

**Net.Data**



**解説書**



**Net.Data**



**解説書**

**ご注意**

本書の情報およびそれによってサポートされる製品を使用する前に、341ページの『付録D. 特記事項』に記載する一般情報をお読みください。

原 典： Net Data Reference  
Network Installation Management Guide and Reference

発 行： 日本アイ・ピー・エム株式会社

担 当： ナショナル・ランゲージ・サポート

第1刷 1999.6

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体\*)は、(財)日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注\* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、  
平成角ゴシック体™W5、平成角ゴシック体™W7

# 目次

まえがき . . . . .	ix
Net.Data について . . . . .	ix
本書について . . . . .	x
本書の対象読者 . . . . .	x
本書の例について . . . . .	x
構文図の読み取り方法 . . . . .	x
<b>第1章 Net.Data マクロ言語構成要素 . . . . .</b>	<b>1</b>
Net.Data マクロ構文 . . . . .	1
共通の構文要素 . . . . .	5
変数名 . . . . .	5
変数参照 . . . . .	5
ストリング . . . . .	6
マクロ言語構成要素 . . . . .	7
コメント・ブロック . . . . .	9
DEFINE ブロックまたはステートメント . . . . .	11
ENVVAR ステートメント . . . . .	15
EXEC ブロックまたはステートメント . . . . .	16
FUNCTION ブロック . . . . .	19
関数呼び出し (@) . . . . .	27
HTML ブロック . . . . .	30
IF ブロック . . . . .	33
INCLUDE ステートメント . . . . .	40
INCLUDE_URL ステートメント . . . . .	43
LIST ステートメント . . . . .	46
MACRO_FUNCTION ブロック . . . . .	48
MESSAGE ブロック . . . . .	52
REPORT ブロック . . . . .	57
ROW ブロック . . . . .	60
TABLE ステートメント . . . . .	63
WHILE ブロック . . . . .	65
<b>第2章 変数 . . . . .</b>	<b>69</b>
ユーザー定義変数 . . . . .	70
条件変数 . . . . .	70
環境変数 . . . . .	71
実行可能変数 . . . . .	72
隠蔽変数 . . . . .	73
リスト変数 . . . . .	74
表変数 . . . . .	75
Net.Data 表処理変数 . . . . .	76
Nn . . . . .	77
NLIST . . . . .	78
NUM_COLUMNS . . . . .	79
NUM_ROWS . . . . .	80
ROW_NUM . . . . .	81
TOTAL_ROWS . . . . .	82
V_columnName . . . . .	83
VLIST . . . . .	84

Vn . . . . .	85
Net.Data レポート変数 . . . . .	86
ALIGN . . . . .	87
DTW_DEFAULT_REPORT . . . . .	88
DTW_HTML_TABLE . . . . .	89
RPT_MAX_ROWS . . . . .	90
START_ROW_NUM . . . . .	92
Net.Data 言語環境変数 . . . . .	95
DATABASE. . . . .	96
DB_CASE . . . . .	98
DB2PLAN . . . . .	99
DB2SSID. . . . .	100
DTW_APPLET_ALTTEXT. . . . .	101
DTW_EDIT_CODES . . . . .	102
DTW_PAD_PGM_PARMS. . . . .	103
DTW_SAVE_TABLE_IN . . . . .	104
DTW_SET_TOTAL_ROWS . . . . .	105
LOCATION . . . . .	107
LOGIN . . . . .	108
NULL_RPT_FIELD . . . . .	109
PASSWORD. . . . .	110
SHOWSQL . . . . .	111
SQL_STATE . . . . .	112
TRANSACTION_SCOPE . . . . .	113
Net.Data の各種変数 . . . . .	115
DTW_CURRENT_FILENAME . . . . .	116
DTW_CURRENT_LAST_MODIFIED . . . . .	117
DTW_DEFAULT_MESSAGE. . . . .	118
DTW_LOG_LEVEL . . . . .	119
DTW_MACRO_FILENAME . . . . .	120
DTW_MACRO_LAST_MODIFIED. . . . .	121
DTW_MBMODE . . . . .	122
DTW_MP_PATH . . . . .	124
DTW_MP_VERSION . . . . .	125
DTW_PRINT_HEADER. . . . .	126
DTW_REMOVE_WS. . . . .	127
RETURN_CODE . . . . .	128
<b>第3章 Net.Data 組み込み関数 . . . . .</b>	<b>129</b>
関数名. . . . .	129
入出力パラメーター. . . . .	129
関数結果の形式化 . . . . .	130
関数パラメーターの規則 . . . . .	130
汎用関数. . . . .	131
DTW_ADDQUOTE . . . . .	132
DTW_CACHE_PAGE . . . . .	134
DTW_DATE. . . . .	138
DTW_EXIT . . . . .	140
DTW_GETCOOKIE . . . . .	141
DTW_GETENV . . . . .	143
DTW_GETINIDATA. . . . .	145
DTW_HTMLLENCODE . . . . .	146

DTW_QHTMLENCODE . . . . .	148
DTW_SENDMAIL . . . . .	149
DTW_SETCOOKIE . . . . .	154
DTW_SETENV. . . . .	158
DTW_TIME. . . . .	160
DTW URLESCSEQ . . . . .	162
数学関数. . . . .	164
DTW_ADD . . . . .	165
DTW_DIVIDE . . . . .	167
DTW_DIVREM . . . . .	169
DTW_FORMAT . . . . .	171
DTW_INTDIV . . . . .	175
DTW_MULTIPLY. . . . .	177
DTW_POWER . . . . .	179
DTW_SUBTRACT . . . . .	181
ストリング関数 . . . . .	183
DTW_ASSIGN. . . . .	184
DTW_CHARTOHEX. . . . .	185
DTW_CONCAT . . . . .	186
DTW_DELSTR. . . . .	188
DTW_HEXTOCHAR. . . . .	190
DTW_INSERT . . . . .	192
DTW_LASTPOS . . . . .	194
DTW_LENGTH . . . . .	196
DTW_LOWERCASE. . . . .	197
DTW_POS . . . . .	199
DTW_REPLACE . . . . .	201
DTW_REVERSE . . . . .	203
DTW_STRIP . . . . .	204
DTW_SUBSTR. . . . .	206
DTW_TRANSLATE . . . . .	208
DTW_UPPERCASE . . . . .	210
ワード関数. . . . .	212
DTW_DELWORD. . . . .	213
DTW_SUBWORD. . . . .	215
DTW_WORD . . . . .	217
DTW_WORDINDEX. . . . .	219
DTW_WORDLENGTH . . . . .	221
DTW_WORDPOS. . . . .	222
DTW_WORDS . . . . .	224
表関数. . . . .	225
DTW_TB_APPENDROW . . . . .	226
DTW_TB_COLS . . . . .	228
DTW_TB_DELETECOL . . . . .	229
DTW_TB_DELETEROW . . . . .	230
DTW_TB_DLIST . . . . .	232
DTW_TB_DUMPH . . . . .	235
DTW_TB_DUMPV . . . . .	236
DTW_TB_GETN . . . . .	238
DTW_TB_GETV . . . . .	240
DTW_TB_HTMLENCOD. . . . .	242
DTW_TB_INPUT_CHECKBOX . . . . .	244

DTW_TB_INPUT_RADIO . . . . .	246
DTW_TB_INPUT_TEXT . . . . .	248
DTW_TB_INSERTCOL . . . . .	250
DTW_TB_INSERTTROW . . . . .	251
DTW_TB_LIST . . . . .	253
DTW_TB_QUERYCOLNONJ. . . . .	255
DTW_TB_ROWS . . . . .	257
DTW_TB_SELECT . . . . .	258
DTW_TB_SETCOLS. . . . .	260
DTW_TB_SETN . . . . .	261
DTW_TB_SETV . . . . .	263
DTW_TB_TABLE. . . . .	265
DTW_TB_TEXTAREA . . . . .	267
フラット・ファイル・インターフェース関数 . . . . .	269
フラット・ファイル・データ・ソースへのアクセス . . . . .	269
フラット・ファイル・インターフェース区切り文字 . . . . .	272
ファイルのロック . . . . .	273
DTWF_APPEND . . . . .	274
DTWF_CLOSE. . . . .	277
DTWF_DELETE . . . . .	278
DTWF_INSERT . . . . .	280
DTWF_OPEN . . . . .	283
DTWF_READ . . . . .	285
DTWF_REMOVE . . . . .	288
DTWF_SEARCH . . . . .	290
DTWF_UPDATE . . . . .	293
DTWF_WRITE. . . . .	296
Web レジストリー関数 . . . . .	299
DTWR_ADDENTRY. . . . .	300
DTWR_CLEARREG . . . . .	302
DTWR_CLOSEREG . . . . .	303
DTWR_CREATEREG . . . . .	304
DTWR_DELENTY . . . . .	306
DTWR_DELREG . . . . .	308
DTWR_LISTREG . . . . .	309
DTWR_LISTSUB . . . . .	311
DTWR_OPENREG . . . . .	313
DTWR_RTVENTRY . . . . .	315
DTWR_UPDATEENTRY . . . . .	317
永続的なマクロ関数. . . . .	319
DTW_ACCEPT. . . . .	320
DTW_COMMIT . . . . .	322
DTW_ROLLBACK . . . . .	323
DTW_RTVHANDLE. . . . .	324
DTW_STATIC . . . . .	325
DTW_TERMINATE . . . . .	327
 付録A. Net.Data 技術ライブラリー . . . . .	 329
 付録B. DB2 WWW Connection . . . . .	 331
EXEC_SQL . . . . .	331
HTML_INPUT . . . . .	331



HTML_REPORT . . . . .	331
SQL . . . . .	331
SQL_MESSAGE . . . . .	332
SQL_REPORT . . . . .	332
SQL_CODE . . . . .	333
<b>付録C. Net.Data のオペレーティング・システムごとの参照 . . . . .</b>	<b>335</b>
<b>付録D. 特記事項 . . . . .</b>	<b>341</b>
商標 . . . . .	342
用語集. . . . .	343
索引 . . . . .	345



---

## まえがき

動的 Web ページを作成するための IBM® 開発ツールである Net.Data® をお選びいただきましてありがとうございます。Net.Data を使用すると、各種データ・ソースからのデータを取り込んだり、すでにお使いになっているプログラミング言語を使用して、動的な内容の Web ページを迅速に開発することができます。

---

## Net.Data について

IBM の Net.Data プロダクトを用いると、リレーショナル・データベース管理システムおよび非リレーショナル・データベース管理システム (DBMS) (DB2、IMS、ODBC 対応のデータベース、および DRDA 経由でアクセス可能なデータベースを含む) の両方のデータを使用して、また Java、JavaScript、Perl、C、C++、および REXX などプログラミング言語で作成されたアプリケーションを使用して、動的 Web ページを作成することができます。Net.Data ファミリー・プロダクトは、Windows NT、AIX、OS/2、OS/390、OS/400、HP-UX、Sun Solaris、Santa Cruz オペレーティング・システム (SCO)、および Linux オペレーティング・システムを実行するマシン上で、同様の機能を提供します。

Net.Data は、Web サーバー上のミドルウェアとして実行するマクロ処理プログラムです。マクロと呼ばれる Net.Data アプリケーション・プログラムを作成することができます。Net.Data は、これを解釈して、ユーザー、データベースの現状、その他のデータ・ソース、既存のビジネス論理、およびマクロ中に組み込む予定のその他の係数からの入力データに基づいて、カスタマイズされた内容の動的 Web ページを作成します。

要求は URL (uniform resource locator) の形式で Netscape Navigator または Internet Explorer などのブラウザから Web サーバーに流れ、Web サーバーはその要求を Net.Data に送って実行します。Net.Data はこのマクロを探し出して実行し、Web ページを作成して、ユーザーが作成した関数に基づいてその Web ページをカスタマイズします。これらの関数は以下のことを行うことができます。

- C、C++、RPG、COBOL、JAVA、Perl、または REXX プログラム言語 (これらに限定されるものではありません) によって作成されたアプリケーション内にビジネス論理をカプセル化する
- DB2 などのデータベースにアクセスする
- フラット・ファイルなどの他のデータ・ソースにアクセスする

Net.Data はその Web ページを Web サーバーに渡し、サーバーは、ページをネットワークに転送し、ブラウザ上で表示ができます。

Net.Data は、HTTP (HyperText Transfer Protocol) およびコモン・ゲートウェイ・インターフェース (CGI) などのインターフェースを使用するよう構成されたサーバー環境で使用できます。HTTP は、ブラウザと Web サーバー間で対話を行うための業界標準インターフェースで、CGI は Net.Data などのゲートウェイ・アプリケーションを Web サーバーが起動するための業界標準インターフェースです。これらのインターフェースを使用することで、使い慣れたブラウザまたは Web サーバーを選んで Net.Data に使用することができます。

パフォーマンス向上のため、Net.Data は各種 Web サーバー・アプリケーション・プログラミング・インターフェース (API) をサポートします。また、Net.Data を Java サブレットとして立ち上げることもできます。

---

## 本書について

本書では、Net.Data 言語構成要素、変数および関数の構文と使用法について説明します。

本書では、発表されてはいるもののまだ使用可能になっていない製品や機能を参照する可能性があります。

詳しい情報や、サンプル Net.Data マクロ、デモ、および本書の最新コピーは、以下の WWW サイトから得ることができます。

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

## 本書の対象読者

Net.Data アプリケーションの計画と作成に取り組む方は、本書の情報を使用して、Net.Data の提供する言語構成要素、変数、および関数を理解することができます。

本書で説明している概念について理解するには、Web サーバー、単純な SQL ステートメント、および HTML (HTML 形式の使用法を含む) と、*Net.Data* 管理およびプログラミングの手引き を参照してください。

## 本書の例について

本書で使用している例は、特定の概念を説明するために単純化されたものです。Net.Data 構成要素を使用できるすべてのケースを示しているわけではありません。また、例の中には断片化したコードがありますが、これのみでの実行はできません。

## 構文図の読み取り方法

本書で使用する構文図には、以下の規則が適用されます。

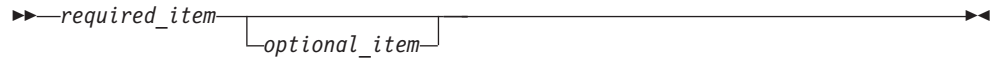
- 構文図は、行のパスに従って、左から右、上から下に読み取ります。
  - ▶— 記号は、ステートメントの開始を示します。
  - ▶ 記号は、ステートメント構文が次の行に継続することを示します。
  - ▶— 記号は、ステートメントが前の行からの継続であることを示します。
  - ▶ 記号は、ステートメントの終了を示します。

完全なステートメントではない構文単位の図は、▶— 記号で始まり、—▶ 記号で終了します。

- 必須項目は水平線 (メインパス) 上に表示されます。

▶—required\_item—▶

- オプション項目はメインパスの下部に表示されます。

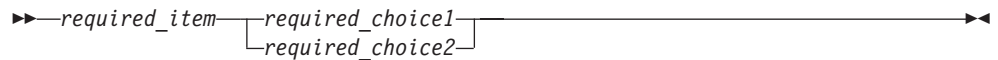


オプション項目がメインパスの上部に表示されている場合、その項目はステートメントの実行に影響を与えません。読み取りやすさのためだけに使用されます。

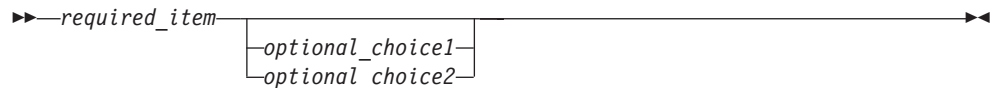


- 複数の項目からの選択が可能の場合、これらの項目は縦方向に重ねて表示されます。

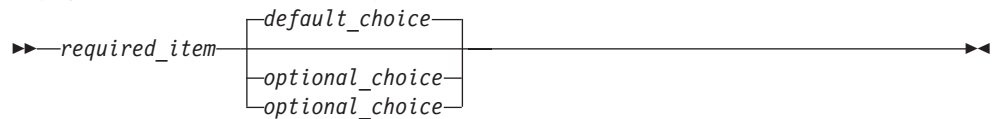
項目の中の 1 つを選択しなければならない場合、スタックの 1 つの項目がメインパス上に表示されます。



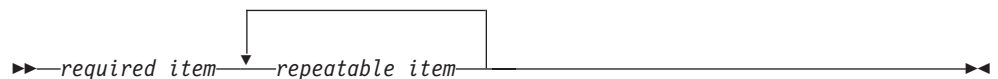
項目の中の 1 項目の選択がオプションの場合、すべてのスタックはメインパスの下部に表示されます。



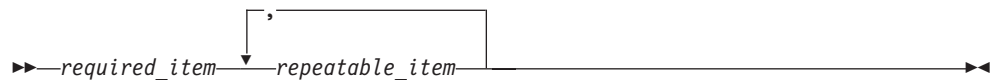
項目の 1 つがデフォルトの場合、その項目はメインパスの上部に表示され、残りの選択項目はメインパスの下部に表示されます。



- メイン行の上部で左に戻る矢印は、繰り返すことのできる項目を示します。



繰り返しの矢印に句読点が含まれている場合、繰り返し項目を指定の句読点で分ければなりません。



スタックの上部での繰り返し矢印は、スタックの項目を繰り返すことができることを示します。

- キーワードは大文字で表示されます (たとえば, FROM)。Net.Data では、キーワードは大文字でも小文字でもかまいません。キーワードではない用語は小文字で表示されます (たとえば, column-name)。これらの用語はユーザー提供の名前または値です。
- 句読記号、括弧、算術演算子、あるいはその他の記号が表示されている場合、それらの記号を構文の一部として入力しなければなりません。



---

## 第1章 Net.Data マクロ言語構成要素

この章では、Net.Data マクロで使用する Net.Data マクロ構文と言語構成要素について説明します。言語構成要素は、Net.Data マクロのキーワードとステートメントまたはブロックで構成されます。そして、さまざまな変数型を指定し、ファイルの組み込みなどのその他の特別な作業を行います。

この章では、以下のことを説明します。

- 『Net.Data マクロ構文』
- 5ページの『共通の構文要素』
- 7ページの『マクロ言語構成要素』

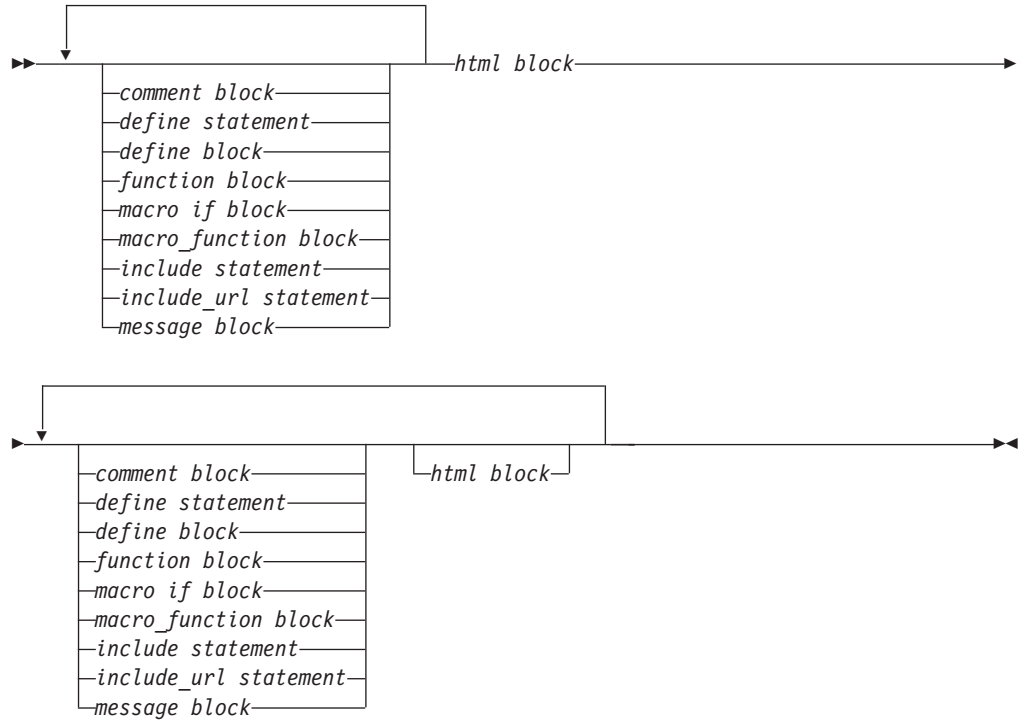
---

### Net.Data マクロ構文

Net.Data マクロは、以下のことを行う一連の Net.Data マクロ言語構成要素から構成されるテキスト・ファイルです。

- Web ページのレイアウトを指定する
- 変数と関数を定義する
- マクロで定義した関数、あるいは Net.Data が処理のために言語環境に渡す関数を呼び出す

それぞれのステートメントは、1 つまたは複数の言語構成要素で構成されます。さらに、これらの言語構成要素は、キーワード、特殊文字、ストリング、名前、および変数で構成されます。以下の図は、構文的に有効な Net.Data マクロのグローバルな構造を示します。グローバルな構造の各要素の詳しい構文については、7ページの『マクロ言語構成要素』を参照してください。



Net.Data マクロには、宣言パーツと表示パーツという 2 つのパーツが含まれています。これらのパーツは、任意の順序で何度でも使用することができます。

- 宣言パーツ には、マクロの変数と関数の定義が含まれます。
- 表示パーツ には、 Web ページのレイアウトを指定する HTML ステートメントが入った HTML ブロックが含まれます。このパーツはレポート・セクションを組み込みます。

3ページの図1 では、マクロの宣言および表示パーツを示します。



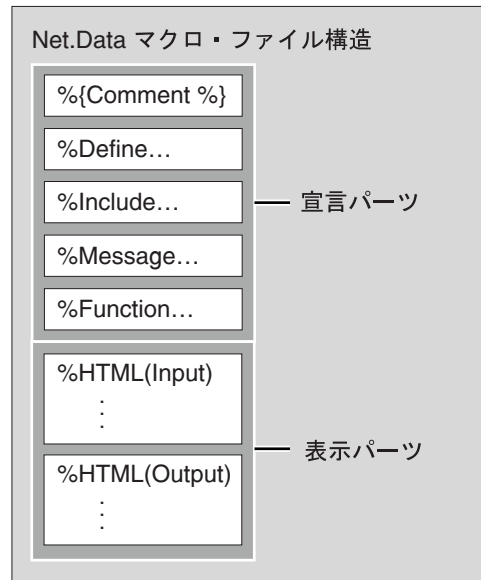


図 1. マクロの構造

宣言または表示パーツで使用する変数と関数は、変数参照または関数呼び出しで使用する前に、あらかじめ定義しておかなければなりません。

4ページの図2 では、マクロのパーツを示します。宣言パーツには、**DEFINE** および **FUNCTION** 定義ブロックが含まれています。**HTML** ブロックは、入出力ブロックとして機能します。

```

%{ ***** 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:
<INPUT NAME="input_data" TYPE="text" SIZE="30">
<p>
<INPUT TYPE="submit" VALUE="Enter">

<hr>
<p>[<a href="/">Home page]
</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>
}%

```

図2. マクロ・テンプレート形式

Net.Data マクロ言語は自由形式の言語であるため、マクロを柔軟に作成することができます。特に断りのない限り、余計な空白文字は無視されます。それぞれの Net.Data マクロ言語構成要素については、構成要素を定義するのに使用するその他のいくつかの要素と共に、以下のセクションで説明します。Net.Data マクロ言語は、逆方向の互換性のために、DB2 WWW Connection 言語要素をサポートします。これらの言語要素については 331ページの『付録B. DB2 WWW Connection』で説明していますが、Net.Data 言語構成要素を使用することをお勧めします。

例では、マクロで言語構成要素、変数、関数、およびその他の要素を使用できる方法の一部を示します。さらに広範囲の例については、以下の `Net.Data Web` ページからサンプルとデモをダウンロードすることができます。

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

---

## 共通の構文要素

以下の構文要素は、言語構成要素の説明で頻繁に使用されます。

- 『変数名』
- 『変数参照』
- 6ページの『ストリング』

## 変数名

目的：

変数を識別します。変数は、マクロの実行中に値を変更できるオブジェクトのことです。

変数名は、文字または下線 (`_`) で開始し、任意の英数字、下線、ハッシュ・マーク (`#`)、またはピリオド (`.`) を含む必要があります。 `N_columnName` と `V_columnName` 以外のすべての変数名には、大文字小文字の区別があります (これらの 2 つの例外については、76ページの『`Net.Data` 表処理変数』を参照してください)。

## 変数参照

目的：

変数を戻します。`$` と `()` を使って指定します。たとえば、`VAR = 'abc'` の場合、`$(VAR)` は値 `'abc'` を戻します。変数参照は、実行時に評価されます。変数が `EXEC` ステートメントまたはブロックで定義されていると、`Net.Data` は、変数参照を読み取るときに指定のアクションを実行します。

変数参照内に、変数参照、ストリングおよび関数呼び出しを組み込むことにより、変数名を動的に作成することができます。動的に生成された変数で、変数名の規則に従わない変数を参照する場合、`Net.Data` は参照を空ストリングに解決します。

変数参照では前後の空白文字は無視され、関数呼び出し、ストリング、および変数参照間に空白文字を使用することはできません。関数呼び出し、ストリング、および変数参照の間に改行文字が検出された場合、エラー・メッセージが発行されます。その他の空白文字を含む参照変数は、空ストリングを戻します。

構文 :



注:

1. スtringには、変数名で使用可能な文字のみを含むことができます。これには英数字、下線 ( \_ )、ハッシュ・マーク ( # )、またはピリオド ( . ) があります。

#### 例 1: 変数参照

変数 homeURL を定義した場合:

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

ホーム・ページを \$(homeURL) として参照し、リンクを作成します。

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

#### 例 2: 動的に生成された変数参照

変数参照を動的に生成することにより、行内のフィールド値を動的に参照することができます。

```
%WHILE (INDEX < NUM_COLS) {
  $(V$(INDEX))
  @DTW_ADD(INDEX, "1", INDEX)
%}
```

#### 例 3: ネストされた変数参照および関数呼び出しを持つ動的な変数参照

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

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

変数参照は、値 hey を戻します。

## スString

任意の順序の英字、数字、および句読点です。スStringが二重引用符内にある場合、改行文字は使用できません。言語構成要素で使用する時の制約事項については、それぞれの言語構成要素のスString・パラメーター説明を参照してください。

引用符 ( " ) 内のスStringには、改行文字以外のすべての文字を含めることができます。大括弧内のスString ( { % } ) の場合は、改行文字を含むすべての文字を含めることができます。たとえば、次のようにします。

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

引用符で囲まれたストリングの中に二重引用符を指定するためには、二重引用符を 2 対使用してください。関数実引き数として、あるいは比較式中の用語として使われるストリングには、二重引用符を含めることができます。たとえば、ストリングを以下のように定義する場合があります。

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

`result` の値は以下のようになります。

```
"Hello world!"
```

HTML ステートメントはストリングです。

関数実引き数、用語、および可変値として使用されるストリングには、変数参照および関数呼び出しを含めることができます。次の例の関数呼び出し `myfunc2` には、変数参照と関数呼び出しを含むパラメーターが使用されています。

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

`Net.Data` は、変数参照 `$(var1)` と関数呼び出し `@myfunc()` を、ストリングの一部として文字どおりに解釈するのではなく、それらを解決してから、そのストリングを関数 `myfunc2` に渡します。

---

## マクロ言語構成要素

このセクションでは、`Net.Data` マクロで使用する言語構成要素について説明します。

それぞれの言語構成要素の説明には、以下の情報が記載されていることがあります。

**目的** `Net.Data` マクロで言語構成要素を使用する理由を定義します。

**構文** 言語構成要素の論理構造図を示します。

**パラメーター**

構文図のすべての要素を定義し、ほかの言語構成要素の構文および例への相互参照を行います。

**コンテキスト**

`Net.Data` マクロ構造内のどこで言語構成要素を使用できるかを説明します。

**制約事項**

含むことのできる要素を定義し、使用におけるすべての制約事項を指定します。

**例** `Net.Data` マクロ内部でキーワード・ステートメントやブロックを使用するための簡単な例や説明を記載します。

マクロでは以下の構成要素を使用します。構文と例については、それぞれの構成要素の説明を参照してください。

- 9ページの『コメント・ブロック』
- 11ページの『`DEFINE` ブロックまたはステートメント』
- 15ページの『`ENVVAR` ステートメント』

- 16ページの『EXEC ブロックまたはステートメント』
- 19ページの『FUNCTION ブロック』
- 27ページの『関数呼び出し (@)』
- 30ページの『HTML ブロック』
- 33ページの『IF ブロック』
- 40ページの『INCLUDE ステートメント』
- 43ページの『INCLUDE\_URL ステートメント』
- 46ページの『LIST ステートメント』
- 48ページの『MACRO\_FUNCTION ブロック』
- 52ページの『MESSAGE ブロック』
- 57ページの『REPORT ブロック』
- 60ページの『ROW ブロック』
- 63ページの『TABLE ステートメント』
- 65ページの『WHILE ブロック』

## コメント・ブロック

### 目的

Net.Data マクロの機能を説明します。 COMMENT ブロックはマクロのどこでも使用できるため、他の構文図ではこのブロックを記載しません。

COMMENT ブロックは、Net.Data 初期設定ファイルでも使用できます。

### 構文

►►—%{—*text*—}%—

### 値

**text** 1 行または複数行での任意のストリング。 Net.Data は、すべてのコメントの内容を無視します。

### コンテキスト

コメントは、Net.Data マクロの Net.Data 言語構成要素間、または Net.Data 初期設定ファイルのどこにでも配置することができます。

### 制約

どのようなテキストや文字でも使用できますが、コメント・ブロックをネストすることはできません。

### 例

#### 例 1: 基本のコメント・ブロック

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

#### 例 2: FUNCTION ブロックのコメント

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

#### 例 3: HTML ブロックのコメント

```
%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>
%}

```

#### 例 4: DEFINE ブロックのコメント

```

%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 %}
%}

```

#### Example 5: Net.Data 初期設定ファイル中のコメント

```

%{ 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) appl11 ()
ENVIRONMENT (DTW_SYSTEM) sysd11 ()

```



## DEFINE ブロックまたはステートメント

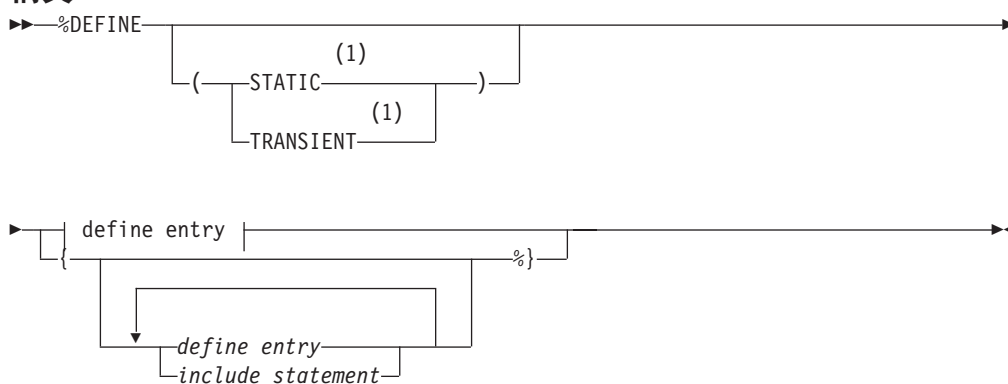
### 目的

DEFINE セクションでは、マクロの宣言パーツに変数名を定義します。このセクションは、以下のとおりステートメントでもブロックでもかまいません。

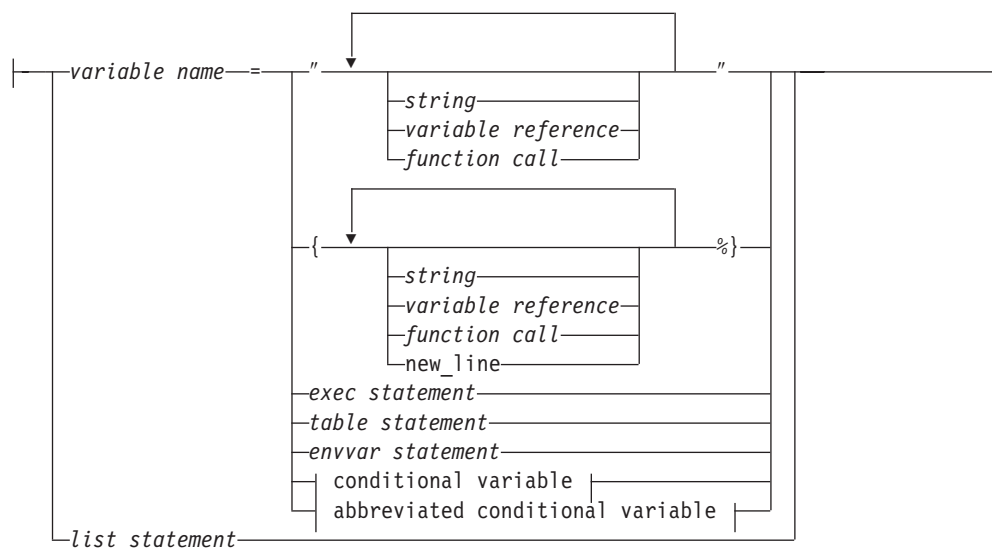
- 一度に 1 つの変数を定義するときにはステートメントを使用
- 数個の変数を定義するときはブロックを使用

変数定義は、二重引用符 (") を使用して単一行にしたり、大括弧とパーセント記号 ({ %}) を使って複数行にわたるようにすることができます。変数を定義すると、マクロのどこでもその変数を参照できるようになります。

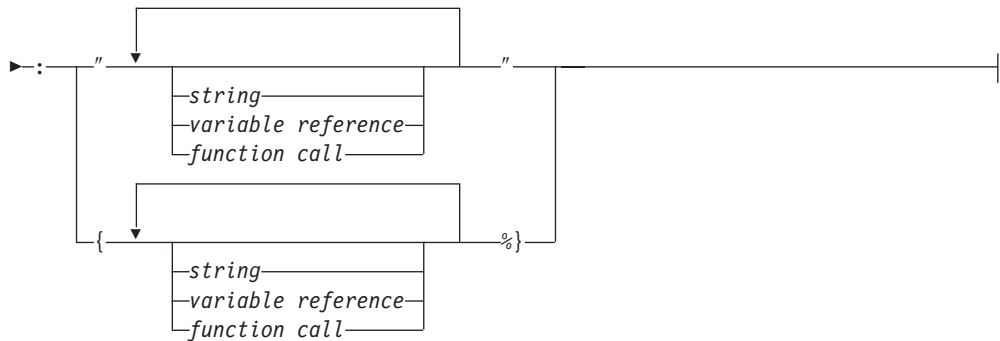
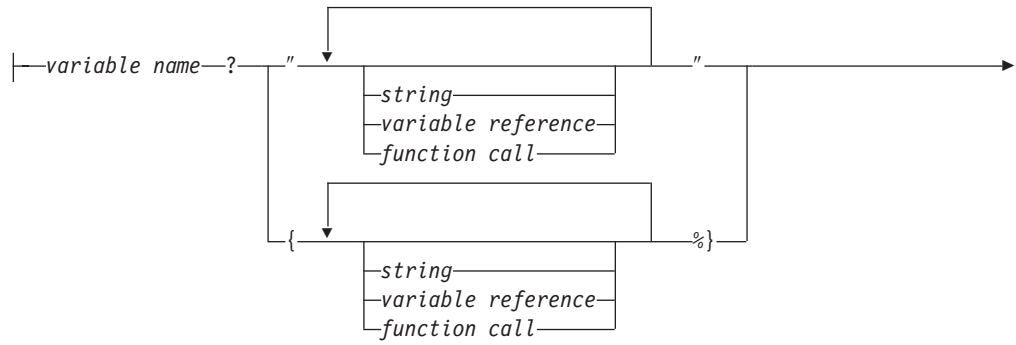
### 構文



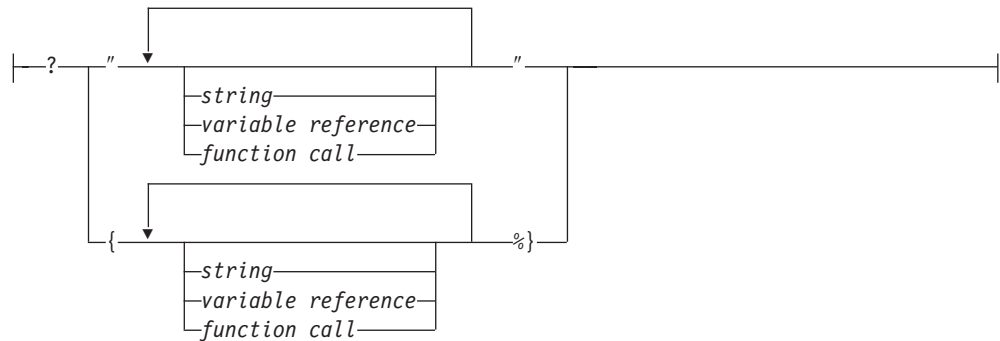
### define entry:



### conditional variable:



#### abbreviated conditional variable:



#### 注:

1. **STATIC** および **TRANSIENT** は、永続的なマクロのキーワードです。これらのマクロは、現在、OS/400 オペレーティング・システムだけで使用することができます。

#### 値

##### %DEFINE

変数を定義するキーワード。

##### STATIC

変数の値が永続的なトランザクション内の複数のマクロ起動を通じて維持されることを指定するキーワード。これは、永続的なマクロの場合のデフォルトです。

## TRANSIENT

この変数の値が複数のマクロ起動を通じて維持されないことを指定するキーワード。これは、非永続的なマクロの場合のデフォルトです。

### define entry:

#### variable name

変数を識別する名前です。構文情報については、5ページの『変数名』を参照してください。

#### string

任意の順序の英字、数字、および句読点です。ストリングが二重引用符内にある場合、改行文字は使用できません。

#### variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

#### function call

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した Net.Data 組み込み関数を呼び出します。構文と例については、27ページの『関数呼び出し (@)』を参照してください。

#### exec statement

EXEC ステートメントです。変数の参照時または関数の呼び出し時に実行される外部プログラム名です。構文と例については、16ページの『EXEC ブロックまたはステートメント』を参照してください。

#### table statement

TABLE ステートメントです。同一レコードまたは同一行の配列、および各行のフィールドを記述する列名の配列が含まれている関連データの集合を定義します。構文と例については、63ページの『TABLE ステートメント』を参照してください。

#### envvar statement

ENVVAR ステートメントです。環境変数を参照します。構文と例については、15ページの『ENVVAR ステートメント』を参照してください。

#### conditional variable

別の変数またはストリングの値に基づいて、変数の値を設定します。

#### abbreviated conditional variable

別の変数またはストリングの値に基づいて、変数の値を設定します。条件変数の簡易形式です。

#### list statement

LIST ステートメントです。値の区切りリストを作成するのに使用する変数を定義します。構文と例については、46ページの『LIST ステートメント』を参照してください。

### include statement

INCLUDE ステートメントです。ファイルを読み取って Net.Data マクロに取り込みます。構文と例については、40ページの『INCLUDE ステートメント』を参照してください。

## コンテキスト

DEFINE ブロックまたはステートメントは、IF ブロックの中か、Net.Data マクロの宣言パーツのほかのすべてのブロックの外側に指定しなければなりません。

### 制約

- 以下の要素を含めることができます。
  - コメント・ブロック
  - 条件変数
  - LIST ステートメント
  - TABLE ステートメント
  - 変数参照
  - INCLUDE ステートメント
  - EXEC ステートメント
  - 関数呼び出し
  - ENVVAR ステートメント
- 変数の定義そのものの中でその変数を使うことはできません。たとえば、以下の変数定義は許可されていません。

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

### 例

#### 例 1: 単純な変数の定義

```
%DEFINE var1 = "orders"
%DEFINE var2 = "${var1}.html"
```

実行時に変数参照 `$(var2)` は、`orders.html` と評価されます。

#### 例 2: ストリング内部の引用符

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

表示されると、変数 `hi` の値は `say "hello"` となります。変数 `empty` は空です。

#### 例 3: 複数の変数の定義

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

#### 例 4: 変数の複数行にわたる定義

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

例 5: この条件変数の例は、結果の値が NULL 値を含まない場合に、変数 `var` が引用符 ("" ) 内に結果の値を入れる方法を示したものです。

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

## ENVVAR ステートメント

### 目的

DEFINE ブロックで変数を環境変数として定義します。 ENVVAR 変数を参照すると、Net.Data は同じ名前の環境変数の現行値を戻します。

### 構文

▶▶—%ENVVAR—▶▶

### コンテキスト

ENVVAR ステートメントは、DEFINE ブロックまたはステートメントで指定することができます。

### 値

#### %ENVVAR

DEFINE ブロックで変数を環境変数として定義するためのキーワードです。この変数は、マクロのどこでも環境変数の値を入手することができます。

### 制約

ENVVAR ステートメントにはその他の要素を含めることができません。

### 例

**例 1:** この例では、ENVVAR は、参照時に環境変数 SERVER\_SOFTWARE の現行値、つまり Web サーバー名を戻す変数を定義します。

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

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

## EXEC ブロックまたはステートメント

### 目的

変数の参照時または関数の呼び出し時に実行される外部プログラムを指定します。

Net.Data がマクロ内で実行可能変数を見つけると、参照された実行可能プログラムを以下の方法で探します。

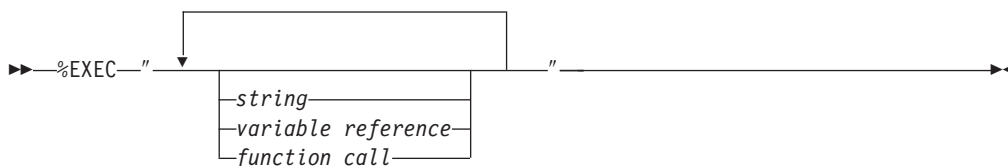
1. Net.Data 初期設定ファイルの中で EXEC\_PATH を探します。EXEC\_PATH についての詳細は、*Net.Data* 管理およびプログラミングの手引き の構成の章を参照してください。
2. プログラムが見つからない場合、Net.Data はシステムの PATH 環境変数で定義されているディレクトリーを探します。実行可能プログラムが見つかったら、Net.Data はそのプログラムを実行します。

