=$(ID)
%SQL_MESSAGE{
100: "<b>The selected URL no longer exists in the table</b>." :
continue
%}
%}

%HTML_INPUT{
<html>
...
%EXEC_SQL
</html>
%}

%HTML_REPORT{
<html>
...
</html>
%}
```

Ejemplo 1: Una macro Net.Data equivalente

```
%FUNCTION(DTW_SQL) URLquery(){
UPDATE $(dbtbl) SET URL='$(URL)' WHERE ID=$(ID)
%MESSAGE{
100: "<b>The selected URL no longer exists in the table</b>." :
continue
%}
%}

%HTML(INPUT){
<html>
...
@URLquery
</html>
%}

%HTML(REPORT){
<html>
...
</html>
%}
```

SQL_MESSAGE

Una construcción de lenguaje de DB2 WWW Connection que equivale a la sentencia MESSAGE de Net.Data. Consulte el apartado “Bloque MESSAGE” en la página 54 para ver un ejemplo.

SQL_REPORT

Una construcción de lenguaje de DB2 WWW Connection que equivale a la sentencia REPORT de Net.Data. Consulte el apartado “Bloque REPORT” en la página 59 para ver un ejemplo.

SQL_CODE

Una construcción de lenguaje de DB2 WWW Connection que equivale a “RETURN_CODE” en la página 142.

Apéndice C. Referencia del sistema operativo de Net.Data

No todas las características de Net.Data están soportadas en cada uno de los sistemas operativos. Esta sección muestra las características que están soportadas para su sistema operativo. Una **X** indica que la característica está soportada.

Tabla 221. Entornos de lenguajes de Net.Data

Entorno de lenguajes	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
Llamada directa						X			
Interfaz de archivo plano	X	X	X	X	X	X	X	X	X
IMS Web	X			X				X	X
Applets Java	X	X	X	X	X	X	X	X	X
Aplicaciones Java	X			X		X		X	X
ODBC	X	X		X	X			X	X
Oracle	X								X
Perl	X	X	X	X	X		X	X	X
REXX	X		X	X	X	X		X	X
SQL	X	X	X	X	X	X	X	X	X
Sistema	X	X	X	X	X	X	X	X	X
Registro Web	X			X		X			X

Tabla 222. Variables de configuración de Net.Data

Variable de configuración	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
DB2INSTANCE	X	X	X	X			X	X	X
DB2MSG5					X				
DB2PLAN					X				
DB2SSID					X				
DefaultDBCp					X				
DSNAOINI					X				
DTW_CACHE_HOST	X								X
DTW_CACHE_MACRO					X				
DTW_CACHE_MANAGEMENT_INTERVAL					X				
DTW_CACHE_PAGE					X				
DTW_CACHE_PORT	X								X
DTW_CM_PORT	X	X		X					X
DTW_DEFAULT_ERROR_MESSAGE	X	X	X	X	X	X	X	X	X
DTW_DEFAULT_MACRO						X			
DTW_DIRECT_REQUEST	X	X		X	X			X	X
DTW_DO_NOT_CACHE_MACRO					X				
DTW_ERROR_LOG_DIR					X				
DTW_ERROR_LOG_LEVEL					X				
DTW_INST_DIR	X		X	X			X	X	X
DTW_LOB_LIFETIME					X				
DTW_LOG_DIR	X	X	X	X			X	X	X
DTW_LOG_LEVEL	X	X	X	X			X	X	X
DTW_MBMODE	X	X	X	X	X		X	X	X
DTW_REMOVE_WS	X	X	X	X	X	X	X	X	X

Tabla 222. Variables de configuración de Net.Data (continuación)

Variable de configuración	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
DTW_SHOWSQL	X	X	X	X	X	X	X	X	X
DTW_SMTP_CHARSET						X			
DTW_SMTP_SERVER	X	X	X	X		X	X	X	X
DTW_SQL_ISOLATION						X			
DTW_SQL_NAMING_MODE						X			
DTW_TRACE_LOG_DIR					X				
DTW_TRACE_LOG_LEVEL					X				
DTW_UNICODE	X	X	X	X			X	X	X
DTW_VARIABLE_SCOPE	X	X	X	X			X	X	X

Tabla 223. Variables de Net.Data

Variable	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
ALIGN	X	X	X	X	X	X	X	X	X
DATABASE	X	X	X	X		X	X	X	X
DB_CASE	X	X	X	X	X	X	X	X	X
DB2PLAN					X				
DB2SSID					X				
DTW_APPLET_ALTTEXT	X	X	X	X	X	X	X	X	X
DTW_CURRENT_FILENAME	X	X	X	X	X	X	X	X	X
DTW_CURRENT_LAST_MODIFIED	X	X	X	X	X	X	X	X	X
DTW_DEFAULT_MESSAGE	X	X	X	X	X	X	X	X	X
DTW_DEFAULT_REPORT	X	X	X	X	X	X	X	X	X
DTW_EDIT_CODES						X			
DTW_HTML_TABLE	X	X	X	X	X	X	X	X	X
DTW_LOG_LEVEL	X	X	X	X			X	X	X
DTW_MACRO_FILENAME	X	X	X	X	X	X	X	X	X
DTW_MACRO_LAST_MODIFIED	X	X	X	X	X	X	X	X	X
DTW_MBMODE	X	X	X	X	X		X	X	X
DTW_MP_PATH	X	X	X	X	X	X	X	X	X
DTW_MP_VERSION	X	X	X	X	X	X	X	X	X
DTW_PAD_PGM_PARMS						X			
DTW_PRINT_HEADER	X	X	X	X	X	X	X	X	X
DTW_REMOVE_WS	X	X	X	X	X	X	X	X	X
DTW_SAVE_TABLE_IN	X	X	X	X	X	X	X	X	X
DTW_SET_TOTAL_ROWS	X	X	X	X	X	X	X	X	X
LOCATION					X				

Tabla 223. Variables de Net.Data (continuación)

[illegible]

Tabla 224. Funciones de Net.Data