**許可のヒント:** Net.Data が実行されるユーザー ID が、EXEC ステートメントまたはブロックによって参照されるどのファイルについてもアクセス権限を持つようにしてください。詳しくは、*Net.Data* 管理およびプログラミングの手引き の構成の章にある、Net.Data ファイルへの Web サーバーのアクセス権限の指定に関するセクションを参照してください。

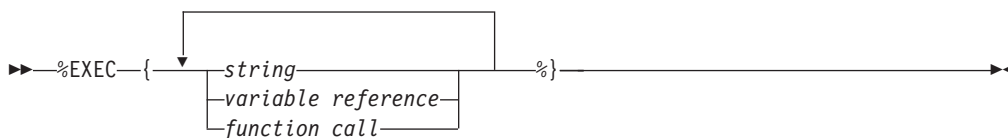
EXEC ステートメントおよびブロックは、使用する場所に応じて 2 つの異なるコンテキストで使用し、その構文も異なります。DEFINE ブロックでは EXEC ステートメントを使用し、FUNCTION ブロックでは EXEC ブロックを使用します。

### 構文

DEFINE ブロックで使用する際の EXEC ステートメント構文：



FUNCTION ブロックで使用する際の EXEC ブロック構文：



### 値

#### %EXEC

変数の参照時または関数の呼び出し時に実行される外部プログラム名を指定するキーワードです。Net.Data が EXEC ステートメントで定義された変数参照を検出すると、Net.Data は EXEC ステートメントが変数に宣言した内容进行处理します。

## string

任意の順序の英字、数字、および句読点です。ストリングが二重引用符内にある場合、改行文字は使用できません。

## variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

## function call

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した Net.Data 組み込み関数を呼び出します。構文と例については、27ページの『関数呼び出し (@)』を参照してください。

## コンテキスト

EXEC ブロックまたはステートメントは、以下のコンテキストで検出することができます。

- DEFINE ブロック
- FUNCTION ブロック

## 制約

EXEC ブロックまたはステートメントには、以下の要素を含めることができます。

- コメント・ブロック
- ストリング
- 変数参照
- 関数呼び出し

以下の Net.Data に提供される言語環境では、EXEC ステートメントがサポートされます。

- REXX
- システム
- Perl

## 例

### 例 1: 変数により参照される実行可能ファイル

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

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

この例では、変数 mycall を参照するたびに MYEXEC.EXE を実行します。

### 例 2: 関数により参照される実行可能ファイル

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

この例は、関数 `my_rexx_pgm` の呼び出し時に `mypgm.cmd` を実行します。



## FUNCTION ブロック

### 目的

Net.Data がマクロから呼び出すサブルーチンを定義します。FUNCTION ブロックの実行可能ステートメントは、言語環境により直接解釈されるインライン・ステートメントにしたり、外部プログラムの呼び出しにすることができます。

**Function ブロック内の EXEC ブロック:** FUNCTION ブロック内部で EXEC ブロックを使用する場合、その EXEC ブロックは、FUNCTION ブロック内の唯一の実行可能ステートメントでなければなりません。実行可能ステートメントを言語環境に渡す前に、Net.Data は、初期設定ファイルの EXEC\_PATH 構成ステートメントで判別されるパス名に、EXEC ブロックのプログラムのファイル名を追加します。結果ストリングは、実行される言語環境に渡されます。

言語環境が EXEC ブロックを処理するのに使用する方法は、特定の言語環境により異なります。REXX、System、および Perl の Net.Data 提供言語環境だけが EXEC ブロックをサポートします。

**言語ステートメント内での特殊文字の使用:** Net.Data 言語構成要素構文に一致する文字が、構文的に有効な組み込みプログラム・コード (たとえば、REXX または Perl) の一部として、関数ブロックの言語ステートメント・セクションで使用されると、それらの文字が誤って Net.Data 言語構成要素として解釈され、マクロでエラーが生じたり、予測できない結果が生じたりすることがあります。

たとえば、Perl 関数が COMMENT ブロックの区切り文字として %{ を使用場合があります。このマクロが実行されると、%{ という文字は COMMENT ブロックの先頭と解釈されます。そして Net.Data は、COMMENT ブロックの終わりを探し、この関数ブロックの終わりに達したときに COMMENT ブロックの終わりが見つかったものと判定します。その後で Net.Data はさらに関数ブロックの終わりを探し、それが見つからない場合には、エラーを発行します。

Net.Data 特殊文字を組み込みプログラム・コードの一部として使用し、Net.Data によって特殊文字として解釈されないようにするためには、以下の方法を使用してください。

- プログラム・コードをインラインに組み込むのではなく、EXEC ステートメントを使用してそのコードを呼び出す。
- 変数参照を使用して特殊文字を指定する。

たとえば、次の Perl 関数には、Perl 言語ステートメントの一部として、COMMENT ブロックの区切り文字を表す文字 %{ が含まれています。

```
%function(DTW_PERL) func() {  
    ...  
    for $num_words (sort bynumber keys %{ $Rtitles{$num} }) {  
        &make_links($Rtitles{$num}{$num_words});  
    }  
    ...  
    %}
```

Net.Data が %{ 文字を Net.Data の COMMENT ブロック区切り文字としてでなく、Perl ソース・コードとして解釈するようにするには、次のいずれかの方法でこの関数を書き直してください。

- %EXEC ステートメントを使用する。

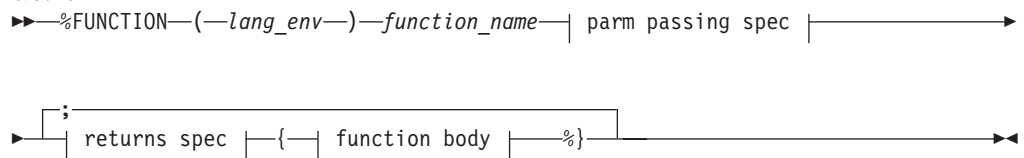
```
%function(DTW_PERL) func() {
    %EXEC{ func.prl %}
%}
```

- 変数参照を使用して %{ 文字を指定する。

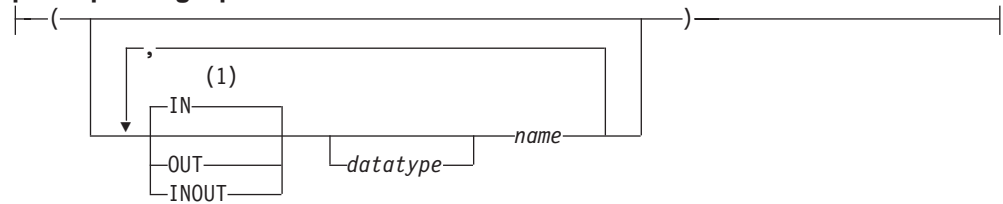
```
%define percent_openbrace = "%{"

%function(DTW_PERL) func() {
    ...
    for $num_words (sort bynumber keys ${percent_openbrace} $Rtitles{$num} }) {
        &make_links($Rtitles{$num}{$num_words});
    }
    ...
%}
```

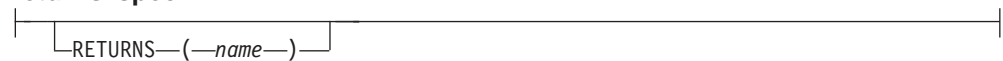
## 構文



### parm passing spec:

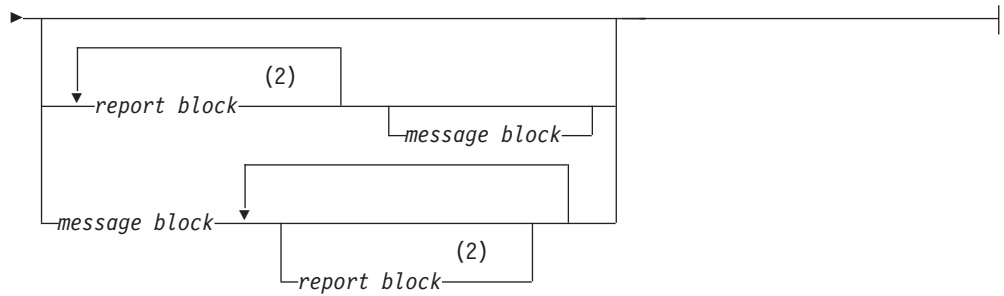


### returns spec:



### function body:





#### 注:

1. パラメーター・リストの先頭でパラメーター型を指定しない場合、デフォルトのパラメーター型の IN が適用されます。パラメーター型を指定していないパラメーターは、パラメーター・リストで最新に指定された型を使用するか、型がそれまでに指定されていない場合には型 IN を使用します。たとえば、パラメーター・リスト (*parm1*, INOUT *parm2*, *parm3*, OUT *parm4*, *parm5*) において、パラメーター *parm1*、*parm3*、および *parm5* にはパラメーター型が指定されていません。パラメーター *parm1* の型は、初期パラメーターが指定されていないため IN になります。パラメーター *parm3* の型は INOUT になりますが、これは、指定された最新のパラメーター型が INOUT であるからです。同様に、パラメーター・リストで指定された最新の型が OUT であるため、パラメーター *parm5* の型は OUT です。
2. 繰り返される report block は、次の場合に有効です。
  - OS/390 オペレーティング・システムでは、複数の結果セットを戻すストアード・プロシージャを処理する際の SQL および ODBC 言語環境。
  - OS/400、OS/2、Windows NT、および UNIX オペレーティング・システムでは、任意の言語環境を呼び出す関数。

#### 値

##### %FUNCTION

Net.Data がマクロ・ファイルから呼び出すサブルーチンを指定するキーワードです。

##### lang\_env

関数本体を処理する言語環境です。詳しくは、*Net.Data* 管理およびプログラミングの手引き を参照してください。

##### function\_name

定義されている関数の名前です。この名前は、英字または下線で開始し、英字、数字、あるいは下線文字を任意の組み合わせで含む英字または数字ストリングにすることができます。

##### parm passing spec:

**IN** Net.Data が入力データを言語環境に渡すことを指定します。IN はデフォルトです。

##### OUT

言語環境が出力データを Net.Data に戻すことを指定します。

## INOUT

Net.Data が入力データを言語環境に渡し、言語環境が出力データを Net.Data に戻すことを指定します。

## datatype

パラメーターのデータ型を指定します。パラメーターのデータ型です。ストアド・プロシージャでサポートされるデータ型の詳細は、*Net.Data* 解説書の「付録 C. Net.DATA のオペレーティング・システムごとの参照」を参照してください。

## name

英字または下線で開始し、英字、数値、あるいは下線文字を任意の組み合わせで含んでいる英字または数値ストリングです。

## returns spec:

## RETURNS

関数が完了した後で、言語環境により割り当てられる関数値を含む変数を宣言します。

## function body:

### inline statement block

関数定義で指定された REXX、SQL、Perl などの言語環境で構文的に有効なステートメントです。使用している言語環境の説明は、*Net.Data* 管理およびプログラミングの手引き を参照してください。構文と使用法については、プログラム言語のプログラミング解説書を参照してください。インライン・ステートメント・ブロックを表すストリングには、*Net.Data* 変数参照と関数呼び出しを含めることができます。これらの変数参照と関数呼び出しは、インライン・ステートメント・ブロック (プログラム) の実行前に評価されます。

### exec block

EXEC ブロックです。変数の参照時または関数の呼び出し時に実行される外部プログラム名です。構文と例については、16ページの『EXEC ブロックまたはステートメント』 を参照してください。

### report block

REPORT ブロックです。関数呼び出しの出力用の形式化指示です。レポート用にヘッダーおよびフッター情報を使用することができます。構文と例については、57ページの『REPORT ブロック』 を参照してください。

### message block

MESSAGE ブロックです。関数呼び出しが戻されるとき、戻りコードの集合、関連メッセージ、および Net.Data が行うアクションです。構文と例については、52ページの『MESSAGE ブロック』 を参照してください。

## コンテキスト

FUNCTION ブロックは、以下のコンテキストで検出することができます。

- IF ブロック
- Net.Data マクロの宣言パーツのすべてのブロックまたはステートメントの外側。

## 制約

- FUNCTION ブロックには、以下の要素を含めることができます。
  - コメント・ブロック
  - EXEC ブロック
  - MESSAGE ブロック
  - REPORT ブロック
  - インライン・ステートメント・ブロック
- Net.Data 変数参照や関数呼び出しの指定がない連続する最長のインライン・ステートメント・ブロック・ストリングは、以下の長さの制限を受けます。
  - OS/2、Windows NT、または UNIX の場合: 無制限
  - OS/390 の場合 : 256KB
  - OS/400 の場合 : 256KB
- インライン・ステートメント・ブロックの SQL ステートメントは、以下の長さをとることができます。データベースによっては異なる制限のある場合があります。データベースのドキュメンテーションを参照して、データベースに制限がないか判断してください。IBM DB2 データベースについて、Net.Data の制限とはことなる点を以下にリストします。
  - OS/2、Windows NT、および UNIX の場合: 64 KB  
DB2 には以下の制限があります。
    - DB2 ユニバーサル・データベース V6 以降: 64 KB
    - DB2 ユニバーサル・データベース V5.2 以前: 32 KB
  - OS/390: 32 KB
  - OS/400: 32 KB

## 例

以下の例は一般的なものであり、すべての言語環境をカバーするわけではありません。特定の言語環境での FUNCTION ブロックの使用についての詳細は、*Net.Data 言語環境解説書* を参照してください。

### 例 1: REXX サブストリング関数

```
%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 %}
```

*a* が評価されると、@substring 関数呼び出しが検索され、サブストリング FUNCTION ブロックが実行されます。変数が FUNCTION ブロックの実行可能ステートメントで置換されてから、テキスト・ストリングの *s* = substr("longstring", 1, 4) が REXX インタープリターに渡されて実行されます。RETURNS 文節が指定してあるため、*a* の評価において @substring 関数呼び出しの値は、"long" (つまり *s* の値) に置き換えられます。

### 例 2: 外部 REXX プログラムの呼び出し

- Net.Data マクロ :

```
%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)
  %}
```

変数 *w* および *x* は、関数の INOUT パラメーター *a* と *b* に対応します。これらの値と *y* の値 (IN パラメーター *c* に対応) は、HTML 形式入力または DEFINE ステートメントで定義しておく必要があります。変数 *a* および *b* には、パラメーター *a* および *b* が値を戻すときに、新規の値が割り当てられます。変数 *z* は、OUT パラメーター *d* が値を戻すときに定義されます。

- REXX プログラム 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;
```

- mypgm.cmd からの出力 :

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

EXEC ステートメントは、REXX インタープリターが外部 REXX プログラム mypgm.cmd を実行するように REXX 言語環境 に命令します。REXX 言語環境は、Net.Data 変数を REXX プログラムと共用できるため、mypgm.cmd を実行する前に、REXX 変数 *a*、*b*、および *c* に Net.Data 変数 *w*、*x*、および *y* の値を割り当てることができます。mypgm.cmd は、変数 *a*、*b*、および *c* を REXX ステートメントで直接使用することができます。プログラムが終了すると、REXX 変数 *a*、*b*、および *d* は REXX プログラムから取り出され、これらの値は Net.Data 変数 *w*、*x*、および *z* に割り当てられます。RETURNS 文節は my\_rexx\_pgm FUNCTION ブロックの定義で使用されていないため、@my\_rexx\_pgm 関数呼び出しの値は、ヌル・ストリング "" (戻りコードが 0 の場合) か、REXX プログラムの戻りコードの値 (戻りコードがゼロ以外の場合) です。

### 例 3: SQL 照会およびレポート

```
%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)
%}

```

@query\_1 関数呼び出しは、SELECT ステートメントで \$(x) を IBM、\$(y) を 12345 に置き換えます。SQL 関数 query\_1 の定義では出力表変数を指定しないため、デフォルトの表が使用されます (詳しくは、TABLE 変数ブロックを参照してください)。REPORT ブロックで参照される NLIST および Vi 変数は、デフォルトの表定義で定義されています。REPORT ブロックで作成されるレポートは、query\_1 関数が呼び出される出力 HTML に置かれます。

#### 例 4: Perl スクリプトを実行するシステム呼び出し

- Net.Data マクロ :

```

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

```

- Perl プログラム today.pr1:

```

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

```

System 言語環境は、C 言語の system() 関数呼び出しを使用して FUNCTION ブロックの実行可能ステートメントをオペレーティング・システムに渡すことによって、これらの実行可能ステートメントを解釈します。この方式では、REXX 言語環境が行うように、Net.Data 変数を実行可能ステートメントへ直接渡したり取り出したりすることができません。このため、System 言語環境は、以下のように変数の受け渡しと取り出しを行います。

- putenv() 関数を使用して入力パラメーターをシステム環境変数として渡し、実行プログラムでこれらの入力パラメーターを取り出すことができます。言語によって変数の参照は異なります。UNIX の cshell スクリプトでは、\$x のように、環境変数名の前に '\$' を付けることによって環境変数を参照します。Perl の言語スクリプトでは、%ENV{'x'} のように結合配列 %ENV を参照することによって環境変数を参照します。DOS のバッチ (.BAT) ファイルでは、%x% のように、パーセント記号で囲んだ変数名を参照します。
- OS/400 プラットフォーム以外では、出力パラメーターは、環境変数 DTWPIPE で名前が渡されるパイプに書き込むことによって言語環境に戻されます。OS/400 プラットフォームでは、出力パラメーターは、システム環境変数として言語環境に戻されます。名前付きパイプに書き込まれるデータの形式は、DEFINE ステートメントの場合と同様に、name="value" です。出力パラメーターに対応する変数名がこの方式で書き込まれると、現行値が新しい値によって置換されます。出力パラメーターに対応しない変数名が書き込まれると、それは無視されます。

@today 関数呼び出しが検出されると、Net.Data は実行可能ステートメントで変数置換を行います。この例では実行可能ステートメントに Net.Data 変数がないため、変数



置換は行われません。実行可能ステートメントとパラメーターは System 言語環境に渡されます。この言語環境は、名前付きパイプを作成し、環境変数 DTWPIPE をパイプ名に設定します。

次に、外部プログラムが C system() 関数呼び出しで呼び出されます。外部プログラムは、書き込み専用としてパイプをオープンし、パイプが標準ストリーム・ファイルであるかのように、出力パラメーターの値をパイプに書き込みます。外部プログラムは、STDOUT に書き込むことにより HTML 出力を生成します。この例では、システム日付プログラムの出力が変数結果に割り当てられます。これは、FUNCTION ブロックの RETURNS 文節で指定される変数です。この変数結果の値は、HTML ブロックの @today() 関数呼び出しを置き換えます。

#### 例 5: Perl 言語環境

```
%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()
}%
```

ブロックの使用法を調べるため、この例と 例 4 を比較します。例 4 では、System 言語環境は Perl プログラムの解釈方法を理解していませんが、言語環境は外部プログラムの呼び出し方法を理解しています。EXEC ブロックは、外部プログラムとして perl という名前のプログラムを呼び出すことを言語環境に命令します。実際の Perl 言語ステートメントは、外部 Perl プログラムによって解釈されます。例 5 には EXEC ブロックがありません。これは、Perl 言語環境が Perl 言語ステートメントを直接解釈することができるからです。

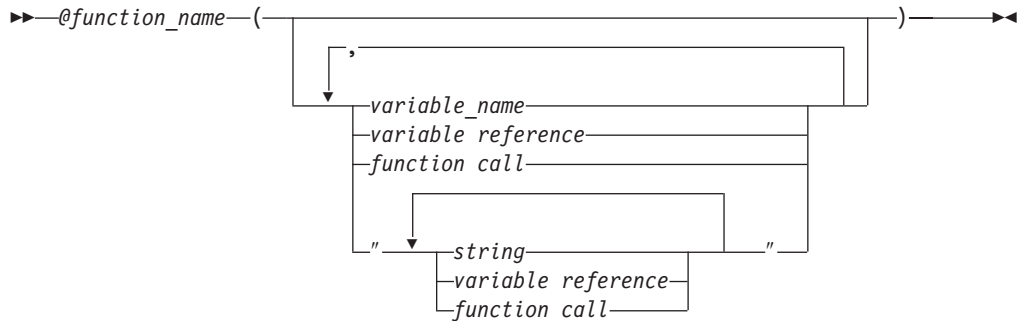


## 関数呼び出し (@)

### 目的

FUNCTION ブロック、MACRO\_FUNCTION ブロック、あるいは引き数を指定した組み込み関数を呼び出します。関数が組み込み関数ではない場合、関数呼び出しを指定する前に、あらかじめ Net.Data マクロでその関数を定義しておかなければなりません。

### 構文



### 値

#### @function\_name

任意の既存の関数名です。英字または下線で開始し、英字、数字、あるいは下線文字を任意の組み合わせで含んでいる英字または数字ストリングです。

#### variable name

変数を識別する名前です。構文情報については、5ページの『変数名』を参照してください。

#### string

任意の順序の英字、数字、および句読点です (改行文字は除く)。

#### variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

#### function call

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した Net.Data 組み込み関数を呼び出します。

### コンテキスト

関数呼び出しは、以下のコンテキストで検出することができます。

- HTML ブロック
- REPORT ブロック
- ROW ブロック
- DEFINE ブロック
- IF ブロック

- MACRO\_FUNCTION ブロック
- MESSAGE ブロック
- WHILE ブロック
- 関数呼び出しステートメント
- Net.Data マクロの宣言パーツのすべてのブロックの外側

## 制約

- 関数呼び出しには、以下の要素を含めることができます。
  - コメント・ブロック
  - スtring
  - 関数呼び出し
  - 変数参照
- OUT または INOUT パラメーター値には、変数参照、関数呼び出し、またはリテラル・Stringを含むことはできません。

## 例

### 例 1: SQL 関数 formQuery の呼び出し

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

%HTML (input){
<P>Which columns of $(tableName) do you want to see?
<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">
%}

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

### 例 2: 入出力パラメーターを指定した REXX 関数の呼び出し

```
%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)
%}
```

### 例 3: 変数参照と関数呼び出しを使用し、入力パラメーターを指定した REXX 関数の呼び出し

```
%FUNCTION(DTW_REXX) my_rexx_pgm(IN a, b, c, d, OUT e) {
  ...
%}

%HTML(INPUT){
  <p> @my_rexx_pgm($(myA), @getB(), @retrieveC(), $(myD), myE)
%}
```

#### 例 4: INOUT パラメーターの使用方を示したマクロ

```
%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>
%}
```

#### 出力結果:

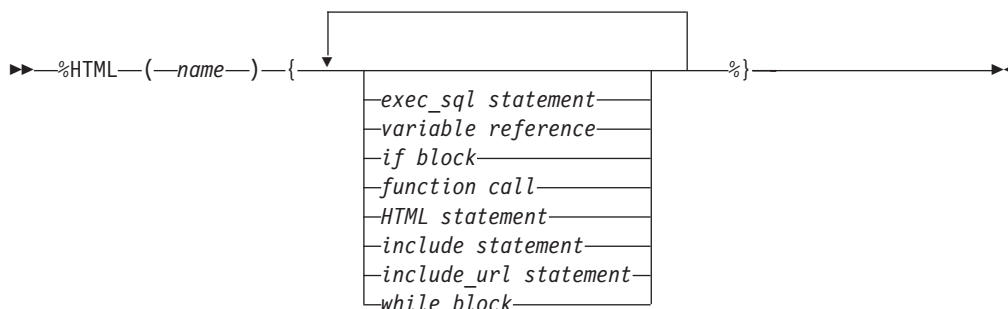
```
initial values:
a = initial value of a
value at start of function:
x = initial value of a
value at start of report block:
x = new value of a
value at end of report block:
x = newest value of a
value after function call:
a = newest value of a
```

## HTML ブロック

### 目的

Web ページの表示方法を定義します。実行する HTML ブロックの名前は、Net.Data の起動時に URL に指定されます。HTML ブロックには、ほとんどの Net.Data マクロ言語ステートメント、および HTML や Javascript などの有効な表示ステートメントを含むことができます。

### 構文



### 値

#### %HTML

クライアントのブラウザーに表示される HTML タグとテキストが入ったブロックを指定するキーワードです。

#### name

英字または下線で開始し、英字、数字、あるいは下線文字 (ピリオドを含む) を任意の組み合わせで含んでいる英字または数字ストリングです。

#### exec\_sql statement

互換性のためにサポートされている DB2WWW リリース 1 の言語要素です。331ページの『付録B. DB2 WWW Connection』または DB2 World Wide Web リリース 1 の資料を参照してください。

#### variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

#### if block

IF ブロックです。条件付きストリング処理を行います。条件リストのストリング値が整数を表すストリングであり、しかもこれらのストリングの先頭または末尾に空白文字がない場合、これらの値は比較のための数値として処理されます。これらの値の先頭に、プラス (+) または マイナス (-) 記号を 1 つ付けることができます。構文と例については、33ページの『IF ブロック』を参照してください。

#### function call

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した Net.Data 組み込み関数を呼び出します。構文と例については、27ページの『関数呼び出し (@)』を参照してください。

## HTML statements

クライアントのブラウザー用に形式化される HTML タグだけでなく、任意の英字または数字を組み込みます。

### include statement

INCLUDE ステートメントです。ファイルを読み取って Net.Data マクロに取り込みます。構文と例については、40ページの『INCLUDE ステートメント』を参照してください。

### include\_url statement

INCLUDE\_URL ステートメントです。別のファイルを読み取って、ステートメントが指定された Net.Data Web マクロに取り込みます。指定されたファイルは、ローカルまたはリモート・サーバーに存在します。構文と例については、43ページの『INCLUDE\_URL ステートメント』を参照してください。

### while block

WHILE ブロックです。条件付きストリング処理を伴うループを実行します。構文と例については、65ページの『WHILE ブロック』を参照してください。

## コンテキスト

HTML ブロックは、以下のコンテキストで検出することができます。

- IF ブロック
- Net.Data マクロの宣言パーツのすべてのブロックの外側

## 制約

HTML ブロックには、以下の要素を含めることができます。

- コメント・ブロック
- EXEC\_SQL ステートメント
- IF ブロック
- HTML ステートメント
- INCLUDE ステートメント
- INCLUDE\_URL ステートメント
- WHILE ブロック
- 変数参照
- 関数呼び出し

## 例

**例 1:** ヘッダーおよびフッター用の組み込みファイルを指定した HTML ブロック

```
%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"  
%}
```

**例 2:** ピリオドを含む名前の HTML ブロック

```
%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:  
@xmp1()  
%INCLUDE"footer.html"  
%}
```

## IF ブロック

### 目的

条件付きストリング処理を行います。IF ブロックにより、1 つまたは複数の条件を処理することができ、続いて条件テストの結果に基づいて、ステートメントのブロックを処理することができます。IF ブロックは、Net.Data マクロの宣言パーツ、HTML ブロック、MACRO\_FUNCTION ブロック、REPORT ブロック、WHILE ブロック、ROW ブロックで使用することも、別の IF ブロックの内部でネストさせることもできます。

条件リストのストリング値が整数を表すストリングであり、しかもこれらのストリングの先頭または末尾に空白文字がない場合、これらの値は比較のための数値として処理されます。これらの値の先頭に、プラス (+) または マイナス (-) 記号を 1 つ付けることができます。

**制約事項:** Net.Data は非整数の数値比較をサポートしません。例、浮動小数点数。

**ネストされた IF ブロック:** IF ブロック構文の規則は、マクロにおけるブロックの位置によって決定されます。IF ブロックが、宣言パーツのほかのすべてのブロックの外側にある IF ブロックの内部でネストされている場合、ネストされた IF ブロックは、外側のブロックが使用することのできる要素をすべて使用できます。IF ブロックが、IF ブロックの中の別のブロック内でネストされている場合、そのネストされた IF ブロックが入っているブロックの構文規則に従います。

以下の例では、ネストされた IF ブロックは、HTML ブロック内で使用される規則に従わなければなりません。

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

IF ブロックは最大 1024 までネストできます。

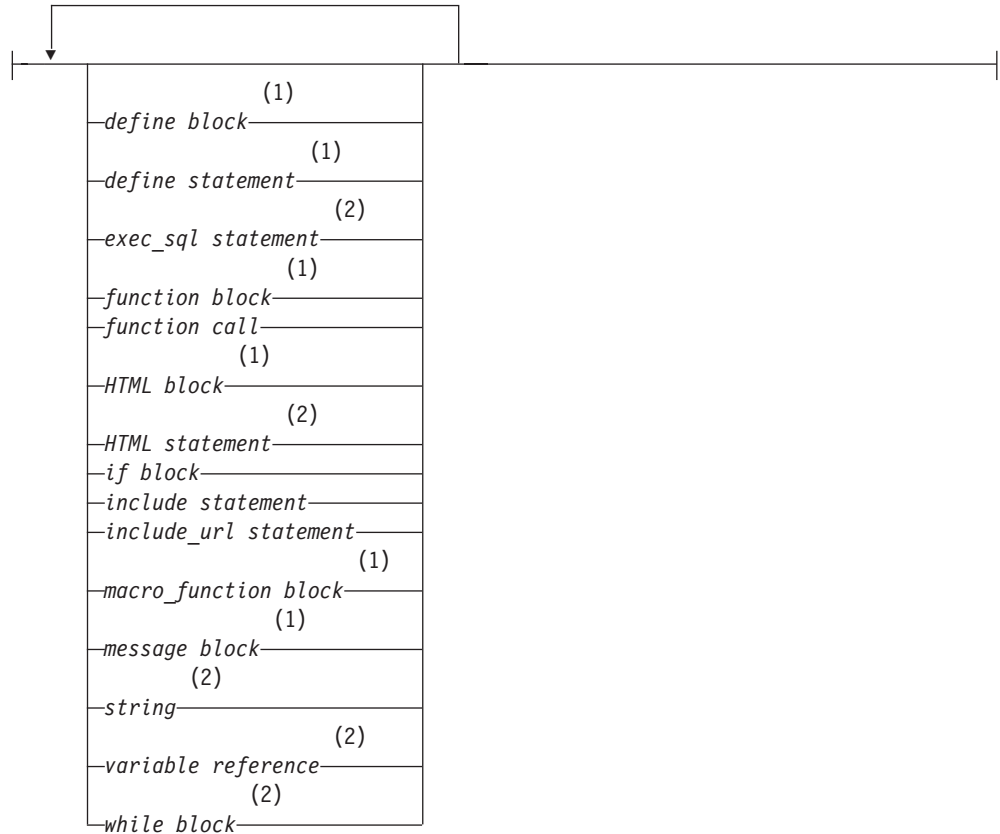
### 構文

►►%IF | condition list | statement\_block | else\_if spec | %ENDIF◄◄

#### condition list:

```
| ( | ( condition list | ) |
| condition list && condition list |
| condition list || condition list |
| ! condition list |
| condition |
| term |
```

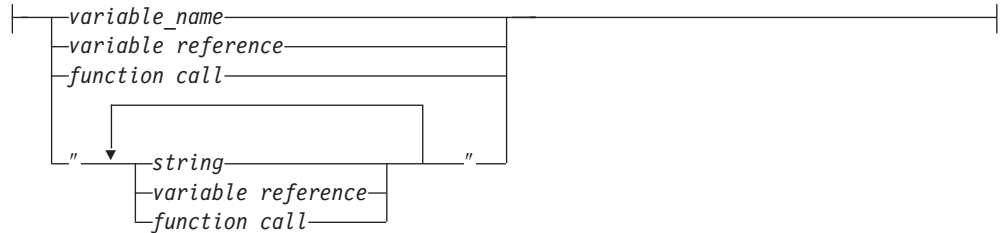
#### statement\_block:



#### condition:

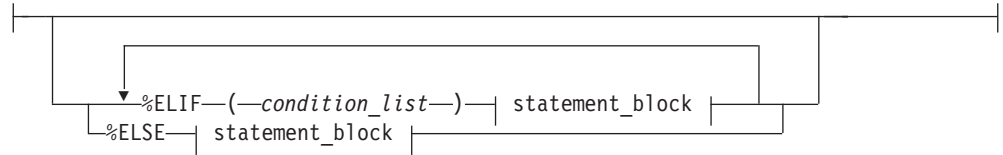


#### term:



#### else\_if spec:





#### 注:

1. この言語構成要素が有効であるのは、IF ブロックがマクロの宣言パーツのほかのすべてのブロックの外側にあるときです。
2. この言語構成要素が有効であるのは、IF ブロックが、HTML ブロック、MACRO\_FUNCTION ブロック、REPORT ブロック、ROW ブロック、または WHILE ブロックに入っているときです。

## 値

### %IF

条件付きストリング処理を指定するキーワードです。

#### condition list

条件と条件の値を比較します。条件リストは、ブール演算子を使って接続することができます。条件リストは、別の条件リスト内でネストすることができます。

#### statement\_block

有効な Net.Data マクロ構成要素は以下のとおりです。図の注および制約事項を参照して、マクロ構成要素が有効になるコンテキストを判別してください。

#### define statement

DEFINE ブロックまたはステートメントです。変数を定義して、構成変数を設定します。変数名は、文字または下線 ( \_ ) で開始し、任意の英数字または下線を含む必要があります。構文と例については、11ページの『DEFINE ブロックまたはステートメント』を参照してください。

#### exec\_sql statement

互換性のためにサポートされている DB2WWW リリース 1 の言語要素です。331ページの『付録B. DB2 WWW Connection』または DB2 World Wide Web リリース 1 の資料を参照してください。

#### function block

Net.Data マクロから呼び出すことのできるサブルーチンを指定するキーワードです。FUNCTION ブロックの実行可能ステートメントは、言語環境により直接解釈される言語ステートメントを含んだり、外部プログラムの呼び出しを指示することができます。構文と例については、19ページの『FUNCTION ブロック』を参照してください。

#### function call

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した Net.Data 組み込み関数を呼び出します。構文と例については、27ページの『関数呼び出し (@)』を参照してください。

#### HTML block

クライアントのブラウザー用に形式化される HTML タグだけでなく、任意の英字または数字を組み込みます。

### HTML statement

任意の英字や数字と、クライアントのブラウザ用に形式化される HTML タグを組み込みます。

### if block

IF ブロックです。条件付きストリング処理を行います。条件リストのストリング値が整数を表すストリングであり、しかもこれらのストリングの先頭または末尾に空白文字がない場合、これらの値は比較のための数値として処理されます。これらの値の先頭に、プラス (+) または マイナス (-) 記号を 1 つ付けることができます。

### include statement

INCLUDE ステートメントです。ファイルを読み取って Net.Data マクロに取り込みます。構文と例については、40ページの『INCLUDE ステートメント』を参照してください。

### include\_url statement

INCLUDE\_URL ステートメントです。別のファイルを読み取って、ステートメントが指定された Net.Data Web マクロに取り込みます。指定されたファイルは、ローカルまたはリモート・サーバーに存在します。構文と例については、43ページの『INCLUDE\_URL ステートメント』を参照してください。

### macro\_function block

Net.Data マクロから呼び出すことのできるサブルーチンを指定するキーワードです。MACRO\_FUNCTION ブロックの実行可能ステートメントには、Net.Data マクロ言語ソース・ステートメントを含めることができます。構文と例については、48ページの『MACRO\_FUNCTION ブロック』を参照してください。

### message block

MESSAGE ブロックです。関数呼び出しが戻されるときの、戻りコードの集合、関連メッセージ、および Net.Data が行うアクションです。構文と例については、52ページの『MESSAGE ブロック』を参照してください。

### string

任意の順序の英字、数字、および句読点です。ストリングが条件リストの条件に含まれている場合、そのストリングには、改行文字以外のすべての文字を含めることができます。ストリングが実行可能ブロックのコードに含まれている場合、そのストリングには、改行文字を含むすべての文字を含めることができます。

### variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

### while block

WHILE ブロックです。条件付きストリング処理を伴うループを実行します。構文と例については、65ページの『WHILE ブロック』を参照してください。

### condition

比較演算子を使用した、2 つの条件の間の比較です。以下の両方の条件が当てはまる場合、IF 条件は数値比較として処理されます。

- 条件演算子が以下のいずれかである：<、<=、>、>=、==、!=
- 条件とストリングが両方とも有効整数を表している。ここで、有効整数とは、オプションでプラス (+) またはマイナス (-) 記号が先頭に付くことがあり、ほかに空白文字が入っていない数字のストリングのことです。

どちらかの条件が当てはまらない場合、標準のストリング比較が行われます。

#### **term**

変数名、ストリング、変数参照、または関数呼び出しです。

#### **%ELIF**

代替処理パスを開始するキーワードのことで、このキーワードには、条件リストとほとんどの Net.Data マクロ・ステートメントを含めることができます。

#### **%ENDIF**

%IF ブロックをクローズするキーワードです。

#### **%ELSE**

ほかのすべての条件リストが満たされない場合、関連ステートメントを実行するキーワードです。

### **コンテキスト**

IF ブロックは、以下のコンテキストで検出することができます。

- Net.Data マクロの宣言パーツのほかのすべてのブロックの外側
- HTML ブロック
- IF ブロック
- MACRO\_FUNCTION ブロック
- REPORT ブロック
- ROW ブロック
- WHILE ブロック

### **制約**

IF ブロックが Net.Data マクロの宣言パーツのほかのすべてのブロックの外側にあるときには、IF ブロックに以下の要素を含めることができます。

- コメント・ブロック
- DEFINE ブロック
- DEFINE ステートメント
- FUNCTION ブロック
- 関数呼び出し
- HTML ブロック
- IF ブロック
- INCLUDE ステートメント
- INCLUDE\_URL ステートメント
- MACRO\_FUNCTION ブロック
- MESSAGE ブロック
- 変数参照

IF ブロックが、Net.Data マクロの HTML ブロック、MACRO\_FUNCTION ブロック、REPORT ブロック、ROW ブロック、または WHILE ブロックに入っているときには、IF ブロックに以下の要素を含めることができます。

- コメント・ブロック
- EXEC\_SQL ステートメント
- 関数呼び出し
- IF ブロック
- INCLUDE ステートメント
- INCLUDE\_URL ステートメント
- HTML ステートメント
- スtring
- 変数参照
- WHILE ブロック

IF ブロックは最大 1024 までネストできます。

## 例

**例 1:** Net.Data マクロの宣言パーツにある IF ブロック

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

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

**例 2:** HTML ブロック内にある IF ブロック

```
%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
%}
```

**例 3:** 数値比較

```
%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
```

暗黙の表変数 `ROW_NUM` は常に整数値を戻し、比較対象の値も整数であるため、数値比較が行われます。

#### 例 4: ネストされた IF ブロック

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

# INCLUDE ステートメント

## 目的

ファイルを読み取って、ステートメントが指定された Net.Data マクロに取り込みます。

Net.Data は、初期設定ファイルの INCLUDE\_PATH ステートメントで指定したディレクトリを検索して、組み込みファイルを見つけます。

ほとんどの高水準言語で使用するのと同じ方法で、組み込みファイルを使用することができます。組み込みファイルでは、共通ヘッダーおよびフッターを挿入したり、共通の変数の集合を定義したり、FUNCTION ブロック定義の共通サブルーチン・ライブラリーを Net.Data マクロに取り込むことができます。

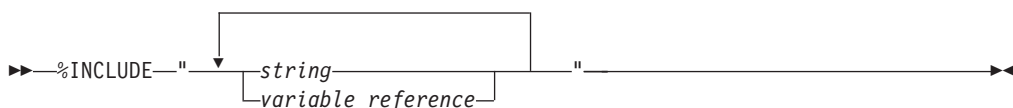
Net.Data は、マクロを処理しているときに一度だけ INCLUDE ステートメントを実行し、マクロ内の INCLUDE ステートメントのロケーションに組み込みファイルの内容を挿入します。組み込みファイルの名前の変数参照が解決されるのは、INCLUDE ステートメントを最初に行うときで、組み込みファイルの内容を実行するときではありません。

INCLUDE ステートメントが ROW ブロックまたは WHILE ブロック内にあるときには、Net.Data は INCLUDE ステートメントを繰り返し実行しません。Net.Data は、最初に ROW ブロックまたは WHILE ブロックを実行するときに、INCLUDE ステートメントを実行し、組み込みファイルの内容をブロックに取り込み、その後、組み込みファイルの内容をもつ ROW ブロックまたは WHILE ブロックを繰り返し実行します。

**許可のヒント** Net.Data が実行されるユーザー ID が、INCLUDE ステートメントによって参照されるどのファイルについてもアクセス権を持つようにしてください。詳しくは、*Net.Data* 管理およびプログラミングの手引き の構成の章にある、Net.Data ファイルへの Web サーバーのアクセス権限の指定に関するセクションを参照してください。

**ヒント** ローカル Web サーバーから HTML ファイルを組み込みたい場合には、INCLUDE\_URL の例 3 に示すように INCLUDE\_URL 構成要素を使用してください。表示されている構文を使用することにより、すでに Web サーバーに認識されているディレクトリを指定するために、Net.Data 初期設定ファイルの INCLUDE\_PATH を更新する必要はありません。

## 構文



## 値

### %INCLUDE

ファイルを読み取って Net.Data マクロに取り込むことを示すキーワードです。

### name

英字または下線で開始し、英字、数値、あるいは下線文字を任意の組み合わせで含んでいる英字または数値ストリングです。

### string

任意の順序の英字、数字、および句読点です (改行文字は除く)。

### variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

## コンテキスト

INCLUDE ステートメントは、以下のコンテキストで検出することができます。

- DEFINE ブロック
- HTML ブロック
- REPORT ブロック
- ROW ブロック
- IF ブロック
- MESSAGE ブロック
- MACRO\_FUNCTION ブロック
- WHILE ブロック
- Net.Data マクロの宣言パーツのすべてのブロックの外側

## 制約

INCLUDE ステートメントには、以下の要素を含めることができます。

- コメント・ブロック
- ストリング
- 変数参照

ストリング内での関数呼び出しは許可されません。

INCLUDE ステートメントは最大 10 までネストできます。

## 例

**例 1:** HTML ブロック内の INCLUDE ステートメント

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

**例 2:** REPORT ブロック内の INCLUDE ステートメント

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

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

**例 3:** INCLUDE ステートメント内の変数参照

```
%define library = "/qsys.lib/mylib.lib/"  
%define filename = "macros.file/incfile.mbr"  
  
%include "$(library)$(filename)"
```



## INCLUDE\_URL ステートメント

### 目的

別のファイルを読み取って、ステートメントが指定された Net.Data 生成の出力に取り込みます。指定されたファイルは、ローカルまたはリモート・サーバーに存在します。

INCLUDE\_URL ステートメントを使用すると、アプリケーション・ユーザーがサブミット・ボタンを選択しなくても、別のマクロから 1 つのマクロを呼び出すことができます。

Net.Data は、マクロを処理しているときに一度だけ INCLUDE\_URL ステートメントを実行し、マクロ内の INCLUDE\_URL ステートメントのロケーションに組み込みファイルの内容を挿入します。組み込みファイルの名前の変数参照が解決されるのは、INCLUDE\_URL ステートメントを最初に実行するときで、組み込みファイルの内容を実行するときではありません。

INCLUDE\_URL ステートメントが ROW ブロックまたは WHILE ブロック内にあるときには、Net.Data は INCLUDE\_URL ステートメントを繰り返し実行しません。Net.Data は、最初に ROW ブロックまたは WHILE ブロックを実行するときに、INCLUDE\_URL ステートメントを実行し、組み込みファイルの内容をブロックに取り込み、その後、組み込みファイルの内容をもつ ROW ブロックまたは WHILE ブロックを繰り返し実行します。

### 構文



### 値

#### %INCLUDE\_URL

ローカルまたはリモート・サーバーからファイルを読み取って、そのファイルを Net.Data マクロに取り込むことを示すキーワードです。

#### string

任意の順序の英字、数字、および句読点です (改行文字は除く)。

#### variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5 ページの『変数参照』を参照してください。

### コンテキスト

INCLUDE\_URL ステートメントは、以下のコンテキストで検出することができます。

- DEFINE ブロック
- HTML ブロック

- IF ブロック
- MACRO\_FUNCTION ブロック
- MESSAGE ブロック
- REPORT ブロック
- ROW ブロック
- WHILE ブロック
- Net.Data マクロの宣言パーツのすべてのブロックの外側

## 制約

INCLUDE\_URL ステートメントには、以下の要素を含めることができます。

- コメント・ブロック
- スtring
- 変数参照

OS/390 では、INCLUDE\_URL ファイルは最大 256 KB とすることができます。他のオペレーティング・システムでは制限はありません。

INCLUDE\_URL ステートメントを使用する場合、現行のマクロ・ファイルを再帰的に呼び出してマクロ要求の無限シーケンスを開始しないようにしてください。

INCLUDE\_URL は、OS/400 環境ではサポートされていません。

## 例

**例 1:** 別のサーバーからの HTML ファイルの組み込み

```
%include_url "http://www.ibm.com/path/myfile.html"
```

**例 2:** サーバー名を呼び出すことによる、リモート・サーバーからの HTML ファイルの組み込み

```
%include_url "myserver/path/myfile.html"
```

ここで、myserver はサーバー名です。

**例 3:** ローカル Web サーバーからの HTML ファイルの組み込み

```
%include_url "/path/myfile.html"
```

**ヒント:** この方法を使うことにより、すでに Web サーバーに認識されているディレクトリーを指定するために、Net.Data 構成ファイルの INCLUDE\_URL パスを更新する必要はありません。string がスラッシュで始まっていない場合、Net.Data は、String がサーバー名であると推定し、対応する名前のサーバーからファイルを取り出そうとします。

**例 4:** リモート・サーバーからのほかの Net.Data マクロの組み込み

```
%REPORT{
<P>Current hot pick as of @DTW_rTIME():
%include_url "http://www.ibm.com/cgi-bin/db2www/hotpic.mac/report?custno=$(custno)"
```

この例では、マクロ hotpic.mac が呼び出され、custno が変数として送られます。string がスラッシュで始まっている場合、Net.Data は、ローカル Web サーバーから

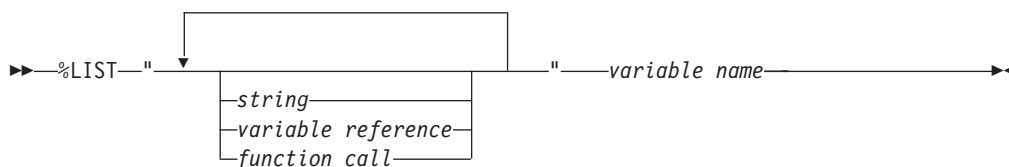
INCLUDE ファイルを取り出します。

## LIST ステートメント

### 目的

値の区切りリストを作成します。一部の WHERE または HAVING 文節に見られるように、複数の項目を指定して SQL 照会を作成するときに、LIST ステートメントを使用することができます。

### 構文



### 値

#### %LIST

値の区切りリストを作成するのに使用する変数を指定するキーワードです。

#### string

任意の順序の英字、数字、および句読点です (改行文字は除く)。

#### variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

#### function call

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した Net.Data 組み込み関数を呼び出します。構文と例については、27ページの『関数呼び出し (@)』を参照してください。

#### variable name

変数を識別する名前です。構文情報については、5ページの『変数名』を参照してください。

### コンテキスト

LIST ステートメントは、以下のコンテキストで検出することができます。

- DEFINE ステートメント

### 制約

LIST ステートメントには、以下の要素を含めることができます。

- コメント・ブロック
- 変数参照
- 関数呼び出し
- スtring

## 例

### 例 1: 変数のリスト

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

## MACRO\_FUNCTION ブロック

### 目的

Net.Data マクロから呼び出すことのできるサブルーチンを定義します。  
MACRO\_FUNCTION ブロックの実行可能ステートメントは、Net.Data マクロ言語ソース・ステートメントでなければなりません。

### 構文

▶▶ `%MACRO_FUNCTION` *function\_name* | parm passing spec |

(4)  
| returns spec | { | function body | %} |

(3)  
| report block | %} |

#### parm passing spec:

(1)  
| IN |  
| OUT |  
| INOUT |  
| name |

#### returns spec:

(4)  
| RETURNS ( *name* ) |

#### function body:

(2)  
| exec\_sql statement |  
| variable reference |  
| if block |  
| function call |  
| HTML statement |  
| include statement |  
| include\_url statement |  
| while block |

**注:**

1. パラメーター・リストの先頭でパラメーター型を指定しない場合、デフォルトのパラメーター型の IN が適用されます。パラメーター型を指定していないパラメーターは、パラメーター・リストで最新に指定された型を使用するか、型がそれまでに指定されていない場合には型 IN を使用します。たとえば、パラメーター・リスト (*parm1*, INOUT *parm2*, *parm3*, OUT *parm4*, *parm5*) において、パラメーター *parm1*、*parm3*、および *parm5* にはパラメーター型が指定されていません。パラメーター *parm1* の型は、初期パラメーターが指定されていないため IN になります。パラメーター *parm3* の型は INOUT になりますが、これは、指定された最新のパラメーター型が INOUT であるからです。同様に、パラメーター・リストで指定された最新の型が OUT であるため、パラメーター *parm5* の型は OUT です。
2. OS/400 では INCLUDE\_URL ステートメントをサポートしません。
3. MACRO\_FUNCTION ブロックにおける REPORT ブロックは、OS/400、OS/2、Windows NT および UNIX でサポートされます。
4. RETURNS ステートメントは、OS/400 のみのサポートです。

**値**

**%MACRO\_FUNCTION**

Net.Data マクロから呼び出すことのできるサブルーチンを指定するキーワードです。MACRO\_FUNCTION ブロックの実行可能ステートメントには、Net.Data が直接解釈する言語ステートメントを含めなければなりません。

**function\_name**

定義されている関数の名前です。英字または下線で開始し、英字、数字、あるいは下線文字を任意の組み合わせで含んでいる英字または数字ストリングです。

**parm passing spec:**

**IN** Net.Data が入力データを言語環境に渡すことを指定します。IN はデフォルトです。

**OUT**

言語環境が出力データを Net.Data に戻すことを指定します。

**INOUT**

Net.Data が入力データを言語環境に渡し、言語環境が出力データを Net.Data に戻すことを指定します。

**name**

英字または下線で開始し、英字、数値、あるいは下線文字を任意の組み合わせで含んでいる英字または数値ストリングです。*name* は、Net.Data 表または結果セットを表すことができます。

**returns spec:**

**RETURNS**

関数が完了した後で、関数値を含む変数を宣言します。

**function body:**

**exec\_sql**

互換性のためにサポートされている DB2WWW リリース 1 の言語要素です。

331ページの『付録B. DB2 WWW Connection』 または DB2 World Wide Web リリース 1 の資料を参照してください。

#### **variable reference**

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』 を参照してください。

#### **if block**

IF ブロックです。条件付きストリング処理を行います。条件リストのストリング値が、先頭または末尾に空白文字がない整数を表す場合、これらの値は比較のための数値として処理されます。これらの値の先頭に、プラス (+) または マイナス (-) 記号を 1 つ付けることができます。

#### **function call**

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した Net.Data 組み込み関数を呼び出します。構文と例については、27ページの『関数呼び出し (@)』 を参照してください。

#### **HTML statement**

クライアントのブラウザ用に形式化される HTML タグだけでなく、任意の英字または数字を組み込みます。

#### **include statement**

INCLUDE ステートメントです。ファイルを読み取って Net.Data マクロに取り込みます。構文と例については、40ページの『INCLUDE ステートメント』 を参照してください。

#### **include\_url statement**

INCLUDE\_URL ステートメントです。別のファイルを読み取って、ステートメントが指定された Net.Data マクロに取り込みます。指定されたファイルは、ローカルまたはリモート・サーバーに存在します。構文と例については、43ページの『INCLUDE\_URL ステートメント』 を参照してください。

#### **while block**

WHILE ブロックです。条件付きストリング処理を伴うループを実行します。構文と例については、65ページの『WHILE ブロック』 を参照してください。

#### **report block**

REPORT ブロックです。関数呼び出しの出力用の形式化指示です。レポート用にヘッダーおよびフッター情報を使用することができます。構文と例については、57ページの『REPORT ブロック』 を参照してください。

## **コンテキスト**

MACRO\_FUNCTION ブロックは、以下のコンテキストで検出することができます。

- IF ブロック
- Net.Data マクロの宣言パーツのすべてのブロックの外側

## **制約**

MACRO\_FUNCTION ブロックには、以下の要素を含めることができます。

- コメント・ブロック



- EXEC\_SQL ステートメント
- HTML ステートメント
- IF ブロック
- INCLUDE ステートメント
- INCLUDE\_URL ステートメント  
OS/400 ではサポートなし
- REPORT ブロック
- WHILE ブロック
- 変数参照
- 関数呼び出し

## 例

### 例 1: メッセージ処理を指定するマクロ関数

```
%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
%}
```

### 例 2: ヘッダー情報を指定するマクロ関数

```
%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
%}
```

## MESSAGE ブロック

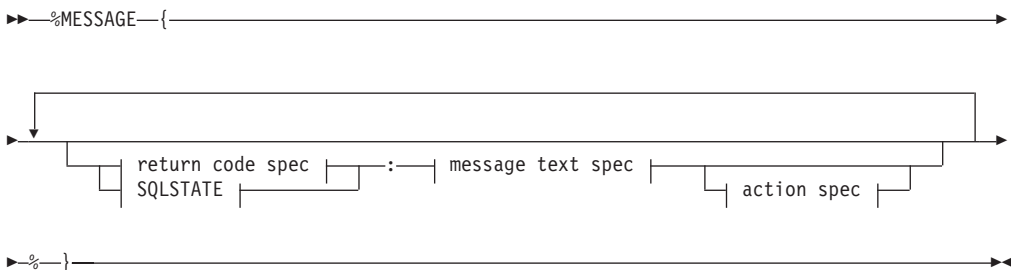
### 目的

関数からの戻りコードに基づいて、表示するメッセージと行うアクションを指定します。

戻りコードのセットと、それらのコードに対応するメッセージとアクションとを MESSAGE ブロックに定義します。関数呼び出しが完了すると、*Net.Data* は、その戻りコードと MESSAGE ブロックで定義された戻りコードとを比較します。関数の戻りコードが MESSAGE ブロックの戻りコードと一致すると、*Net.Data* はメッセージを表示してアクションを評価して、*Net.Data* マクロの処理を続行するか終了するかを判別します。

MESSAGE ブロックは、その効力範囲をグローバルにすることも、1つの FUNCTION ブロック対象にローカルにすることもできます。MESSAGE ブロックが最外部のマクロ・レイヤーで定義してある場合、そのブロックの効力範囲はグローバルであると考えられます。複数のグローバル MESSAGE ブロックが定義されている場合、処理された最新のブロックだけがアクティブであると考えられます。MESSAGE ブロックが FUNCTION ブロック内で定義されている場合、そのブロックの効力範囲は、定義されている FUNCTION ブロック対象にローカルとなります。戻りコードの処理規則については、*Net.Data* 管理およびプログラミングの手引き の MESSAGE ブロックのセクションを参照してください。

### 構文



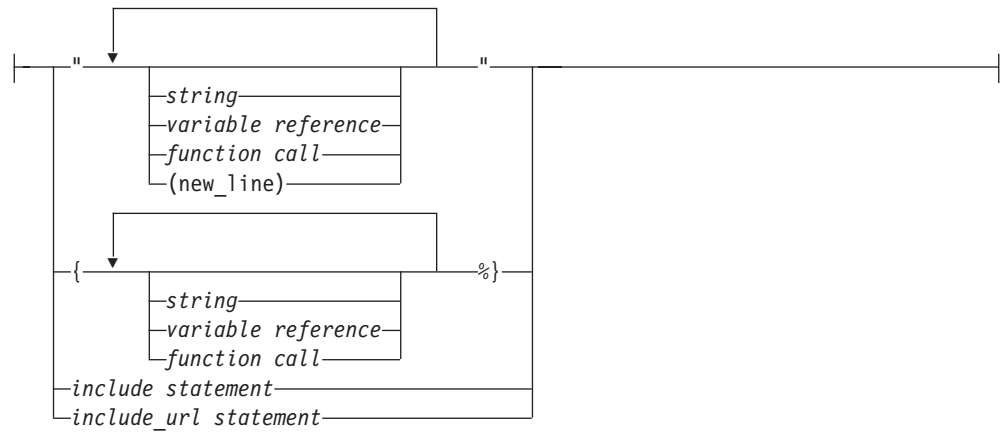
#### return code spec:



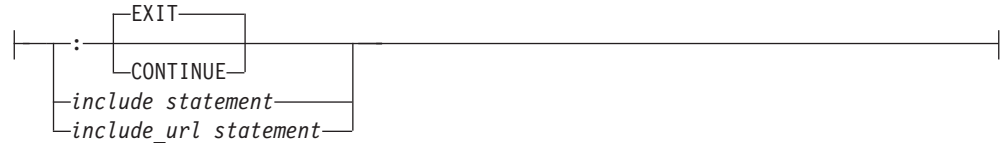
#### SQLSTATE:



### message text spec:



### action spec:



## 値

### %MESSAGE

関数呼び出しが戻されるときの、戻りコードの集合、関連メッセージ、および Net.Data が行うアクションを定義するブロック用のキーワードです。

### return code spec

正または負の整数です。 Net.Data RETURN\_CODE 変数の値が *return code spec* 値に一致すると、メッセージ・ステートメントの残りの情報を使用して、関数呼び出しを処理します。 MESSAGE ブロックで特に入力されない戻りコードのメッセージを指定することもできます。

### +DEFAULT

デフォルトの正のメッセージ・コードを指定するのに使用するキーワードです。 Net.Data は、RETURN\_CODE がゼロ (0) よりも大きく、しかも完全に一致するコードが指定されていない場合に、このメッセージ・ステートメントの情報を使用して関数呼び出しを処理します。

### -DEFAULT

デフォルトの負のメッセージ・コードを指定するのに使用するキーワードです。 Net.Data は、RETURN\_CODE がゼロ (0) よりも小さく、しかも完全に一致するコードが指定されていない場合に、このメッセージ・ステートメントの情報を使用して関数呼び出しを処理します。

### DEFAULT

デフォルトのメッセージ・コードを指定するキーワードです。 Net.Data は、以下のすべての条件が満たされた場合に、このメッセージ・ステートメントの情報を使用して、関数呼び出しを処理します。

- RETURN\_CODE がゼロではなく、ゼロより大きい小さい

- 戻りコードに完全に一致するものが指定されていない
- RETURN\_CODE がゼロより大きい小さいときのために、+DEFAULT や -DEFAULT 値が指定されていない

#### **msg\_code**

処理中に起こる可能性のあるエラーと警告を指定するメッセージ・コードです。0 から 9 までの値の数字のストリングです。

#### **SQLSTATE**

アプリケーション・プログラムに共通エラー条件用の共通コードを提供するキーワードです。SQLSTATE 値は SQL 標準に含まれている SQLSTATE 指定に基づいていて、コード体系は IBM のすべての SQL の設定で同じです。

#### **state\_id**

SQLSTATE です。ccsss 形式をとる 5 文字 (バイト) の英数字です。cc はクラス、sss はサブクラスを表します。

#### **message text spec**

RETURN\_CODE が現行メッセージ・ステートメントの *return\_code* 値と一致した場合に、Web ブラウザーに送られるストリングです。

#### **string**

任意の順序の英字、数字、および句読点です。ストリングが二重引用符内にある場合、改行文字は使用できません。

#### **variable reference**

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5 ページの『変数参照』を参照してください。

#### **function call**

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した Net.Data 組み込み関数を呼び出します。構文と例については、27 ページの『関数呼び出し (@)』を参照してください。

#### **action spec**

RETURN\_CODE が現行メッセージ・ステートメントの *return\_code* 値と一致した場合に、Net.Data が行うアクションを決定します。

#### **EXIT**

指定されたメッセージ・コードに対応するエラーまたは警告が生じたときに、マクロを即時に終了することを指定するキーワードです。この値がデフォルトです。

#### **CONTINUE**

指定されたメッセージ・コードに対応するエラーまたは警告が生じたときに、処理を続行することを指定するキーワードです。

#### **include statement**

INCLUDE ステートメントです。ファイルを読み取って Net.Data マクロに取り込みます。INCLUDE ステートメントは、MESSAGE のどこでも指定することができます。構文と例については、40 ページの『INCLUDE ステートメント』を参照してください。

#### **include\_url statement**

INCLUDE\_URL ステートメントです。別のファイルを読み取って、ステート

メントが指定された Net.Data マクロに取り込みます。指定されたファイルは、ローカルまたはリモート・サーバーに存在します。構文と例については、43ページの『INCLUDE\_URL ステートメント』を参照してください。

## コンテキスト

MESSAGE ブロックは、以下のコンテキストで検出することができます。

- FUNCTION ブロック
- IF ブロック
- Net.Data マクロの宣言パーツのすべてのブロックまたはステートメントの外側

## 制約

MESSAGE ブロックには、以下の要素を含めることができます。

- コメント・ブロック
- 関数呼び出し
- 変数参照
- HTML ステートメント
- スtring
- INCLUDE ステートメント
- INCLUDE\_URL ステートメント

**OS/390、OS/2、Windows NT、および UNIX オペレーティング・システムの場合:**  
SQL 関数内部から SQL 関数を呼び出すことはできません。

## 例

### 例 1: ローカル MESSAGE ブロック

```
%{ 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  
    %}
```

### 例 2: グローバル MESSAGE ブロック

```
%{ 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
%}

```

### 例 3: INCLUDE ステートメントを含む MESSAGE ブロック

```

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

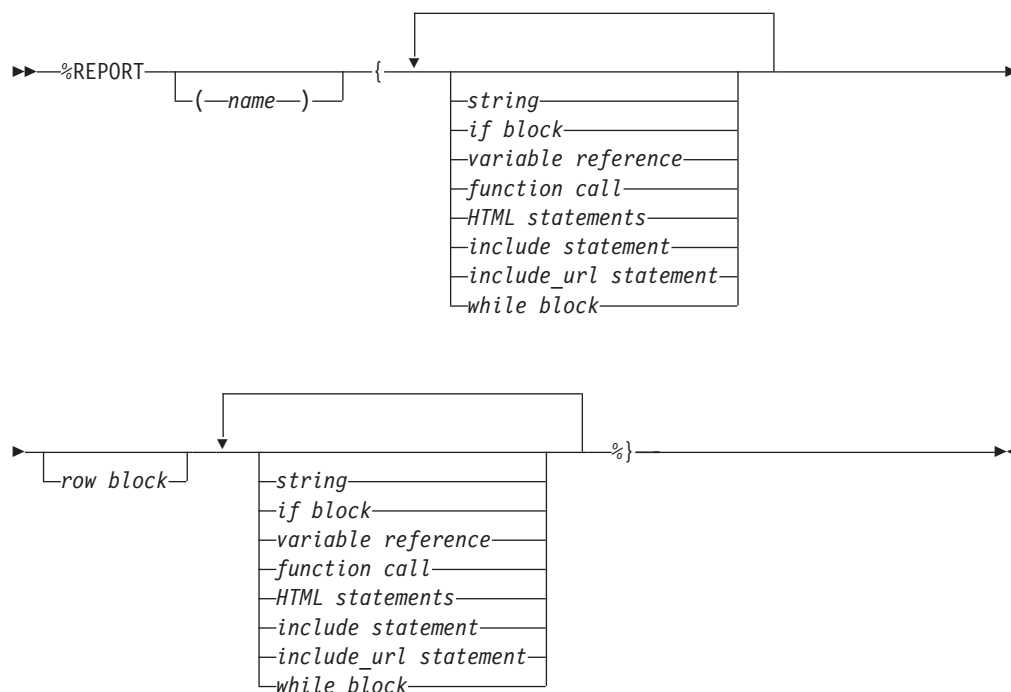
```

# REPORT ブロック

## 目的

関数呼び出しからの出力を形式化します。表名パラメーターを入力すると、指定された表でレポートがデータを使用するのを指定することができます。そうしないと、レポートは、関数パラメーター・リストで最初に検出された出力表で生成されます。また、リストに表名がない場合、レポートはデフォルトの表データで生成されます。

## 構文



## 値

### %REPORT

関数呼び出しの出力用の形式化指示を指定するキーワードです。レポート用にヘッダーおよびフッター情報を使用することができます。

### name

この値は、Net.Data 表または結果セットを表します。詳細については、*Net.Data* 管理およびプログラミングの手引き の Report ブロックのセクションを参照してください。

### string

任意の順序の英字、数字、および句読点です。

### if block

IF ブロックです。条件付きストリング処理を行います。条件リストのストリング値が、先頭または末尾に空白文字がない整数を表す場合、これらの値は比較のた

めの数値として処理されます。これらの値の先頭に、プラス (+) または マイナス (-) 記号を 1 つ付けることができます。構文と例については、33ページの『IF ブロック』を参照してください。

#### **variable reference**

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

#### **function call**

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した Net.Data 組み込み関数を呼び出します。構文と例については、27ページの『関数呼び出し (@)』を参照してください。

#### **HTML statements**

クライアントのブラウザー用に形式化される HTML タグだけでなく、任意の英字または数字を組み込みます。

#### **include statement**

INCLUDE ステートメントです。ファイルを読み取って Net.Data マクロに取り込みます。構文と例については、40ページの『INCLUDE ステートメント』を参照してください。

#### **include\_url statement**

INCLUDE\_URL ステートメントです。別のファイルを読み取って、ステートメントが指定された Net.Data マクロに取り込みます。指定されたファイルは、ローカルまたはリモート・サーバーに存在します。構文と例については、43ページの『INCLUDE\_URL ステートメント』を参照してください。

#### **row block**

ROW ブロックです。関数呼び出しから戻されるデータの行ごとに、HTML 形式化データを 1 度表示します。構文と例については、60ページの『ROW ブロック』を参照してください。

#### **while block**

WHILE ブロックです。条件付きストリング処理を伴うループを実行します。構文と例については、65ページの『WHILE ブロック』を参照してください。

### **コンテキスト**

REPORT ブロックは、以下のコンテキストで検出することができます。

- FUNCTION ステートメントまたはブロック

#### **制約**

REPORT ブロックには、以下の要素を含めることができます。

- コメント・ブロック
- IF ブロック
- INCLUDE ステートメント
- INCLUDE\_URL ステートメント
- ROW ブロック
- WHILE ブロック



- 関数呼び出し

**OS/390、OS/2、Windows NT、および UNIX オペレーティング・システムの場合:** SQL 関数内部から SQL 関数を呼び出すことはできません。

- HTML ステートメント
- スtring
- 変数参照

**OS/390 の場合:** REPORT ブロックは、MACRO\_FUNCTION ブロックでは使用できません。

## 例

**例 1:** 名前と場所のリストを表示する 2 列の HTML 表

```
%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=$(V1)&location=$(V2)">$(V1)</a></TD>
    <TD>$(V2)</TD>
    %}
    </TABLE>
  %}
```

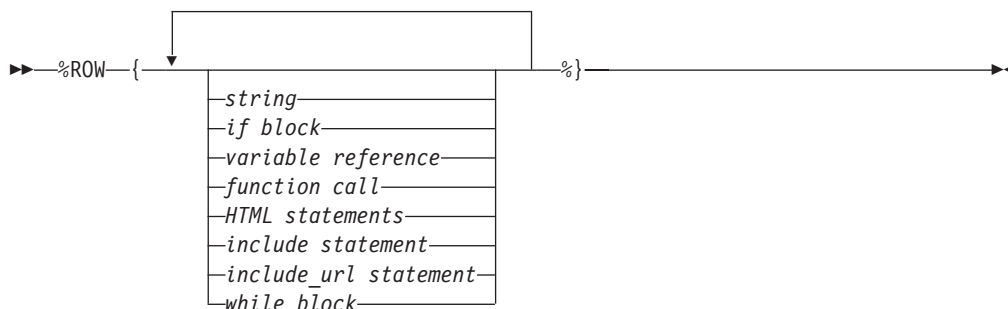
表の名前を選択すると、*name.mac* Net.Data マクロの *details* HTML ブロックが呼び出され、URL のパーツとして 2 つの値が送信されます。この例では、*name.mac* で値を使用すると、名前に関する詳細をルックアップすることができます。

## ROW ブロック

### 目的

関数呼び出しから戻されるそれぞれの表行を処理します。Net.Data は、それぞれの行ごとに ROW ブロック内のステートメントを 1 度処理します。

### 構文



### 値

#### %ROW

関数呼び出しから戻されるデータの行ごとに、HTML 形式化データを 1 度表示することを指定するキーワードです。

#### string

任意の順序の英字、数字、および句読点です。

#### if block

IF ブロックです。条件付きストリング処理を行います。条件リストのストリング値が整数を表すストリングであり、しかもこれらのストリングの先頭または末尾に空白文字がない場合、これらの値は比較のための数値として処理されます。これらの値の先頭に、プラス (+) または マイナス (-) 記号を 1 つ付けることができます。構文と例については、33ページの『IF ブロック』を参照してください。

#### variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

#### function call

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した組み込み関数を呼び出します。構文と例については、27ページの『関数呼び出し (@)』を参照してください。

#### HTML statements

クライアントのブラウザー用に形式化される HTML タグだけでなく、任意の英字または数字を組み込みます。

#### include statement

INCLUDE ステートメントです。ファイルを読み取って Net.Data マクロに取り込みます。構文と例については、40ページの『INCLUDE ステートメント』を参照してください。

### include\_url statement

INCLUDE\_URL ステートメントです。別のファイルを読み取って、ステートメントが指定された Net.Data マクロに取り込みます。指定されたファイルは、ローカルまたはリモート・サーバーに存在します。構文と例については、43ページの『INCLUDE\_URL ステートメント』を参照してください。

### while block

WHILE ブロックです。条件付きストリング処理を伴うループを実行します。構文と例については、65ページの『WHILE ブロック』を参照してください。

## コンテキスト

ROW ブロックは、以下のコンテキストで検出することができます。

- REPORT ブロック

### 制約

ROW ブロックには、以下の要素を含めることができます。

- コメント・ブロック
- IF ブロック
- INCLUDE ステートメント
- INCLUDE\_URL ステートメント
- WHILE ブロック
- 関数呼び出し

**OS/390、OS/2、Windows NT、および UNIX オペレーティング・システムの場合:** SQL 関数内部から SQL 関数を呼び出すことはできません。

- 変数参照
- HTML ステートメント
- ストリング

### 例

**例 1:** 名前と場所のリストを表示する 2 列の HTML 表

```
%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>
%}
```

表の名前を選択すると、*name.mac* Net.Data マクロの *details* HTML ブロックが呼び出され、URL のパーツとして 2 つの値が送信されます。この例では、*name.mac* で

値を使用すると、名前に関する詳細をルックアップすることができます。

1  
1  
1  
1  
1

関連データの集合である変数を定義します。変数には行および列のセットが含まれ、これにはそれぞれの行のフィールドを説明する列見出し行が含まれます。表ステートメントは、**DEFINE** ステートメントまたはブロックでしか指定できません。

11

►►%TABLE| upper limit |

$$\left( \frac{\text{number}}{\text{ALL}} \right)$$

**%TABLE**

**upper limit**

number

ALL

## コンテキスト

- DEFINE ステートメント

TABLE ステートメントには、以下の要素を含めることができます。

- 第1章 Net.Data マクロ言語構成要素 63

## 例

**例 1:** 上限を 30 行に指定した Net.Data 表

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

**例 2:** デフォルトであるすべての行を使用する Net.Data 表

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

**例 3:** すべての行を指定した Net.Data 表

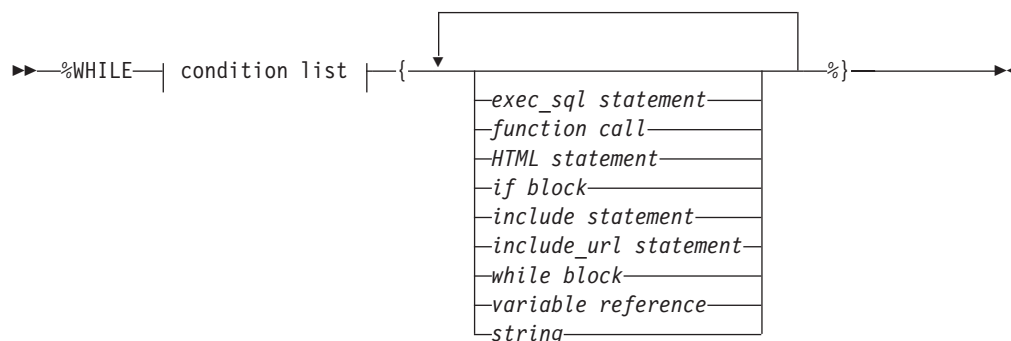
```
%DEFINE myTable3=%TABLE(ALL)
```

## WHILE ブロック

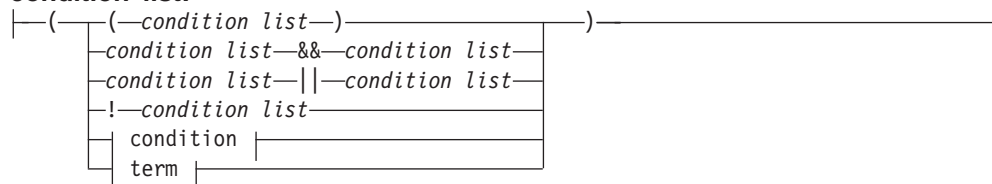
### 目的

条件付きstring処理に基づいて、ループ構成要素を提供します。 WHILE ブロックは、HTML ブロック、REPORT ブロック、ROW ブロック、 IF ブロック、および MACRO\_FUNCTION ブロックで使用することができます。条件リストのstring値が整数を表し、しかもstringの先頭または末尾に空白文字がない場合、これらの値は比較のための数値として処理されます。また、値の先頭にはプラス (+) またはマイナス (-) 記号を 1 つ付けることができます。

### 構文



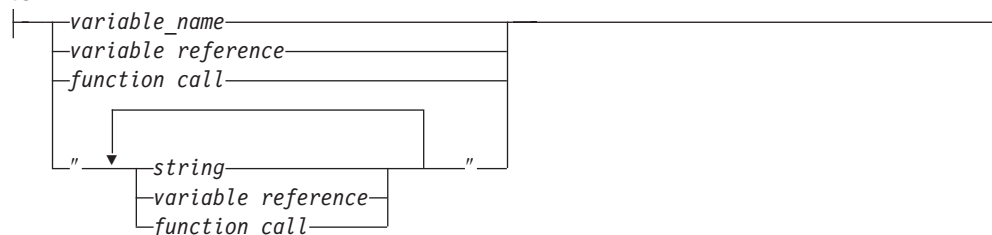
#### condition list:



#### condition:



#### term:



## 値

### %WHILE

ループ処理を指定するキーワードです。

### condition list

条件と条件の値を比較します。条件リストは、ブール演算子を使って接続することができます。条件リストは、別の条件リスト内でネストすることができます。

### condition

比較演算子を使用した、2 つの条件の間の比較です。以下の両方の条件が当てはまる場合、IF 条件は数値比較として処理されます。

- 条件演算子が以下のいずれかである : <、<=、>、>=、==、!=
- 条件とストリングが両方とも有効整数を表している。ここで、有効整数とは、オプションでプラス (+) またはマイナス (-) 記号が先頭に付くことがあり、ほかに空白文字が入っていない数字のストリングのことです。

どちらかの条件が当てはまらない場合、標準のストリング比較が行われます。

### term

変数名、ストリング、変数参照、または関数呼び出しです。

### exec\_sql statement

互換性のためにサポートされている DB2WWW リリース 1 の言語要素です。331ページの『付録B. DB2 WWW Connection』 または DB2 World Wide Web リリース 1 の資料を参照してください。

### function call

1 つまたは複数の FUNCTION または MACRO\_FUNCTION ブロックか、引き数を指定した組み込み関数を呼び出します。構文と例については、27ページの『関数呼び出し (@)』 を参照してください。

### HTML statement

クライアントのブラウザー用に形式化される HTML タグだけでなく、任意の英字または数字を組み込みます。

### if block

IF ブロックです。条件付きストリング処理を行います。条件リストのストリング値が、先頭または末尾に空白文字がない整数を表す場合、これらの値は比較のための数値として処理されます。これらの値の先頭に、プラス (+) または マイナス (-) 記号を 1 つ付けることができます。構文と例については、33ページの『IF ブロック』 を参照してください。

### include statement

INCLUDE ステートメントです。ファイルを読み取って Net.Data マクロに取り込みます。構文と例については、40ページの『INCLUDE ステートメント』 を参照してください。

### include\_url statement

INCLUDE\_URL ステートメントです。別のファイルを読み取って、ステートメントが指定された Net.Data Web マクロに取り込みます。指定されたファイルは、ローカルまたはリモート・サーバーに存在します。構文と例については、43ページの『INCLUDE\_URL ステートメント』 を参照してください。



### while block

WHILE ブロックです。条件付きストリング処理を伴うループを実行します。構文と例については、65ページの『WHILE ブロック』を参照してください。

### variable reference

変数を戻します。\$ と () を使って指定します。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を戻します。構文情報については、5ページの『変数参照』を参照してください。

### string

任意の順序の英字、数字、および句読点です。ストリングが条件リストの条件に含まれている場合、そのストリングには、改行文字以外のすべての文字を含めることができます。

### variable name

変数を識別する名前です。構文情報については、5ページの『変数名』を参照してください。

## コンテキスト

WHILE ブロックは、以下のコンテキストで検出することができます。

- HTML ブロック
- REPORT ブロック
- ROW ブロック
- MACRO\_FUNCTION ブロック
- IF ブロック
- WHILE ブロック

## 制約

WHILE ブロックには、以下の要素を含めることができます。

- コメント・ブロック
- EXEC\_SQL ステートメント
- IF ブロック
- WHILE ブロック
- ストリング
- HTML ステートメント
- 関数呼び出し
- 変数参照
- INCLUDE ステートメント
- INCLUDE\_URL ステートメント

## 例

例 1: 表に行を生成する WHILE ブロック

```
%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)
%}
%}

```

## 第2章 変数

Net.Data はユーザー定義変数と Net.Data 変数の、2 つのタイプの変数を提供します。

### 70ページの『ユーザー定義変数』

ユーザーがアプリケーションのために定義する変数です。以下のタスクを実行する変数を定義できます。

- 70ページの『条件変数』

別の変数またはストリングの値に基づいて変数値を割り当てます。

- 71ページの『環境変数』

環境変数を参照するために ENVVAR 言語構成要素を使用します。

- 72ページの『実行可能変数』

変数参照から他のプログラムを呼び出すために、EXEC 言語構成要素を使用します。

- 73ページの『隠蔽変数』

HTML ソースから変数参照を隠します。

- 74ページの『リスト変数』

LIST 言語構成要素を使って、区切られた値のストリングを組み立てます。

- 75ページの『表変数』

関数との間で値の配列を受け渡しします。レポート出力に使用できます。

### Net.Data 変数

各種の処理およびファイル操作、表処理、レポート形式設定、および言語環境のための変数です。

変数の中にはユーザーが定義または変更できる値を持つものと、Net.Data によって定義されるものがあります。変数の説明では、ユーザーが値を定義できるか否かが示されています。値がどのように定義されるかについては、変数の説明を参照してください。

以下の変数タイプは Net.Data によって提供されます。

- 76ページの『Net.Data 表処理変数』

ユーザーが Net.Data 表を処理できるようにするために、Net.Data によって定義されます。SQL 照会および関数呼び出しからデータをアクセスするために、これらの変数を使用します。これらは特に指定されていない限り、REPORT または ROW ブロックの内部でのみ認識されます。

- 86ページの『Net.Data レポート変数』

関数からレポートをカスタマイズするのに役立ちます。どの Net.Data マクロ・ブロックでも、レポート変数を定義または参照することができます。

- 95ページの『Net.Data 言語環境変数』

言語環境を使って FUNCTION ブロックを処理する方法をカスタマイズするのに役立ちます。

- 115ページの『Net.Data の各種変数』

Net.Data 処理に影響を与え、関数呼び出しの状況を検出し、またデータベース照会の結果セットに関する情報を入手するために、Net.Data によって定義されます。各種の変数が Net.Data によって設定され、変更することはできません。

多くの Net.Data 変数の出力は、それが実行しているオペレーティング・システムによって異なり多種多様です。

Net.Data マクロでは定数は最大 256KB です。したがって、マクロ内で長さが 256 KB より長い変数は、初期設定したりデフォルト値を設定したりすることはできません。

この章では、各変数ごとのオペレーティング・システム・サポートが示されています。以下のリストは、オペレーティング・システムの省略形を定義するものです。

<b>HP-UX</b>	Hewlett Packard UNIX オペレーティング・システム
<b>SCO</b>	Santa Cruz オペレーション OpenServer
<b>SUN</b>	Solaris オペレーティング・システム
<b>Win NT</b>	Microsoft Windows NT オペレーティング・システム

---

## ユーザー定義変数

このセクションでは、ユーザー定義変数について説明します。これらの変数はマクロ・ファイルの中で定義します。

## 条件変数

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

条件変数の値は、別の変数またはストリングの値に基づいて条件付きで設定されます。これは 3 進演算とも言います。

条件変数の構文は、次のとおりです。

```
test ? trueValue : falseValue
```

ここで、

**test**     テストする条件です。

**trueValue**

テスト結果が真であった場合に使用する値です。

**falseValue**

テスト結果が偽であった場合に使用する値です。

**例 1:** 2 つの可能な値とともに定義される条件変数

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

varB が存在する場合は varA=value\_1 で、存在しない場合は varA=value\_2 です。

### 例 2: 変数参照とともに定義される条件変数

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

このケースでは、*value\_1* が NULL の場合は *varname* は NULL で、そうでない場合は *varname* が *value\_1* に設定されます。

### 例 3: LIST ステートメントおよび WHERE 文節とともに使用される条件変数