Función	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
DTW_ACCEPT						X			
DTW_ADD	X	X	X	X	X	X	X	X	X
DTW_ADDQUOTE	X	X	X	X	X	X	X	X	X
DTW_ASSIGN	X	X	X	X	X	X	X	X	X
DTW_CACHE_PAGE	X								X
DTW_CHARTOHEX	X	X	X	X	X	X	X	X	X
DTW_COMMIT						X			
DTW_CONCAT	X	X	X	X	X	X	X	X	X
DTW_DATE	X	X	X	X	X	X	X	X	X
DTW_DELSTR	X	X	X	X	X	X	X	X	X
DTW_DELWORD	X	X	X	X	X	X	X	X	X
DTW_DIVIDE	X	X	X	X	X	X	X	X	X
DTW_DIVREM	X	X	X	X	X	X	X	X	X
DTW_EXIT	X	X	X	X	X	X	X	X	X
DTW_FORMAT	X	X	X	X	X	X	X	X	X
DTW_GETCOOKIE	X	X	X	X	X	X	X	X	X
DTW_GETENV	X	X	X	X	X	X	X	X	X
DTW_GETINIDATA	X	X	X	X	X	X	X	X	X
DTW_HEXTOCHAR	X	X	X	X	X	X	X	X	X
DTW_HTMLLENCODE	X	X	X	X	X	X	X	X	X
DTW_INSERT	X	X	X	X	X	X	X	X	X
DTW_INTDIV	X	X	X	X	X	X	X	X	X
DTW_LASTPOS	X	X	X	X	X	X	X	X	X
DTW_LENGTH	X	X	X	X	X	X	X	X	X

Tabla 224. Funciones de Net.Data (continuación)

Función	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
DTW_TB_DUMPV	X	X	X	X	X	X	X	X	X
DTW_TB_GETN	X	X	X	X	X	X	X	X	X
DTW_TB_GETV	X	X	X	X	X	X	X	X	X
DTW_TB_HTMLENCODE	X	X	X	X	X	X	X	X	X
DTW_TB_INPUT_CHECKBOX	X	X	X	X	X	X	X	X	X
DTW_TB_INPUT_RADIO	X	X	X	X	X	X	X	X	X
DTW_TB_INPUT_TEXT	X	X	X	X	X	X	X	X	X
DTW_TB_INSERTCOL	X	X		X	X	X		X	X
DTW_TB_INSERTROW	X	X		X	X	X		X	X
DTW_TB_LIST	X	X	X	X	X	X	X	X	X
DTW_TB_MAXROWS						X			
DTW_TB_QUERYCOLNONJ	X	X		X	X	X		X	X
DTW_TB_ROWS	X	X	X	X	X	X	X	X	X
DTW_TB_SELECT	X	X	X	X	X	X	X	X	X
DTW_TB_SETCOLS	X	X		X	X	X		X	X
DTW_TB_SETN	X	X		X	X	X		X	X
DTW_TB_SETV	X	X		X	X	X		X	X
DTW_TB_TABLE	X	X	X	X	X	X	X	X	X
DTW_TB_TEXTAREA	X	X	X	X	X	X	X	X	X
DTW_TERMINATE						X			
DTW_TIME	X	X	X	X	X	X	X	X	X
DTW_TRANSLATE	X	X	X	X	X	X	X	X	X
DTW_UPPERCASE	X	X	X	X	X	X	X	X	X
DTW_URLESCSEQ	X	X	X	X	X	X	X	X	X

Tabla 224. Funciones de Net.Data (continuación)

Función	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
DTW_USE_DB2_PREPARE_CACHE	X	X	X	X			X	X	X
DTW_WORD	X	X	X	X	X	X	X	X	X
DTW_WORDINDEX	X	X	X	X	X	X	X	X	X
DTW_WORDLENGTH	X	X	X	X	X	X	X	X	X
DTW_WORDPOS	X	X	X	X	X	X	X	X	X
DTW_WORDS	X	X	X	X	X	X	X	X	X
DTWF_APPEND	X	X	X	X	X	X	X	X	X
DTWF_CLOSE	X	X	X	X	X	X	X	X	X
DTWF_DELETE	X	X	X	X	X	X	X	X	X
DTWF_INSERT	X	X	X	X	X	X	X	X	X
DTWF_OPEN	X	X	X	X	X	X	X	X	X
DTWF_READ	X	X	X	X	X	X	X	X	X
DTWF_REMOVE	X	X	X	X	X	X	X	X	X
DTWF_SEARCH	X	X	X	X	X	X	X	X	X
DTWF_UPDATE	X	X	X	X	X	X	X	X	X
DTWF_WRITE	X	X	X	X	X	X	X	X	X
DTWR_ADDENTRY	X			X		X			X
DTWR_CLEARREG	X			X		X			X
DTWR_CLOSEREG						X			
DTWR_CREATEREG	X			X		X			X
DTWR_DELENTY	X			X		X			X
DTWR_DELREG	X			X		X			X
DTWR_LISTREG	X			X		X			X
DTWR_LISTSUB	X			X					X

Tabla 224. Funciones de Net.Data (continuación)

Función	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
DTWR_OPENREG						X			
DTWR_RTENTRY	X			X		X			X
DTWR_UPDATEENTRY	X			X		X			X

Tabla 225. Interfaces de Net.Data

Tipo de interfaz	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
FastCGI	X							X	
CGI	X	X	X	X	X	X	X	X	X
Java Beans									X
API de Internet Connection (ICAPI)	X			X	X				X
API de servidor de Internet (ISAPI)									X
Live Connection	X			X				X	X
Servidor de Lotus Domino Go Web (GWAPI)	X			X	X				X
API de Netscape (NSAPI)								X	X
Servlets	X				X			X	X

Tabla 226. Herramientas de Net.Data

Herramienta	AIX	HP	LINUX	OS/2	OS/390	OS/400	PTX	SUN	Win NT
Herramienta de administración	X			X					X
Plug-ins de fusión de NetObjects									X
Asistentes	X			X				X	X

Avisos

Es posible que IBM no ofrezca en todos los países los productos, servicios o características descritos en este documento. Consulte al representante local de IBM para obtener información sobre los productos y servicios disponibles actualmente en su zona. Las referencias hechas a productos, programas o servicios de IBM no pretenden afirmar ni implicar que sólo puedan utilizarse esos productos, programas o servicios de IBM. En su lugar puede utilizarse cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario la evaluación y verificación del funcionamiento de cualquier producto, programa o servicio no de IBM.