```
%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 prodtbale $(where_clause)
%}
```

条件変数と LIST 変数を一緒に使用すると最も効果的です。上記の例は、DEFINE ブロック内での WHERE 文節のセットアップ方法を示します。変数 *cust\_inp* および *prod\_inp* は HTML 入力変数で、Web ブラウザーから通常 HTML 形式で渡されます。変数 *where\_list* は 2 つの条件ステートメントから成る LIST 変数です。それぞれの条件ステートメントには、Web ブラウザーから受け取った変数が含まれます。

Web ブラウザーが *cust\_inp* と *prod\_inp* の両方の変数に対して、たとえば IBM と 755C という値を戻した場合、*where\_clause* は以下ようになります。

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

*cust\_inp* または *prod\_inp* の変数のいずれかが NULL であるかあるいは定義されていない場合、WHERE 文節は NULL 値を省略するように変更されます。たとえば、*prod\_inp* が NULL の場合の WHERE は以下ようになります。

```
WHERE custid = IBM
```

両方の値とも NULL であるかあるいは定義されていない場合、変数 *where\_clause* は NULL で、*\$(where\_clause)* を含む SQL 照会には WHERE 文節は現れません。

## 環境変数

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

環境変数を使用すると、Net.Data が実行しているプロセスに存在する環境変数を参照するために、Net.Data ENVVAR 言語構成要素を使用できます。

### 例 1: 環境変数の値を割り当てられた変数

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

```
...
```

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

環境変数 *SERVER\_NAME* には現在のサーバー名の値が含まれます。この例では *www.software.ibm.com* です。

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

ENVVAR ステートメントについての詳細は、15ページの『ENVVAR ステートメント』を参照してください。

## 実行可能変数

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

実行可能変数を使用すると、実行可能変数フィーチャーを使って変数参照から他のプログラムを呼び出せます。実行可能変数は `Net.Data` マクロで、EXEC 言語要素を使って定義します。EXEC 言語要素についての詳細は、16ページの『EXEC ブロックまたはステートメント』を参照してください。

`Net.Data` がマクロ内で実行可能変数を見つけると、参照された実行可能プログラムを以下の方法で探します。

1. `Net.Data` 初期設定ファイルの中で `EXEC_PATH` を探します。`EXEC_PATH` についての詳細は、*Net.Data* 管理およびプログラミングの手引き の構成の章を参照してください。
2. プログラムが見つからない場合、`Net.Data` はシステムの `PATH` 環境変数で定義されているディレクトリーを探します。実行可能プログラムが見つかったら、`Net.Data` はそのプログラムを実行します。

### 例 1: 実行可能変数の定義

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

変数 `runit` は、実行可能プログラム `testProg` を実行するために定義されます。`runit` は実行可能変数になります。

`Net.Data` は、`Net.Data` マクロ内で実行可能変数の参照が見つかったときに、実行可能プログラムを実行します。たとえば、プログラム `testProg` は、`Net.Data` マクロ内で変数 `runit` に対して実行可能変数の参照が行われたときに実行されます。

簡単な方法は、別の変数定義から実行変数を参照することです。例 2 はこの方法を示します。変数 `date` が実行可能変数として定義され、さらにその実行可能変数を含む `dateRpt` が変数参照として定義されています。

### 例 2: 変数参照としての実行可能変数

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

`Net.Data` が変数参照 `$(dateRpt)` を解決すると、`Net.Data` は実行可能な `date` を検索し、プログラムを実行し、以下のものを戻します。

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

実行可能変数は、自分が呼び出した実行可能プログラムの出力値に設定されることはありません。前の例を使うと、`date` の値は `NULL` です。実行可能変数の値を別の変数に割り当てるために、`DTW_ASSIGN` 関数呼び出しで実行可能変数を使用した場合、割り当て後の新しい変数値も `NULL` です。実行可能変数の唯一の目的は、定義したプログラムを呼び出すことです。

変数定義でプログラム名と一緒にパラメーターを指定することによって、実行されるプログラムにパラメーターを渡すこともできます。

**例 3:** パラメーターを指定した実行可能変数

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

distance と time の値がプログラム calcMPH に渡されます。

# 隠蔽変数

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

隠蔽変数によって、HTML ソースで実際の変数値を隠しているときに変数を参照できます。隠蔽変数を使用するためには、以下のようにします。

1. 隠したいストリングごとに変数を定義します。
2. 変数を参照するためには、変数が参照されている HTML ブロック内でドル記号 1 つの代わりにドル記号 2 つを使います。たとえば、\$(X) の代わりに \$\$\$(X) です。

動的に構成された変数名を持つ隠蔽変数の参照を行わないようにしてください。

**例 1:** HTML 形式での隠蔽変数

```
%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
%}
.
.
.
```

HTML 形式が Web ブラウザーに表示されると、\$\$\$(name) および \$\$\$(addr) がそれぞれ \$(name) および \$(addr) に置き換えられるので、実際の表名および列名は HTML 形式には決して現れず、本来の変数名が隠されていることは誰にも分かりません。この形式を実行依頼すると、HTML(REPORT) ブロックが呼び出されます。@mySelect() が FUNCTION ブロックを呼び出すと、SQL ステートメントの中で \$(Field) が置き換えられ、SQL 照会では customer.name または customer.addr になります。

## リスト変数

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

リスト変数によって、区切られた値のストリングを組み立てます。これは、幾つかの WHERE または HAVING 文節に含まれるような複数の項目を持つ、SQL 照会を構成するときに特に役立ちます。

ブランクは有効です。通常、値の両側にブランク・スペースを使用します。ほとんどの照会はブール演算子または数値演算子 (たとえば、AND、OR、および >) を使用します。構文および詳細については、46ページの『LIST ステートメント』を参照してください。

### 例 1: 条件変数、隠蔽変数、およびリスト変数の使用

```
%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()
%}
```

HTML 形式でボックスがどれもチェックされていない場合 *conditions* は NULL なので、照会の中の *whereClause* も NULL です。そうでない場合、*whereClause* には選択された値がブール演算子 OR で区切られて入ります。たとえば、3 つのすべての都市が選択された場合、SQL 照会は以下ようになります。

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

### 例 2: 値区切り記号

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



この例の表処理変数 VLIST は、値区切り記号として 2 つの引用符と 1 つの OR 線 ( | ) を使用しています。値のストリングは、引用符の中の値で区切られます。

## 表変数

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

表変数は、関連データの集合を定義します。列見出し行を含む行および列のセットが含まれます。関数に値のグループを渡すために、表変数を使用します。関数の REPORT ブロック中で、または表組み込み関数を使用して、表の個別の要素 (行) を参照できます。表変数は、SQL 関数からの出力およびレポートへの入力のために使用されることが多いですが、IN、OUT、または INOUT パラメーターとして SQL 以外の関数に渡すこともできます。表は、OUT パラメーターとして SQL 関数にのみ渡すことができます。構文および詳細については、63ページの『TABLE ステートメント』を参照してください。

TABLE 変数を参照する際、Net.Data は、表の内容をプレーンの文字表、または HTML 表 (DTW\_HTML\_TABLE 変数を YES に設定した場合) のいずれかに表示します。

### 例 1: REXX プログラムに渡される SQL の結果セット

```
%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>
}%
```

HTML の REPORT ブロックが SQL 照会を呼び出し、結果を表変数に保管し、その後その変数を REXX 関数に渡します。

---

## Net.Data 表処理変数

Net.Data は、特に記述されていない限り、これらの変数を REPORT および ROW ブロックで使用するために定義します。これらの変数を使って、照会が戻した値を参照します。

- 77ページの『 $N_n$ 』
- 78ページの『NLIST』
- 79ページの『NUM\_COLUMNS』
- 80ページの『NUM\_ROWS』
- 81ページの『ROW\_NUM』
- 82ページの『TOTAL\_ROWS』
- 83ページの『 $V\_columnName$ 』
- 84ページの『VLIST』
- 85ページの『 $V_n$ 』

## Nn

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

### 目的

関数呼び出しまたは列 `n` の照会によって戻される列名。

`Nn` は `REPORT` および `ROW` ブロックで参照することができます。

### 例

#### 例 1: 列名のための変数参照

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

#### 例 2: `DTW_ASSIGN` を使って `REPORT` ブロックの外で使用するために列名の値を保管する

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

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

この例は、`DTW_ASSIGN` を使って `REPORT` ブロックの外でこの変数を使用する方法を示しています。詳細については、184ページの『`DTW_ASSIGN`』を参照してください。

#### 例 3: 列名を定義するための HTML 表の中の `Nn`

```
%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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

関数呼び出しまたは照会の結果の、すべての列名のリストを含みます。デフォルトの区切り記号はスペースです。

NLIST は REPORT および ROW ブロックで参照することができます。

### 例

**例 1:** ALIGN を指定した列名のリスト

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

列名のリストでは、ALIGN が YES に設定され、列名と列名の間にスペースが使用されます。

**例 2:** 区切り記号を " | " に変更する %LIST 変数

```
%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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

Net.Data が REPORT ブロックで処理する表の列の数。列は、関数呼び出しまたは照会によって戻されます。

NUM\_COLUMNS は REPORT および ROW ブロックで参照することができます。

### 例

**例 1:** NLIST を指定した変数参照として使用される NUM\_COLUMNS

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

## NUM\_ROWS

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

### 目的

Net.Data が REPORT ブロックで処理する表の行数。表の行数は、データを保持している Net.Data 表に対して定義された *upper limit* パラメーターの値の影響を受けます。たとえば、*upper limit* が 30 に設定されていると、SELECT ステートメントが 1000 行を戻しても NUM\_ROWS の値は 30 です。さらに、*upper limit* が 30 に設定されていると、SELECT ステートメントが 20 行を戻しても NUM\_ROWS は 20 です。TABLE ステートメントおよび *upper limit* パラメーターについての詳細は、63ページの『TABLE ステートメント』を参照してください。

NUM\_ROWS は、START\_ROW\_NUM が言語環境に渡されない限り、START\_ROW\_NUM の値の影響を受けません。たとえば、START\_ROW\_NUM が 5 に設定されていて (Web ページに表示される表は 5 行目から移植される)、SELECT ステートメントが 25 行を戻した場合、NUM\_ROWS は 21 ではなく 25 に設定されます。最初の 4 行は表から廃棄されますが、NUM\_ROWS の値には含まれます。しかし、START\_ROW\_NUM が言語環境に渡された場合は、NUM\_ROWS には、START\_ROW\_NUM で指定された行で始まる行数のみが含まれます。上記の例では、NUM\_ROWS は 21 に設定されます。

NUM\_ROWS は、REPORT および ROW ブロックで参照することができます。

### 例

**例 1:** REPORT ブロックで処理される名前の数を表示する

```
%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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

Net.Data 表の中で行が 1 行処理されるたびに Net.Data が値を増やす表変数。この変数はカウンターとして使用でき、その値は処理されている現在行の番号です。

RPT\_MAX\_ROWS は ROW\_NUM の値に影響を与える場合があります。たとえば、表に 100 行含まれていて RPT\_MAX\_ROWS を 20 に設定している場合、最後に処理された行は 20 行目なので、最終的な ROW\_NUM の値は 20 になります。

ROW\_NUM は、ROW ブロック内だけで参照することができます。

### 例

**例 1:** 表の中の各行にラベルを付けるために、ROW\_NUM を使って HTML 出力の中に列を移植する

```
%REPORT{
<TABLE BORDER=1>
<TR><TD> Row Number </TD> <TD> Customer </TD>
%ROW{
<TR><TD> $(ROW_NUM) </TD> <TD> $(V_custname) </TD>
%}
</TABLE>
%}
```

REPORT ブロックは、下記に示すものと同じような表を作成します。

Row Number	Customer
1	Jane Smith
2	Jon Chiu
3	Frank Nguyen
4	Mary Nichols

## TOTAL\_ROWS

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

### 目的

照会が戻す行数の合計。 TABLE 言語構成要素の *upper\_limit* の値には無関係です。たとえば、RPT\_MAX\_ROWS が最大 20 行を表示するように設定されていて、照会が 100 行を戻した場合、ROW 処理の後でこの値は 100 に設定されます。

### オペレーティング・システムによる相違点:

- OS/400 オペレーティング・システムでは、REPORT または ROW ブロック内のどこでもこの変数を参照できます。
- OS/390、OS/2、Windows NT、および UNIX オペレーティング・システムでは、REPORT フッターの中でのみこの変数を参照できます。

**言語環境の制約事項:** この変数は、以下のデータベース言語環境でのみ使用します。

- SQL
- ODBC
- Oracle
- Sybase

**必須事項:** この変数を使用するためには、DTW\_SET\_TOTAL\_ROWS を YES に設定しなければなりません。詳しくは、105ページの『DTW\_SET\_TOTAL\_ROWS』を参照してください。

### 例

**例 1:** 見つかった名前の数の合計を表示する

```
%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\_columnName

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

### 目的

現在行の、指定された列名の値。この変数は、定義されていない列名に対しては設定されません。同じ名前を持つ 2 つの列名を含む照会では、結果は予想できません。重複した列の名前を変更するために、SQL で AS 文節を使用することを検討してください。

V\_columnName は、ROW ブロック内だけで参照することができます。

### 値

V\_columnName

表 1. V\_columnName の値

値	説明
columnName	データベース表の現在行の中の列名。

### 例

**例 1:** V\_columnName を変数参照として使用する

```
%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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

ROW ブロックで処理されている現在行のすべてのフィールド値のリスト。

VLIST は、ROW ブロック内だけで参照することができます。デフォルトの区切り記号はスペースです。

### 例

**例 1:** 照会の結果を表示するためにリスト・タグを使用する

```
%DEFINE ALIGN="YES"

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

**例 2:** 区切り記号を <P> に変更するためにリスト変数を使用する

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

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

## V<sub>n</sub>

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

### 目的

現在行の指定された列番号  $n$  の値。

V<sub>n</sub> は、ROW ブロック内だけで参照することができます。

Net.Data は、表の中の各フィールドに対して変数を割り当てます。参照したいフィールドの番号を指定して、変数参照の中で変数を使用します。ブロックの外でこの変数を使用するためには、前もって定義したグローバル変数、または OUT あるいは INOUT 関数のパラメーター変数に、V<sub>n</sub> の値を割り当てます。

### 例

#### 例 1: HTML 表を表示するレポート

```
%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>
%}
```

2 番目の列は e-mail アドレスを示します。リンクをクリックすることによって、その人にメッセージを送ることができます。

---

## Net.Data レポート変数

これらの変数は、レポートをカスタマイズするのに役立ちます。変数にはそれぞれデフォルト値があります。変数に新規の値を割り当てて、デフォルト値をオーバーライドすることができます。

- 87ページの『ALIGN』
- 88ページの『DTW\_DEFAULT\_REPORT』
- 89ページの『DTW\_HTML\_TABLE』
- 90ページの『RPT\_MAX\_ROWS』
- 92ページの『START\_ROW\_NUM』

# ALIGN

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

## 目的

表処理変数 `NLIST` および `VLIST` で使用される前後のスペースを制御します。

**パフォーマンスのヒント:** `ALIGN` は必要な場合にだけ使用してください。 `ALIGN` を使用するときには、必要な埋め込みを計算するために、 `Net.Data` が表の中のすべての列の最大列長を判別する必要があるからです。このプロセスは、パフォーマンスに悪影響を与える可能性があります。

`YES` に設定されている場合、`ALIGN` によって、表示用に表処理変数を位置合わせするための埋め込みが提供されます。`HTML` リンクまたはフォーム・アクションに照会結果を組み込みたい場合、デフォルト値の `NO` を使って、`Net.Data` がレポート変数に前後のスペースを付けないようにします。

`DEFINE` ステートメントまたは `@DTW_ASSIGN()` 関数を使用して、この変数の値を指定してください。

## 値

`ALIGN="YES"|"NO"`

表 2. `ALIGN` の値

値	説明
<code>YES</code>	<code>Net.Data</code> はレポート変数に、表示用の位置合わせのためのスペースとともに前後のスペースを付加する。
<code>NO</code>	<code>Net.Data</code> は前後のスペースを付加しない。 <code>NO</code> はデフォルト。

## 例

**例 1:** 各列をスペースで分離するために `ALIGN` 変数を使用する

```
%DEFINE ALIGN="YES"
<P>Your query was on these columns: $(NLIST)
```

## DTW\_DEFAULT\_REPORT

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

### 目的

Net.Data が REPORT ブロックのない関数のためにデフォルト・レポートを生成するかどうかを判別します。この変数を YES に設定すると、Net.Data はデフォルト・レポートを生成します。NO に設定すると、Net.Data はデフォルト・レポートを生成しません。デフォルト・レポートを生成しないことは、たとえば関数呼び出しの結果を表変数で受け取って、その結果を別の関数に渡して処理したい場合に役立ちます。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

DTW\_DEFAULT\_REPORT="YES"|"NO"|"MULTIPLE"

表 3. DTW\_DEFAULT\_REPORT の値

値	説明
YES	Net.Data は REPORT ブロックのない関数のためにデフォルト・レポートを生成し、結果をブラウザに表示する。YES はデフォルト。
NO	Net.Data は REPORT ブロックのない関数のためのデフォルト・レポートを破棄する。
MULTIPLE	Net.Data は、複数の REPORT ブロックを持つ関数について、REPORT ブロックに割り当てられていない結果セットまたは出力表のデフォルトのレポートを作成する。

### 例

例 1: Net.Data で生成されたデフォルト・レポートをオーバーライドする

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

## DTW\_HTML\_TABLE

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

### 目的

結果の表をテキスト・タイプ形式で表示する代わりに HTML 表で表示します (つまり、PRE タグではなく TABLE タグを使います)。

生成された TABLE タグには、ボーダーおよびセル埋め込み指定が含まれます。

```
<TABLE BORDER CELLPADDING=2>
```

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

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

表 4. DTW\_HTML\_TABLE の値

値	説明
YES	表データを、HTML 表タグを使って表示する。
NO	表データを、PRE タグを使ってテキスト形式で表示する。NO はデフォルト。

### 例

例 1: SQL 関数の結果を HTML タグで表示する

```
%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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

関数の REPORTS ブロックにおいて、またはデフォルト・レポートの生成中に (REPORT ブロックが指定されていない場合) 処理をする表内の行数を指定します。

データベース言語環境はこの変数を使用することにより、戻される行数を制限し、これにより大きな結果セットに対するパフォーマンスが大幅に向上します。この変数は START\_ROW\_NUM と使用して、大きな結果セットを持つ照会をより小さい表に分割し、それぞれを自身の HTML ページに収めます。

**OS/400、Windows NT、OS/2、および UNIX ユーザーの場合:** この変数を言語環境に渡すには、Net.Data 初期設定ファイルのデータベース言語環境の ENVIRONMENT ステートメントに IN 変数としてこの変数を組み込みます。データベース言語環境ステートメントの詳細については、*Net.Data 管理およびプログラミングの手引き* の構成の章を参照してください。

**OS/390 ユーザーの場合:** RPT\_MAX\_ROWS は、マクロで定義する際に、暗黙的にデータベース言語環境に渡されます。

この変数の値は、DEFINE ステートメントを使って、または @DTW\_ASSIGN() 関数で指定できます。

### 値

RPT\_MAX\_ROWS="ALL"|"0"|"number"

表 5. RPT\_MAX\_ROWS の値

値	説明
ALL	関数呼び出しで生成する表の中で表示する行数に制限がないことを示す。すべての行が表示される。
0	表の中のすべての行を表示するように指定する。この値は ALL を指定するのと同じ。
number	関数呼び出しで生成する表の中で表示する行の、最大行数を示す正整数。  FUNCTION ブロックに REPORT および ROW ブロックが含まれる場合、この数字は ROW ブロックが実行される回数を指定する。

### 例

**例 1:** DEFINE ステートメントで RPT\_MAX\_ROWS を定義する

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

上記の方法では、関数が戻す行数を 20 行に制限します。

**例 2:** 変数を HTML 形式で定義するために HTML 入力を使用する



Maximum rows to return (0 for no limit):  
<INPUT TYPE="text" NAME="RPT\_MAX\_ROWS" SIZE=3>

上記の例の行は、アプリケーション・ユーザーが照会から受け取りたい行数を設定できるように、 **FORM** タグに置くことができます。

## START\_ROW\_NUM

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

### 目的

関数の REPORT ブロック中、またはデフォルト・レポートの生成中に (REPORT ブロックが指定されていない場合) 処理をする表内の開始行数を指定します。

データベース言語環境はこの変数を使用して、結果セット内の処理を開始する行を決定します。大きな結果セットにおけるパフォーマンスを大幅に向上させるには、この変数を RPT\_MAX\_ROWS と使用して、大きな結果セットを持つ照会をより小さい表に分割します。

**OS/400、Windows NT、OS/2、および UNIX ユーザーの場合:** この変数を言語環境に渡すには、Net.Data 初期設定ファイルのデータベース言語環境の ENVIRONMENT ステートメントに IN パラメーターとしてこの変数を組み込みます。データベース言語環境ステートメントの詳細については、*Net.Data 管理およびプログラミングの手引き* の構成の章を参照してください。

**OS/390 ユーザーの場合:** START\_ROW\_NUM は、マクロで定義する際に、暗黙的にデータベース言語環境に渡されます。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

START\_ROW\_NUM="number"

表 6. START\_ROW\_NUM の値

値	説明
<i>number</i>	レポートの表示を始める行番号を示す正整数。デフォルト値は 1 です。  初期設定ファイルの中のデータベース言語環境の ENVIRONMENT ステートメントで START_ROW_NUM が指定されている場合、この数字が、データベース言語環境によって処理される結果セットの行数を指定する。  START_ROW_NUM が言語環境に渡されない場合、この数字は、レポートを表示するために使用する Net.Data 表の行数を指定する。

### 例

例 1: HTML 形式の Next および Previous ボタンでスクロールする

```
%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 %}
        @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

    %{ 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>
%}
```

---

## Net.Data 言語環境変数

これらの変数を関数と一緒に使用すると、言語環境が FUNCTION ブロックを処理する方法をカスタマイズするのに役立ちます。変数にはそれぞれデフォルト値があります。変数に新規の値を割り当てて、デフォルト値をオーバーライドすることができます。

- 96ページの『DATABASE』
- 98ページの『DB\_CASE』
- 99ページの『DB2PLAN』
- 100ページの『DB2SSID』
- 101ページの『DTW\_APPLET\_ALTTEXT』
- 102ページの『DTW\_EDIT\_CODES』
- 103ページの『DTW\_PAD\_PGM\_PARMS』
- 104ページの『DTW\_SAVE\_TABLE\_IN』
- 105ページの『DTW\_SET\_TOTAL\_ROWS』
- 107ページの『LOCATION』
- 108ページの『LOGIN』
- 109ページの『NULL\_RPT\_FIELD』
- 110ページの『PASSWORD』
- 111ページの『SHOWSQL』
- 112ページの『SQL\_STATE』
- 113ページの『TRANSACTION\_SCOPE』

## DATABASE

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

### 目的

データベース関数を呼び出したときにアクセスする、データベースまたは ODBC データ・ソースを指定します。複数のデータベースまた ODBC データ・ソースをアクセスするために、この変数を 1 つのマクロ内で複数回変更することができます。

**OS/400 オペレーティング・システム:** この変数はオプションです。デフォルトでは、Net.Data は DATABASE="\*LOCAL" を指定します。DTW\_SQL 言語環境は、ローカルのリレーショナル・データベースのディレクトリー項目を使用します。

**Windows NT、OS/2、および UNIX オペレーティング・システム:** データベース関数を呼び出す前にこの変数を定義します。ただし、DTW\_ORA (Oracle) 言語環境を使用しているときは例外です。さらに、同じ HTML ブロックから同じ言語環境を介して複数のデータベースにアクセスする場合は、Live コネクションを使用しなければなりません。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

DATABASE="dbname"

表 7. DATABASE の値

値	説明
dbname	Net.Data が接続するデータベースの名前。

### 例

**例 1:** SQL 操作で CELDIAL データベースに接続することを指定する

```
%DEFINE DATABASE="CELDIAL"

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

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

関数 getRpt が呼び出されたとき、データベース CELDIAL がアクセスされます。

**例 2:** 前の DATABASE 定義を DTW\_ASSIGN でオーバーライドする

```
%DEFINE DATABASE="DB2C1"
...
%HTML(monthRpt){
  @DTW_ASSIGN(DATABASE, "DB2D1")
}
```

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

DATABASE の前の値が何であったかにかかわらず、HTML ブロックはデータベース DB2D1 を照会します。

## DB\_CASE

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

### 目的

SQL コマンドで大文字小文字のどちらを使用するか指定し、すべての文字を大文字または小文字のいずれかに変換します。この変数が定義されていない場合、デフォルトでは SQL コマンドの文字を変換しません。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

DB\_CASE="UPPER"|"LOWER"

表 8. DB\_CASE の値

値	説明
UPPER	SQL コマンドのすべての文字を大文字に変換する。
LOWER	SQL コマンドのすべての文字を小文字に変換する。

### 例

例 1: すべての SQL コマンドに大文字を指定する

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



## DB2PLAN

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

### 目的

ローカル DB2 サブシステムへの接続のためのプランを割り当てます。この変数は、Net.Data がアクセスするローカル DB2 サブシステムでの Net.Data SQL 言語環境のプラン名を指定します。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

**要件:** この変数の値をマクロで有効にするには、変数を SQL 言語環境の ENVIRONMENT ステートメントにリストする必要があります。

### 値

DB2PLAN="*plan\_name*"

表 9. DB2PLAN の値

値	説明
<i>plan_name</i>	DB2 プランの名前。名前は 8 文字以内です。

### 例

**例 1:** DEFINE ステートメントにプランを指定する

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

## DB2SSID

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

### 目的

ローカル DB2 サブシステムへの接続を確立します。この変数は、Net.Data がアクセスするローカル DB2 サブシステムのサブシステム ID を指定します。各マクロで許されるローカル・データベース接続は 1 つだけです。

**要件:** この変数の値をマクロで有効にするには、変数を SQL 言語環境の ENVIRONMENT ステートメントにリストする必要があります。

### 値

```
DB2PLAN="subsytem_id"
```

表 10. DB2SSID の値

値	説明
<i>subsystem_id</i>	DB2 サブシステムの名前。名前は 8 文字以内です。

### 例

**例 1:** DEFINE ステートメントにサブシステム ID を指定する

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

## DTW\_APPLET\_ALTTEXT

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

### 目的

アプレット言語環境で使用されている APPLET タグを認識しないブラウザに、HTML タグとテキストを表示します。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

DTW\_APPLET\_ALTTEXT="HTML\_text\_and\_tags"

表 11. DTW\_APPLET\_ALTTEXT の値

値	説明
HTML_text_and_tags	APPLET タグを認識しないブラウザ用の HTML タグとテキスト。

### 例

例 1: Web ブラウザーの制約事項を示す代替テキスト

```
%DEFINE DTW_APPLET_ALTTEXT="<P>Sorry, your browser is not java-enabled."
```

## DTW\_EDIT\_CODES

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

### 目的

DTW\_SQL 言語環境の SQL 操作の結果として戻される NUMERIC、DECIMAL、INTEGER、および SMALLINT のデータ型を変換します。変数 DTW\_EDIT\_CODES は、DTW\_SQL LE が構築する表の結果の列に対応する文字ストリングです。たとえば、その列がサポートされている型の 1 つである場合、DTW\_EDIT\_CODES の中の 5 番目の文字は、結果セットの 5 つ目の列に適用します。この単一の文字は、*Data Description Specification Reference* で定義された、サポートされるシステム提供編集コードのいずれかです。

たとえば DECIMAL(6,0) フィールドは通常、文字ストリング '112698' として表示されます。その列に関して変数 DTW\_EDIT\_CODES で編集コード 'Y' を指定すると、日付 '11/26/98' を表す文字ストリングとして、結果表内の対応列が表示されます。

**ヒント:** 結果が数値でない文字 (コンマ、または通貨記号など) を含む文字ストリングになる列にユーザー提供の編集コードを適用すると、Net.Data マクロ内の後続の処理のために文字ストリングがサーバーに送り戻された場合に、構文エラーになります。たとえば、後続の DTW\_SQL 関数呼び出しで数値でない列値が数値の比較に使用されると、構文エラーになります。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

DTW\_EDIT\_CODES="edit\_code"

表 12. DTW\_EDIT\_CODES の値

値	説明
edit_code	SQL 言語環境が構築する、表の結果の列に対応する文字ストリングを指定する。

### 例

#### 例 1:

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

## DTW\_PAD\_PGM\_PARMS

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

### 目的

文字パラメーター (データ型 CHAR または CHARACTER) をプログラムまたはストアド・プロシージャに渡す際、ブランクで埋め込むかどうかを言語環境に対し指示します。

IN または INOUT パラメーターの場合、指定された精度よりもパラメーター値の長さが短い場合、パラメーター値の右にブランクが挿入され、パラメーター値の長さが精度と同じになるようにします。

OUT パラメーターの場合、パラメーター値は 精度 ブランクに設定されます。

プログラムまたはストアド・プロシージャへの呼び出し後、OUT および INOUT パラメーター値からすべての後書きブランクが削除されます。

すべてのユーザー・マクロに対して値を指定するためには、Net.Data 初期設定ファイルの中でこの変数を設定します。マクロ中でこの変数を定義すると、値をオーバーライドすることができます。DTW\_PAD\_PGM\_PARMS がマクロで定義されていない場合は、Net.Data 初期設定ファイルの中の値を使用します。

DTW\_PAD\_PGM\_PARMS は、直接呼び出しおよび SQL 言語環境でサポートされます。

### 値

DTW\_PAD\_PGM\_PARMS="YES"|"NO"

表 13. DTW\_PAD\_PGM\_PARMS 値

値	説明
YES	パラメーターをプログラムまたはストアド・プロシージャに渡す前に、すべての IN および INOUT 文字パラメーター値が左寄せされ、パラメーターに定義された精度と同じになるようブランクが埋め込まれます。プログラムまたはストアド・プロシージャへの呼び出し後に、後書きブランクが削除されます。
NO	パラメーターをプログラムまたはストアド・プロシージャに渡す際、文字パラメーター値に埋め込みは追加されません。値はヌルで終了します。プログラムまたはストアド・プロシージャへの呼び出し後に、後書きブランクは削除されません。

### 例

**例 1:** パラメーターをブランクで埋め込みます。

DTW\_PAD\_PGM\_PARMS="YES"

## DTW\_SAVE\_TABLE\_IN

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

### 目的

SQL 言語環境が照会からの表データを保管するために使用する、表変数を指定します。この表は後で、たとえば表データを分析する REXX プログラムなどで使用できます。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

DTW\_SAVE\_TABLE\_IN="table\_name\_var"

表 14. DTW\_SAVE\_TABLE\_IN の値

値	説明
table_name_var	SQL 言語環境が照会からの表データを保管するための表名。

### 例

**例 1:** REXX 呼び出しで使用する、事前定義された表変数

```
%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)
}%
```

REXX FUNCTION ブロックが REXX プログラム anzTbl.cmd を呼び出し、このプログラムは表変数 theTable を使って表の中のデータを分析します。変数 theTable は、前の SQL 関数呼び出しで戻されたものです。

# DTW\_SET\_TOTAL\_ROWS

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

## 目的

照会の結果セット内の行の合計数を TOTAL\_ROWS に割り当てるように、データベース言語環境に指定します。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

**OS/400、OS/2、Windows NT、および UNIX ユーザーの場合:** この変数を言語環境に渡すには、Net.Data 初期設定ファイルファイルのデータベース言語環境の ENVIRONMENT ステートメントに IN 変数としてこの変数を組み込みます。データベース言語環境ステートメントの詳細については、Net.Data 管理およびプログラミングの手引き の構成の章を参照してください。

**OS/390 ユーザーの場合:** DTW\_SET\_TOTAL\_ROWS は、マクロで定義する際に、暗黙的にデータベース言語環境に渡されます。

**パフォーマンス上のヒント:** DTW\_SET\_TOTAL\_ROWS を YES に設定すると、合計行を判別するためにデータベース言語環境がすべての行を検索しなければならないので、パフォーマンスが影響を受けます。

## 値

DTW\_SET\_TOTAL\_ROWS="YES"|"NO"

表 15. DTW\_SET\_TOTAL\_ROWS の値

値	説明
YES	行の合計数の値を TOTAL_ROWS 変数に割り当てる。 <b>重要:</b> 照会から戻された行の数を判別するために変数 TOTAL_ROWS を参照したい場合は、この値を設定しなければならない。
NO	Net.Data は TOTAL_ROWS 変数を設定せず、マクロで TOTAL_ROWS を参照することはできない。NO はデフォルト。

## 例

**例 1:** TOTAL\_ROWS を使用するために DTW\_SET\_TOTAL\_ROWS を定義する

```
%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.  
%}  
%}
```



# LOCATION

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

## 目的

リモート・データベース・サーバーへの接続を確立します。変数は、ローカル DB2 サブシステムがリモート・サーバーを認識するための名前を指定します。LOCATION の値は、通信データベース (CDB) の SYSIBM.SYSLOCATIONS 表に定義しなければなりません。この変数がマクロ内で定義されていない場合、マクロによって発行される SQL 要求はすべてローカル DB2 サブシステムで実行されます。

**要件:** この変数の値をマクロで有効にするには、変数を SQL 言語環境の ENVIRONMENT ステートメントにリストする必要があります。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

## 値

LOCATION="remote\_dbase\_name"

表 16. LOCATION の値

値	説明
remote_dbase_name	CDB の SYSIBM.SYSLOCATIONS 表で定義されている、有効なリモート・データベース・サーバーの名前。名前は 8 文字以内です。

## 例

**例 1:** DEFINE ステートメントでリモート・データベース・ロケーションを定義する  
%DEFINE LOCATION="QMFDJ00"

# LOGIN

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

## 目的

データベース言語環境にユーザー ID を渡すことによって、保護されたデータへのアクセスを提供します。この変数は PASSWORD と一緒に使用して、DB2 の機密保護アルゴリズムを取り込みます。

**OS/400 ユーザー:** DATABASE 変数が定義されていない場合、あるいは `"*LOCAL"` という値に設定されている場合には、OS/400 は LOGIN と PASSWORD をともに無視します。データベース・アクセスは、Net.Data が実行されているユーザー・プロファイルを介して経路指定されます。

**機密保護のヒント:** この値を Net.Data マクロにコーディングすることもできますが、アプリケーション・ユーザーに HTML 形式でユーザー ID を入力させることをお勧めします。さらに、Web サーバー ID のデフォルト値を使用すると、機密保護要件に合わない可能性のあるアクセス・レベルを提供します。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

## 値

LOGIN="database\_user\_id"

表 17. LOGIN の値

値	説明
database_user_id	有効なデータベース・ユーザー ID。デフォルトでは、Web サーバーを開始したユーザー ID を使用します。

## 例

**例 1:** ユーザー ID DB2USER へのアクセスを制限する

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

**例 2:** HTML 形式の入力行の使用

```
USERID&#58; <INPUT TYPE="text" NAME="LOGIN" SIZE=6>
```

この例は、アプリケーション・ユーザーが自分のユーザー ID を入力できるように、HTML 形式の一部として含めることができる行を示します。

## NULL\_RPT\_FIELD

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

### 目的

SQL の結果セットの中に戻される NULL 値を表すために、ユーザーが DTW\_SQL 言語環境に提供できるストリングを指定します。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

NULL\_RPT\_FIELD="null\_char"

表 18. NULL\_RPT\_FIELD の値

値	説明
null_char	SQL の結果セットの中に戻される NULL 値を表すストリングを指定する。デフォルトは空ストリング。

### 例

例 1: SQL 言語環境で NULL 値を表すストリングを指定する

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

## PASSWORD

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

### 目的

データベース言語環境にパスワードを渡すことによって、保護されたデータへのアクセスを提供します。この変数は LOGIN と一緒に使用して、DB2 の機密保護アルゴリズムを取り込みます。

**OS/400 ユーザー:** DATABASE 変数が定義されていない場合、あるいは `"*LOCAL"` という値に設定されている場合には、OS/400 は LOGIN と PASSWORD をともに無視します。データベース・アクセスは、Net.Data が実行されているユーザー・プロファイルを介して経路指定されます。

**機密保護のヒント:** この値を Net.Data マクロにコーディングすることもできますが、アプリケーション・ユーザーに HTML 形式でパスワードを入力させることをお勧めします。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

PASSWORD=`"password"`

表 19. PASSWORD の値

値	説明
<code>password</code>	データベース言語環境への自動アクセスを提供する、有効なパスワードを指定する。

### 例

**例 1:** パスワード NETDATA を持つアプリケーション・ユーザーへのアクセスを制限する

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

**例 2:** HTML 形式の入力行

```
PASSWORD&#58; <INPUT TYPE="password" NAME="PASSWORD" SIZE=8>
```

この例は、アプリケーション・ユーザーが自分のパスワードを入力できるように、HTML 形式の一部として含めることができる行を示します。

## SHOWSQL

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

### 目的

Web ブラウザーで使用される照会の SQL を隠すかまたは表示します。テスト中に SQL を表示すると、Net.Data マクロをデバッグするのに特に便利です。SHOWSQL は、Net.Data 構成ファイルで DTW\_SHOWSQL が YES に設定されている場合のみです。DTW\_SHOWSQL 構成変数に関する詳細は、*Net.Data 管理およびプログラミングの手引き* の構成の章を参照してください。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

SHOWSQL="YES"|"NO"

表 20. SHOW\_SQL の値

値	説明
YES	データベースに送信される照会の SQL を表示する。
NO	データベースに送信される照会の SQL を隠す。NO はデフォルト。

### 例

**例 1:** すべての SQL 照会を表示する。

構成ファイル中、以下を行います。

```
DTW_SHOWSQL YES
```

マクロ中、以下を行います。

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

**例 2:** HTML 形式の入力を使って、SQL を表示すべきか否かを指定する。

構成ファイル中、以下を行います。

```
DTW_SHOWSQL YES
```

マクロ中、以下を行います。

```
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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

データベースから戻された SQL 状態値をアクセスまたは表示します。

この変数は事前定義変数であり、その値を変更することはできません。この変数は変数参照として使用してください。

### 例

**例 1:** REPORT ブロックで SQL 状態を表示する

```
%FUNCTION (DTW_SQL) val1() {  
  select * from customer  
%REPORT {  
  ...  
%ROW {  
  ...  
%}  
  SQLSTATE=$(SQL_STATE)  
%}
```

## TRANSACTION\_SCOPE

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

### 目的

Net.Data が各 SQL コマンドの後、または HTML ブロックのすべての SQL コマンドが正常に終了した後のどちらで COMMIT を出すかを決定する、SQL コマンドのトランザクション効力範囲を指定します。コミットの前にすべての SQL コマンドが正常に終了する必要があると指定した場合、SQL コマンドが 1 つでも失敗すると、そのブロックで同じデータベースに対してそれ以前に実行されたすべての SQL がロールバックされます。

TRANSACTION\_SCOPE 変数を有効とするには、Net.Data 構成ファイルの ENVIRONMENT ステートメントにこの変数を組み込みます。これで、この変数の値は、DEFINE ステートメントを使って、または @DTW\_ASSIGN() 関数で指定できます。

**整合性の考慮事項:** OS/400 と OS/390 以外のオペレーティング・システムでは、以下の条件がすべて満たされる場合、データベースへの更新が失敗したという応答を受け取ると、同じ HTML ブロックでアクセスされる他のデータベースへの更新がコミットされても、失敗した更新はロールバックされる可能性があります。

- TRANSACTION\_SCOPE = "MULTIPLE" が指定されている。
- 1 つの HTML ブロックで複数のデータベースにアクセスしている (これは Live コネクションを使用すると可能)。
- 失敗したという応答が SQL 要求から戻された。

OS/400 で、または IBM の DataJoiner を使って、Net.Data から複数のデータベースにアクセスしている場合、Net.Data から更新すると、複数データベースの更新が調整され、整合性が保たれます。

OS/400 および OS/390 では、TRANSACTION\_SCOPE = "MULTIPLE" を指定すると、1 つの HTML ブロックから出されたすべての IBM データベース更新が一緒にコミットまたはロール・バックされます。

OS/400 以外のオペレーティング・システムでは、REXX、Perl、および Java 言語環境は、独自の別々のオペレーティング・システム・プロセスで実行します。したがって、これらの言語環境から発行されたデータベース更新はすべて、Net.Data の TRANSACTION\_SCOPE 値に関係なく、Net.Data マクロで発行されたデータベース更新とは別々にコミットまたはロールバックされます。

### 値

TRANSACTION\_SCOPE="SINGLE"|"MULTIPLE"

表 21. TRANSACTION\_SCOPE の値

値	説明
SINGLE	Net.Data は、HTML ブロック内のそれぞれの SQL コマンドが正常に終了した後で COMMIT を発行する。

表 21. TRANSACTION\_SCOPE の値 (続き)

値	説明
MULTIPLE	Net.Data は、HTML ブロック内のすべての SQL コマンドが正常に終了した後でのみ COMMIT を発行する。MULTIPLE はデフォルト。