IBM puede tener patentes o solicitudes de patente pendientes que cubran temas tratados en este documento. La adquisición de este documento no confiere ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
Estados Unidos

Para consultas sobre licencias relacionadas con información de doble byte (DBCS), póngase en contacto con el Departamento de propiedad intelectual de IBM de su país o envíe sus consultas, por escrito, a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún país donde las disposiciones en él expuestas sean incompatibles con la legislación local:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN “TAL CUAL” SIN GARANTÍAS DE NINGÚN TIPO, YA SEAN EXPLÍCITAS O IMPLÍCITAS, INCLUYENDO, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UNA FINALIDAD DETERMINADA. Algunas legislaciones no contemplan la exclusión de garantías implícitas o explícitas en determinadas transacciones, por lo que es posible que esta declaración no sea de aplicación en su caso.

Esta publicación puede contener imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán en nuevas ediciones de la publicación. IBM se reserva el derecho a realizar, si lo considera oportuno, cualquier modificación en los productos o programas que se describen en el presente manual.

Las referencias que se hacen en esta información a sitios Web no IBM se facilitan únicamente por cortesía y en ningún modo deben interpretarse como un respaldo de dichos sitios Web. Los materiales de los sitios Web mencionados no son parte de los materiales para este producto IBM y el usuario los utiliza por su cuenta y riesgo.

IBM puede utilizar o distribuir la información que se le remita en la forma que considere más adecuada sin incurrir por ello en ninguna obligación con el remitente.

Los usuarios con licencia de este programa que deseen obtener información sobre el mismo para permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) el uso mutuo de la información intercambiada, deberán ponerse en contacto con:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADÁ

Esta información puede estar disponible sujeta a los términos y condiciones apropiados incluyendo, en algunos casos, el pago de una tarifa.

IBM proporciona el programa bajo licencia descrito en esta publicación, y todo el material con licencia disponible para el mismo, bajo los términos del Contrato de Cliente de IBM, del Contrato internacional de programas bajo licencia de IBM o de cualquier contrato equivalente entre ambas partes.

Los datos de rendimiento aquí contenidos se han determinado en un entorno controlado. Por tanto, es posible que los resultados que se obtengan en otros entornos operativos puedan variar significativamente. Es posible que se hayan efectuado algunas mediciones en sistemas en su fase de desarrollo y no se garantiza que dichas mediciones serán iguales en los sistemas que estén disponibles normalmente. Además, es posible que algunas mediciones se hayan estimado a través de una extrapolación. Los resultados reales pueden ser diferentes. Los usuarios de este documento deberían verificar los datos de aplicación para su entorno específico.

La información acerca de productos no de IBM se ha obtenido de los suministradores de dichos productos, de anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha sometido a prueba dichos productos y no puede confirmar la precisión de su funcionamiento, su compatibilidad ni ninguna otra afirmación relacionada con productos no de IBM. Las preguntas sobre las posibilidades de los productos no de IBM deberán dirigirse a los suministradores de dichos productos.

Todas las declaraciones al respecto de las intenciones futuras de IBM pueden cambiarse o retirarse sin previo aviso y únicamente representan objetivos.

Esta información puede contener ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlas lo mejor posible, los ejemplos incluyen los nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con los nombres y direcciones que utilice una compañía real es mera coincidencia.

LICENCIA DE COPYRIGHT:

Esta información puede contener programas de aplicación de ejemplo en lenguaje fuente que ilustran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin efectuar pago alguno a IBM, con fines de desarrollo, uso, comercialización o distribución de programas de aplicación que se ajusten a la interfaz de programas de aplicación de la plataforma operativa para la que se han escrito los programas de ejemplo. Estos ejemplos no han sido probados exhaustivamente bajo todas las condiciones. Por consiguiente, IBM no puede garantizar ni confirmar la fiabilidad, utilidad o funcionamiento de estos programas.

Cualquier copia o parte de estos programas de ejemplo o trabajo derivado de los mismos debe incluir un aviso de copyright como el siguiente:

© (nombre de su empresa) (año). Parte de este código proviene de Programas de ejemplo de IBM Corp.. © Copyright IBM Corp. _entre el año o años_. Reservados todos los derechos.

Marcas registradas

Los siguientes términos, que pueden estar indicados por medio de un asterisco (*), son marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países.

ACF/VTAM	IBM
AISPO	IMS
AIX	IMS/ESA
AIX/6000	LAN DistanceMVS
AIXwindows	MVS/ESA
AnyNet	MVS/XA
APPN	Net.Data
AS/400	OS/2
BookManager	OS/390
CICS	OS/400
C Set++	PowerPC
C/370	QBIC
DATABASE 2	QMF
DataHub	RACF
DataJoiner	RISC System/6000
DataPropagator	RS/6000
DataRefresher	S/370
DB2	SP
DB2 Connect	SQL/DS
DB2 Extenders	SQL/400
DB2 OLAP Server	System/370
DB2 Universal Database	System/390
Distributed Relational Database Architecture	SystemView
DRDA	VisualAge
eNetwork	VM/ESA
Extended Services	VSE/ESA
FFST	VTAM
First Failure Support Technology	WebExplorer
	WIN-OS/2

Los términos siguientes son marcas comerciales o marcas registradas de otras empresas:

Microsoft, Windows y Windows NT son marcas comerciales o marcas registradas de Microsoft Corporation.

Java o todas las marcas registradas y logotipos basados en Java y Solaris son marcas registradas de Sun Microsystems, Inc. en Estados Unidos y/o en otros países.

Tivoli y NetView son marcas registradas de Tivoli Systems, Inc. en Estados Unidos y/o en otros países.