## 例

**例 1:** 各トランザクションの後に COMMIT を発行するように指定する

```
%DEFINE TRANSACTION_SCOPE="SINGLE"
```



---

## Net.Data の各種変数

これらの変数は Net.Data によって定義される変数で、Net.Data 処理に影響を与え、関数呼び出しの状況を検出し、またデータベース照会の結果セットに関する情報を入手するために使用できます。さらに、ファイルの場所および日付に関する情報を判別することもできます。ユーザー自身で関数を作成する際に、またはユーザーの Net.Data マクロをテストする際に使用すると便利です。

- 116ページの『DTW\_CURRENT\_FILENAME』
- 117ページの『DTW\_CURRENT\_LAST\_MODIFIED』
- 118ページの『DTW\_DEFAULT\_MESSAGE』
- 119ページの『DTW\_LOG\_LEVEL』
- 120ページの『DTW\_MACRO\_FILENAME』
- 121ページの『DTW\_MACRO\_LAST\_MODIFIED』
- 122ページの『DTW\_MBMODE』
- 124ページの『DTW\_MP\_PATH』
- 125ページの『DTW\_MP\_VERSION』
- 126ページの『DTW\_PRINT\_HEADER』
- 127ページの『DTW\_REMOVE\_WS』
- 128ページの『RETURN\_CODE』

## DTW\_CURRENT\_FILENAME

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

### 目的

現行入力ファイルの名前と拡張子。入力ファイルは、Net.Data マクロ、または INCLUDE ステートメントで指定されたファイルのいずれかです。

この変数は事前定義変数であり、その値を変更することはできません。この変数は変数参照として使用してください。

### 例

```
<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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

現行ファイルが最後に変更された日付と時刻。現行ファイルは、`Net.Data` マクロ、または `INCLUDE` ステートメントで指定されたファイルのいずれかです。出力形式は、`Net.Data` が実行しているシステムによって決まります。

この変数は事前定義変数であり、その値を変更することはできません。この変数は変数参照として使用してください。

### 例

```
<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	SCO	SUN	Win NT
X	X	X	X		X	X	X	X

### 目的

呼び出しから組み込み関数へ、またはエラーが発生した場合は言語環境へ戻されるメッセージ・テキストを含みます。

DTW\_DEFAULT\_MESSAGE 変数は、Net.Data マクロの中のどこでも使用できます。

この変数は事前定義変数で、その値を変更することはお勧めできません。この変数は、変数参照として使用してください。

### 例

**例 1:** 関数が正常に終了したかどうかを示すメッセージ

```
@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
```

**例 2:** 関数が非ゼロのリターン・コードを戻した場合のデフォルト・テキスト

```
%MESSAGE{
default: {<h2>Net.Data received return code: $(RETURN_CODE).
Error message is $(DTW_DEFAULT_MESSAGE)</h2> %} : continue
%}
```

関数が 0 以外のリターン・コードを戻した場合、デフォルトのエラー・メッセージが表示されます。

## DTW\_LOG\_LEVEL

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

### 目的

Net.Data がログ・ファイルに書き込むメッセージのレベル。

この変数の値は、DEFINE ステートメントを使って、または @DTW\_ASSIGN() 関数で指定できます。

**要件:** ログングを初期化するために Net.Data 初期設定ファイル内で DTW\_LOG\_DIR を定義してください。そうしないと、マクロで DTW\_LOG\_LEVEL 変数を指定しても、Net.Data はメッセージをログに記録しません。

### 値

DTW\_LOG\_LEVEL="OFF|ERROR|WARNING"

表 22. DTW\_LOG\_LEVEL の値

値	説明
OFF	Net.Data はエラーをログに記録しない。OFF はデフォルト。
ERROR	Net.Data はエラー・メッセージをログに記録する。
WARNING	Net.Data はエラー・メッセージと警告メッセージをログに記録する。

### 例

```
%DEFINE DTW_LOG_LEVEL="ERROR"
```

## DTW\_MACRO\_FILENAME

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

### 目的

現行 Net.Data マクロ・ファイルの名前と拡張子。

この変数は事前定義変数であり、その値を変更することはできません。この変数は変数参照として使用してください。

### 例

```
<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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

Net.Data マクロが最後に変更された日付と時刻。出力形式は、Net.Data が実行しているシステムによって異なります。

この変数は事前定義変数であり、その値を変更することはできません。この変数は変数参照として使用してください。

### 例

```
<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	SCO	SUN	Win NT
X	X	X	X	X		X	X	X

### 目的

デフォルトの言語環境で使用するストリングおよびワードの関数のために、複数バイト文字セット (MBCS) のサポートを提供します。この変数は Net.Data 初期設定ファイルで設定できますが、この変数をマクロ内で使用しても、設定および現在の設定のオーバーライドができます。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

この変数は Net.Data 初期設定ファイルで設定できますが、この変数をマクロ内で使用しても、設定および現在の設定のオーバーライドができます。

**OS/400 ユーザー:** Net.Data for OS/400 は MBCS サポートの関数を自動的に使用可能にするので、この変数は必要ありません。Net.Data for OS/400 は、OS/400 オペレーティング・システムに移行されたマクロ・ファイルの中のこの変数を無視します。

この構成変数は、DTW\_UNICODE 構成変数とともに機能します。DTW\_UNICODE がデフォルト値 NO を使用する場合、値 DTW\_MBMODE が使用されます。

DTW\_UNICODE が NO 以外の値に設定された場合は、その値が使用されます。表23に、これら 2 つの変数の設定による、組み込み関数のストリング処理の決定方法を示します。

表 23. DTW\_UNICODE および DTW\_MBMODE の設定の関連

DTW_UNICODE の設定	DTW_MBMODE=YES の場合	DTW_MBMODE=NO の場合
NO	DBCS および SBCS をサポート	SBCS のみをサポート
UTF8	UTF-8 をサポート	UTF-8 をサポート

### 値

DTW\_MBMODE="YES" | "NO"

表 24. DTW\_MBMODE の値

値	説明
YES	ストリングおよびワードの関数のための MBCS サポートを指定する。
NO	ストリングおよびワードの関数が MBCS サポートを持たないことを指定する。NO はデフォルト。



## 例

例 1: INI ファイル内の値をオーバーライドする

INI ファイル:

```
DTW_MBMODE NO
```

マクロ:

```
%DEFINE DTW_MBMODE = "YES"
```

## DTW\_MP\_PATH

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

### 目的

Net.Data 実行可能ファイルのパスと名前。出力の表示はご使用のシステムによって異なりますが、以下のパスと名前の例のようになります。

/usr/lpp/internet/server\_root/cgi-bin/db2www

この変数は事前定義変数であり、その値を変更することはできません。この変数は変数参照として使用してください。

### 例

The Net.Data executable file is \$(DTW\_MP\_PATH).

## DTW\_MP\_VERSION

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

### 目的

サーバー上で実行している Net.Data のバージョンとリリース番号。

この変数は事前定義変数であり、その値を変更することはできません。この変数は変数参照として使用してください。

### 例

This Web application uses \$(DTW\_MP\_VERSION).

## DTW\_PRINT\_HEADER

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

### 目的

HTTP ヘッダーのテキストを指定します。

この変数は、Web ブラウザーに送られるテキストを Net.Data が処理する前に設定しなければなりません。なぜなら、Net.Data はテキストを表示する前に 1 度この変数を読み取るだけで、その後は参照しないからです。Net.Data がテキストをブラウザに送った後は、DTW\_PRINT\_HEADER 変数の変更は無視されます。

ユーザー独自のヘッダーを生成するために DTW\_PRINT\_HEADER を使用している場合は (DTW\_PRINT\_HEADER = "NO")、DTW\_REMOVE\_WS に "NO" を設定しなければなりません。

DEFINE ステートメントまたは @DTW\_ASSIGN() 関数を使用して、この変数の値を指定してください。

### 値

DTW\_PRINT\_HEADER="YES"|"NO"

表 25. DTW\_PRINT\_HEADER の値

値	説明
YES	Net.Data は、HTTP ヘッダーとしてテキスト Content-type: text/html を印刷する。YES はデフォルト。
NO	Net.Data は HTTP ヘッダーを印刷しない。ユーザー独自の HTTP ヘッダー情報を生成できる。

### 例

この変数が最も共通して使われる目的の 1 つは、Net.Data マクロが cookie を送信できるようにすることです。cookie を送信するためには DTW\_PRINT\_HEADER 変数を NO に設定し、最初の 3 行は Content-type ヘッダー、Set-Cookie ステートメント、そして空白行でなければなりません。

**例 1:** Net.Data が cookie を送信できるようにする

```
%DEFINE DTW_PRINT_HEADER="NO"
```

```
%HTML(cookie1) {  
Content-type: text/html  
Set-Cookie: UsrId=56, expires=Friday, 12-Dec-99, 12:00:00 GMT; path=/  
  
<P>  
Any text  
%}
```

## DTW\_REMOVE\_WS

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

### 目的

タブ、空白文字、および改行文字によって発生する余分なスペースを圧縮することによって、動的に生成される Web ページのサイズを小さくします。

この変数の値は DEFINE ブロックで指定します。

**<PRE></PRE> タグの使用:** この変数を YES に定義すると、印刷される空白文字の量とタイプに影響を与えます。変数を YES に設定した場合、<PRE></PRE> のタグを使用している HTML ページの一部が、予定どおりには表示されなくなります。

ユーザー独自のヘッダーを生成するために DTW\_PRINT\_HEADER を使用している場合は (DTW\_PRINT\_HEADER = "NO")、DTW\_REMOVE\_WS に "NO" を設定しなければなりません。

**OS/390 ユーザー:** すべてのユーザー・マクロに対して値を指定するためには、Net.Data 初期設定ファイルの中でこの変数を設定します。マクロ中でこの変数を定義すると、値をオーバーライドすることができます。DTW\_REMOVE\_WS がマクロで定義されていない場合は、初期設定ファイルの中の値を使用します。

### 値

DTW\_REMOVE\_WS="YES" | "NO"

表 26. DTW\_REMOVE\_WS の値

値	説明
YES	Net.Data は、2 つ以上の空白文字を 1 つの改行文字に圧縮し、より短い HTML の結果ページを生成する。
NO	Net.Data は空白文字を圧縮しない。NO はデフォルト。

### 例

**例 1:** 空白文字の圧縮

DTW\_REMOVE\_WS="YES"

## RETURN\_CODE

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

### 目的

組み込み関数または言語環境の呼び出しによって戻されるリターン・コード。Net.Data は、MESSAGE ブロックを処理するためにこの値を使用します。この変数を使って、関数呼び出しが正常終了したかまたは失敗したかを判別できます。値がゼロの場合は、関数呼び出しが正常終了したことを示します。

RETURN\_CODE 変数は、Net.Data マクロの中のどこでも参照できます。

この値は事前定義されています。値を変更することはお勧めできません。この変数は、変数参照として使用してください。

### 例

**例 1:** 関数が正常に終了したかどうかを示すメッセージ

```
@function1()
%IF ("$(RETURN_CODE)" == "0")
  The function completed successfully.
%ELSE
  The function failed with the return code $(RETURN_CODE).
%ENDIF
```

**例 2:** リターン・コードが 0 でない場合のデフォルトのメッセージ

```
%MESSAGE{
default: "<h2>Net.Data received return code: $(RETURN_CODE)</h2>" : continue
%}
```

関数が 0 以外のリターン・コードを戻した場合、デフォルトのエラー・メッセージが表示されます。

---

## 第3章 Net.Data 組み込み関数

Net.Data には、広範囲にわたる各種の関数が用意されており、ユーザーは自分で FUNCTION ブロックを作成しなくてもこれらの関数を使用することができます。Net.Data 組み込み関数は、以下のカテゴリーに分類されます。

- **汎用関数**は、Net.Data による Web ページの作成を援助する関数であり、他のカテゴリーには適合しません。131ページの『汎用関数』を参照してください。
- **数学関数**は数学演算を行います。164ページの『数学関数』を参照してください。
- **ストリング処理関数**は、ストリングと文字を変更します。183ページの『ストリング関数』を参照してください。
- **ワード処理関数**は、ワードまたはワード・セットを変更します。212ページの『ワード関数』を参照してください。
- **表処理関数**は、ユーザーが表データからフォームとレポートを生成するのを援助します。225ページの『表関数』を参照してください。
- **フラット・ファイル・インターフェース関数**は、ファイルの入出力を実行します。269ページの『フラット・ファイル・インターフェース関数』を参照してください。
- **Web レジストリー関数**は、Web レジストリーの操作を行います。299ページの『Web レジストリー関数』を参照してください。
- **永続マクロ関数**は、Net.Data におけるトランザクション処理をサポートします。319ページの『永続的なマクロ関数』を参照してください。

関数パラメーターには、**整数** または **浮動** 型を持つとされるものがありますが、これらの用語はそれぞれ整数値または浮動値を表すストリングを指すために使用されています。

---

### 関数名

Net.Data 組み込み関数は、予約済み接頭部である DTW で始まっています。ユーザー定義の関数は、この接頭部を使用することはできません。

Net.Data 組み込み関数以外の関数に DTW 接頭部を使用すると、予測できない振る舞いが生じることがあります。

組み込み関数名は、大文字小文字の区別がありません。

---

### 入出力パラメーター

関数には、Net.Data がそのパラメーターを入力、または出力、あるいはその両方に使用するかを判別する指定を渡すパラメーターを指定することができます。指定を渡すこれらのパラメーターは、以下のキーワードによって指定します。

**IN**      パラメーターが入力データを Net.Data から言語環境に渡すことを指定します。

**OUT** パラメーターが出力データを言語環境から Net.Data に戻すことを指定します。

**INOUT**

パラメーターが入力データを Net.Data から言語環境に渡し、出力データを言語環境から Net.Data に戻すことを指定します。

---

## 関数結果の形式化

多くの関数は、以下の 1 つまたは複数の形式になっています。

- DTW\_r、DTWF\_r、および DTWR\_r で始まる関数は、結果を関数呼び出しに戻します。このため、出力パラメーターはありません。次の例は、サーバー時刻を示しています。

```
Current local time is @DTW_rTIME().
```

- DTW\_m で始まる関数は、複数のパラメーターで指定された機能を実行します。各パラメーターは、入力パラメーターとしても出力パラメーターとしても働きます。パラメーターで指定された機能が実行され、結果はパラメーターに戻されます。次の例は、3 つの入力パラメーターをすべて大文字に変換して、表示画面上で一貫性のある形にしています。

```
@DTW_mUPPERCASE(model, style, shipNo)  
Shipment $(shipNo) contains $(quantity) of model $(model) $(style).
```

- DTW\_、DTWF\_、および DTWR\_ で始まるその他の関数は、結果を出力パラメーターに戻します。ユーザーは、出力パラメーターを指定する必要があります。次の例は、サーバー時刻を示しています。

```
@DTW_TIME(nowTime)  
Current local time is $(nowTime).
```

---

## 関数パラメーターの規則

関数パラメーターを正しい順序にします。すべての入力パラメーターを指定してから最後の入力パラメーターを指定する、またはヌル (『』) を指定してデフォルトを受け入れる必要があります。たとえば、次の例のように DTW\_TB\_INPUT\_TEXT を呼び出すことができます。

```
@DTW_TB_INPUT_TEXT(myTable, "1", "2", "", "", "32")
```

上の例では、4 番目と 5 番目のパラメーターがデフォルトの値を使用しています。これらのパラメーターをヌルとして組み込んで、"32" が、生成された HTML の MAXLENGTH の値であることを示します。最終パラメーターが指定されていないため、デフォルトの値が使用されます。MAXLENGTH のデフォルト値と前の 2 つのパラメーターを受け入れたい場合は、次の例のようにそれらを省略してください。

```
@DTW_TB_INPUT_TEXT(myTable, "1", "2")
```

後続の非ヌル入力パラメーターが存在する場合は、入力パラメーターのパラメーター・リストに中間ヌル値を指定する必要があります。最終出力パラメーターを指定するまでは、中間ヌル入力パラメーターを指定する必要はありません。



---

## 汎用関数

汎用関数は、Net.Data による Web ページの作成を援助する関数であり、他のカテゴリには適合しません。汎用関数は次のとおりです。

- 132ページの『DTW\_ADDQUOTE』
- 134ページの『DTW\_CACHE\_PAGE』
- 138ページの『DTW\_DATE』
- 140ページの『DTW\_EXIT』
- 141ページの『DTW\_GETCOOKIE』
- 143ページの『DTW\_GETENV』
- 145ページの『DTW\_GETINIDATA』
- 146ページの『DTW\_HTMLENCODER』
- 148ページの『DTW\_QHTMLENCODER』
- 149ページの『DTW\_SENDMAIL』
- 154ページの『DTW\_SETCOOKIE』
- 158ページの『DTW\_SETENV』
- 160ページの『DTW\_TIME』
- 162ページの『DTW\_URLESCSEQ』

## DTW\_ADDQUOTE

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

### 目的

入力ストリングの単一引用符を 2 つの単一引用符と置き換えます。

### 構文

@DTW\_ADDQUOTE(stringIn, stringOut)

@DTW\_rADDQUOTE(stringIn)

@DTW\_mADDQUOTE(stringMult, stringMult2, ..., stringMultn)

### 値

表 27. DTW\_ADDQUOTE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。 DTW_mADDQUOTE は複数の入力ストリングをもつことができます。
ストリング	<i>stringOut</i>	OUT	<i>stringIn</i> の変更形式が含まれている変数。
ストリング	<i>stringMult</i>	INOUT	<ul style="list-style-type: none"><li>入力の場合: ストリングが含まれている変数。</li><li>出力の場合: すべての単一引用符 (') 文字が 2 つの単一引用符文字と置き換えられた入力ストリングが含まれている変数。</li></ul>

### 戻りコード

表 28. DTW\_ADDQUOTE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 使用上の注意

入力が Web ブラウザーから取得される場合に、この関数をすべての SQL INPUT ステートメントに使用するものと考えてみます。たとえば、以下の例のように 0'Brien をラストネームとして入力すると、単一引用符がエラーになることがあります。

```
INSERT INTO USER1.CUSTABLE (LNAME, FNAME)
VALUES ('0'Brien', 'Patrick')
```

DTW\_ADDQUOTE 関数を使用すると、SQL ステートメントを変更してエラーを防止することができます。

```
INSERT INTO USER1.CUSTABLE (LNAME, FNAME)
VALUES ('O'Brien', 'Patrick')
```

## 例

**例 1:** エクストラ単一引用符を OUT パラメーターに追加します。

```
@DTW_ADDQUOTE(string1,string2)
```

- 入力: string1="John's Web page"
- 戻り: string2="John''s Web page"

**例 2** エクストラ単一引用符を関数呼び出しの戻り値に追加します。

```
@DTW_rADDQUOTE("The title of the article is 'Once upon a time'")
```

- 戻り: "The title of the article is ''Once upon a time''"

**例 3:** エクストラ単一引用符を関数呼び出しの各 INOUT パラメーターに追加します。

```
@DTW_mADDQUOTE(string1,string2)
```

- 入力: string1="Joe's bag", string2=""to be or not to be"
- 戻り: string1="Joe''s bag", string2=""'to be or not to be''"

**例 4:** DB2 表に挿入するデータにエクストラ単一引用符を挿入します。

```
%FUNCTION(DTW_SQL) insertName(){
INSERT INTO USER1.CUSTABLE (LNAME,FNAME)
VALUES ('@DTW_rADDQUOTE(lastname)', '@DTW_rADDQUOTE(firstname)')
%}
```

- 入力: lastname="O'Brien", firstname="Patrick"
- 戻り: "O''Brien", "Patrick"

## DTW\_CACHE\_PAGE

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

### 目的

マクロ処理の結果として生成された、部分的または完全な Web ページをキャッシュします。

### 構文

@DTW\_CACHE\_PAGE(cacheid, url, age, status)

### 値

表 29. DTW\_CACHE\_PAGE のパラメーター

パラメーター	用途	説明
<i>cache_id</i>	IN	ページを入れるキャッシュを識別するストリング変数。
<i>cached_page_ID</i>	IN	後続の DTW_CACHE_PAGE キャッシュ要求でキャッシュ・ページを見付けるために使用する識別子が含まれているストリング変数。このストリングを URL にすることができます。
<i>age</i>	IN	時間の長さ (秒数) が含まれているストリング変数。このパラメーターは、ページの有効期限が切れたかどうかを決定します。ページが <i>age</i> よりも古い場合には、このページはブラウザに送信されません。  <i>age</i> が -1 として指定され、ページがキャッシュに入っていると、Net.Data は、その経過日数とは無関係に、それをキャッシュから直接 Web ブラウザーに送信します。Net.Data は、キャッシュ内でページの置換は行いません。

表 29. DTW\_CACHE\_PAGE のパラメーター (続き)

パラメーター	用途	説明
<i>status</i>	OUT	<p>キャッシュ・ページの状態を示すストリング変数。使用できる値は小文字です。</p> <ul style="list-style-type: none"> <li>• <b>ok:</b> 出力ページは、マクロの実行が終了したときにキャッシュされます。</li> <li>• <b>new:</b> ページはキャッシュに入っていません。</li> <li>• <b>renew:</b> ページはキャッシュに入っていますが、有効期限が切れています。</li> <li>• <b>no_cache:</b> 指定されたキャッシュ識別子が存在しません。このキャッシュ識別子は、キャッシュ構成ファイルに定義されていなければなりません。ユーザー・マクロは、ページのキャッシングを行わなくても実行を継続することができます。</li> <li>• <b>inactive:</b> ユーザーが指定したキャッシュが、非活動状態としてマークされています。ユーザー・マクロは、ページのキャッシングを行わなくても実行を継続することができます。</li> <li>• <b>busy:</b> ユーザー・マクロがこの実行の前に DTW_CACHE_PAGE 組み込み関数を発行しました。ユーザー・マクロは実行を継続することができます。</li> <li>• <b>error:</b> キャッシュと通信しようとしているときにエラーが起きました。</li> </ul>

## 戻りコード

表 30. DTW\_CACHE\_PAGE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

## 使用上の注意

1. DTW\_CACHE\_PAGE() を起動すると、この関数は指定されたページをキャッシュから取り出し、それを、あたかもマクロから生成された出力ページであるかのように、Web ページに送信しようとします。このページが見付かり、まだ有効期限が切れていなければ、Net.Data はマクロ処理を停止し、マクロを終了し、キャッシュ・ページを Web ブラウザーに送信します。

要求されたページがキャッシュに入っていないか、または既存のキャッシュ・ページが *age* の値よりも古ければ、Net.Data は新規の出力ページを生成します。マクロが正常に完了すると、Net.Data は新規ページをブラウザーに送信し、このページをキャッシュします。

2. ほとんどのキャッシング・アプリケーションでは、マクロの先頭で DTW\_CACHE\_PAGE() を指定して、マクロ実行時に生成されるすべての Web ページをキャッシュします。この手法により、マクロの更新時におけるマクロの保守が容易になります。たとえば、関数がマクロの中間に入っていたりすると、マクロの始めの部分で HTML レポート・セクションを追加するときに、この関数を見過ごす可能性があります。Net.Data は新規のレポート出力をキャッシュしません。さらに、この方法を使用するとパフォーマンスも向上します。それは、ページのキャッシングを決定するときに、Net.Data がそれ以降のすべての処理を停止するからです。

拡張キャッシング・アプリケーションの場合は、マクロの先頭ではなく、処理時の特定の時点でキャッシングを決定しなければならないときに、関数をマクロの指定位置に配置することができます。たとえば、照会または関数呼び出しからいくつの行が戻されたかに基づいて、キャッシングの決定を行わなければならない場合があります。

## 例

**例 1:** マクロの先頭に DTW\_CACHE\_PAGE() 関数を入れて、すべての HTML 出力を取り込みます。

```
%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>

%}
```

**例 2:** キャッシングの決定が HTML 出力の予期サイズに依存しているため、関数を HTML ブロックに入れます。

```
%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>
}%

```

この例では、HTML の予期サイズに基づいて、ページのキャッシングまたは検索を行います。HTML 出力ページは、データベース表に 101 行以上含まれているときにのみキャッシングの価値があると考えられます。Net.Data は、マクロを実行した後、常に、OUTPUT ブロックのテキスト This is the customer list をブラウザに送信します。関数呼び出しの後に続く行 @count\_rows() は、IF ブロックの条件が満たされたときにキャッシングまたは検索されます。両方の部分が組み合わさって、完全な Net.Data 出力ページが形成されます。

**例 3:** キャッシュ ID とキャッシュ・ページ ID を動的に検索します。

```

%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!
  </body>
  </html>

}%

```

## DTW\_DATE

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

### 目的

指定された形式の現行システム日付を戻します。

### 構文

@DTW\_DATE(format, stringOut)

@DTW\_DATE(stringOut)

@DTW\_rDATE(format)

@DTW\_rDATE()

### 値

表 31. DTW\_DATE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>format</i>	IN	データ形式を指定する変数またはリテラル・ストリング。有効な形式としては、以下のものがあります。  D - 年通算日 (001-366) E - 欧州日付形式 (dd/mm/yy) N - 一般日付形式 (dd mon yyyy) O - 順序日付形式 (yy/mm/dd) S - 標準日付形式 (yyyymmdd) U - 米国日付形式 (mm/dd/yy)  デフォルトは N です。
ストリング	<i>stringOut</i>	OUT	指定形式の日付が含まれている変数。

### 戻りコード

表 32. DTW\_DATE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。



## 例

### 例 1: 一般日付形式

```
@DTW_DATE(results)
```

- 戻り: results = "25 Apr 1997"

### 例 2: 欧州日付形式

```
@DTW_DATE("E", results)
```

- 戻り: results="25/04/97"

### 例 3: 米国日付形式

```
%HTML(report){  
<P>This report created on @DTW_rDATE("U").
```

- 戻り: 04/25/97

## DTW\_EXIT

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

### 目的

即時にマクロから出ることを指定します。Net.Data は、DTW\_EXIT() が Web ブラウザーに呼び出される前に生成された Web ページをいずれも送信します。

### 構文

@DTW\_EXIT()

### 戻りコード

表 33. DTW\_EXIT の戻りコード

戻りコード	説明
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。

### 使用上の注意

1. DTW\_EXIT() を使用することにより、マクロ処理を即時に停止します。この手法を用いると、Net.Data がファイル全体の処理に費やす時間が短縮できます。
2. DTW\_EXIT() 関数を追加する前に、マクロ全体が構文的に正しいことを確認してください。DTW\_EXIT() を使用すると、Net.Data は、この関数への呼び出しを検出したときにマクロの処理を停止するようになります。これにより、DTW\_EXIT() 関数が処理された後で発生したエラーをキャッチできなくなることがあります。

### 例

例 1: マクロを終了します。

```
%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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

指定された Cookie の値を戻します。

### 構文

@DTW\_GETCOOKIE(IN cookie\_name, OUT cookie\_value)

@DTW\_rGETCOOKIE(IN cookie\_name)

### 値

表 34. DTW\_GETCOOKIE のパラメーター

データ型	パラメーター	用途	説明
ストリング	cookie_name	IN	cookie の名前を指定する変数またはリテラル・ストリング。
ストリング	cookie_value	OUT	ユーザー状態情報などの関数が検索する cookie の値を含む変数。  <b>OS/400 および OS/390 ユーザーの場合:</b> cookie の値が URL スタイルのエンコードを含む場合 (たとえば "%20")、cookie の値は、値が戻される前に復号されます。  <b>ワークステーション・ユーザーの場合:</b> cookie の値が URL スタイルのエンコードを含む場合 (たとえば "%20")、cookie の値は、値が戻される前に復号されることはありません。

### 戻りコード

表 35. DTW\_GETCOOKIE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
8000	cookie が検出できません。

## 使用上の注意

2 つの異なる HTTP 要求にて cookie を定義し検索してください。 cookie が表示されるのは、クライアントに送信された後だけであるため、同じ HTTP 要求で定義された cookie を入手しようとする、予期しない結果を受け取る可能性があります。

### 例

**例 1:** ユーザー ID 情報とパスワード情報を含む cookie を検索します。

```
@DTW_GETCOOKIE("mycookie_name_for_userID", userID)
@DTW_GETCOOKIE("mycookie_name_for_password", password)
```

**例 2:** ユーザー情報を集める前に、ユーザーのための cookie が存在するかどうかを判別します。

```
%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.
    %ENDIF
    </body>
    </html>
    %}
```

HTML ウェルカム・セクションでは、cookie NDC\_name が存在するかどうかを検査します。 cookie が存在する場合には、ブラウザーに個別設定されたあいさつが表示されます。 cookie が存在しない場合には、ユーザー名を要求するプロンプトがブラウザーに表示され、そのユーザー名を HTML remember セクションに通知します。以下のように、このセクションはユーザー名を cookie NDC\_name に設定します。

```
%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><a href="welcome">Come back</a>
    </body>
    </html>
    %}
```

## DTW\_GETENV

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

### 目的

指定された環境変数の値を返します。

### 構文

@DTW\_GETENV(envVarName, envVarValue)

@DTW\_rGETENV(envVarName)

### 値

表 36. DTW\_GETENV のパラメーター

データ型	パラメーター	用途	説明
ストリング	envVarName	IN	変数またはリテラル・ストリング。
ストリング	envVarValue	OUT	envVarName に指定された環境変数の値。 この値が見付からないと、ヌル・ストリングが返されます。

### 戻りコード

表 37. DTW\_GETENV の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 使用上の注意

ENVVAR ステートメントを使用して、環境変数の値を参照することもできます。詳しくは、15ページの『ENVVAR ステートメント』を参照してください。

### 例

**例 1:** OUT パラメーター上の PATH ステートメントの値を返します。

```
@DTW_GETENV(myEnvVarName, myEnvVarValue)
```

- 入力: myEnvVarName = "PATH"
- 戻り: myEnvVarValue = "/usr/bin"

**例 2** PATH ステートメントの値を返します。

```
@DTW_rGETENV(myPath)
```

- 入力: myPath = "PATH"
- 戻り: "/usr/bin"

**例 3:** サーバー・プロトコルの値を戻します。

The server is @DTW\_rGETENV("SERVER\_PROTOCOL").

- 戻り: "HTTP/1.0"

## DTW\_GETINIDATA

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

### 目的

指定された構成変数の値を返します。

### 構文

```
@DTW_GETINIDATA(iniVarName, iniVarValue)
```

```
@DTW_rGETINIDATA(iniVarName)
```

### 値

表 38. DTW\_GETINIDATA のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>iniVarName</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>iniVarValue</i>	OUT	<i>iniVarName</i> に指定された構成変数の値。

### 戻りコード

表 39. DTW\_GETINIDATA の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 使用上の注意

1. 構成ファイルにない構成変数が指定されていると、Net.Data は空ストリングを返します。
2. **OS/390、OS/2、Windows NT、および UNIX ユーザーの場合:** ENVIRONMENT ステートメントと同様、構成パス変数 (MACRO\_PATH、EXEC\_PATH、および INCLUDE\_PATH) も、この呼び出しで検索することはできません。
3. **OS/400 ユーザーの場合:** ENVIRONMENT ステートメントは、この呼び出しで検索することはできません。

### 例

例 1: Net.Data パス変数値を返します。

```
@DTW_GETINIDATA(myEnvVarName, myEnvVarValue)
```

- 入力: myEnvVarName = "FFI\_PATH"
- 戻り: myEnvVarValue = "D:¥FFI"

## DTW\_HTMLENCODER

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

### 目的

選択した文字を、HTML 文字エスケープ・コードを使用してエンコードします。

### 構文

@DTW\_HTMLENCODER(stringIn, stringOut)

@DTW\_rHTMLENCODER(stringIn)

### 値

表 40. DTW\_HTMLENCODER のパラメーター

データ型	パラメーター	用途	説明
ストリング	stringIn	IN	変数またはリテラル・ストリング。
ストリング	stringOut	OUT	特定の文字が HTML 文字エスケープ・コードによって置き換えられた変更入力ストリングが含まれている変数。

### 戻りコード

表 41. DTW\_HTMLENCODER の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 使用上の注意

- この関数を使用し、Web ブラウザーに HTML として解釈させたくない文字データをエンコードすることができます。たとえば、適切なエスケープ・コードを使用することにより、「より小 (<)」および「より大 (>)」などの文字を表示することができます。通常ブラウザは、これらの文字を HTML タグの構成要素として解釈します。
- 表42 は、DTW\_HTMLENCODER 関数によってエンコードされた文字を示しています。

表 42. HTML の文字エスケープ・コード

文字	名前	コード
SPACE	スペース	&#32;
"	二重引用符	&#34;



表 42. HTML の文字エスケープ・コード (続き)

#	番号記号	&#35;
%	パーセント	&#37;
&	アンパーサンド	&#38;
[	左大括弧	&#40;
]	右大括弧	&#41;
+	プラス	&#43;
/	スラッシュ	&#47;
:	コロン	&#58;
;	セミコロン	&#59;
<	より小 (LT)	&#60;
=	等しい	&#61;
>	より大 (GT)	&#62;
?	疑問符	&#63;
@	アットマーク	&#64;
¥	円記号	&#92;
^	caret	&#94;
{	左中括弧	&#123;
	縦線	&#124;
}	右中括弧	&#125;
~	ティルド	&#126;

## 例

例 1: スペース文字をエンコードします。

```
@DTW_HTML_ENCODE(string1,string2)
```

- 入力: string1 = "Jim's dog"
- 戻り: string2 = "Jim's&#32;dog"

例 2: スペース、'より小' 符号、および等号をエンコードします。

```
@DTW_rHTML_ENCODE("X <= 10")
```

- 戻り: "X&#32;&#60;&#61;&#32;10"

## DTW\_QHTMLENCODE

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

### 目的

@DTW\_HTMLENCODE と同じ機能を実行しますが、単一引用符 (') を &#39; としてエンコードもします。DTW\_QHTMLENCODE が使用する HTML 文字エスケープ・コードが、146ページの表42 に示されています。

### 構文

```
@DTW_QHTMLENCODE(stringIn, stringOut)
```

```
@DTW_rQHTMLENCODE(stringIn)
```

### 値

表 43. DTW\_QHTMLENCODE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>stringOut</i>	OUT	特定の文字が HTML 文字エスケープ・コードによって置き換えられた、変更形式 <i>stringIn</i> が含まれている変数。

### 戻りコード

表 44. DTW\_QHTMLENCODE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 例

例 1: アポストロフィとスペースをエンコードします。

```
@DTW_QHTMLENCODE(string1,string2)
```

- 入力: string1 = "Jim's dog"
- 戻り: string2 = "Jim&#39;s&#32;dog"

例 2: アポストロフィ、スペース、およびアンパーサンドをエンコードします。

```
@DTW_rQHTMLENCODE("John's & Jane's")
```

- 戻り: "John&#39;s&#32;&#38;&#32;Jane&#39;s"

## DTW\_SENDMAIL

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

### 目的

電子メール (e-mail) メッセージを動的に作成し、送信します。

### 構文

@DTW\_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject, IN CarbonCopy, IN BlindCarbonCopy, IN ReplyTo, IN Organization)

@DTW\_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject, IN CarbonCopy, IN BlindCarbonCopy, IN ReplyTo)

@DTW\_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject, IN CarbonCopy, IN BlindCarbonCopy)

@DTW\_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject, IN CarbonCopy)

@DTW\_SENDMAIL(IN Sender, IN Recipient, IN Message, IN Subject)

@DTW\_SENDMAIL(IN Sender, IN Recipient, IN Message)

### 値

表 45. DTW\_SENDMAIL のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>sender</i>	IN	作成者のアドレスを指定する変数またはリテラル・ストリング。このパラメーターは必須です。有効な形式は、次のとおりです。 <ul style="list-style-type: none"><li>名前 &lt;user@domain&gt;</li><li>&lt;user@domain&gt;</li><li>user@domain</li></ul>
ストリング	<i>recipient</i>	IN	このメッセージの送信先の電子メール・アドレスを指定する変数またはリテラル・ストリング。この値には、コンマ (,) で区切られた複数の受信側を含むことができます。このパラメーターは必須です。有効な <i>recipient</i> の形式は次のとおりです。 <ul style="list-style-type: none"><li>名前 &lt;user@domain&gt;</li><li>&lt;user@domain&gt;</li><li>user@domain</li></ul>
ストリング	<i>message</i>	IN	電子メール・メッセージのテキストを含む変数またはリテラル・ストリング。このパラメーターは必須です。

表 45. DTW\_SENDMAIL のパラメーター (続き)

データ型	パラメーター	用途	説明
ストリング	<i>subject</i>	IN	対象行のテキストを含む変数またはリテラル・ストリング。これはオプション・パラメーターです。ヌル・ストリング ("" ) を指定して、追加のパラメーターを指定する必要があります。
ストリング	<i>CarbonCopy</i>	IN	電子メール・アドレスまたは追加の受信側の名前と電子メールアドレスを含む変数またはリテラル・ストリング。この値には、コンマ (,) で区切られた複数の追加の受信側を含むことができます。有効な受信側形式については、 <i>Recipient</i> パラメーターを参照してください。これはオプション・パラメーターです。ヌル・ストリング ("" ) を指定して、追加のパラメーターを指定する必要があります。
ストリング	<i>BlindCarbonCopy</i>	IN	電子メール・アドレスまたは追加の受信側の名前と電子メールアドレスを含む変数またはリテラル・ストリング。ただし、受信側は電子メールのヘッダーには表示されません。この値には、コンマ (,) で区切られた複数の追加の受信側を含むことができます。有効な受信側形式については、 <i>Recipient</i> パラメーターを参照してください。これはオプション・パラメーターです。ヌル・ストリング ("" ) を指定して、追加のパラメーターを指定する必要があります。
ストリング	<i>ReplyTo</i>	IN	このメッセージの応答を返す電子メール・アドレスを含む変数またはリテラル・ストリング。これはオプション・パラメーターです。ヌル・ストリング ("" ) を指定して、追加のパラメーターを指定する必要があります。有効な <i>ReplyTo</i> の形式は次のとおりです。 <ul style="list-style-type: none"> <li>• 名前 &lt;user@domain&gt;</li> <li>• &lt;user@domain&gt;</li> <li>• user@domain</li> </ul>
ストリング	<i>Organization</i>	IN	<i>sender</i> の編成名を含む変数またはリテラル・ストリング。これはオプション・パラメーターです。

## 戻りコード

表 46. DTW\_SENDMAIL の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てたための Net.Data 要求を処理できませんでした。

表 46. DTW\_SENDMAIL の戻りコード (続き)

戻りコード	説明
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
7000	指定した SMTP サーバーに Net.Data が接続できません。
7001	Net.Data が電子メール・メッセージを指定した SMTP サーバーに中継しようとした際に、SMTP エラーが発生しました。
7002	指定された SMTP サーバーが、拡張シンプル・メール転送プロトコル (SMTP) をサポートしません。

## 使用上の注意

- オプションの構成変数 DTW\_SMTP\_SERVER を使用して、電子メール・メッセージを送信するために使用する SMTP サーバーを指定することができます。このパラメーター値は、ホスト名、IP アドレス、またはノード名といずれかとすることができます。この変数を定義しないと、Net.Data は SMTP サーバーとしてローカル・ホストを使用します。この変数に関する詳細については、*Net.Data 管理およびプログラミングの手引き* の構成の章を参照してください。
- OS/400、OS/2、Windows NT、および UNIX ユーザーの場合:** SMTP (標準シンプル・メール転送プロトコル) サーバーは、7 ビット・データ (米国 ASCII 文字など) しか受け入れません。メッセージに 8 ビット文字が入っている場合は、ESMTP (拡張シンプル・メール転送プロトコル) サーバーを指定することをお勧めします。ESMTP は、8 ビット文字を受け入れます。Net.Data は、8 ビット・データを 7 ビット・データにエンコードしません。ESMTP サーバーにアクセスできない場合は、電子メール・メッセージから 8 ビット文字をすべて取り除いてください。

Net.Data for OS/390 ユーザーの場合、SMTP サーバーへの電子メール・メッセージを変更する必要はありません。

- 文字セット・サポート:
  - OS/400 ユーザーの場合:** オプションの構成変数 DTW\_SMTP\_CHARSET を使用して、メッセージを EBCDIC から ASCII に変換する際に使用する ASCII 文字を指定することができます。DTW\_SMTP\_CHARSET が指定されない場合、デフォルトの文字セットは iso-8859-1 となります。この変数およびサポートされる文字セットの詳細については、*Net.Data 管理およびプログラミングの手引き OS/400* の構成の章を参照してください。
  - OS/2、Windows NT、および UNIX ユーザーの場合:** 表47 にサポートされる文字セットを示します。

表 47. Net.Data がサポートする文字セット

ロケール	文字セット	OS/2 または UNIX コード・ページ	Windows NT コー ド・ページ
U.S、西欧	"iso-8859-1"	819	1252

表 47. Net.Data がサポートする文字セット (続き)

ロケール	文字セット	OS/2 または UNIX コード・ページ	Windows NT コー ド・ページ
日本	"x-sjis"	943	932
中国 (簡体字)	"gb2312"	1381	936
韓国	"euc-kr"	970	949
中国 (繁体字)	"big5"	950	950

4. 以下のリストでは、Net.Data が電子メール・メッセージを送信しない条件について説明しています。

- ・ 指定された SMTP サーバーに到達することができない。
- ・ 指定された SMTP サーバーが拡張単一メール・プロトコル (ESMTP) をサポートしないが、指定された電子メール・メッセージに非 U.S. ASCII 文字が含まれている。

## 例

**例 1：** 単一の電子メール・メッセージを作成し送信する関数呼び出し

```
@DTW_SENDMAIL("<ibmuser1@ibm.com>", "<ibmuser2@ibm.com>", "There is a meeting at 9:30.",
"Status meeting")
```

DTW\_SENDMAIL 関数は、以下の情報を含む電子メール・メッセージを送信します。

```
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.

Date の情報はシステム日時関数を使用して構成され、SMTP 固有のデータ形式で形式化されます。

**例 2：** 複数の受信者側、カーボン・コピーと受信停止カーボン・コピーの受信者側、および会社名を含む電子メール・メッセージを作成し送信する関数呼び出し

```
@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")
```

DTW\_SENDMAIL 関数は、以下の情報を含む電子メール・メッセージを送信します。

```
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.

**例 3：** Web 形式インターフェースを経由して電子メールを作成し送信するマクロ

```
%HTML(start) {
<html>
<body>
<h1>Net.Data E-Mail Example</h1>
<form method="post" action="sendemail">
```

```

<p>To:<br><input name="recipient"><p>
Subject:<br><input name="subject"><p>
Message:<br><textarea name=message rows=20 cols=40>
</textarea><p>
<input type="submit" value="Send E-mail"><br>
</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.
</body>
</html>
%}

```

このマクロは、Web 形式インターフェースを経由して電子メールを送信します。HTML 開始セクションには、受信側の電子メール・アドレス、主題、およびメッセージを入力することができる形式が表示されます。ユーザーが「**電子メールの送信**」ボタンをクリックすると、HTML (sendemail) セクションで指定される受信側に、メッセージが送信されます。このセクションでは DTW\_SENDMAIL を呼び出し、Web 形式から獲得したパラメーターを使用して、送信側と受信側だけでなく、電子メールの内容も判別します。電子メールを送信してしまうと、確認通知が表示されます。

#### 例 4：SQL 照会を使用して、受信側のリストを判別するマクロ

```

%Function(DTW_SQL) mailing_list(IN message) {
  SELECT EMAIL_ADDRESS FROM CUSTOMERS WHERE ZIPCODE='CA'
%REPORT {
  Sending out product information to all customers who live 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>
  %}
%}
%}

```

このマクロは、顧客データベースから出された SQL 照会の結果で判別される顧客の指定したグループに、自動化された電子メール・メッセージを送信します。SQL 照会は、顧客の電子メール・アドレスも検索します。電子メールの内容は *message* の値で判別され、静的または動的であることができます (たとえば、ほかの SQL 照会を使用して、製品のバージョン番号またはさまざまなオファリングの価格を動的に指定することができます)。

## DTW\_SETCOOKIE

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

### 目的

クライアント・システムで cookie の設定をする JavaScript コードを生成します。

### 構文

@DTW\_SETCOOKIE(IN cookie\_name, IN cookie\_value, IN advanced\_options)

@DTW\_SETCOOKIE(IN cookie\_name, IN cookie\_value)

### 値

表 48. DTW\_SETCOOKIE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>cookie_name</i>	IN	cookie の名前を指定する変数またはリテラル・ストリング
ストリング	<i>cookie_value</i>	IN	cookie の値を指定する変数またはリテラル・ストリング  セミコロン、コンマ、およびスペースを <i>cookie_value</i> のパーツとして使用しないでください。これらが必要な場合は、 Net.Data 関数 DTW_rURLESCSEQ を使用して、 DTW_SETCOOKIE に渡す前に、特殊文字を含むストリングを 処理します。たとえば、次のようにします。  @DTW_SETCOOKIE("my_cookie_name", @DTW_rURLESCSEQ("my cookie value"))



表 48. DTW\_SETCOOKIE のパラメーター (続き)

データ型	パラメーター	用途	説明
ストリング	<i>advanced_options</i>	IN	<p>セミコロンで区切られた任意選択の属性を含むストリングで、cookie を定義するために使用します。属性には以下のものがあります。</p> <p><b>expires = date</b>  cookie の有効な存続時間を定義する日付ストリングを指定します。有効期限が切れた cookie は、保管されたり検索されたりしなくなります。構文:  <b>weekday, DD-month-YYYY HH:MM:SS GMT</b></p> <p>ここで、  <b>weekday</b>  曜日の完全な名前を指定します。  <b>DD</b> その月の日付を数値で指定します。  <b>month</b>  その月の省略語を 3 文字で指定します。  <b>YYYY</b>  西暦を 4 文字の数字で指定します。  <b>HH:MM:SS</b>  タイム・スタンプを時間、分、秒の順序で指定します。</p> <p><b>domain = domain_name</b>  cookie のドメイン属性を指定し、ドメイン属性のマッチングに使用します。</p> <p><b>path = path</b>  cookie が有効なドメイン内で URL のサブセットを指定します。</p> <p><b>secure</b> cookie がセキュア・チャネルだけを經由して HTTPS サーバーに送信されることを指定します。</p> <p><b>secure</b> オプションが指定されていないと、cookie は非セキュア・チャネルを經由して送信される場合があります。セキュア・オプションを指定すれば、ブラウザは cookie を暗号化する必要も、また DTW_SETCOOKIE ステートメントを含むページが SSL 上で送信されることを確認する必要もありません。</p> <p>拡張オプションに関する追加情報は、  <a href="http://home.netscape.com">http://home.netscape.com</a> にある Netscape cookie 仕様を参照してください。</p>

## 戻りコード

表 49. DTW\_SETCOOKIE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。

## 使用上の注意

1. クライアントである Web ブラウザーが Java Script をサポートしていない場合には、ブラウザーは cookie を設定しません。
2. DTW\_SETCOOKIE が Java Script コードを生成するので、<SCRIPT> または <NOSCRIPT> HTML 要素の内側で DTW\_SETCOOKIE を呼び出さないでください。
3. cookie を検索するには、DTW\_GETCOOKIE() 関数を使用します。cookie の定義方法については、141ページの『DTW\_GETCOOKIE』を参照してください。
4. 2 つの異なる HTTP 要求しい cookie を定義し検索します。cookie が表示されるのは、クライアントに送信された後だけであるため、同じ HTTP 要求で定義された cookie を入手しようとする、予期しない結果を受け取る可能性があります。

## 例

**例 1:** セキュア高機能オプションとともに、ユーザー ID 情報およびパスワード情報を含む cookie を定義します。

```
@DTW_SETCOOKIE("mycookie_name_for_userID", "User1")
@DTW_SETCOOKIE("mycookie_name_for_password", "sd3dT", "secure")
```

**例 2:** 有効期限高機能オプションを含む cookie を定義します。

```
@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")
```

関数呼び出しは 1 行に指定しなくてはなりません。この例では、フォーマットの都合上、改行してあります。

**例 3:** ユーザー情報を集める前に、ユーザーのための cookie が存在するかどうかを定義します。

```
%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>
%}

```

HTML (welcome) セクションでは、cookie NDC\_name が存在するかどうかを検査します。 cookie が存在する場合には、ブラウザーに個別設定されたあいさつが表示されます。 cookie が存在しない場合には、ユーザー名を要求するプロンプトがブラウザーに表示され、そのユーザー名を HTML (remember) セクションに通知します。このセクションでは、以下のように、ユーザー名を cookie NDC\_name に記録します。

```

%HTML(remember) {
    <html>
    <body>
    <H1>Net.Data Club>
    @DTW_SETCOOKIE("NDC_name", name, "expires=Wednesday, 01-Dec-2010 00:00:00;path=/")
    <p>Thank you.
    <p><a href="welcome">Come back</a>
    </body>
    </html>
%}

```

## DTW\_SETENV

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

### 目的

指定値をもつ環境変数を割り当て、前の値を戻します。

### 構文

@DTW\_SETENV(envVarName, envVarValue, prevValue)

@DTW\_rSETENV(envVarName, envVarValue)

### 値

表 50. DTW\_SETENV のパラメーター

データ型	パラメーター	用途	説明
ストリング	envVarName	IN	環境変数を表す変数またはリテラル・ストリング。
ストリング	envVarValue	IN	環境変数が割り当てられた値が含まれている変数またはリテラル・ストリング。
ストリング	prevValue	OUT	前の環境変数値が含まれている変数。

### 戻りコード

表 51. DTW\_SETENV の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 使用上の注意

環境変数の以前の値が検出できなければ、空ストリングが戻されます。

### 例

例 1: 前のパスの値を戻します。

```
@DTW_SETENV("PATH", "myPath", prevValue)
```

- 入力: envVarName = "PATH", envVarValue = "myPath"
- 戻り: prevValue = "myPreviousPath"

例 2: 前のパスの値を戻し、PATH の値を割り当てます。

```
@DTW_rSETENV("PATH",  
"myPath")
```

- 入力: envVarName = "PATH", envVarValue = "myPath"
- 戻り: "myPreviousPath", PATH = "myPath"

## DTW\_TIME

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

### 目的

現行システム時刻を指定された形式で戻します。

### 構文

@DTW\_TIME(stringIn, stringOut)

@DTW\_TIME(stringOut)

@DTW\_rTIME(stringIn)

@DTW\_rTIME()

### 値

表 52. DTW\_TIME のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	時刻形式を指定する変数またはリテラル・ストリング。有効な形式は、次のとおりです。  C - 常用時間 (12 時間クロックを使用する hh:mmAM/PM) L - 現地時間 (hh:mm:ss) N - 標準時間 (24 時間クロックを使用する hh:mm:ss。デフォルト) X - 拡張時間 (24 時間クロックを使用する hh:mm:ss.ccc。ccc はミリ秒数) H - 午前 0 時以降の時間数 M - 午前 0 時以降の分数 S - 午前 0 時以降の秒数
ストリング	<i>stringOut</i>	OUT	指定形式の時刻が含まれている変数。

### 戻りコード

表 53. DTW\_TIME の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

## 例

例 1: 24 時間クロック形式。

```
@DTW_TIME(results)
```

- 戻り: results = "10:30:53"

例 2: 常用時刻形式。

```
@DTW_TIME("C", results)
```

- 戻り: results = "10:30AM"

例 3: 関数呼び出しにより、午前 0 時以降の分数を戻します。

```
@DTW_rTIME("M")
```

- 戻り: "630"

例 4: 関数呼び出しにより、デフォルトの時刻とデータ形式を戻します。

```
%REPORT{  
<P>This report was created at @DTW_rTIME(), @DTW_rDATE().  
%}
```

- 戻り: This report was created 15:04:39, 01 May 1997.

## DTW\_URLESCSEQ

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

### 目的

URL に使用できない選択文字をエスケープ値 (URL エンコード・コードとも呼ばれます) と置き換えます。

### 構文

```
@DTW_URLESCSEQ(stringIn, stringOut)
```

```
@DTW_rURLESCSEQ(stringIn)
```

### 値

表 54. DTW\_URLESCSEQ のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>stringOut</i>	OUT	対応する 16 進エスケープ値に置き換えられる URL に使用できない文字をもつ入力ストリングが含まれている変数。

### 戻りコード

表 55. DTW\_URLESCSEQ の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 使用上の注意

この関数を使用して、表56 にリストされている任意の文字を他のマクロまたは HTML ブロックに渡します。

表 56. URL の文字エスケープ値

文字	名前	コード
SPACE	スペース	%20
"	二重引用符	%22
#	番号記号	%23
%	パーセント	%25
&	アンパーサンド	%26
+	プラス	%2B



表 56. URL の文字エスケープ値 (続き)

¥	円記号	%2F
:	コロン	%3A
;	セミコロン	%3B
<	より小 (LT)	%3C
=	等しい	%3D
>	より大 (GT)	%3E
?	疑問符	%3F
@	アットマーク	%40
[	左大括弧	%5B
/	スラッシュ	%5C
]	右大括弧	%5D
^	caret	%5E
{	左中括弧	%7B
	縦線	%7C
}	右中括弧	%7D
~	ティルド	%7E

## 例

**例 1:** `string1` 内のスペースとアンパーサンドをそのエスケープ値に置き換え、結果を `string2` に割り当てます。

```
@DTW_URLESCSEQ(string1,string2)
```

- 入力: `string1 = "Guys & Dolls"`
- 戻り: `string2 = "Guys%20%26%20Dolls"`

**例 2:** スペースおよびアンパーサンド文字をそのエスケープ・コードに置き換えます。

```
@DTW_rURLESCSEQ("Guys & Dolls")
```

- 戻り: `"Guys%20%26%20Dolls"`

**例 3:** ROW ブロックの `DTW_rURLESCSEQ` を使用し、スペースとアットマークをそのエスケープ・コードに置き換えます。

```
%ROW{
<P><a href="fullrpt.mac/input?name=@DTW_rURLESCSEQ(V1)&email=@DTW_rURLESCSEQ(V2)">
$(V1)</a>
%}
```

- 入力: `V1="Patrick O'Brien", V2="obrien@ibm.com"`
- 戻り:

```
<P><a href="fullrpt.mac/input?name=Patrick%20'O'Brien&email="obrien%40ibm.com">
Patrick O'Brien</a>
```

アプリケーション・ユーザーが名前 "Patrick O'Brien" をクリックすると、名前と電子メール・アドレスに指定された値が URL の照会ストリング内にフローし、`Net.Data` が `fullrpt.mac` マクロの入力セクションを実行します。

---

## 数学関数

これらの関数を使用すると、数値計算を行うことができます。

**数学関数の場合の NLS の考慮事項:** Net.Data は、Net.Data の実行環境下で Web サーバーで指定されている地域設定値に基づいて、数値内の小数点を表示します。たとえば、Web サーバーで小数点がコンマ (,) として指定されている場合、Net.Data はコンマを使用して 10 進データを形式設定します。Net.Data は、以下の設定を使用して、小数点を指定するために使用する文字を判別します。

**OS/390、Windows NT、OS/2、および UNIX オペレーティング・システムの場合:**  
Web サーバーを実行するロケール

**OS/400 オペレーティング・システムの場合:**

- V4R2 またはそれ以降のリリース: プロセスが実行されているユーザー・プロファイルで指定。
- V4R1 またそれ以前のリリース: QDECFMT システム値から取得。

数値計算には、以下の関数を利用することができます。

- 165ページの『DTW\_ADD』
- 167ページの『DTW\_DIVIDE』
- 169ページの『DTW\_DIVREM』
- 171ページの『DTW\_FORMAT』
- 175ページの『DTW\_INTDIV』
- 177ページの『DTW\_MULTIPLY』
- 179ページの『DTW\_POWER』
- 181ページの『DTW\_SUBTRACT』

# DTW\_ADD

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

## 目的

2 つの数を追加します。

## 構文

```
@DTW_ADD(number1, number2, precision, result)
@DTW_ADD(number1, number2, result)
@DTW_rADD(number1, number2, precision)
@DTW_rADD(number1, number2)
```

## 値

表 57. DTW\_ADD のパラメーター

データ型	パラメーター	用途	説明
浮動	<i>number1</i>	IN	数値を表す変数またはリテラル・ストリング。
浮動	<i>number2</i>	IN	数値を表す変数またはリテラル・ストリング。
整数	<i>precision</i>	IN	結果の精度を指定する正の整数を表す変数またはリテラル・ストリング。デフォルトは 9 です。
浮動	<i>result</i>	OUT	<i>number1</i> と <i>number2</i> の合計値が含まれている変数。

## 戻りコード

表 58. DTW\_ADD の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
4000	パラメーターに無効な整数値が含まれています。
4001	パラメーターに無効な値が含まれています。
4002	算術演算の結果に、サポートされる -999,999,999 から +999,999,999 の範囲外の指数が含まれています。

## 例

### 例 1:

```
@DTW_ADD(NUM1, NUM2, "2", result)
```

- 入力: NUM1 = "105", NUM2 = "3"
- 戻り: result = "1.1E+2"

### 例 2:

```
@DTW_rADD("12", NUM2,  
"5")
```

- 入力: NUM2 = "7.00"
- 戻り: "19.00"

## DTW\_DIVIDE

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

### 目的

ある数を他の数で除算します。

### 構文

@DTW\_DIVIDE(number1, number2, precision, result)

@DTW\_DIVIDE(number1, number2, result)

@DTW\_rDIVIDE(number1, number2, precision)

@DTW\_rDIVIDE(number1, number2)

### 値

表 59. DTW\_DIVIDE のパラメーター

データ型	パラメーター	用途	説明
浮動	<i>number1</i>	IN	除算する数を表す変数またはリテラル・ストリング。
浮動	<i>number2</i>	IN	数値を表す変数またはリテラル・ストリング。
整数	<i>precision</i>	IN	結果の精度を指定する正の整数を表す変数またはリテラル・ストリング。デフォルトは 9 です。
浮動	<i>result</i>	OUT	<i>number1</i> を <i>number2</i> で割った結果が含まれている変数。

### 戻りコード

表 60. DTW\_DIVIDE 戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
4000	パラメーターに無効な整数値が含まれています。
4001	パラメーターに無効な値が含まれています。
4002	算術演算の結果に、サポートされる -999,999,999 から +999,999,999 の範囲外の指数が含まれています。

## 例

### 例 1:

```
@DTW_DIVIDE("8.0", NUM2,  
result)
```

- 入力: NUM2 = "2"
- 戻り: result = "4"

### 例 2:

```
@DTW_rDIVIDE("1", NUM2,  
"5")
```

- 入力: "1", NUM2 = "3"
- 戻り: "0.33333"

### 例 3:

```
@DTW_rDIVIDE(NUM1, "2",  
"5")
```

- 入力: NUM1 = "5"
- 戻り: "2.5"

## DTW\_DIVREM

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

### 目的

ある数を他の数で除算し、剰余を戻します。

### 構文

@DTW\_DIVREM(number1, number2, precision, result)

@DTW\_DIVREM(number1, number2, result)

@DTW\_rDIVREM(number1, number2, precision)

@DTW\_rDIVREM(number1, number2)

### 値

表 61. DTW\_DIVREM のパラメーター

データ型	パラメーター	用途	説明
浮動	<i>number1</i>	IN	除算する数を表す変数またはリテラル・ストリング。
浮動	<i>number2</i>	IN	数値を表す変数またはリテラル・ストリング。
整数	<i>precision</i>	IN	結果の精度を指定する正の整数を表す変数またはリテラル・ストリング。デフォルトは 9 です。
浮動	<i>result</i>	OUT	<i>number1</i> を <i>number2</i> で割ったときの剰余が含まれている変数。

### 戻りコード

表 62. DTW\_DIVIDEREM の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
4000	パラメーターに無効な整数値が含まれています。
4001	パラメーターに無効な値が含まれています。
4002	算術演算の結果に、サポートされる -999,999,999 から +999,999,999 の範囲外の指数が含まれています。

## 使用上の注意

剰余が非ゼロであれば、その符号は、最初のパラメーターの符号と同じです。

### 例

#### 例 1:

```
@DTW_DIVREM(NUM1, NUM2,  
result)
```

- 入力: NUM1 = "2.1", NUM2 = "3"
- 戻り: result = "2.1"

#### 例 2:

```
@DTW_rDIVREM("10",  
NUM2)
```

- 入力: NUM2 = "0.3"
- 戻り: "0.1"

#### 例 3:

```
@DTW_rDIVREM("3.6",  
"1.3")
```

- 戻り: "1.0"

#### 例 4:

```
@DTW_rDIVREM("-10",  
"3")
```

- 戻り: "-1"



## DTW\_FORMAT

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

### 目的

数値の形式化をカスタマイズします。

### 構文

@DTW\_FORMAT(number, before, after, exp, expt, precision, result)

@DTW\_FORMAT(number, before, after, exp, expt, result)

@DTW\_FORMAT(number, before, after, exp, result)

@DTW\_FORMAT(number, before, after, result)

@DTW\_FORMAT(number, before, result)

@DTW\_FORMAT(number, result)

@DTW\_rFORMAT(number, before, after, exp, expt, precision)

@DTW\_rFORMAT(number, before, after, exp, expt)

@DTW\_rFORMAT(number, before, after, exp)

@DTW\_rFORMAT(number, before, after)

@DTW\_rFORMAT(number, before)

@DTW\_rFORMAT(number)

### 値

表 63. DTW\_FORMAT のパラメーター

データ型	パラメーター	用途	説明
浮動	<i>number</i>	IN	数値を表す変数またはリテラル・ストリング。
整数	<i>before</i>	IN	正の整数を表す変数またはリテラル・ストリング。これはオプション・パラメーターです。ヌル・ストリング ("" ) を入力して追加のパラメーターを指定する必要があります。
整数	<i>after</i>	IN	正の整数を表す変数またはリテラル・ストリング。これはオプション・パラメーターです。ヌル・ストリング ("" ) を入力して追加のパラメーターを指定する必要があります。
整数	<i>exp</i>	IN	正の整数を表す変数またはリテラル・ストリング。ヌル・ストリング ("" ) を入力して追加のパラメーターを指定する必要があります。
整数	<i>expt</i>	IN	正の整数を表す変数またはリテラル・ストリング。ヌル・ストリング ("" ) を入力して追加のパラメーターを指定する必要があります。

表 63. DTW\_FORMAT のパラメーター (続き)

データ型	パラメーター	用途	説明
整数	<i>precision</i>	IN	結果の精度を指定する正の整数を表す変数またはリテラル・ストリング。デフォルトは 9 です。
浮動	<i>result</i>	OUT	丸めと形式化が指定された数値が含まれている変数。

## 戻りコード

表 64. DTW\_FORMAT の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
4000	パラメーターに無効な整数値が含まれています。
4001	パラメーターに無効な値が含まれています。

## 使用上の注意

1. *number* パラメーターのみを指定すると、結果は、@DTW\_rADD(*number*, 『0』) を実行した場合と同じように形式化されます。
2. *before* および *after* パラメーターは、それぞれ *result* パラメーターの整数部分と小数部分に何文字を使用しているかを記述します。これらのパラメーターのいずれか、または両方を省略すると、該当部分に使用される文字の数は必要なだけの数になります。
3. *before* パラメーターが数値の整数部分 (および、負の数の符号) を含めるだけの大きさをもっていない場合は、エラーになります。 *before* パラメーターがその部分に必要な大きさを越えている場合は、*number* パラメーター値は、左側にブランクが埋め込まれます。 *after* パラメーターが *number* パラメーターの小数部分と同じサイズでない場合は、同じサイズになるように数値が丸められます (または、ゼロで拡張されます)。 0 を指定すると、数値は丸められて整数になります。
4. *expp* および *expt* パラメーターは結果の指数部分を制御します。 *expp* パラメーターは、指数部分の桁数を設定します。デフォルトでは、必要なだけの桁数を使用します (ゼロの場合もあります)。 *expt* パラメーターは、指数表記に使用するトリガー・ポイントを設定します。デフォルトは、精度パラメーターのデフォルトの値です。
5. *expp* が 0 であれば、指数は提供されず、数値は、必要なだけゼロが追加された単純な形式で表されます。 *expp* が指数を含めるだけの大きさをもっていない場合は、エラーになります。

6. 整数部分または小数部分に必要な桁数が、それぞれ *expt* または 2 倍の *expt* を超えている場合は、指数表記を使用します。 *expt* が 0 であれば、指数が 0 でない限り、指数表記が常に使用されます。 (*expp* が 0 であれば、*expt* の 0 値をオーバーライドします。) 非ゼロの *expp* が指定されているときに指数が 0 であれば、*expp*+2 ブランクが結果の指数部分として提供されます。指数が 0 で、*expp* が指定されていない場合は、単純な形式が使用されます。

## 例

### 例 1:

```
@DTW_FORMAT(NUM, BEFORE, result)
```

- 入力: NUM = "3", BEFORE = "4"
- 戻り: result = "3"

### 例 2:

```
@DTW_FORMAT("1.73", "4", "0", result)
```

- 戻り: result = "2"

### 例 3:

```
@DTW_FORMAT("1.73", "4", "3", result)
```

- 戻り: result = "1.730"

### 例 4:

```
@DTW_FORMAT(" - 12.73", "", "4", result)
```

- 戻り: result = "-12.7300"

### 例 5:

```
@DTW_FORMAT("12345.73", "", "", "2", "2", result)
```

- 戻り: result = "1.234573E+04"

### 例 6:

```
@DTW_FORMAT("1.234573", "", "3", "", "0", result)
```

- 戻り: result = "1.235"

### 例 7:

```
@DTW_rFORMAT(" - 12.73")
```

- 戻り: " - 12.73"

### 例 8:

```
@DTW_rFORMAT("0.000")
```

- 戻り: "0"

### 例 9:

```
@DTW_rFORMAT("12345.73", "", "", "3", "6")
```

- 戻り: "12345.73"

**例 10:**

```
@DTW_rFORMAT("1234567e5", "", "3", "0")
```

- 戻り: "123456700000.000"

**例 11:**

```
@DTW_rFORMAT("12345.73", "", "3", "", "0")
```

- 戻り: "1.235E+4"

# DTW\_INTDIV

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

## 目的

ある数を他の数で除算し、結果の整数部分を戻します。

## 構文

@DTW\_INTDIV(number1, number2, precision, result)

@DTW\_INTDIV(number1, number2, result)

@DTW\_rINTDIV(number1, number2, precision)

@DTW\_rINTDIV(number1, number2)

## 値

表 65. DTW\_INTDIV のパラメーター

データ型	パラメーター	用途	説明
浮動	<i>number1</i>	IN	除算する数を表す変数またはリテラル・ストリング。
浮動	<i>number2</i>	IN	数値を表す変数またはリテラル・ストリング。
整数	<i>precision</i>	IN	結果の精度を指定する正の整数を表す変数またはリテラル・ストリング。デフォルトは 9 です。
浮動	<i>result</i>	OUT	<i>number1</i> を <i>number2</i> で割った結果の整数部分が含まれている変数。

## 戻りコード

表 66. DTW\_INTDIV の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
4000	パラメーターに無効な整数値が含まれています。
4001	パラメーターに無効な値が含まれています。
4002	算術演算の結果に、サポートされる -999,999,999 から +999,999,999 の範囲外の指数が含まれています。

## 例

### 例 1:

```
@DTW_INTDIV(NUM1, NUM2,  
result)
```

- 入力: NUM1 = "10", NUM2 = "3"
- 戻り: result = "3"

### 例 2:

```
@DTW_rINTDIV("2",  
NUM2)
```

- 入力: NUM2 = "3"
- 戻り: "0"

## DTW\_MULTIPLY

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

### 目的

2 つの数を乗算します。

### 構文

@DTW\_MULTIPLY(number1, number2, precision, result)

@DTW\_MULTIPLY(number1, number2, result)

@DTW\_rMULTIPLY(number1, number2, precision)

@DTW\_rMULTIPLY(number1, number2)

### 値

表 67. DTW\_MULTIPLY のパラメーター

データ型	パラメーター	用途	説明
浮動	<i>number1</i>	IN	数値を表す変数またはリテラル・ストリング。
浮動	<i>number2</i>	IN	数値を表す変数またはリテラル・ストリング。
整数	<i>precision</i>	IN	結果の精度を指定する正の整数を表す変数またはリテラル・ストリング。デフォルトは 9 です。
浮動	<i>result</i>	OUT	<i>number1</i> と <i>number2</i> の積が含まれている変数。

### 戻りコード

表 68. DTW\_MULTIPLY の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
4000	パラメーターに無効な整数値が含まれています。
4001	パラメーターに無効な値が含まれています。
4002	算術演算の結果に、サポートされる -999,999,999 から +999,999,999 の範囲外の指数が含まれています。

## 例

### 例 1:

```
@DTW_MULTIPLY(NUM1, NUM2, result)
```

- 入力: NUM1 = "4", NUM2 = "5"
- 戻り: result = "20"

### 例 2:

```
@DTW_rMULTIPLY("0.9",  
NUM2)
```

- 入力: NUM2 = "0.8"
- 戻り: "0.72"



## DTW\_POWER

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

### 目的

整数をその累乗にします。

### 構文

@DTW\_POWER(number1, number2, precision, result)

@DTW\_POWER(number1, number2, result)

@DTW\_rPOWER(number1, number2, precision)

@DTW\_rPOWER(number1, number2)

### 値

表 69. DTW\_POWER のパラメーター

データ型	パラメーター	用途	説明
浮動	<i>number1</i>	IN	累乗する数を表す変数またはリテラル・ストリング。
浮動	<i>number2</i>	IN	数値を表す変数またはリテラル・ストリング。
整数	<i>precision</i>	IN	結果の精度を指定する正の整数を表す変数またはリテラル・ストリング。デフォルトは 9 です。
浮動	<i>result</i>	OUT	<i>number1</i> を <i>number2</i> 乗した結果が含まれている変数。

### 戻りコード

表 70. DTW\_POWER 戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
4000	パラメーターに無効な整数値が含まれています。
4001	パラメーターに無効な値が含まれています。
4002	算術演算の結果に、サポートされる -999,999,999 から +999,999,999 の範囲外の指数が含まれています。

## 例

### 例 1:

```
@DTW_POWER(NUM1, NUM2,  
result)
```

- 入力: NUM1 = "2", NUM2 = "-3"
- 戻り: result = "0.125"

### 例 2:

```
@DTW_rPOWER("1.7", NUM2, precision)
```

- 入力: NUM2 = "8", precision = "5"
- 戻り: "69.758"

## DTW\_SUBTRACT

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

### 目的

ある数を他の数から減算します。

### 構文

@DTW\_SUBTRACT(number1, number2, precision, result)

@DTW\_SUBTRACT(number1, number2, result)

@DTW\_rSUBTRACT(number1, number2, precision)

@DTW\_rSUBTRACT(number1, number2)

### 値

表 71. DTW\_SUBTRACT のパラメーター

データ型	パラメーター	用途	説明
浮動	<i>number1</i>	IN	減算の対象とする数を表す変数またはリテラル・ストリング。
浮動	<i>number2</i>	IN	数値を表す変数またはリテラル・ストリング。
整数	<i>precision</i>	IN	結果の精度を指定する正の整数を表す変数またはリテラル・ストリング。デフォルトは 9 です。
浮動	<i>result</i>	OUT	<i>number1</i> と <i>number2</i> の差が含まれている変数。

### 戻りコード

表 72. DTW\_SUBTRACT の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
4000	パラメーターに無効な整数値が含まれています。
4001	パラメーターに無効な値が含まれています。
4002	算術演算の結果に、サポートされる -999,999,999 から +999,999,999 の範囲外の指数が含まれています。

## 例

### 例 1:

```
@DTW_SUBTRACT(NUM1, NUM2, comp)
%IF(comp > "0")
<P>$(NUM1) is larger than $(NUM2).
%ENDIF
```

- 入力: NUM2 = "2.07"
- 戻り: "-0.77"

この例は、Net.Data のストリングである数値の比較方法を示しています。

### 例 2:

```
@DTW_SUBTRACT(NUM1, NUM2, result)
• 入力: NUM1 = "1.3, NUM2 = "1.07"
• 戻り: result = "0.23"
```

### 例 3:

```
@DTW_rSUBTRACT("1.3",
NUM2)
• 入力: NUM2 = "2.07"
• 戻り: "-0.77"
```

---

## ストリング関数

以下の関数は、Net.Data によってサポートされる標準のストリング関数です。

- 184ページの『DTW\_ASSIGN』
- 185ページの『DTW\_CHARTOHEX』
- 186ページの『DTW\_CONCAT』
- 188ページの『DTW\_DELSTR』
- 190ページの『DTW\_HEXTOCHAR』
- 192ページの『DTW\_INSERT』
- 194ページの『DTW\_LASTPOS』
- 196ページの『DTW\_LENGTH』
- 197ページの『DTW\_LOWERCASE』
- 199ページの『DTW\_POS』
- 201ページの『DTW\_REPLACE』
- 203ページの『DTW\_REVERSE』
- 204ページの『DTW\_STRIP』
- 206ページの『DTW\_SUBSTR』
- 208ページの『DTW\_TRANSLATE』
- 210ページの『DTW\_UPPERCASE』

### **MBCS サポート (OS/390、OS/2、Windows NT、および UNIX の場合):**

DTW\_MBMODE 構成値をもつワード関数とストリング関数に対して、複数バイト文字セット (MBCS) サポートを指定することができます。この値を Net.Data 初期設定ファイルに指定します。デフォルトでは、サポートがありません。DTW\_MBMODE 変数を Net.Data マクロに設定することにより、初期設定ファイルの値をオーバーライドすることができます。詳しくは、*Net.Data 管理およびプログラミングの手引き* の構成変数に関する節、および 122ページの『DTW\_MBMODE』を参照してください。

**MBCS サポート (OS/400 の場合):** DBCS サポートは自動的に提供されるため、この変数は必要ありません。

## DTW\_ASSIGN

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

### 目的

変数に値を割り当てます。

### 構文

```
@DTW_ASSIGN(stringOut, stringIn)
```

### 値

表 73. DTW\_ASSIGN のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringOut</i>	OUT	<i>stringIn</i> と同じリテラル・ストリングが含まれている変数。
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。

### 戻りコード

表 74. DTW\_ASSIGN の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 例

#### 例 1:

```
@DTW_ASSIGN(RC, "0")
```

- RC を "0" に設定します。

#### 例 2:

```
@DTW_ASSIGN(string1,  
string2)
```

- *string1* を *string2* の値に設定します。

## DTW\_CHARTOHEX

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

### 目的

文字列内の各文字を、2 つの 16 進文字に変換します。

### 構文

@DTW\_CHARTOHEX(stringIn, stringOut)

@DTW\_rCHARTOHEX(stringIn)

### 値

表 75. DTW\_CHARTOHEX のパラメーター

データ型	パラメーター	用途	説明
文字列	stringIn	IN	変換する変数またはリテラル・文字列。
文字列	stringOut	OUT	16 進形式で表した stringIn が含まれている変数。

### 戻りコード

表 76. DTW\_CHARTOHEX の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (文字列変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・文字列が渡されました。

### 使用上の注意

16 進文字はそれぞれ入力文字の 4 ビット部分を表します (文字は 8 ビットで表されます)。

### 例

例 1: EBCDIC オペレーティング・システム

@DTW\_rCHARTOHEX("12345")

- 戻り: "F1F2F3F4F5"

例 2: ASCII オペレーティング・システム

@DTW\_rCHARTOHEX("12345")

- 戻り: "C1C2C3C4C5"

## DTW\_CONCAT

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

### 目的

2 つのストリングを連結します。

### 構文

```
@DTW_CONCAT(stringIn1, stringIn2, stringOut)
```

```
@DTW_rCONCAT(stringIn1, stringIn2)
```

### 値

表 77. DTW\_CONCAT のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn1</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>stringIn2</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>stringOut</i>	OUT	ストリング ' <i>stringIn1stringIn2</i> ' が含まれている変数。ここで、 <i>string1</i> は <i>string2</i> と連結されています。

### 戻りコード

表 78. DTW\_CONCAT の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 例

#### 例 1:

```
@DTW_CONCAT("This", " is a test.", result)
```

- 戻り: result = "This is a test."

#### 例 2:

```
@DTW_CONCAT(string1, "1-2-3", result)
```

- 入力: string1 = "Testing "
- 戻り: result = "Testing 1-2-3"

#### 例 3:

```
@DTW_rCONCAT("This", " is a test.")
```



- 戻り: "This is a test."

## DTW\_DELSTR

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

### 目的

$n$  番目の文字から、 $length$  文字だけ、ストリングのサブストリングを削除します。

### 構文

@DTW\_DELSTR(stringIn, n, length, stringOut)

@DTW\_DELSTR(stringIn, n, stringOut)

@DTW\_rDELSTR(stringIn, n, length)

@DTW\_rDELSTR(stringIn, n)

### 値

表 79. DTW\_DELSTR のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。
整数	$n$	IN	削除するサブストリングが始まる文字の位置。 $n$ が <i>stringIn</i> の長さよりも大きい場合は、 <i>stringOut</i> が <i>stringIn</i> の値に設定されます。
整数	<i>length</i>	IN	削除するサブストリングの長さ。デフォルトでは、 <i>stringIn</i> の終わりまでのすべての文字が削除されます。
ストリング	<i>stringOut</i>	OUT	<i>stringIn</i> の変更形式が含まれている変数。

### 戻りコード

表 80. DTW\_DELSTR の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

## 例

### 例 1:

```
@DTW_DELSTR("abcde", "3", "2", result)
```

- 戻り: result = "abe"

### 例 2:

```
@DTW_rDELSTR("abcde", "4", "1")
```

- 戻り: "abce"

## DTW\_HEXTOCHAR

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

### 目的

ストリング内の文字をそれぞれ文字値に変換します。

### 構文

@DTW\_HEXTOCHAR(stringIn, stringOut)

@DTW\_rHEXTOCHAR(stringIn)

### 値

表 81. DTW\_HEXTOCHAR Parameters

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変換する変数またはリテラル・ストリング。
ストリング	<i>stringOut</i>	OUT	文字形式で表した <i>stringIn</i> が含まれている変数。

### 戻りコード

表 82. DTW\_HEXTOCHAR 戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

### 使用上の注意

入力ストリングの 16 進文字はそれぞれ結果文字ストリングの 4 ビット部分を表します（文字は 8 ビットで表されます）。入力ストリングは、偶数の 16 進文字を含む必要があります、これには 0-9、A-F、および a-f を含むことができます。

### 例

例 1: EBCDIC オペレーティング・システム

```
@DTW_rHEXTOCHAR("F0F1F2")
```

• 戻り: "123"

例 2: ASCII オペレーティング・システム

```
@DTW_rHEXTOCHAR("303132")
```

|  
|

- 戻り: "123"

## DTW\_INSERT

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

### 目的

あるストリングを別のストリングの *n* 番目の文字の後から挿入します。

### 構文

@DTW\_INSERT(stringIn1, stringIn2, n, length, pad, stringOut)

@DTW\_INSERT(stringIn1, stringIn2, n, length, stringOut)

@DTW\_INSERT(stringIn1, stringIn2, n, stringOut)

@DTW\_INSERT(stringIn1, stringIn2, stringOut)

@DTW\_rINSERT(stringIn1, stringIn2, n, length, pad)

@DTW\_rINSERT(stringIn1, stringIn2, n, length)

@DTW\_rINSERT(stringIn1, stringIn2, n)

@DTW\_rINSERT(stringIn1, stringIn2)

### 値

表 83. DTW\_INSERT のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn1</i>	IN	<i>stringIn2</i> に挿入する変数またはリテラル・ストリング。
ストリング	<i>stringIn2</i>	IN	変数またはリテラル・ストリング。
整数	<i>n</i>	IN	<i>stringIn1</i> の挿入を開始する、 <i>stringIn2</i> の文字位置。 <i>n</i> が <i>stringIn2</i> の長さよりも大きい場合は、それが十分な文字数になるまで、埋め込み文字 <i>pad</i> が埋め込まれます。デフォルトでは、 <i>stringIn2</i> の先頭から挿入されます。
整数	<i>length</i>	IN	挿入する <i>stringIn1</i> の文字数。このパラメーターが <i>stringIn1</i> の長さよりも大きい場合は、埋め込み文字 <i>pad</i> がストリングに埋め込まれます。デフォルトは <i>stringIn1</i> の長さです。
整数	<i>pad</i>	IN	<i>n</i> や <i>length</i> の場合と同じ埋め込み文字。デフォルトの埋め込み文字はブランクです。
ストリング	<i>stringOut</i>	OUT	<i>stringIn1</i> の一部またはすべてを挿入して変更された <i>stringIn2</i> が含まれている変数。

### 戻りコード

表 84. DTW\_INSERT の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。

表 84. DTW\_INSERT の戻りコード (続き)

戻りコード	説明
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を 超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするため に必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須である パラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

## 例

### 例 1:

```
@DTW_INSERT("123", "abc", result)
```

- 戻り: result = "123abc"

### 例 2:

```
@DTW_INSERT("123", "abc", "5", result)
```

- 戻り: result = "abc 123 "

### 例 3:

```
@DTW_INSERT("123", "abc", "5", "6", result)
```

- 戻り: result = "abc 123 "

### 例 4:

```
@DTW_INSERT("123", "abc", "5", "6", "/", result)
```

- 戻り: result = "abc//123///"

### 例 5:

```
@DTW_rINSERT("123", "abc", "5", "6", "+")
```

- 戻り: "abc++123++"

## DTW\_LASTPOS

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

### 目的

あるストリングが別のストリングに最後に出現した位置を、*n* 番目の文字から逆方向に（つまり、右から左へ）スキャンして戻します。

### 構文

@DTW\_LASTPOS(stringIn1, stringIn2, n, position)

@DTW\_LASTPOS(stringIn1, stringIn2, position)

@DTW\_rLASTPOS(stringIn1, stringIn2, n)

@DTW\_rLASTPOS(stringIn1, stringIn2)

### 値

表 85. DTW\_LASTPOS のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn1</i>	IN	<i>stringIn2</i> 内の検索対象変数またはリテラル・ストリング。
ストリング	<i>stringIn2</i>	IN	変数またはリテラル・ストリング。
整数	<i>n</i>	IN	<i>stringIn1</i> の検索を開始する <i>stringIn2</i> 内の文字位置。デフォルトでは、最後の文字から検索が開始され、逆方法に（つまり、右から左へ）スキャンされます。
整数	<i>position</i>	OUT	<i>stringIn1</i> が <i>stringIn2</i> 内で最後に出現した位置。出現が見付からないと、0 が戻されます。

### 戻りコード

表 86. DTW\_LASTPOS の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。



## 例

### 例 1:

```
@DTW_LASTPOS(" ", "abc def ghi", result)
```

- 戻り: result = "8"

### 例 2:

```
@DTW_LASTPOS(" ", "abc def ghi", "10", result)
```

- 戻り: result = "8"

### 例 3:

```
@DTW_rLASTPOS(" ", "abc def ghi", "7")
```

- 戻り: "4"

## DTW\_LENGTH

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

### 目的

ストリングの長さを戻します。

### 構文

```
@DTW_LENGTH(stringIn, length)
```

```
@DTW_rLENGTH(stringIn)
```

### 値

表 87. DTW\_LENGTH のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。
整数	<i>length</i>	OUT	<i>stringIn</i> 内の文字の数が含まれているシンボル。

### 戻りコード

表 88. DTW\_LENGTH の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 例

#### 例 1:

```
@DTW_LENGTH("abcdefgh",  
result)
```

- 戻り: result = "8"

#### 例 2:

```
@DTW_rLENGTH("")
```

- 戻り: "0"

## DTW\_LOWERCASE

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

### 目的

文字列をすべて小文字で戻します。

### 構文

@DTW\_LOWERCASE(stringIn, stringOut)

@DTW\_rLOWERCASE(stringIn)

@DTW\_mLOWERCASE(stringMult1, stringMult2, ..., stringMultn)

### 値

表 89. DTW\_LOWERCASE のパラメーター

データ型	パラメーター	用途	説明
文字列	<i>stringIn</i>	IN	任意のケースの文字をもつ変数またはリテラル・文字列。
文字列	<i>stringOut</i>	OUT	すべての文字が小文字になった <i>stringIn</i> が含まれている変数。
文字列	<i>stringMult</i>	INOUT	<ul style="list-style-type: none"><li>入力の場合: 文字列が含まれている変数。</li><li>出力の場合: 小文字に変換された入力文字列が含まれている変数。</li></ul>

### 戻りコード

表 90. DTW\_LOWERCASE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (文字列変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・文字列が渡されました。

### 例

#### 例 1:

```
@DTW_LOWERCASE("This", stringOut)
```

- 戻り: stringOut = "this"

#### 例 2:

```
@DTW_rLOWERCASE(string1)
```

- 入力: string1 = "Hello"

- 戻り: "hello"

**例 3:**

@DTW\_mLOWERCASE(string1, string2, string3)

- 入力: string1 = "THIS", string2 = "IS", string3 = "LOWERCASE"
- 戻り: string1 = "this", string2 = "is", string3 = "lowercase"

## DTW\_POS

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

### 目的

あるストリングが別のストリングに初めて出現した位置を、下方検索パターンを使用して戻します。

### 構文

@DTW\_POS(stringIn1, stringIn2, n, nOut)

@DTW\_POS(stringIn1, stringIn2, nOut)

@DTW\_rPOS(stringIn1, stringIn2, n)

@DTW\_rPOS(stringIn1, stringIn2)

### 値

表 91. DTW\_POS のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn1</i>	IN	検索対象の変数またはリテラル・ストリング。
ストリング	<i>stringIn2</i>	IN	検索対象の変数またはリテラル・ストリング。
整数	<i>n</i>	IN	検索を開始する <i>stringIn2</i> 内の文字位置。デフォルトの値では、 <i>stringIn2</i> の先頭文字から検索が開始されます。
整数	<i>nOut</i>	OUT	<i>stringIn1</i> が <i>stringIn2</i> に初めて出現した位置が含まれている変数。出現が見付からないと、0 が戻されます。

### 戻りコード

表 92. DTW\_POS の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

## 例

### 例 1:

```
@DTW_POS("day", "Saturday", result)
```

- 戻り: result = "6"

### 例 2:

```
@DTW_POS("a", "Saturday", "3", result)
```

- 戻り: result = "7"

### 例 3:

```
@DTW_rPOS(" ", "abc def ghi", "5")
```

- 戻り: "8"

## DTW\_REPLACE

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

### 目的

ストリング内の文字を置換します。

### 構文

@DTW\_REPLACE(stringIn, stringFrom, stringTo, n, option, stringOut)

@DTW\_REPLACE(stringIn, stringFrom, stringTo, n, stringOut)

@DTW\_REPLACE(stringIn, stringFrom, stringTo, stringOut)

@DTW\_rREPLACE(stringIn, stringFrom, stringTo, n, option)

@DTW\_rREPLACE(stringIn, stringFrom, stringTo, n)

@DTW\_rREPLACE(stringIn, stringFrom, stringTo)

### 値

表 93. DTW\_REPLACE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	検索する変数またはリテラル・ストリング。
ストリング	<i>stringFrom</i>	IN	置換する変数またはリテラル・ストリング。
ストリング	<i>stringTo</i>	IN	<i>stringFrom</i> のオカレンスを置換する変数またはリテラル・ストリング。
整数	<i>n</i>	IN	検索を開始する文字の位置。
ストリング	<i>option</i>	IN	すべてのオカレンス、または最初のオカレンスのみを置換するかを指定します。以下の値のいずれかとします。  <b>A または a</b> すべてのオカレンスを置換します。デフォルトは A です。  <b>F または f</b> 最初のオカレンスのみを置換します。
ストリング	<i>stringOut</i>	OUT	<i>stringFrom</i> のオカレンスを <i>stringTo</i> で置換した <i>stringIn</i> が含まれている変数。

### 戻りコード

表 94. DTW\_REPLACE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。

表 94. DTW\_REPLACE の戻りコード (続き)

戻りコード	説明
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を 超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするため に必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須である パラメーターにリテラル・ストリングが渡されました。

## 例

### 例 1:

```
@DTW_rREPLACE("ABCABCABC", "AB", "1234")
```

- 戻り: "1234C1234C1234C"



# DTW\_REVERSE

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

## 目的

最後の文字を最初に、最後から 2 番目の文字を 2 番目に反転し、ストリング全体を反転します。

## 構文

```
@DTW_REVERSE(stringIn, stringOut)
@DTW_rREVERSE(stringIn)
```

## 値

表 95. DTW\_REVERSE のパラメーター

データ型	パラメーター	用途	説明
ストリング	stringIn	IN	反転する変数またはリテラル・ストリング。
ストリング	stringOut	OUT	stringIn の反転形式が含まれている変数。

## 戻りコード

表 96. DTW\_REVERSE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

## 例

例 1:

```
@DTW_REVERSE("This is it.", result)
```

- 戻り: result = ".ti si sihT"

例 2:

```
@DTW_rREVERSE(string1)
```

- 入力: string1 = "reversed"
- 戻り: "desrever"

## DTW\_STRIP

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

### 目的

先行ブランク、後書きブランク、またはその両方をストリングから除去します。

### 構文

@DTW\_STRIP(stringIn, option, stringOut)

@DTW\_STRIP(stringIn, stringOut)

@DTW\_rSTRIP(stringIn, option)

@DTW\_rSTRIP(stringIn)

### 値

表 97. DTW\_STRIP のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>option</i>	IN	<i>stringIn</i> からどのブランクを除去するかを指定します。デフォルトは B です。  B または b - 先行ブランクと後書きブランクを除去します。  L または l - 先行ブランクのみを除去します。  T または t - 後書きブランクのみを除去します。
ストリング	<i>stringOut</i>	OUT	オプションで指定されたとおりにブランクが除去された <i>stringIn</i> が含まれている変数。

### 戻りコード

表 98. DTW\_STRIP の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

## 例

### 例 1:

```
@DTW_STRIP(" day ",  
result)
```

- 戻り: result = " day"

### 例 2:

```
@DTW_STRIP(" day ", "T", result)
```

- 戻り: result = " day"

### 例 3:

```
@DTW_rSTRIP(" a day ",  
"L")
```

- 戻り: "a day "

## DTW\_SUBSTR

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

### 目的

オプションの埋め込み文字をもつ、ストリングのサブストリングを戻します。

### 構文

@DTW\_SUBSTR(stringIn, n, length, pad, stringOut)

@DTW\_SUBSTR(stringIn, n, length, stringOut)

@DTW\_SUBSTR(stringIn, n, stringOut)

@DTW\_rSUBSTR(stringIn, n, length, pad)

@DTW\_rSUBSTR(stringIn, n, length)

@DTW\_rSUBSTR(stringIn, n)

### 値

表 99. DTW\_SUBSTR のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	検索対象の変数またはリテラル・ストリング。
整数	<i>n</i>	IN	サブストリングの先頭文字位置。デフォルトでは、 <i>stringIn</i> の先頭から開始されます。
整数	<i>length</i>	IN	サブストリングの文字数。デフォルトは、ストリングの残りの文字です。
ストリング	<i>pad</i>	IN	<i>n</i> が <i>stringIn</i> の長さよりも大きい場合、または <i>length</i> が <i>stringIn</i> よりも長い場合に使用する埋め込み文字。デフォルトはブランクです。
ストリング	<i>stringOut</i>	OUT	<i>stringIn</i> のサブストリングが含まれている変数。

### 戻りコード

表 100. DTW\_SUBSTR の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

表 100. DTW\_SUBSTR の戻りコード (続き)

戻りコード	説明
1007	パラメーターに無効な値が含まれています。

## 例

### 例 1:

```
@DTW_SUBSTR("abc", "2",
result)
```

- 戻り: result = "bc "

### 例 2:

```
@DTW_SUBSTR("abc", "2", "4", result)
```

- 戻り: result = "bc "

### 例 3:

```
@DTW_SUBSTR("abc", "2", "4", ".", result )
```

- 戻り: result = "bc.."

### 例 4:

```
@DTW_rSUBSTR("abc", "2", "6", ".")
```

- 戻り: "bc...."

## DTW\_TRANSLATE

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

### 目的

それぞれの文字を他の文字に変換したストリング、または未変更のストリングを戻します。

### 構文

@DTW\_TRANSLATE(stringIn, tableO, tableI, default, stringOut)

@DTW\_TRANSLATE(stringIn, tableO, tableI, stringOut)

@DTW\_TRANSLATE(stringIn, tableO, stringOut)

@DTW\_TRANSLATE(stringIn, stringOut)

@DTW\_rTRANSLATE(stringIn, tableO, tableI, default)

@DTW\_rTRANSLATE(stringIn, tableO, tableI)

@DTW\_rTRANSLATE(stringIn, tableO)

@DTW\_rTRANSLATE(stringIn)

### 値

表 101. DTW\_TRANSLATE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>tableO</i>	IN	変換表として使用する変数またはリテラル・ストリング。 <i>tableI</i> または <i>default</i> を指定するには、ヌル ("" ) を使用します。それ以外の場合は、このパラメーターはオプションです。
ストリング	<i>tableI</i>	IN	<i>stringIn</i> 内の検索対象変数またはリテラル・ストリング。 <i>default</i> を指定するには、ヌル ("" ) を使用します。それ以外の場合は、このパラメーターはオプションです。
ストリング	<i>default</i>	IN	使用するデフォルト文字。デフォルトはブランクです。
ストリング	<i>stringOut</i>	OUT	<i>stringIn</i> の変数結果が含まれている変数。

### 戻りコード

表 102. DTW\_TRANSLATE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。

表 102. DTW\_TRANSLATE の戻りコード (続き)

戻りコード	説明
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

## 使用上の注意

1. *tableI*、*tableO*、および *default* 文字がパラメーター・リストに入っていない場合、*stringIn* パラメーターが大文字に変換されます。
2. *tableI* と *tableO* がリストに入っている場合、*tableI* 内の入力ストリングの各文字が検索され、*tableO* 内の対応する文字に変換されます。*tableI* 内の文字が *tableO* 内の文字と対応しない場合は、*default* 文字が代りに使用されます。

## 例

### 例 1:

```
@DTW_TRANSLATE("abbc", result)
```

- 戻り: result = "ABBC"

### 例 2:

```
@DTW_TRANSLATE("abbc", "R", "bc", result)
```

- 戻り: result = "aRR "

### 例 3:

```
@DTW_rTRANSLATE("abcdef", "12", "abcd", ".")
```

- 戻り: "12..ef"

### 例 4:

```
@DTW_rTRANSLATE("abbc", "", "", "")
```

- 戻り: "abbc"

## DTW\_UPPERCASE

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

### 目的

文字列を大文字で戻します。

### 構文

@DTW\_UPPERCASE(stringIn, stringOut)

@DTW\_rUPPERCASE(stringIn)

@DTW\_mUPPERCASE(stringMult1, stringMult2, ..., stringMultn)

### 値

表 103. DTW\_UPPERCASE のパラメーター

データ型	パラメーター	用途	説明
文字列	<i>stringIn</i>	IN	任意のケースの文字をもつ変数またはリテラル・文字列。
文字列	<i>stringOut</i>	OUT	すべての文字が大文字になった <i>stringIn</i> が含まれている変数。
文字列	<i>stringMult</i>	INOUT	<ul style="list-style-type: none"><li>入力の場合: 文字列が含まれている変数。</li><li>出力の場合: 大文字に変換された入力文字列が含まれている変数。</li></ul>

### 戻りコード

表 104. DTW\_UPPERCASE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (文字列変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・文字列が渡されました。

### 例

#### 例 1:

```
@DTW_UPPERCASE("Test", result)
```

- 戻り: result = "TEST"

#### 例 2:

```
@DTW_rUPPERCASE(string1)
```

- 入力: string1 = "Web pages"



- 戻り: "WEB PAGES"

**例 3:**

@DTW\_mUPPERCASE(string1, string2, string3)

- 入力: string1 = "This", string2 = "is", string3 = "uppercase"
- 戻り: string1 = "THIS", string2 = "IS", string3 = "UPPERCASE"

---

## ワード関数

これらの関数は、ワードまたはワード・セットを変更することによりストリング関数を補足します。Net.Data は、ワードを、スペースで区切られたストリング、または両サイドにスペースがあるストリングとして解釈します。次に、いくつか例を示します。

ストリング値	ワード数
one two three	3
one , two , three	5
Part 2: Internet Sales Grow	5

**MBCS サポート (OS/390、OS/2、Windows NT、および UNIX の場合):** DTW\_MBMODE 構成値をもつワード関数とストリング関数に対して、複数バイト文字セット (MBCS) サポートを指定することができます。この値を Net.Data 初期設定ファイルに指定します。デフォルトでは、サポートがありません。DTW\_MBMODE 変数を Net.Data マクロに設定することにより、初期設定ファイルの値をオーバーライドすることができます。詳しくは、*Net.Data 管理およびプログラミングの手引き* の構成変数に関する節、および 122ページの『DTW\_MBMODE』を参照してください。

**MBCS サポート (OS/400 の場合):** DBCS サポートは自動的に提供されるため、この変数は必要ありません。

以下の関数は、Net.Data でサポートされるワード関数です。

- 213ページの『DTW\_DELWORD』
- 215ページの『DTW\_SUBWORD』
- 217ページの『DTW\_WORD』
- 219ページの『DTW\_WORDINDEX』
- 221ページの『DTW\_WORDLENGTH』
- 222ページの『DTW\_WORDPOS』
- 224ページの『DTW\_WORDS』

## DTW\_DELWORD

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

### 目的

*length* で指定されたワード数だけ、ワード *n* からストリング内のワードが削除されます。

### 構文

@DTW\_DELWORD(stringIn, n, length, stringOut)

@DTW\_DELWORD(stringIn, n, stringOut)

@DTW\_rDELWORD(stringIn, n, length)

@DTW\_rDELWORD(stringIn, n)

### 値

表 105. DTW\_DELWORD のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。
整数	<i>n</i>	IN	最初に削除されるワードのワード位置。
整数	<i>length</i>	IN	削除するワードの数。デフォルトでは、 <i>n</i> から <i>stringIn</i> の終わりまでのすべてのワードが削除されます。オプション・パラメーターです。
ストリング	<i>stringOut</i>	OUT	<i>stringIn</i> の変更形式が含まれている変数。

### 戻りコード

表 106. DTW\_DELWORD の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

### 例

#### 例 1

```
@DTW_DELWORD("Now is the time", "5", result)
```

- 戻り: result = "Now is the time"

**例 2:**

```
@DTW_DELWORD("Now is the time", "2", result)
```

- 戻り: result = "Now"

**例 3:**

```
@DTW_DELWORD("Now is the time", "2", "2", result)
```

- 戻り: result = "Now time"

**例 4:**

```
@DTW_rDELWORD("Now is the time.", "3")
```

- 戻り: "Now is"

# DTW\_SUBWORD

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

## 目的

*length* で指定されたワード数だけ *n* から開始するストリングのサブストリングを戻します。

## 構文

```
@DTW_SUBWORD(stringIn, n, length, stringOut)
@DTW_SUBWORD(stringIn, n, stringOut)
@DTW_rSUBWORD(stringIn, n, length)
@DTW_rSUBWORD(stringIn, n)
```

## 値

表 107. DTW\_SUBWORD のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。
整数	<i>n</i>	IN	サブストリングの先頭ワードのワード位置。この値が <i>stringIn</i> のワード数よりも大きい場合は、ヌルが戻されます。
整数	<i>length</i>	IN	サブストリングのワード数。この値が <i>n</i> から <i>stringIn</i> の終わりまでのワード数よりも大きい場合は、 <i>stringIn</i> の終わりまでのすべてのワードが戻されます。デフォルトでは、 <i>n</i> から <i>stringIn</i> の終わりまでのすべてのワードが戻されます。
ストリング	<i>stringOut</i>	OUT	<i>n</i> と <i>length</i> で指定された <i>stringIn</i> のサブストリングが含まれている変数。

## 戻りコード

表 108. DTW\_SUBWORD の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

## 例

### 例 1:

```
@DTW_SUBWORD("Now is the time", "5", result)
```

- 戻り: result = ""

### 例 2:

```
@DTW_SUBWORD("Now is the time", "2", result)
```

- 戻り: result = "is the time"

### 例 3:

```
@DTW_SUBWORD("Now is the time", "2", "2", result)
```

- 戻り: result = "is the"

### 例 4:

```
@DTW_rSUBWORD("Now is the time", "3")
```

- 戻り: "the time"

## DTW\_WORD

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

### 目的

文字列内の  $n$  番目のワードを戻します。

### 構文

```
@DTW_WORD(stringIn, n, stringOut)
```

```
@DTW_rWORD(stringIn, n)
```

### 値

表 109. DTW\_WORD のパラメーター

データ型	パラメーター	用途	説明
文字列	<i>stringIn</i>	IN	変数またはリテラル・文字列。
整数	<i>n</i>	IN	戻されるワードのワード位置。この値が <i>stringIn</i> のワード数よりも大きい場合は、ヌルが戻されます。
文字列	<i>stringOut</i>	OUT	ワード位置 $n$ のワードが含まれている変数。

### 戻りコード

表 110. DTW\_WORD の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (文字列変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・文字列が渡されました。
1007	パラメーターに無効な値が含まれています。

### 例

#### 例 1:

```
@DTW_WORD("Now is the time", "3", result)
```

- 戻り: result = "the"

#### 例 2:

```
@DTW_WORD("Now is the time", "5", result)
```

- 戻り: result = ""

**例 3:**

```
@DTW_rWORD("Now is the time", "4")
```

- 戻り: "time"



## DTW\_WORDINDEX

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

### 目的

文字列の  $n$  番目のワードの先頭文字の文字位置を戻します。

### 構文

```
@DTW_WORDINDEX(stringIn, n, stringOut)
```

```
@DTW_rWORDINDEX(stringIn, n)
```

### 値

表 111. DTW\_WORDINDEX のパラメーター

データ型	パラメーター	用途	説明
文字列	<i>stringIn</i>	IN	変数またはリテラル・文字列。
整数	<i>n</i>	IN	索引に入れるワードのワード位置。この値が入力文字列のワードの数よりも大きい場合は、0 が戻されます。
文字列	<i>stringOut</i>	OUT	<i>stringIn</i> の $n$ 番目のワードの文字位置が含まれている変数。

### 戻りコード

表 112. DTW\_WORDINDEX の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (文字列変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・文字列が渡されました。
1007	パラメーターに無効な値が含まれています。

### 例

#### 例 1:

```
@DTW_WORDINDEX("Now is the time", "3", result)
```

- 戻り: result = "8"

#### 例 2:

```
@DTW_WORDINDEX("Now is the time", "6", result)
```

- 戻り: result = "0"

**例 3:**

```
@DTW_rWORDINDEX("Now is the time", "2")
```

- 戻り: "5"

## DTW\_WORDLENGTH

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

### 目的

文字列の  $n$  番目のワードの長さを返します。

### 構文

@DTW\_WORDLENGTH(stringIn, n, stringOut)

@DTW\_rWORDLENGTH(stringIn, n)

### 値

表 113. DTW\_WORDLENGTH のパラメーター

データ型	パラメーター	用途	説明
文字列	<i>stringIn</i>	IN	変数またはリテラル・文字列。
整数	<i>n</i>	IN	長さを知りたいワードのワード位置。この値が入力文字列のワードの数よりも大きい場合は、0 が返されます。
文字列	<i>stringOut</i>	OUT	<i>stringIn</i> 内の $n$ 番目のワードの長さが含まれている変数。

### 戻りコード

表 114. DTW\_WORDLENGTH の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (文字列変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・文字列が渡されました。
1007	パラメーターに無効な値が含まれています。

### 例

#### 例 1:

```
@DTW_WORDLENGTH("Now is the time", "1", result)
```

- 戻り: result = "3"

#### 例 2:

```
@DTW_rWORDLENGTH("Now is the time", "6")
```

- 戻り: "0"

## DTW\_WORDPOS

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

### 目的

あるストリングが別のストリングに初めて出現したときのワード番号を戻します。

### 構文

@DTW\_WORDPOS(stringIn1, stringIn2, n, stringOut)

@DTW\_WORDPOS(stringIn1, stringIn2, stringOut)

@DTW\_rWORDPOS(stringIn1, stringIn2, n)

@DTW\_rWORDPOS(stringIn1, stringIn2)

### 値

表 115. DTW\_WORDPOS のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn1</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>stringIn2</i>	IN	検索対象の変数またはリテラル・ストリング。
整数	<i>n</i>	IN	検索を開始する <i>stringIn2</i> 内のワード位置。 この値が <i>stringIn2</i> 内のワードの数よりも大きい場合は、0 が戻されます。デフォルトでは、 <i>stringIn2</i> の先頭から検索されます。
ストリング	<i>stringOut</i>	OUT	<i>stringIn2</i> 内の <i>stringIn1</i> のワード位置。

### 戻りコード

表 116. DTW\_WORDPOS の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。

### 使用上の注意

複数のブランクは、比較のために単一のブランクとして扱われます。

## 例

### 例 1:

```
@DTW_WORDPOS("the", "Now is the time", result)
```

- 戻り: result = "3"

### 例 2:

```
@DTW_WORDPOS("The", "Now is the time", result)
```

- 戻り: result = "0"

### 例 3:

```
@DTW_WORDPOS("The", "Now is the time", "5", result)
```

- 戻り: result = "0"

### 例 4:

```
@DTW_WORDPOS("is the", "Now is the time", result)
```

- 戻り: result = "2"

### 例 5:

```
@DTW_rWORDPOS("be", "To be or not to be", "3")
```

- 戻り: "6"

## DTW\_WORDS

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

### 目的

ストリングのワード数を戻します。

### 構文

```
@DTW_WORDS(stringIn, stringOut)
```

```
@DTW_rWORDS(stringIn)
```

### 値

表 117. DTW\_WORDS のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>stringIn</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>stringOut</i>	OUT	<i>stringIn</i> 内のワードの数が含まれている変数。

### 戻りコード

表 118. DTW\_WORDS の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 例

#### 例 1:

```
@DTW_WORDS("Now is the time", result)
```

- 戻り:  
result = "4"

#### 例 2:

```
@DTW_rWORDS(" ")
```

- 戻り: "0"

---

## 表関数

以下の関数は、Net.Data 表の処理を単純化し、ユーザーが REXX、C、または Perl を使用して独自の関数を作成する場合と比較して、より効率的な手法を提供します。

- 226ページの『DTW\_TB\_APPENDROW』
- 228ページの『DTW\_TB\_COLS』
- 230ページの『DTW\_TB\_DELETEROW』
- 229ページの『DTW\_TB\_DELETECOL』
- 232ページの『DTW\_TB\_DLIST』
- 235ページの『DTW\_TB\_DUMP』
- 236ページの『DTW\_TB\_DUMPV』
- 238ページの『DTW\_TB\_GETN』
- 240ページの『DTW\_TB\_GETV』
- 242ページの『DTW\_TB\_HTMLENCODING』
- 244ページの『DTW\_TB\_INPUT\_CHECKBOX』
- 246ページの『DTW\_TB\_INPUT\_RADIO』
- 248ページの『DTW\_TB\_INPUT\_TEXT』
- 250ページの『DTW\_TB\_INSERTCOL』
- 251ページの『DTW\_TB\_INSERTROW』
- 253ページの『DTW\_TB\_LIST』
- 255ページの『DTW\_TB\_QUERYCOLNONJ』
- 257ページの『DTW\_TB\_ROWS』
- 258ページの『DTW\_TB\_SELECT』
- 260ページの『DTW\_TB\_SETCOLS』
- 261ページの『DTW\_TB\_SETN』
- 263ページの『DTW\_TB\_SETV』
- 265ページの『DTW\_TB\_TABLE』
- 267ページの『DTW\_TB\_TEXTAREA』

## DTW\_TB\_APPENDROW

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

### 目的

Net.Data 表の終わりに 1 つまたは複数の行を追加します。

### 構文

@DTW\_TB\_APPENDROW(table, rows)

### 値

表 119. DTW\_TB\_APPENDROW のパラメーター

データ型	パラメーター	用途	説明
表	table	INOUT	行が追加されるマクロ表変数。
整数	rows	IN	table に付加する行数。

### 戻りコード

表 120. DTW\_TB\_APPENDROW の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。
1010	データは表がいっぱいになるまで書き込まれ、余分なデータは廃棄されます。

### 使用上の注意

1. DTW\_TB\_APPENDROW() を呼び出す前に、表内の列番号を設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使うか、あるいは、その表を設定する言語環境に渡すことによって、列番号を設定することができます。
2. 表に行を追加した後 DTW\_TB\_SETV() 関数で新しい行に値を割り当てるか、表を言語環境に渡して処理を行います。
3. 表で利用できる合計行数が制限されている場合、行数を追加することによってその限界を超えると、呼び出し元にエラーが戻されます。



## 例

例 1: 表に 10 行を追加します。

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_APPENDROW(myTable, "10")
```

## DTW\_TB\_COLS

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

### 目的

Net.Data 表内の列数を戻します。

### 構文

```
@DTW_TB_COLS(table, cols)
```

```
@DTW_TB_rCOLS(table)
```

### 値

表 121. DTW\_TB\_COLS のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	列の番号が戻されるマクロ表変数。
整数	<i>cols</i>	OUT	<i>table</i> の列番号が含まれている変数。

### 戻りコード

表 122. DTW\_TB\_COLS の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 例

**例 1:** 列の番号を検索し、その値を *cols* に割り当てます。

```
%DEFINE myTable = %TABLE
%DEFINE cols = ""
...
@FillTable(myTable)
...
@DTW_TB_COLS(myTable, cols)
```

**例 2:** 表内の列の現行番号の値を検索し表示します。

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<P>My table contains @DTW_TB_rCOLS(myTable) columns.
```

# DTW\_TB\_DELETECOL

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

## 目的

Net.data 表の 1 つまたは複数の列を削除します。

## 構文

@DTW\_TB\_DELETECOL(table, after\_col, cols)

## 値

表 123. DTW\_TB\_DELETECOL のパラメーター

データ型	パラメーター	用途	説明
表	table	INOUT	列を削除するマクロ表変数。
整数	after_col	IN	続く列を削除する列の番号。最初の列を削除するには、0 を指定します。
整数	cols	IN	table から削除する列の数。

## 戻りコード

表 124. DTW\_TB\_DELETECOL の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

## 例

例 1: 表の 3 番目および 4 番目の列を削除します。

```
%DEFINE myTable = %TABLE
@DTW_TB_DELETECOL(myTable, "3", "2")
```

例 2: 表の最初の列を削除します。

```
%DEFINE myTable = %TABLE
@DTW_TB_DELETECOL(myTable, "0", "1")
```

## DTW\_TB\_DELETEROW

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

### 目的

Net.Data 表の 1 つまたは複数の行を削除します。

### 構文

```
@DTW_TB_DELETEROW(table, start_row, rows)
```

### 値

表 125. DTW\_TB\_DELETEROW のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	INOUT	行が削除されるマクロ表変数。
整数	<i>start_row</i>	IN	<i>table</i> 内の削除する最初の行の行番号。
整数	<i>rows</i>	IN	<i>table</i> から削除する行数。

### 戻りコード

表 126. DTW\_TB\_DELETEROW の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

### 使用上の注意

DTW\_TB\_DELETEROW() を呼び出す前に、表内の列番号を設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使うか、あるいは、その表を設定する言語環境に渡すことによって、列番号を設定することができます。

### 例

例 1: 表の行 10 から開始して 5 行を削除します。

```
%DEFINE myTable = %TABLE  
  
@DTW_TB_DELETEROW(myTable, "10", "5")
```

**例 2:** 表のすべての行を削除します。

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_DELETE_ROW(myTable, "1", @DTW_TB_rROWS(myTable))
```

## DTW\_TB\_DLIST

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

### 目的

Net.Data 表から HTML 定義リストを生成します。

### 構文

```
@DTW_TB_DLIST(table, term, def, termstyle, defstyle, link, link_u, image, image_u)
@DTW_TB_DLIST(table, term, def, termstyle, defstyle, link, link_u, image)
@DTW_TB_DLIST(table, term, def, termstyle, defstyle, link, link_u)
@DTW_TB_DLIST(table, term, def, termstyle, defstyle, link)
@DTW_TB_DLIST(table, term, def, termstyle, defstyle)
@DTW_TB_DLIST(table, term, def, termstyle)
@DTW_TB_DLIST(table, term, def)
@DTW_TB_DLIST(table, term)
@DTW_TB_DLIST(table)
```

### 値

表 127. DTW\_TB\_DLIST のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	HTML 定義リストとして表示するマクロ表変数を指定するシンボル。
整数	<i>term</i>	IN	<i>term</i> 名前値 (<DT> タグの後に続くテキスト) が含まれている <i>table</i> 内の列番号。デフォルトでは、先頭の列が使用されます。
整数	<i>def</i>	IN	<i>term</i> 定義値 (<DD> タグの後に続くテキスト) が含まれている <i>table</i> 内の列番号。デフォルトでは、2 番目の列が使用されます。
ストリング	<i>termstyle</i>	IN	<i>term</i> 名前値の HTML 要素のリストが含まれている変数またはリテラル・ストリング。デフォルトでは、スタイル・タグは使用されません。
ストリング	<i>defstyle</i>	IN	<i>term</i> 定義値の HTML 要素のリストが含まれている変数またはリテラル・ストリング。デフォルトでは、スタイル・タグは使用されません。
ストリング	<i>link</i>	IN	どの HTML 要素に対して HTML リンクが生成されるかを指定します。有効な値は DT と DD です。デフォルトでは、HTML リンクは生成されません。

表 127. DTW\_TB\_DLIST のパラメーター (続き)

データ型	パラメーター	用途	説明
整数	<i>link_u</i>	IN	HTML 参照の URL が含まれている <i>table</i> 内の列番号。デフォルトでは、HTML リンクは生成されません。
ストリング	<i>image</i>	IN	どの HTML 要素に対してインライン・イメージが生成されるかを指定します。有効な値は DT と DD です。デフォルトでは、インライン・イメージ (DT) は生成されません。
整数	<i>image_u</i>	IN	インライン・イメージのための URL が含まれている <i>table</i> 内の列番号。デフォルトでは、インライン・イメージは生成されません。

## 戻りコード

表 128. DTW\_TB\_DLIST の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

## 例

例 1: 表データに基づいて以下の HTML を生成する定義リストを作成します。

```
@DTW_TB_DLIST(Mytable,"3","4","b i","strong","DD","2","DT","1")
```

結果:

```
<DL>
<DT>
<IMG SRC="http://www.mycompany.com/images/image1.gif" ALT=""><b><i>image1text</i></b>
<DD>
<A HREF="http://www.mycompany.com/link1.html"><strong>link1text</strong></A>
<DT>
<IMG SRC="http://www.mycompany.com/images/image2.gif" ALT=""><b><i>image2text</i></b>
<DD>
<A HREF="http://www.mycompany.com/link2.html"><strong>link2text</strong></A>
<DT>
<IMG SRC="http://www.mycompany.com/images/image3.gif" ALT=""><b><i>image3text</i></b>
<DD>
<A HREF="http://www.mycompany.com/link3.html"><strong>link3text</strong></A>
<DT>
<IMG SRC="http://www.mycompany.com/images/image4.gif" ALT=""><b><i>image4text</i></b>
```

```
<DD>  
<A HREF="http://www.mycompany.com/link4.html"><strong>link4text</strong></A>  
</DT>  
</DL>
```



# DTW\_TB\_DUMP

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

## 目的

HTMLの <PRE> タグを使用し、Net.Data 表の内容を印刷します。表の各行は、1 行に表示されます。

## 構文

@DTW\_TB\_DUMP(table)

## 値

表 129. DTW\_TB\_DUMP のパラメーター

データ型	パラメーター	用途	説明
表	table	IN	表示するマクロ表変数を指定するシンボル。

## 戻りコード

表 130. DTW\_DB\_DUMP の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。

## 使用上の注意

Net.Data 表が空の場合、エラーが戻されます。

## 例

例 1:

@DTW\_TB\_DUMP(Mytable)

この例で生成される HTML は、次のようになります。

```
<PRE>
Name           Department      Position
Jack Smith     Internet Technologies  Software Engineer
Helen Williams Database             Development Manager
Alex Jones      Manufacturing         Industrial Engineer
Tom Baker       Procurement           Sales Rep
</PRE>
```

## DTW\_TB\_DUMPV

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

### 目的

HTML <PRE> タグを使用し、Net.Data 表の内容を印刷します。表の各フィールドは、1 行に表示されます。

### 構文

@DTW\_TB\_DUMPV(table)

### 値

表 131. DTW\_TB\_DUMPV のパラメーター

データ型	パラメーター	用途	説明
表	table	IN	表示するマクロ表変数を指定するシンボル。

### 戻りコード

表 132. DTW\_TB\_DUMPV の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てたための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。

### 使用上の注意

Net.Data 表が空の場合、エラーが戻されます。

### 例

#### 例 1:

@DTW\_TB\_DUMPV(Mytable)

この例で生成される HTML は、次のようになります。

```
<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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

Net.Data 表から列見出しを戻します。

### 構文

@DTW\_TB\_GETN(table, col, name)

@DTW\_TB\_rGETN(table, col)

### 値

表 133. DTW\_TB\_GETN のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	列名が戻されるマクロ表変数。
整数	<i>col</i>	IN	その名前が戻される列の列番号。
ストリング	<i>name</i>	OUT	<i>col</i> で指定される列名を含む変数。

### 戻りコード

表 134. DTW\_TB\_GETN の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

### 使用上の注意

DTW\_TB\_GETN() を呼び出す前に、表内の列番号を設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使うか、あるいは、その表を設定する言語環境に渡すことによって、列番号を設定することができます。

### 例

例 1: 列 4 の列名を検索します。

```
%DEFINE myTable = %TABLE
%DEFINE name = ""
...
@FillTable(myTable)
...
@DTW_TB_GETN(myTable, "4", name)
```

**例 2:** 表内の最後の列の列名を検索します。

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<P>The column name of the last column is @DTW_TB_rGETN(myTable, @DTW_TB_rCOLS(myTable))
```

## DTW\_TB\_GETV

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

### 目的

Net.Data 表の指定した行および列の値を戻します。

### 構文

@DTW\_TB\_GETV(table, row, col, value)

@DTW\_TB\_rGETV(table, row, col)

### 値

表 135. DTW\_TB\_GETV のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	表値が戻されるマクロ表変数。
整数	<i>row</i>	IN	戻される値の行番号。
整数	<i>col</i>	IN	戻される値の列番号。
ストリング	<i>value</i>	OUT	<i>row</i> および <i>col</i> で指定された行と列に値を含む変数。

### 戻りコード

表 136. DTW\_TB\_GETV の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

### 使用上の注意

DTW\_TB\_GETV() を呼び出す前に、表内の列番号を設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使うか、あるいは、その表を設定する言語環境に渡すことによって、列番号を設定することができます。

## 例

例 1: 行 6、列 3 の表値を検索します。

```
%DEFINE myTable = %TABLE
%DEFINE value = ""
...
@FillTable(myTable)
...
@DTW_TB_GETV(myTable, "6", "3", value)
```

例 2: 行 1、列 1 の表値を検索します。

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<P>The table value of row 1, column 1 is @DTW_TB_rGETV(myTable, "1", "1").
```

## DTW\_TB\_HTMLENCODE

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

### 目的

Net.Data 表に配置されているデータの特定の文字を、その対応する HTML 文字エスケープ・コードで置換します。

### 構文

@DTW\_TB\_HTMLENCODE(table, collist)

@DTW\_TB\_HTMLENCODE(table)

### 値

表 137. DTW\_TB\_HTMLENCODE のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	INOUT	変更するマクロ表変数。
ストリング	<i>collist</i>	IN	エンコードする <i>table</i> 内の列番号。デフォルトでは、すべての列がエンコードされます。

### 戻りコード

表 138. DTW\_TB\_HTMLENCODE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

### 使用上の注意

置換される文字を以下の表に示します。

名前	文字	コード
アンパーサンド	&	&#38;
二重引用符	"	&#34;
より大 (GT)	>	&#62;
より小 (LT)	<	&#60;



## 例

### 例 1:

```
@DTW_TB_HTMLencode(Mytable, "3 4")
```

指定された表の列 3 および 4 の特殊文字は、それぞれ対応するエンコード形式に置き換えられます。

## DTW\_TB\_INPUT\_CHECKBOX

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

### 目的

Net.Data 表から 1 つまたは複数の HTML チェック・ボックス入力タグを生成します。

### 構文

@DTW\_TB\_INPUT\_CHECKBOX(table, prompt, namecol, valuecol, rows, checkedrows)

@DTW\_TB\_INPUT\_CHECKBOX(table, prompt, namecol, valuecol, rows)

@DTW\_TB\_INPUT\_CHECKBOX(table, prompt, namecol, valuecol)

@DTW\_TB\_INPUT\_CHECKBOX(table, prompt, namecol)

### 値

表 139. DTW\_TB\_INPUT\_CHECKBOX のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	チェック・ボックス入力タグとして表示するマクロ表変数。
ストリング	<i>prompt</i>	IN	チェック・ボックスの次に表示するテキストが含まれている、 <i>table</i> またはストリング内の列番号。このパラメーターは必須ですが、ヌル ("" ) 値にすることもできます。 <i>prompt</i> がヌルであれば、使用される値は <i>namecol</i> に定義された値になります。
ストリング	<i>namecol</i>	IN	入力フィールド名が含まれている、 <i>table</i> またはストリング内の列番号。
整数	<i>valuecol</i>	IN	入力フィールド値が含まれている <i>table</i> 内の列番号。デフォルトは 1 です。
整数	<i>rows</i>	IN	入力フィールドが生成される <i>table</i> 内の行のリスト。デフォルトでは、すべての行が使用されます。
整数	<i>checkedrows</i>	IN	<i>table</i> のどの <i>rows</i> を検査するかを指定する行のリスト。デフォルトでは、フィールドは検査されません。

### 戻りコード

表 140. DTW\_TB\_INPUT\_CHECKBOX の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。

表 140. DTW\_TB\_INPUT\_CHECKBOX の戻りコード (続き)

戻りコード	説明
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

## 例

**例 1:** 3 つのチェック・ボックス入力タグのための HTML を生成します。

```
@DTW_TB_INPUT_CHECKBOX(Mytable,"3","4","", "2 3 4", "1 3 4")
```

結果:

```
<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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

Net.Data 表から HTML ラジオ・ボタン入力タグを生成します。

### 構文

@DTW\_TB\_INPUT\_RADIO(table, prompt, namecol, valuecol, rows, checkedrows)

@DTW\_TB\_INPUT\_RADIO(table, prompt, namecol, valuecol, rows)

@DTW\_TB\_INPUT\_RADIO(table, prompt, namecol, valuecol)

### 値

表 141. DTW\_TB\_INPUT\_RADIO のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	ラジオ・ボタン入力タグとして表示するマクロ表変数。
ストリング	<i>prompt</i>	IN	ラジオ・ボタンの次に表示するテキストが含まれている、 <i>table</i> またはストリング内の列番号。このパラメーターは必須ですが、ヌル ("" ) 値にすることもできます。 <i>prompt</i> がヌルであれば、 <i>valuecol</i> の値が使用されます。
ストリング	<i>namecol</i>	IN	入力フィールド名が含まれている、 <i>table</i> またはストリング内の列番号。
整数	<i>valuecol</i>	IN	入力フィールド値が含まれている <i>table</i> 内の列番号。
ストリング	<i>rows</i>	IN	入力フィールドが生成される <i>table</i> 内の行のリスト。デフォルトでは、すべての行が使用されます。
整数	<i>checkedrows</i>	IN	対応する検査済みラジオ・ボタンを表示する <i>table</i> 内の行番号。1 つの値しか使用できません。