UNIX es una marca registrada en Estados Unidos y/o en otros países cuya licencia se obtiene exclusivamente a través de X/Open Company Limited.

Otros nombres de empresas, productos y servicios, que pueden estar indicadas con un doble asterisco (**), pueden ser marcas registradas o de servicio de otras compañías.

Glosario

antememoria. Parte de la memoria o del espacio de disco que contiene los datos a los que se ha accedido recientemente, diseñados para acelerar los accesos sucesivos a los mismos datos. La antememoria se utiliza a menudo para que contenga una copia local de los datos que se utilizan con frecuencia y a la que puede accederse a través de una red.

API. Vea *interfaz de programas de aplicación*.

applet. Un programa de Java incluido en una página HTML. Los applets funcionan con navegadores habilitados por Java, por ejemplo, Netscape Navigator y se cargan cuando se procesa la página HTML.

base de datos. Conjunto de tablas, o conjunto de espacios de tablas y de espacios de índices.

BLOB. Objeto binario grande.

CGI. Interfaz de pasarela común.

cliette. Proceso de ejecución larga de Net.Data Live Connection que sirve peticiones procedentes del servidor Web. El Gestor de conexiones planifica procesos cliette para servir estas peticiones.

CLOB. Objeto de carácter grande.

Connection Manager. Un archivo ejecutable, dtwcm, de Net.Data que se necesita para dar soporte a Live Connection.

control de confirmación. El establecimiento de un límite dentro del proceso en el que se está ejecutando Net.Data en el que las operaciones sobre los recursos forman parte de una unidad de trabajo.

cookie. Paquete de información que envía un servidor HTTP a un navegador Web y después devuelve el navegador cada vez que accede a dicho servidor. Los cookies pueden contener cualquier información arbitraria que elija el

servidor y se utilizan para mantener el estado entre transacciones HTTP que de otro modo no tienen estado. *Free Online Dictionary of Computing*

cortafuegos. Sistema con software que protege una red interna frente al acceso externo no autorizado.

DATALINK. Tipo de datos de DB2 que permite consultas lógicas desde la base de datos a un archivo almacenado fuera de la base de datos.

DBCLOB. Objeto de carácter de doble byte grande.

DBMS. Sistema de gestión de base de datos.

directorio de trabajo actual. El directorio por omisión de un proceso desde el que se resuelven todos los nombres de vía de acceso relativos.

entorno de lenguaje. Módulo que facilita acceso desde una macro Net.Data a una fuente de datos externa, como por ejemplo, DB2 o un lenguaje de programación, como por ejemplo Perl.

Gestor de antememoria. El programa que gestiona una antememoria para una máquina. Puede gestionar varias antememorias.

GWAPI. API de servidor Go Web.

HTML. Lenguaje de marcación de hipertexto.

HTTP. Protocolo de transferencia de hipertexto.

interfaz de archivo plano. Conjunto de funciones incorporadas de Net.Data que le permiten leer y escribir datos procedentes de archivos de texto plano.

Interfaz de pasarela común (CGI). Forma estándar en la que un servidor Web pasa el control a un programa de aplicación y recibe datos del mismo.

interfaz de programas de aplicación. Interfaz que proporciona el sistema operativo o un programa bajo licencia que puede solicitarse por separado que permite a un programa de aplicación escrito en un lenguaje de alto nivel utilizar funciones o datos específicos del sistema operativo o programa bajo licencia. Net.Data da soporte a las siguientes API de servidor de la Web de propietario para el rendimiento mejorado a través de procesos de CGI: GWAPI, ISAPI y NSAPI.

Internet. Red de sistema TCP/IP pública internacional.

Intranet. Red TCP/IP en la parte interna de un cortafuegos de empresa.

ISAPI. API de servidor de Internet de Microsoft.

Java. Lenguaje de programación orientada a objeto independiente del sistema operativo que resulta especialmente útil para las aplicaciones de Internet.

lenguaje de marcación de hipertexto. Lenguaje de código utilizado para escribir documentos Web.

Live Connection. Componente de Net.Data que consta de un Connection Manager y de varios cliettes. Live Connection gestiona la reutilización de la base de datos y las conexiones de máquina virtual Java.

LOB. Objeto grande.

middleware. Software que media entre un programa de aplicación y una red. Gestiona la interacción entre un programa de aplicación de cliente y un servidor a través de la red.

nombre de vía de acceso. Indica al sistema el modo de localizar un objeto. El nombre de vía de acceso se expresa como una secuencia de nombres de directorio seguidos por el nombre del objeto. Los directorios individuales y el nombre de objeto están separados por un carácter de barra inclinada (/) o de barra inclinada invertida (\).

nombre de vía de acceso relativa. Nombre de vía de acceso que no comienza en el nivel superior, o directorio "root". El sistema asume que el nombre de vía de acceso comienza en el directorio de trabajo actual del proceso.

NSAPI. API de Netscape.

nulo. Valor especial que indica la ausencia de información.

Perl. Lenguaje de programación interpretado.

permanencia. El estado de conservar un valor asignado para toda una transacción, en la que una transacción abarca varias invocaciones de Net.Data. Sólo las variables pueden ser permanentes. Además, las operaciones sobre los recursos afectados por el control de confirmación permanecen activas hasta que se efectúa una acción de confirmación o retroacción, o hasta el momento en que finaliza la transacción.

poner en antememoria. El proceso de almacenar resultados que se utilizan con frecuencia procedentes de una petición al servidor Web localmente para una rápida recuperación, hasta que llega el momento de renovar la información.

Protocolo de control de transmisiones /

Protocolo Internet. Conjunto de protocolos de comunicaciones que dan soporte a funciones de conectividad de igual a igual para redes de área amplia y locales.

protocolo de transferencia de hipertexto. El protocolo de comunicaciones utilizado entre un navegador y un servidor Web.

puerto. Número de 16 bits utilizado para la comunicación entre TCP/IP y una aplicación o protocolo de nivel superior.

registro. Depósito en el que las series se pueden almacenar y recuperar.

Servidor Domino Go Web. El servidor Web ofrecido por Lotus Corp. e IBM, que ofrece conexiones regulares y protegidas. GWAPI es la interfaz que se proporciona con este servidor.

Servidor Web. Sistema que ejecuta el software de servidor de HTTP, por ejemplo, Internet Connection.

sistema de gestión de base de datos (DBMS). Sistema de software que controla la creación, organización y modificación de una base de datos y el acceso a los datos almacenados en la misma.

TCP/IP. Protocolo de control de transmisiones / Protocolo Internet

tipo de datos. Atributo de columnas y literales.

transacción. Una invocación de Net.Data. Si se utiliza Net.Data permanente, una transacción puede abarcar múltiples invocaciones de Net.Data.

ubicación uniforme de recursos. Dirección que nombra un servidor HTTP y opcionalmente un nombre de archivo y directorio, por ejemplo:
<http://www.ibm.com/software/data/net.data/index.html>.

unidad de trabajo. Secuencia recuperable de operaciones que se trata como operación atómica. Pueden completarse todas las operaciones que hay dentro de la unidad de trabajo (confirmadas) o deshacerse (retrotraídas) como si las operaciones fueran una única operación. Sólo pueden confirmarse o retrotraerse las operaciones sobre los recursos que resulten afectadas por el control de confirmación.

URL. Localizador uniforme de recursos.

vía de acceso. Ruta de búsqueda utilizada para localizar archivos.

vía de acceso absoluta. El nombre de vía de acceso completa de un objeto. Los nombres de vía acceso absoluta comienzan en el nivel más alto, o en el directorio "root" (que se identifica por medio del carácter de barra inclinada (/) o de barra inclinada invertida (\)).

Índice

A

- acceso a archivos planos 309
- ALIGN 94
- archivos include 44
- archivos planos
 - acceso 309
 - bloqueo de archivos 313
 - coincidencia con FFI_PATH 311
 - creación en el directorio
 - actual 311
 - definición 309
 - delimitadores 312
 - fuentes de datos 309
 - normas de configuración 311
 - recomendaciones de seguridad 311
 - recomendaciones para el acceso 311
 - requisito de autorización 312
 - ubicación
 - directorio actual 310
 - FFI_PATH 310
 - vías de acceso absolutas 311

B

- bloque COMMENT
 - descripción 10
 - sintaxis 10
- bloque DEFINE
 - descripción 12
 - sintaxis 12
- bloque EXEC
 - descripción 18
 - sintaxis 18
- bloque FUNCTION
 - descripción 21
 - sintaxis 22
- bloque HTML
 - descripción 34
 - sintaxis 34
- bloque HTML_INPUT 385
- bloque HTML_REPORT 385
- bloque IF
 - descripción 37
 - sintaxis 37
- bloque MACRO_FUNCTION
 - sintaxis 49
- Bloque MACRO_FUNCTION
 - descripción 49

- bloque MESSAGE
 - descripción 54
 - sintaxis 54
- bloque REPORT
 - ALIGN 94
 - descripción 59
 - DTW_DEFAULT_REPORT 95
 - DTW_HTML_TABLE 96
 - Nn 84
 - NLIST 85
 - NUM_COLUMNS 86
 - NUM_ROWS 87
 - RPT_MAX_ROWS 97
 - sintaxis 59
 - START_ROW_NUM 99
 - TOTAL_ROWS 89
 - variables de tabla 82
- bloque ROW
 - descripción 62
 - Nn 84
 - NLIST 85
 - NUM_COLUMNS 86
 - NUM_ROWS 87
 - ROW_NUM 88
 - sintaxis 62
 - TOTAL_ROWS 89
 - V_Nombrecolumna 90
 - Vn 91, 92
- bloque SQL 385
- bloque SQL_MESSAGE 387
- bloque SQL_REPORT 387
- bloque WHILE
 - descripción 67
 - sintaxis 67
- bloque XML
 - descripción 71
 - sintaxis 71
- bloqueo de archivos, funciones FFI 313
- botón Anterior,
 - RPT_MAX_ROWS 100
- botón Siguiente,
 - RPT_MAX_ROWS 100

C

- cabeceras 44
- código de APPLET, texto
 - alternativo 109
- coherencia de la base de datos,
 - ámbito de transacción 126

- cómo liberar archivos, funciones FFI 313
- cómo llamar al entorno de lenguaje de FFI 309
- comparación numérica de series 37, 67
- conectar a una base de datos,
 - variable DATABASE 104
- conectar con un subsistema DB2
 - ID de subsistema 108
 - plan de DB2 107
 - ubicación 119
- configuración del entorno de lenguaje de FFI 311
- construcciones de lenguaje
 - bloque COMMENT 10
 - bloque FUNCTION 21
 - bloque HTML 34
 - bloque IF 37
 - bloque MACRO_FUNCTION 49
 - bloque MESSAGE 54
 - bloque o sentencia DEFINE 12
 - bloque o sentencia EXEC 18
 - bloque REPORT 59
 - bloque ROW 62
 - bloque WHILE 67
 - bloque XML 71
- elementos de sintaxis
 - comunes 5
- llamadas de función 30
- macro
 - descripción 8
 - sintaxis 1
- nombre de variable 5
- referencia variable 5
- sentencia ENVVAR 17
- sentencia INCLUDE 44
- sentencia LIST 47
- sentencia TABLE 65
- series 7

- cookies
 - DTW_GETCOOKIE 157
 - DTW_PRINT_HEADER 139
 - DTW_SETCOOKIE 177
- enviar 139

D

- DATABASE 104
- DB_CASE 106
- DB2PLAN 107

- DB2SSID 108
- delimitadores, entorno de lenguaje de FFI
 - ASCIITEXT 312
 - DELIMITED 312
- desplazamiento, con los botones
 - Siguiente y Anterior 100
- directorio actual, determinación de
 - archivos planos 310
- DTW_ACCEPT 374
- DTW_ADD 188
- DTW_ADDQUOTE 146
- DTW_APPLET_ALITEXT 109
- DTW_ASSIGN 84, 207, 208
- DTW_CACHE_PAGE 148
- DTW_COMMIT 376
- DTW_CONCAT 211
- DTW_CURRENT_FILENAME 129
- DTW_CURRENT_LAST_MODIFIED 130
- DTW_CHARTOHEX 209
- DTW_DATE 153
- DTW_DEFAULT_MESSAGE 131
- DTW_DEFAULT_REPORT 95
- DTW_DELSTR 213
- DTW_DELWORD 240
- DTW_DIVIDE 191
- DTW_DIVREM 193
- DTW_EDIT_CODES 110
- DTW_FORMAT 195
- DTW_GETCOOKIE 157
- DTW_GETENV 160
- DTW_GETINIDATA 162
- DTW_HEXTOCHAR 215
- DTW_HTML_TABLE 96
- DTW_HTMLLENCODE 164
- DTW_INSERT 217
- DTW_INTDIV 199
- DTW_LASTPOS 220
- DTW_LENGTH 222
- DTW_LOG_LEVEL 132
- DTW_LOWERCASE 224
- DTW_MACRO_FILENAME 133
- DTW_MACRO_LAST_MODIFIED 134
- DTW_MBMODE 135
- DTW_MP_PATH 137
- DTW_MP_VERSION 138
- DTW_MULTIPLY 201
- DTW_PAD_PGM_PARMS 112
- DTW_POS 226
- DTW_POWER 203
- DTW_PRINT_HEADER 139
- DTW_QHTMLENCODE 167
- DTW_REMOVE_WS 141
- DTW_REPLACE 228
- DTW_REVERSE 230
- DTW_ROLLBACK 377
- DTW_RTVHANDLE 378
- DTW_SAVE_TABLE_IN 114
- DTW_SENDMAIL 169
- DTW_SET_TOTAL_ROWS 115
- DTW_SETCOOKIE 177
- DTW_SETENV 181
- DTW_STATIC 380
- DTW_STRIP 232
- DTW_SUBSTR 234
- DTW_SUBTRACT 205
- DTW_SUBWORD 243
- DTW_TB_APPENDROW 256
- DTW_TB_COLS 258
- DTW_TB_deleteCOL 260
- DTW_TB_DELETEROW 262
- DTW_TB_DLIST 264
- DTW_TB_DUMPH 267
- DTW_TB_DUMPV 269
- DTW_TB_GETN 271
- DTW_TB_GETV 273
- DTW_TB_HTMLLENCODE 275
- DTW_TB_INPUT_CHECKBOX 277
- DTW_TB_INPUT_RADIO 279
- DTW_TB_INPUT_TEXT 281
- DTW_TB_INSERTCOL 284
- DTW_TB_INSERTROW 286
- DTW_TB_LIST 284
- DTW_TB_QUERYCOLNONJ 291
- DTW_TB_ROWS 293
- DTW_TB_SELECT 295
- DTW_TB_SETCOLS 298
- DTW_TB_SETN 300
- DTW_TB_SETV 302
- DTW_TB_TABLE 304
- DTW_TB_TEXTAREA 307
- DTW_TERMINATE 382
- DTW_TIME 183
- DTW_TRANSLATE 236
- DTW_UPPERCASE 238
- DTW_URLESCSEQ 185
- DTW_USE_DB2_PREPARE_CACHE 117
- DTW_WORD 245
- DTW_WORDINDEX 247
- DTW_WORDLENGTH 249
- DTW_WORDPOS 251
- DTW_WORDS 253
- DTWF_APPEND 313
- DTWF_CLOSE 313, 318
- DTWF_DELETE 320
- DTWF_INSERT 323
- DTWF_OPEN 313, 326
- DTWF_READ 329
- DTWF_READFILE 333
- DTWF_REMOVE 336
- DTWF_SEARCH 338
- DTWF_UPDATE 342
- DTWF_WRITE 346
- DTWR_ADDENTRY 350
- DTWR_CLEARREG 353
- DTWR_CLOSEREG 355
- DTWR_CREATEREG 357
- DTWR_DELENTY 359
- DTWR_DELREG 361
- DTWR_LISTREG 363
- DTWR_LISTSUB 365
- DTWR_OPENREG 367
- DTWR_RTVENTRY 369
- DTWR_UPDATEENTRY 371

E

- entorno de lenguaje de FFI
 - acceso a archivos 309
 - delimitadores 312
 - directorio actual 310
 - normas de configuración 311
 - recomendaciones de seguridad 311
 - requisito de autorización 312
 - ubicación de archivos 310
- entorno de lenguaje del Sistema,
 - parámetros de pase 28
- envío de un e-mail (correo electrónico) desde la macro 169
- estado de SQL, visualizar 125
- EXEC_PATH 18
- EXEC_SQL 385

F

- FFI_PATH
 - acceso a archivos planos 309
 - normas de configuración 311
 - recomendaciones de seguridad 311
 - sintaxis 309
 - ubicación del archivo plano 310
 - vías de acceso coincidentes con el parámetro *nombarchivo* 311
- formato UTF-8
 - fecha 153
- formatos de fecha, UTF-8 153
- funciones
 - convenios de denominación 143
 - descripción 143
 - generales 145
 - interfaz de archivo plano (FFI) 309
 - matemáticas 188
 - pasar grupos de valores 82
 - permanentes 373
 - registro Web 350

funciones (continuación)

serie 207
tabla 255
texto 240

Funciones de macros permanentes

DTW_ACCEPT 374
DTW_COMMIT 376
DTW_ROLLBACK 377
DTW_RTVHANDLE 378
DTW_STATIC 380
DTW_TERMINATE 382

funciones de registro Web

DTWR_ADDENTRY 351
DTWR_CLEARREG 353
DTWR_CLOSEREG 355
DTWR_CREATEREG 357
DTWR_DELENTY 359
DTWR_DELREG 361
DTWR_LISTREG 363
DTWR_LISTSUB 365
DTWR_OPENREG 367
DTWR_RTVENTRY 369
DTWR_UPDATEENTRY 371

funciones de serie

DTW_ASSIGN 208
DTW_CONCAT 211
DTW_CHARTOHEX 209
DTW_DELSTR 213
DTW_HEXTOCHAR
DTW_HEXTOCHAR 215
DTW_INSERT
DTW_INSERT 217
DTW_LASTPOS 220
DTW_LENGTH 222
DTW_LOWERCASE 224
DTW_POS 226
DTW_REPLACE 228
DTW_REVERSE 230
DTW_STRIP 232
DTW_SUBSTR 234
DTW_TRANSLATE 236
DTW_UPPERCASE 238
soporte de MBCS 207

funciones de tabla

DTW_TB_APPENDROW 256
DTW_TB_COLS 258
DTW_TB_DELETECOL 260
DTW_TB_DELETEROW 262
DTW_TB_DLIST 264
DTW_TB_DUMPH 267
DTW_TB_DUMPV 269
DTW_TB_GETN 271
DTW_TB_GETV 273
DTW_TB_HTMLENCODE 275

funciones de tabla (continuación)

DTW_TB_INPUT_CHECKBOX 277
DTW_TB_INPUT_RADIO 279
DTW_TB_INPUT_TEXT 281
DTW_TB_INSERTCOL 284
DTW_TB_INSERTROW 286
DTW_TB_LIST 288
DTW_TB_QUERYCOLNONJ 291
DTW_TB_ROWS 293
DTW_TB_SELECT 295
DTW_TB_SETCOLS 298
DTW_TB_SETN 300
DTW_TB_SETV 302
DTW_TB_TABLE 304
DTW_TB_TEXTAREA 307

funciones de texto

DTW_DELWORD 241
DTW_SUBWORD 243
DTW_WORD 245
DTW_WORDINDEX 247
DTW_WORDLENGTH 249
DTW_WORDPOS 251
DTW_WORDS 253
soporte de MBCS 240

funciones FFI

bloqueo de archivos 313
cómo liberar archivos 313
DTWF_APPEND 314
DTWF_CLOSE 318
DTWF_DELETE 320
DTWF_INSERT 323
DTWF_OPEN 326
DTWF_READ 329
DTWF_READFILE 333
DTWF_REMOVE 336
DTWF_SEARCH 338
DTWF_UPDATE 342
DTWF_WRITE 346

funciones generales

DTW_ADDQUOTE 146
DTW_CACHE_PAGE 148
DTW_DATE 153
DTW_EXIT 155
DTW_GETCOOKIE 157
DTW_GETENV 160
DTW_GETINIDATA 162
DTW_HTMLENCODE 164
DTW_QHTMLENCODE 167
DTW_SENDMAIL 169
DTW_SETCOOKIE 177
DTW_SETENV 181
DTW_TIME 183
DTW_URLESCSEQ 185

funciones incorporadas

funciones matemáticas

DTW_ADD 189
DTW_DIVIDE 191
DTW_DIVREM 193
DTW_FORMAT 195
DTW_INTDIV 199
DTW_MULTIPLY 201
DTW_POWER 203
DTW_SUBTRACT 205

G

glosario 407

H

HTML

formato, entrar contraseñas 122
formato, entrar los ID de usuarios 120
ocultación de nombres de variable 79
visualización de resultados de tabla en 96

I

ID de subsistema, conectar con un subsistema DB2 108

INCLUDE_PATH 44

INCLUDE_URL 385

informes

alteración temporal del valor por omisión de Net.Data 95
formatear 59

L

límite superior 65

límites de longitud de línea, macros 5

listado de series delimitadas 80

LOCATION 119

LOGIN 120

LL

llamadas de función

descripción 30
formatear salida 59
proceso de filas de tabla 62
sintaxis 30
utilización de variables INOUT 32

llamar

funciones 30
programas externos 18

M

macros

construcciones de lenguaje 1
detener el proceso 155

macros (*continuación*)
ejemplo 3
elementos de sintaxis
comunes 5
formato 3
límites de longitud de línea 5
parte de declaración 2
parte de presentación 2
sintaxis global 1
manejo de errores 54
mayúsculas, especificar 106
mensajes, texto por omisión 131
minúsculas, especificar 106

N

Nn 84
NLIST 85
nombre de variable 5
NULL_RPT_FIELD 121
NUM_COLUMNS 86
NUM_ROWS 87

O

ocultación de nombres de
variable 79

P

palabra clave IN 24, 50, 144
palabra clave INOUT 24, 50, 144
palabra clave OUT 24, 50, 144
palabra clave RETURNS 24, 51
parámetros, pase 28
parámetros de pase, entorno de
lenguaje del Sistema 28
parte de declaración, macro 2
parte de presentación, macro 2
pasar grupos de valores 82
PASSWORD 122
pies de página 44
plan, conectar con un subsistema
DB2 107
proceso de serie condicional 37, 67

R

recomendaciones de seguridad,
FFI_PATH 311
referencia del sistema operativo 387
referencia del soporte de
plataforma 387
referencia variable 5
rendimiento, DTW_EXIT 155
repetición en bucle 67
requisito de autorización,
FFI_PATH 312
restricción del acceso a la base de
datos 120, 122
RETURN_CODE 142

ROW_NUM 88
RPT_MAX_ROWS 97

S

seguridad
contraseñas 122
ID de inicio de sesión 120
sentencia DEFINE
descripción 12
sintaxis 12
sentencia ENVVAR 78
descripción 17
sintaxis 17
sentencia EXEC 78
descripción 18
sintaxis 18
sentencia INCLUDE
descripción 44
sintaxis 44
sentencia LIST
descripción 47
sintaxis 47
sentencia TABLE 82
descripción 65
sintaxis 65
serie de valores delimitada 81
series
comparaciones numéricas 37, 67
de valores, delimitada 81
descripción 7
proceso condicional 37, 67
SHOWSQL 123
soporte de MBCS para funciones
funciones de serie 207
funciones de texto 240
SQL
mandatos, especificar tipo de
letra 106
ocultar o visualizar 123
SQL_CODE 387
SQL_STATE 125
START_ROW_NUM 99
subsistema DB2 local, ID 108
subsistema DB2 remoto,
ubicación 119

T

tabla de características
soportadas 387
tablas
Net.Data, especificación del
número de filas 97
resultados en HTML 96
tablas Net.Data
definición 65
límite superior 65

texto alternativo, navegadores de la
Web 109
tipo de letra, especificar para
mandatos SQL 106
TOTAL_ROWS 89
TRANSACTION_SCOPE 126

U

ubicación, archivos planos 310
ubicación, conectar con un
subsistema DB2 119

V

V_Nombrecolumna 90
variable INOUT
ejemplo 32
variables
condicionales 76
ejecutables 78
entorno 78
entorno de lenguaje 103
informe 93
lista 80
Net.Data, visión general 75
ocultas 79
tabla 82, 83
varios 128
variables condicionales
con referencias variables 77
con sentencias LIST 77
descripción 76
ejemplo 81
variables de entorno
descripción 78
ejemplo 78
sentencia ENVVAR 17
variables de entorno de lenguaje
DATABASE 104
DB_CASE 106
DB2PLAN 107
DB2SSID 108
descripción 103
DTW_APPLET_ALTTTEXT 109
DTW_EDIT_CODES 110
DTW_MBMODE 135
DTW_PAD_PGM_PARMS 112
DTW_SAVE_TABLE_IN 114
DTW_SET_TOTAL_ROWS 115
DTW_USE_DB2_PREPARE_CACHE 117
LOCATION 119
LOGIN 120
NULL_RPT_FIELD 121
PASSWORD 122
SHOWSQL 123
SQL_STATE 125
TRANSACTION_SCOPE 126

- variables de fecha 128
- variables de informe
 - ALIGN 94
 - descripción 93
 - DTW_DEFAULT_REPORT 95
 - DTW_HTML_TABLE 96
 - RPT_MAX_ROWS 97
 - START_ROW_NUM 99
- variables de lista
 - descripción 80
 - ejemplo 81
 - separadores de valor 81
- variables de proceso de tablas
 - descripción 83
 - especificación del entorno de
 - lenguaje SQL 114
 - Nn 84
 - NLIST 85
 - NUM_COLUMNS 86
 - NUM_ROWS 87
 - ROW_NUM 88
 - TOTAL_ROWS 89
 - V_Nombrecolumna 90
 - Vn 92
 - VLIST 91
- variables de tabla
 - descripción 82
 - ejemplo 82
- variables de ubicación de
 - archivos 128
- variables de varios
 - descripción 128
 - DTW_CURRENT_FILENAME 129
 - DTW_CURRENT_LAST_MODIFIED 130
 - DTW_DEFAULT_MESSAGE 131
 - DTW_MACRO_LAST_MODIFIED 134
 - DTW_MP_PATH 137
 - DTW_MP_VERSION 138
 - DTW_PRINT_HEADER 139
 - DTW_REMOVE_WS 141
 - RETURN_CODE 142
- variables ejecutables
 - como una referencia a
 - variable 79
 - con parámetros 79
 - descripción 78
 - ejemplo 79
- variables ocultas
 - descripción 79
 - ejemplo, en un formato
 - HTML 80
 - pasos 80
- vías de acceso absolutas, para
 - archivos planos 311
- VLIST 91

Cómo ponerse en contacto con IBM

Si tiene un problema técnico, repase y lleve a cabo las acciones que se sugieren en la *Guía de resolución de problemas* antes de ponerse en contacto con el Centro de Asistencia al Cliente de DB2. Dicha guía sugiere información que puede reunir para ayudar al Centro de Asistencia a proporcionarle un mejor servicio.

Para obtener información o para solicitar cualquiera de los productos de DB2 Universal Database, consulte a un representante de IBM de una sucursal local o a un concesionario autorizado de IBM.

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-237-5511 para obtener soporte técnico
- 1-888-426-4343 para obtener información sobre las opciones de servicio técnico disponibles

Información sobre productos

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) o 1-800-3IBM-OS2 (1-800-342-6672) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

<http://www.ibm.com/software/data/>

Las páginas Web de DB2 ofrecen información actual sobre DB2 referente a novedades, descripciones de productos, planes de formación, etc.

<http://www.ibm.com/software/data/db2/library/>

La biblioteca técnica de servicio y de productos DB2 ofrece acceso a las preguntas más frecuentes (FAQ), arreglos de programa, manuales e información técnica actualizada sobre DB2.

Nota: Puede que esta información sólo esté disponible en inglés.

<http://www.elink.ibm.com/pbl/pbl/>

El sitio Web para el pedido de publicaciones internacionales proporciona información sobre cómo hacer pedidos de manuales.

<http://www.ibm.com/education/certify/>

El Programa de homologación profesional contenido en el sitio Web de IBM proporciona información de prueba de homologación para diversos productos de IBM, incluido DB2.

ftp.software.ibm.com

Conéctese como anónimo (anonymous). En el directorio /ps/products/db2 encontrará programas de demostración, arreglos de programa, información y herramientas referentes a DB2 y a muchos otros productos.

comp.databases.ibm-db2, bit.listserv.db2-l

En estos foros de discusión de Internet los usuarios pueden explicar sus experiencias con los productos DB2.

En Compuserve: GO IBMDB2

Entre este mandato para acceder a los foros referentes a la familia de productos DB2. Todos los productos DB2 tienen soporte a través de estos foros.

Para conocer cómo ponerse en contacto con IBM desde fuera de los Estados Unidos, consulte el Apéndice A del manual *IBM Software Support Handbook*. Para acceder a este documento, vaya a la página Web siguiente: <http://www.ibm.com/support/> y luego seleccione el enlace "IBM Software Support Handbook", cerca del final de la página.

Nota: En algunos países, los distribuidores autorizados de IBM deben ponerse en contacto con su organización de soporte en lugar de acudir al Centro de Asistencia de IBM.



Impreso en España