### 戻りコード

表 142. DTW\_TB\_INPUT\_RADIO の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。

表 142. DTW\_TB\_INPUT\_RADIO の戻りコード (続き)

戻りコード	説明
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

## 例

**例 1:** 3 つのラジオ・ボタン入力タグのための HTML を生成します。

```
@DTW_TB_INPUT_RADIO(Mytable,"3","Radio4","4","2 3 4","4")
```

結果:

```
<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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

Net.Data 表の指定行の HTML <INPUT> タグを戻します。

### 構文

@DTW\_TB\_INPUT\_TEXT(table, prompt, namecol, valuecol, size, maxlen, rows)

@DTW\_TB\_INPUT\_TEXT(table, prompt, namecol, valuecol, size, maxlen)

@DTW\_TB\_INPUT\_TEXT(table, prompt, namecol, valuecol, size)

@DTW\_TB\_INPUT\_TEXT(table, prompt, namecol, valuecol)

@DTW\_TB\_INPUT\_TEXT(table, prompt, namecol)

### 値

表 143. DTW\_TB\_INPUT\_TEXT のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	テキスト入力タグとして表示するマクロ表変数。
ストリング	<i>prompt</i>	IN	入力フィールドの次に表示するテキストが含まれている、 <i>table</i> またはストリング内の列番号。 <i>prompt</i> がヌルであれば、テキストは表示されません。
ストリング	<i>namecol</i>	IN	入力フィールド名が含まれている <i>table</i> 内の列番号。
整数	<i>valuecol</i>	IN	デフォルトの入力フィールド値が含まれている <i>table</i> 内の列番号。 INPUT タグの VALUE 属性に対して指定されます。デフォルトでは、VALUE 属性値は生成されません。
整数	<i>size</i>	IN	入力フィールドの文字の数。INPUT タグの SIZE 属性に対して指定されます。デフォルトは、最も長いデフォルト入力値の長さ、またはデフォルト入力がない場合は 10 です。
整数	<i>maxlen</i>	IN	入力ストリングの最大長。INPUT タグの MAXLENGTH 属性に対して指定されます。デフォルトでは、MAXLENGTH 属性値は生成されません。
整数	<i>rows</i>	IN	入力フィールドが生成される <i>table</i> 内の行のリスト。デフォルトでは、すべての行が使用されます。

## 戻りコード

表 144. DTW\_TB\_INPUT\_TEXT の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

## 例

例 1: 3 つの HTML <INPUT> タグを戻します。

```
@DTW_TB_INPUT_TEXT(Mytable,"3","3","4","35","40","1 2 3")
```

結果:

```
<P>image1text
<INPUT TYPE="TEXT" NAME="image1text" VALUE="link1text" SIZE="35" MAXLENGTH="40">
<P>image2text
<INPUT TYPE="TEXT" NAME="image2text" VALUE="link2text" SIZE="35" MAXLENGTH="40">
<P>image3text
<INPUT TYPE="TEXT" NAME="image3text" VALUE="link3text" SIZE="35" MAXLENGTH="40">
```

## DTW\_TB\_INSERTCOL

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

### 目的

Net.Data 表に 1 つまたは複数の列を挿入します。

### 構文

```
@DTW_TB_INSERTCOL(table, after_col, cols)
```

### 値

表 145. DTW\_TB\_INSERTCOL のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	INOUT	列が挿入されるマクロ表変数。
整数	<i>after_col</i>	IN	新しい列が挿入される列の列番号 (この列の後に挿入されます)。この表の始めに列を挿入する場合は、0 を指定します。
整数	<i>cols</i>	IN	<i>table</i> に挿入する列の数。

### 戻りコード

表 146. DTW\_TB\_INSERTCOL の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

### 例

例 1: 表の終わりに 5 列を挿入します。

```
%DEFINE myTable = %TABLE  
  
@DTW_TB_INSERTCOL(myTable, @DTW_TB_rCOLS(myTable), "5")
```

例 2: 表の先頭に 1 列を挿入します。

```
%DEFINE myTable = %TABLE  
  
@DTW_TB_INSERTCOL(myTable, "0", "1")
```



## DTW\_TB\_INSERTROW

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

### 目的

Net.Data 表に 1 つまたは複数の行を挿入します。

### 構文

@DTW\_TB\_INSERTROW(table, after\_row, rows)

### 値

表 147. DTW\_TB\_INSERTROW のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	INOUT	行が挿入されるマクロ表変数。
整数	<i>after_row</i>	IN	新しい行が挿入される行の番号 (この行の後に挿入されます)。この表の始めに行を挿入する場合は、0 を指定します。
整数	<i>rows</i>	IN	<i>table</i> に挿入する行の数。

### 戻りコード

表 148. DTW\_TB\_INSERTROW の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

### 使用上の注意

DTW\_TB\_INSERTROW() を呼び出す前に、表内の列番号を設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使うか、あるいは、その表を設定する言語環境に渡すことによって、列番号を設定することができます。

### 例

例 1: 表の行 5 の後に 1 行挿入します。

```
%DEFINE myTable = %TABLE  
@DTW_TB_INSERTROW(myTable, "5", "1")
```

**例 2:** 表の先頭に 3 行挿入します。

```
%DEFINE myTable = %TABLE  
@DTW_TB_INSERTROW(myTable, "0", "3")
```

## DTW\_TB\_LIST

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

### 目的

Net.Data 表から HTML リストを生成します。

### 構文

@DTW\_TB\_LIST(table, listtype, listitem, itemstyle, link\_u, image\_u)

@DTW\_TB\_LIST(table, listtype, listitem, itemstyle, link\_u)

@DTW\_TB\_LIST(table, listtype, listitem, itemstyle)

@DTW\_TB\_LIST(table, listtype, listitem)

@DTW\_TB\_LIST(table, listtype)

### 値

表 149. DTW\_TB\_LIST のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	HTML リストとして表示するマクロ変数を指定するシンボル。
ストリング	<i>listtype</i>	IN	生成するリストの型。許容値としては、次のものがあります。  DIR MENU OL UL
整数	<i>listitem</i>	IN	リスト値 (<LI> タグの後に続くテキスト) が含まれている <i>table</i> 内の列番号。デフォルトでは、先頭の列が使用されます。
ストリング	<i>itemstyle</i>	IN	term 名前値の HTML 要素のリストが含まれている変数またはリテラル・ストリング。デフォルトでは、スタイル・タグは使用されません。
整数	<i>link_u</i>	IN	HTML リンクの URL が含まれている <i>table</i> 内の列番号。この値が指定されていない場合は、HTML リンクは生成されません。
整数	<i>image_u</i>	IN	インライン・イメージのための URL が含まれている <i>table</i> 内の列番号。この値が指定されていない場合は、インライン・イメージは生成されません。

## 戻りコード

表 150. DTW\_TB\_LIST の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

## 例

例 1: 順序付きリストのための HTML タグを生成します。

```
@DTW_TB_LIST(Mytable,"0L","4","TT U","2","1")
```

結果:

```
<TT><U>
<OL>
<LI><A HREF="http://www.mycompany.com/link1.html">
<IMG SRC="http://www.mycompany.com/images/image1.gif" ALT="">link1text</A>
<LI><A HREF="http://www.mycompany.com/link2.html">
<IMG SRC="http://www.mycompany.com/images/image2.gif" ALT="">link2text</A>
<LI><A HREF="http://www.mycompany.com/link3.html">
<
IMG SRC="http://www.mycompany.com/images/image3.gif" ALT="">link3text</A>
<LI><A HREF="http://www.mycompany.com/link4.html">
<IMG SRC="http://www.mycompany.com/images/image4.gif" ALT="">link4txt</A>
</OL>
</U></TT>
```

## DTW\_TB\_QUERYCOLNONJ

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

### 目的

Net.Data 表の列見出しに関連する列番号を戻します。

### 構文

@DTW\_TB\_QUERYCOLNONJ(table, name, col)

@DTW\_TB\_rQUERYCOLNONJ(table, name)

### 値

表 151. DTW\_TB\_QUERYCOLNONJ のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	列番号が戻されるマクロ表変数。
ストリング	<i>name</i>	IN	列番号が戻される列見出しの名前。その列見出しが表に存在しない場合には、0 が戻されます。
整数	<i>col</i>	OUT	<i>name</i> で名前が指定された列の列番号を含む変数。

### 戻りコード

表 152. DTW\_TB\_QUERYCOLNONJ の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 使用上の注意

- DTW\_TB\_QUERYCOLNONJ() を呼び出す前に、表内の列番号を設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使うか、あるいは、その表を設定する言語環境に渡すことによって、列番号を設定することができます。
- その列見出しが表に存在しない場合には、0 が戻されます。

## 例

**例 1:** SERIAL\_NUMBER で名前が指定された列の列番号を検索します。

```
%DEFINE myTable = %TABLE  
%DEFINE col = ""
```

```
@DTW_TB_QUERYCOLNONJ(myTable, "SERIAL_NUMBER", col)
```

**例 2:** SERIAL\_NUMBER で名前が指定された列の列番号を戻します。

```
%DEFINE myTable = %TABLE  
<P>The "SERIAL_NUMBER" column is column number @DTW_TB_rQUERYCOLNONJ(myTable, "SERIAL_NUMBER")
```

## DTW\_TB\_ROWS

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

### 目的

Net.Data 表内の行数を戻します。

### 構文

@DTW\_TB\_ROWS(table, rows)

@DTW\_TB\_rROWS(table)

### 値

表 153. DTW\_TB\_ROWS のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	行の現行番号が戻されるマクロ表変数。
整数	<i>rows</i>	OUT	<i>table</i> 内の行の現行番号が含まれている変数。

### 戻りコード

表 154. DTW\_TB\_ROWS の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 例

**例 1:** 表内にある現在の列数を検索し、その値を *rows* に割り当てます。

```
%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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

Net.Data 表から HTML 選択リストを生成します。

### 構文

@DTW\_TB\_SELECT(table, name, optioncol, size, multiple, rows, selectedrows, valuecol)

@DTW\_TB\_SELECT(table, name, optioncol, size, multiple, rows, selectedrows)

@DTW\_TB\_SELECT(table, name, optioncol, size, multiple, rows)

@DTW\_TB\_SELECT(table, name, optioncol, size, multiple)

@DTW\_TB\_SELECT(table, name, optioncol, size)

@DTW\_TB\_SELECT(table, name, optioncol)

@DTW\_TB\_SELECT(table, name)

### 値

表 155. DTW\_TB\_SELECT のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	SELECT フィールドとして表示するマクロ表変数。
ストリング	<i>name</i>	IN	SELECT フィールドの NAME 属性の値。
整数	<i>optioncol</i>	IN	SELECT フィールドの OPTION タグに使用する値をもつ <i>table</i> 内の列番号。デフォルトでは、先頭の列が使用されます。
整数	<i>size</i>	IN	SELECT フィールドの OPTION タグに使用する <i>table</i> 内の行の数。デフォルトでは、すべての行が使用されます。
ストリング	<i>multiple</i>	IN	複数の選択が許されるかどうかを指定します。デフォルトは N です。つまり、複数の選択は許されません。
ストリング	<i>rows</i>	IN	SELECT フィールドで使用する <i>table</i> 内の行番号。デフォルトでは、すべての行が使用されます。
ストリング	<i>selectedrows</i>	IN	OPTION タグが検査される表の行リスト。複数の行を指定するには、複数のパラメーターを Y に設定する必要があります。デフォルトでは、先頭の項目が選択されます。
ストリング	<i>valuecol</i>	IN	OPTION タグの VALUE 属性に使用する <i>table</i> の列番号。このパラメーターはオプションです。



## 戻りコード

表 156. DTW\_TB\_SELECT の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

## 例

**例 1:** 複数の選択をもつ HTML SELECT メニューを生成します。

```
@DTW_TB_SELECT(Mytable,"URL6","3","","y","1 2 4","1 4")
```

結果:

```
<SELECT NAME="URL6" SIZE="3" MULTIPLE>
<OPTION SELECTED>image1text
<OPTION>image2text
<OPTION SELECTED>image4text
</SELECT>
```

**例 2:** *valuecol* パラメーターを使用して、値の取得先の列番号を使用した HTML SELECT メニューを生成します。

結果:

```
<SELECT NAME="URL6" SIZE="3" MULTIPLE>
<OPTION VALUE="1" SELECTED>image1text
<OPTION VALUE="2">image2text
<OPTION VALUE="3" SELECTED>image3text
</SELECT>
```

## DTW\_TB\_SETCOLS

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

### 目的

Net.Data 表内の列数を設定します。

### 構文

```
@DTW_TB_SETCOLS(table, cols)
```

### 値

表 157. DTW\_TB\_SETCOLS のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	INOUT	列の数が設定されるマクロ表変数。
整数	<i>cols</i>	IN	<i>table</i> に割り当てる列数の初期値。

### 戻りコード

表 158. DTW\_TB\_SETCOLS の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。

### 使用上の注意

- DTW\_TB\_SETCOLS() 関数は、1 つの表につき 1 回だけです。その後では、DTW\_TB\_DELETECOL() または DTW\_TB\_INSERTCOL() 関数を使用して表の中の列の数を変更してください。
- DTW\_TB\_SETN() 関数を使用して列見出しを指定してください。

### 例

例 1: 表に 3 つの列を割り当て、それらの列の名前を割り当てます。

```
%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	SCO	SUN	Win NT
X	X	X	X	X	X	X	X	X

### 目的

Net.Data 内の列見出しに名前を割り当てます。

### 構文

@DTW\_TB\_SETN(table, name, col)

### 値

表 159. DTW\_TB\_SETN のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	INOUT	列名が設定されるマクロ表変数。
ストリング	<i>name</i>	IN	<i>col</i> で指定された列の列見出しとして割り当てられる文字ストリング。
整数	<i>col</i>	IN	見出しが設定される列の列番号。

### 戻りコード

表 160. DTW\_TB\_SETN の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

### 使用上の注意

- DTW\_TB\_SETN() を呼び出す前に、表内の列番号を設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使うか、あるいは、その表を設定する言語環境に渡すことによって、列番号を設定することができます。
- 列見出しを削除する場合は、列見出し値として NULL を割り当てます。

### 例

例 1: 列見出し 1 から 3 までに名前を割り当てます。

```
%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")
```

**例 2:** 列 2 の列見出しを削除します。これは、まだ定義されていない関数呼び出しで変数を渡すことによって行われます。デフォルトでは、この変数の値は NULL になります。

```
%DEFINE myTable = %TABLE

@DTW_TB_SETN(myTable, nullVar, "2")
```

# DTW\_TB\_SETV

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

## 目的

Net.Data 表の特定の行および列に値を割り当てます。

## 構文

@DTW\_TB\_SETV(table, value, row, col)

## 値

表 161. DTW\_TB\_SETV のパラメーター

データ型	パラメーター	用途	説明
表	table	INOUT	表値が設定されるマクロ表変数。
ストリング	value	IN	row および col で指定された行と列の表値に割り当てられる文字ストリング。
整数	row	IN	設定される値の行番号。
整数	col	IN	設定される値の列番号。

## 戻りコード

表 162. DTW\_TB\_SETV の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

## 使用上の注意

- DTW\_TB\_SETV() を呼び出す前に、表内の列番号を設定しなければなりません。  
DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使うか、あるいは、その表を設定する言語環境に渡すことによって、列番号を設定することができます。
- 表値を削除する場合は、この値として NULL を割り当てます。

## 例

例 1: 行 3 の列 3 に値を割り当てます。

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_SETV(myTable, "value3.3", "3", "3")
```

**Example 2:** 行 4 の列 2 の表値を削除します。これは、まだ定義されていない関数呼び出しで変数を渡すことによって行われます。デフォルトでは、この変数の値は NULL になります。

```
%DEFINE myTable = %TABLE
```

```
@DTW_TB_SETV(myTable, nullVar, "4", "2")
```

## DTW\_TB\_TABLE

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

### 目的

Net.Data 表から HTML 表を生成します。

### 構文

@DTW\_TB\_TABLE(table, options, collist, cellstyle, link\_u, image\_u, url\_text, url\_style)

@DTW\_TB\_TABLE(table, options, collist, cellstyle, link\_u, image\_u, url\_text)

@DTW\_TB\_TABLE(table, options, collist, cellstyle, link\_u, image\_u)

@DTW\_TB\_TABLE(table, options, collist, cellstyle, link\_u)

@DTW\_TB\_TABLE(table, options, collist, cellstyle)

@DTW\_TB\_TABLE(table, options, collist)

@DTW\_TB\_TABLE(table, options)

@DTW\_TB\_TABLE(table)

### 値

表 163. DTW\_TB\_TABLE のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	HTML 表として表示するマクロ表変数。
ストリング	<i>options</i>	IN	TABLE タグ内の表属性。デフォルトでは、属性は使用されません。有効な値としては、以下のものがあります。 <ul style="list-style-type: none"><li>• BORDER</li><li>• CELLSPACING</li><li>• WIDTH</li></ul>
ストリング	<i>collist</i>	IN	HTML 表で使用する <i>table</i> 内の列番号。デフォルトでは、すべての列が使用されます。
ストリング	<i>cellstyle</i>	IN	各 TD タグのテキストに入れる HTML スタイル要素 (たとえば、B や I) のリスト。デフォルトでは、スタイル・タグは使用されません。
整数	<i>link_u</i>	IN	HTML リンクを作成するために使用する URL が含まれている <i>table</i> 内の列番号。 <i>collist</i> 内の列も指定する必要があります。デフォルトでは、HTML リンクは生成されません。
整数	<i>image_u</i>	IN	インライン・イメージを作成するために使用する URL が含まれている <i>table</i> 内の列番号。 <i>collist</i> 内の列も指定する必要があります。デフォルトでは、イメージ・タグは生成されません。

表 163. DTW\_TB\_TABLE のパラメーター (続き)

データ型	パラメーター	用途	説明
整数	<i>url_text</i>	IN	HTML リンクまたはインライン・イメージ用に表示するテキストが含まれている <i>table</i> 内の列番号。デフォルトでは、URL それ自体が使用されます。
ストリング	<i>url_style</i>	IN	<i>url_text</i> に指定されたテキストに関する HTML スタイル要素のリスト。デフォルトでは、スタイル・タグは生成されません。

## 戻りコード

表 164. DTW\_TB\_TABLE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
1008	パラメーターが表境界の外側にあります。

## 例

**例 1:** ボーダーをもち、 B (太字) および I (斜体文字) タグを使用する表のための HTML タグを生成します。

```
@DTW_TB_TABLE(Mytable,"BORDER","4 2 1","i","2","1","4","b")
```

結果:

```
<TABLE BORDER>
<TR>
<TH>TITLE
<TH>LINKURL
<TH>IMAGEURL
<TR>
<TD><i>link1text</i>
<TD><A HREF="http://www.mycompany.com/link1.html"><b>link1text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image1.gif" ALT=""><b>link1text</b>
<TR>
<TD><i>link2text</i>
<TD><A HREF="http://www.mycompany.com/link2.html"><b>link2text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image2.gif" ALT=""><b>link2text</b>
<TR>
<TD><i>link3text</i>
<TD><A HREF="http://www.mycompany.com/link3.html"><b>link3text</b></A>
<TD><IMG SRC="http://www.mycompany.com/images/image3.gif" ALT=""><b>link3text</b>
</TABLE>
```



# DTW\_TB\_TEXTAREA

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

## 目的

Net.Data 表から HTML テキスト領域を生成します。

## 構文

```
@DTW_TB_TEXTAREA(table, name, numrows, numcols, valuecol, rows)
@DTW_TB_TEXTAREA(table, name, numrows, numcols, valuecol)
@DTW_TB_TEXTAREA(table, name, numrows, numcols)
@DTW_TB_TEXTAREA(table, name, numrows)
@DTW_TB_TEXTAREA(table, name)
```

## 値

表 165. DTW\_TB\_TEXTAREA のパラメーター

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	TEXTAREA タグとして表示するマクロ表変数。
ストリング	<i>name</i>	IN	テキスト・エリアの名前。
整数	<i>numrows</i>	IN	テキスト・エリアの高さ (行数で指定)。デフォルトは、 <i>table</i> 内の行数です。
整数	<i>numcols</i>	IN	テキスト・エリアの幅 (列数で指定)。デフォルトは、 <i>table</i> 内の最も長い行の長さです。
整数	<i>valuecol</i>	IN	値がテキスト・エリアに示される <i>table</i> 内の列番号。デフォルトは最初の列です。
ストリング	<i>rows</i>	IN	TEXTAREA タグを生成するために使用する <i>table</i> 内の行のリスト。デフォルトでは、すべての行が使用されます。

## 戻りコード

表 166. DTW\_TB\_TEXTAREA の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。

表 166. DTW\_TB\_TEXTAREA の戻りコード (続き)

戻りコード	説明
1008	パラメーターが表境界の外側にあります。

## 例

**例 1:** HTML TEXTAREA タグを生成し、インクルードする行を指定します。

```
@DTW_TB_TEXTAREA(Mytable,"textarea5","3","70","4","1 3 4")
```

結果:

```
<TEXTAREA NAME="textarea5" ROWS="3" COLS="70">
link1text
link3text
link4text
</TEXTAREA>
```

---

## フラット・ファイル・インターフェース関数

フラット・ファイル・インターフェース (FFI) を使用すれば、フラット・ファイル・ソース (テキスト・ファイル) のデータをオープンし、読み取り、操作することができます。ほか、データをフラット・ファイルに格納することができます。以下のフラット・ファイル・インターフェース組み込み関数が使用可能です。

- 274ページの『DTWF\_APPEND』
- 277ページの『DTWF\_CLOSE』
- 278ページの『DTWF\_DELETE』
- 283ページの『DTWF\_OPEN』
- 285ページの『DTWF\_READ』
- 288ページの『DTWF\_REMOVE』
- 290ページの『DTWF\_SEARCH』
- 293ページの『DTWF\_UPDATE』
- 296ページの『DTWF\_WRITE』

以下のセクションでは、FFI 組み込み関数の使用方法と、フラット・ファイル・ソースへのアクセス方法について説明します。

- 『フラット・ファイル・データ・ソースへのアクセス』
- 272ページの『フラット・ファイル・インターフェース区切り文字』
- 273ページの『ファイルのロック』

## フラット・ファイル・データ・ソースへのアクセス

Net.Data 初期設定ファイルで FFI\_PATH パス構成ステートメントを使用して、FFI 関数を使用するときに指定できるディレクトリーおよびサブディレクトリーをリストしたり、パス・ステートメントに指定されたディレクトリーに含まれていないファイルの機密保護を提供したりします。Net.Data 初期設定ファイルは FFI\_PATH なしで出荷されます。パスの構成方法については、*Net.Data 管理およびプログラミングの手引き* を参照してください。

FFI\_PATH は以下の構文を使用します。

```
FFI_PATH /path1;/path2;/path3...
```

マクロ関数内で FFI 言語環境を呼び出すときに、FFI 関数の *filename* パラメーターを使用して、その FFI 関数が操作するフラット・ファイルへのパスを指定します。たとえば、以下のようにします。

```
%DEFINE myfile = "/macros/myfile.txt" @DTWF_READ(myfile, ...)
```

以下のセクションに分けて説明します。

- 270ページの『Net.Data がフラット・ファイル位置を判別する方法』
- 271ページの『フラット・ファイルの構成規則』
- 271ページの『機密保護の推奨事項』

- 272ページの『許可要件』

## Net.Data がフラット・ファイル位置を判別する方法

Net.Data は、FFI 関数の *filename* パラメーターを使用して、Net.Data 初期設定ファイルファイル内の FFI\_PATH ステートメントを検索し、指定されたディレクトリーを使用するのか、あるいは現行ディレクトリーを使用するのかを判別します。

FFI 関数でファイル名が指定されると、Net.Data は、指定された最初のパスから開始して、FFI\_PATH にリストされたそれぞれのパスを検索して、そのファイルを見付けようとします。Net.Data は、検出した最初のコピーを使用します。ファイルが見つからなかった場合には、Net.Data は、Net.Data が実行されているプロセスまたはスレッドの現行作業ディレクトリーからそのファイルを見付けようとします。

**例:** Net.Data が FFI\_PATH 構成ステートメントを使用してファイルを見付けます。

FFI\_PATH には以下のディレクトリーが含まれています。

```
FFI_PATH /macros;/macros/org1;/macros/org2
```

また、ファイルは現行ディレクトリーおよび /macros/org1 の両方に配置されます。関数呼び出しが以下の場合、

```
DTWF_READ("myfile.txt")
```

Net.Data は /macros/org1/myfile.txt を使用します。

既存のファイルを読み取るために DTWF\_READ 関数を使用されている場合に、myfile.txt というファイル名が指定されていると、Net.Data は、上で指定されたパスのリストが FFI\_PATH に含まれているものと想定して、ディレクトリー /macros、/macros/org1、および /macros/org2 からそのファイルを探索します。

### 現行ディレクトリーの判別:

Net.Data の現行ディレクトリーは、ユーザーの Web サーバーの構成によって異なります。

- CGI を使用している場合、現行ディレクトリーは Net.Data が実行されているディレクトリーです。
- Web サーバー API を使用している場合には、現行ディレクトリーはさまざまに異なる可能性があります。このサーバーのデフォルトの要求ルーティングまたはリソース・マッピングが変更されると、現行ディレクトリーも変更される可能性があります。

### フラット・ファイル・アクセスの指定に関する推奨事項

Net.Data がフラット・ファイル・データ・ソースにアクセスできるようにするために、以下の推奨事項に従ってください。

- フラット・ファイルを作成するために DTWF\_OPEN 関数を使用している場合には、FFI\_PATH に入っているディレクトリー・パスを指定するか、あるいは現行ディレクトリーが分かっているようにする必要があります。ディレクトリーを指定しないと、Net.Data は現行作業ディレクトリーにそのファイルを作成しようとします。

- *filename* パラメーターにディレクトリーを組み込む場合、 FFI\_PATH 内のパスのいずれかと一致するフルパスを指定してください。これは、Net.Data が FFI\_PATH で指定されたディレクトリー内のサブディレクトリーを検索しないためです。
- Web サーバー API を使用している場合には特に、 *filename* パラメーターに絶対パスを使用してください。

## フラット・ファイルの構成規則

Net.Data 初期設定ファイル内で FFI\_PATH を追加または更新するときには、以下の規則を使用してください。

- FFI\_PATH 内のパス・ステートメントには、有効な印刷可能文字を含めなければなりません。 FFI では、疑問符 (?) または二重引用符 (") を含むパスは許されません。
- マクロ内で *filename* パラメーターとともに使用されるすべてのディレクトリーは、 FFI\_PATH で指定されている必要があります。*filename* でリストされているパスのサブディレクトリーは、 FFI\_PATH で明示的に指定されていない限り検索されません。
- FFI\_PATH ステートメントには絶対パスを使用してください。

## 機密保護の推奨事項

FFI 関数がアクセスすることのできるファイルは、 Net.Data 初期設定ファイルの FFI\_PATH ステートメントを使用して指定することができます。 FFI は、このステートメントでリストされたパスだけを検索するため、その他のディレクトリー内のファイルは無許可アクセスから保護されます。

たとえば、パブリックまたはゲスト・ユーザー ID のディレクトリーを指定して、次に示すような FFI\_PATH を指定することができます。

```
FFI_PATH      C:¥public;E:¥WWW;E:¥guest;A:
```

次のリストは、フラット・ファイルをセキュアにするための推奨事項を示しています。

- フラット・ファイル操作に使用するのに適したディレクトリーを選択します。ディレクトリーへの検索を制限するために、それらのディレクトリーを FFI\_PATH に追加する必要があります。
- 他のユーザーがマクロ内で DTWF\_REMOVE またはその他のエクスポート操作を実施するときに、現行ディレクトリーに含まれている可能性のある、拡張子 .dll および .cmd の付いたファイルを除去または変更しないように、注意を徹底します。
- システムにどのようなマクロが追加されるのかを適切に制御して、ファイルを保護するための適切なステップを実施します。
- 匿名 FTP ユーザーがパスに書き込みを行うのを避けるために、 FFI\_PATH でパスを指定しないようにします。パスを指定すると、それまで許可されていなかったアクションを行えるようにする Net.Data マクロを、誰かがシステムに書き込む可能性があります。
- Net.Data 初期設定ファイルのパスを FFI\_PATH に追加しないようにします。

## 許可要件

Net.Data が実行されるユーザー ID が、 FFI 組み込み関数によって使用されるファイルへのアクセス権限を必ず持つようにします。詳しくは、 *Net.Data* 管理およびプログラミングの手引き の構成の章にある、 Net.Data ファイルへの Web サーバーのアクセス権限の指定に関するセクションを参照してください。

## フラット・ファイル・インターフェース区切り文字

パフォーマンスを上げるために、一連の SQL 要求から得られた Net.Data 表形式出力をフラット・ファイルに保管することができます。後続の要求で SQL 要求を再発行しなくても、このフラット・ファイルを取り出すことができます。

Net.Data フラット・ファイルは Net.Data 表から作成することができ、 Net.Data 表はフラット・ファイルから構築することができます。表とフラット・ファイルとの間の変換を行うには、表の列とフラット・ファイルのレコードとの間のマッピングを定義する必要があります。区切り文字 とは、要求された変形に従ってファイルをいくつかの部分 (たとえば、列や行) に分割するときに、 FFI が使用するフラグまたは区切り記号のことです。区切り文字は、どうすればフラット・ファイルのレコードの各部分を分離して表の列にマップすることができるか、またどのようにすれば表の列をフラット・ファイルのレコードにマップすることができるかを定義するための方法を提供します。

区切り文字には、次の 2 つの型があります。

### 改行文字 (ASCITEXT)

表が 1 つの列で構成されているときに、この変換を使用します。 Net.Data は、対応するフラット・ファイルの各レコードを表の単一行にマップします。この場合、フラット・ファイルのレコードを分離する通常の改行文字のみが、区切り文字として使用されます。

### 改行文字および区切り文字ストリング (DELIMITED)

表が複数の列で構成されているときに、この変換を使用します。 Net.Data は、表の行からフラット・ファイル・レコードを作成するとき、分離文字としての区切り文字ストリングを項目間に入れます。フラット・ファイルから表を再作成するとき、Net.Data は、区切り文字ストリングを使用して、各行の何文字を表の列に入れるかを決定します。この場合、通常の改行文字は、表の行に対応するフラット・ファイル・レコードを分離し、区切り文字ストリングは、単一レコード内の各項目を分離します。

読み取り操作の場合、区切り文字は、ファイルの内容を分割して、表の行と列に入るようにします。書き込み操作の場合、区切り文字は、表の行と列の値の終わりを示します。 Net.Data は、区切り文字を Net.Data マクロ・ストリングとして FFI に渡します。 DELIMITER パラメーターで明示的にリストされていない限り、複数の文字の終わりにヌル文字は含まれません。

ヌル文字を区切り文字内で使用するためには、 DELIMITER パラメーターで、 2 つの二重引用符 『""]』 を使用して空ストリングを指定する代わりに、二重引用符で囲まれた円記号とゼロ 『¥0』 を指定してください。 ASCITEXT 変形を指定した場合、 Net.Data は改行文字を区切り文字として使用し、要求された区切り文字を無視します。

書き込み操作と読み取り操作で異なる区切り文字を使用すると、ファイルに対して不適切な変更が行われる可能性があります。 `Net.Data` は、新しい区切り文字を使用してファイルを書き込みます。

区切り文字の最大長は 256 文字です。

## ファイルのロック

`DTWF_OPEN` および `DTWF_CLOSE` 関数を使用してフラット・ファイルをロックすることができます。これらの関数を使用することにより、`Net.Data` はフラット・ファイルを予約し、他のアプリケーションがこのファイルを読み取りまたは更新できないようにします。

ファイルをロックするには、`DTWF_OPEN` 関数を使用してください。この関数を使用すると、そのファイルが他のアプリケーションでは使用できなくなり、ファイルを読み取ってから更新するまでの間にそのファイルが変更されないようにすることができます。

ファイルを解放するには、`DTWF_CLOSE` 関数を使用してください。この関数はファイルを解放し、他のアプリケーションがファイルを読み取りまたは更新できるようにします。

## DTWF\_APPEND

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

### 目的

Net.Data 表の内容をテキスト・ファイルの終わりに書き込みます。

### 構文

@DTWF\_APPEND(filename, transform, delimiter, table, retry, rows)

@DTWF\_APPEND(filename, transform, delimiter, table, retry)

@DTWF\_APPEND(filename, transform, delimiter, table)

### 値

表 167. DTWF\_APPEND のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>filename</i>	INOUT	変数の内容を追加するファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を戻します。
ストリング	<i>transform</i>	IN	ファイルの形式。 <ul style="list-style-type: none"><li>• ASCIITEXT - 列値間に改行文字を入れて表をファイルに書き込み、<i>delimiter</i> パラメーターを無視します。</li><li>• DELIMITED - 区切り文字を <i>delimiter</i> パラメーターに指定して表をファイルに書き込みます。</li></ul> ASCIITEXT および DELIMITED 変形の場合、ファイル内の改行文字は、Net.Data マクロ表の行の終わりを示します。
ストリング	<i>delimiter</i>	IN	値の終わりを示す文字ストリング。このパラメーターでは、大文字小文字が区別されます。 <i>transform</i> が ASCIITEXT であれば、無視されます。
表	<i>table</i>	IN	レコードが読み取られた表変数。  <b>OS/400 以外のユーザー:</b> FFI 表の行の最大長は 16383 文字です。この長さには、Net.Data マクロ表の各列のヌル文字も含まれます。
整数	<i>retry</i>	IN	ファイルを即時に付加できない場合に再試行する回数。デフォルトでは、再試行されません。
整数	<i>rows</i>	IN	<i>table</i> から付加する行の最大数。デフォルトでは、すべての行が付加されます。0 を指定すると、すべての行が付加されます。



## 戻りコード

表 168. DTWF\_APPEND の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
2000	フラット・ファイル・インターフェース組み込み関数が指定ファイルを検出できませんでした。
2001	フラット・ファイル・インターフェース組み込み関数が指定ファイルをオープンできませんでした。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。
2003	フラット・ファイル・インターフェース組み込み関数が、データ行を表変数に読み込めません。行内のバイト数が、サポートされる最大バイト数を超えています。
2004	フラット・ファイル・インターフェース組み込み関数がファイルを検出しようとしたますが、FFI_PATH 構成ファイル変数にあるパスが、サポートされる最大バイト数 (4095) を超えています。
2005	システム機能の呼び出しが失敗しました。
2006	フラット・ファイル・インターフェース組み込み関数が指定ファイルにアクセスできません。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。

## 使用上の注意

ファイルの現行内容 (特に、最終行の最終列の内容) は、DTWF\_APPEND の使用結果に影響を与えます。ファイルの最終行の最終列の値の後に改行文字が続いている場合、追加されたデータは新規行に入ります。それ以外の場合には、追加されたデータはファイルの最終行の一部になります。

## 例

### 例 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
}%  
@DTWF_APPEND(myFile, "DELIMITED", " ;", myTable)
```

**例 2:**

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
%}  
@DTWF_APPEND(myFile, "ASCII TEXT", " ;", myTable)
```

**例 3:**

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
%}  
@DTWF_APPEND(myFile, "ASCII TEXT", " ;", myTable, "0", "10")
```

# DTWF\_CLOSE

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

## 目的

DTWF\_OPEN によってオープンされたファイルをクローズします。

## 構文

@DTWF\_CLOSE(filename, retry)

@DTWF\_CLOSE(filename)

## 値

表 169. DTWF\_CLOSE のパラメーター

データ型	パラメーター	用途	説明
ストリング	filename	INOUT	クローズするファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を戻します。
整数	retry	IN	ファイルを即時にクローズできない場合に再試行する回数。デフォルトでは、再試行されません。

## 戻りコード

表 170. DTWF\_CLOSE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててゐるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
2002	フラット・ファイル・インターフェース組み込み関数が指定ファイルをクローズできません。このファイルはマクロ起動時にオープンされていません。

## 例

例 1:

@DTWF\_CLOSE(myFile, "5")

## DTWF\_DELETE

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

### 目的

テキスト・ファイルから行を削除します。

### 構文

@DTWF\_DELETE(filename, transform, delimiter, retry, rows, startrow)

@DTWF\_DELETE(filename, transform, delimiter, retry, rows)

@DTWF\_DELETE(filename, transform, delimiter, retry)

@DTWF\_DELETE(filename, transform, delimiter)

### 値

表 171. DTW\_DELETE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>filename</i>	INOUT	レコードが削除されるファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を戻します。
ストリング	<i>transform</i>	IN	ファイルの形式。 <ul style="list-style-type: none"><li>• ASCIITEXT - 列値間に改行文字を入れて表をファイルに書き込み、<i>delimiter</i> パラメーターを無視します。</li><li>• DELIMITED - 区切り文字を <i>delimiter</i> パラメーターに指定して表をファイルに書き込みます。</li></ul> ASCIITEXT および DELIMITED 変形の場合、ファイル内の改行文字は、Net.Data マクロ表の行の終わりを示します。
ストリング	<i>delimiter</i>	IN	値の終わりを示す文字ストリング。このパラメーターでは、大文字小文字が区別されます。 <i>transform</i> が ASCIITEXT であれば、無視されます。
整数	<i>retry</i>	IN	ファイルを即時に削除できない場合に再試行する回数。デフォルトでは、再試行されません。
整数	<i>rows</i>	IN	削除する行の最大数。デフォルトでは、すべての行が削除されます。0 を指定すると、すべての行が削除されます。
整数	<i>startrow</i>	INOUT	削除を開始する行番号。1 の値は、最初の行から削除を開始することを意味します。この値がファイルの行数よりも大きい場合は、この値は最終レコードに変更され、エラーが戻されます。デフォルトでは、1 から開始されます。

## 戻りコード

表 172. DTWF\_DELETE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
2000	フラット・ファイル・インターフェース組み込み関数が指定ファイルを検出できませんでした。
2001	フラット・ファイル・インターフェース組み込み関数が指定ファイルをオープンできませんでした。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。
2003	フラット・ファイル・インターフェース組み込み関数が、データ行を表変数に読み込めません。行内のバイト数が、サポートされる最大バイト数を越えています。
2004	フラット・ファイル・インターフェース組み込み関数がファイルを検出しようとしたましたが、 FFI_PATH 構成ファイル変数にあるパスが、サポートされる最大バイト数 (4095) を越えています。
2005	システム機能の呼び出しが失敗しました。
2006	フラット・ファイル・インターフェース組み込み関数が指定ファイルにアクセスできません。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。

## 例

### 例 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "5000"  
    myRows = "2"  
}%  
@DTWF_DELETE(myFile, "Delimited", "|", myWait, myRows)
```

### 例 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	SCO	SUN	Win NT
X			X	X	X		X	X

### 目的

テキスト・ファイルに行を挿入します。

### 構文

@DTWF\_INSERT(filename, transform, delimiter, table, retry, rows, startrow)

@DTWF\_INSERT(filename, transform, delimiter, table, retry, rows)

@DTWF\_INSERT(filename, transform, delimiter, table, retry)

@DTWF\_INSERT(filename, transform, delimiter, table)

### 値

表 173. DTWF\_INSERT のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>filename</i>	INOUT	レコードが挿入されるファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を戻します。
ストリング	<i>transform</i>	IN	ファイルの形式。 <ul style="list-style-type: none"><li>• ASCIITEXT - 列値間に改行文字を入れて表をファイルに書き込み、<i>delimiter</i> パラメーターを無視します。</li><li>• DELIMITED - 区切り文字を <i>delimiter</i> パラメーターに指定して表をファイルに書き込みます。</li></ul> ASCIITEXT および DELIMITED 変形の場合、ファイル内の改行文字は、Net.Data マクロ表の行の終わりを示します。
ストリング	<i>delimiter</i>	IN	値の終わりを示す文字ストリング。このパラメーターでは、大文字小文字が区別されます。 <i>transform</i> が ASCIITEXT であれば、無視されます。
表	<i>table</i>	IN	ファイルに挿入するレコードが含まれている表変数。  <b>OS/400 以外のユーザー:</b> FFI 表の行の最大長は 16383 文字です。この長さには、Net.Data マクロ表の各列のヌル文字も含まれます。
整数	<i>retry</i>	IN	ファイルを即時に書き込めない場合に再試行する回数。デフォルトでは、再試行されません。

表 173. DTWF\_INSERT のパラメーター (続き)

データ型	パラメーター	用途	説明
整数	<i>rows</i>	IN	<i>table</i> から挿入する行の最大数。デフォルトでは、すべての行が挿入されます。0 の値は、すべての行を挿入します。
整数	<i>startrow</i>	INOUT	挿入を開始する行番号。値がファイルの行数よりも大きい場合は、この値は最終レコードに変更され、エラーが戻されます。0 を指定すると、ファイルの先頭の後ろに挿入されます。デフォルトでは、1 から開始されます。

## 戻りコード

表 174. DTWF\_INSERT の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
2000	フラット・ファイル・インターフェース組み込み関数が指定ファイルを検出できませんでした。
2001	フラット・ファイル・インターフェース組み込み関数が指定ファイルをオープンできませんでした。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。
2003	フラット・ファイル・インターフェース組み込み関数が、データ行を表変数に読み込めません。行内のバイト数が、サポートされる最大バイト数を超えています。
2004	フラット・ファイル・インターフェース組み込み関数がファイルを検出しようとしたますが、FFI_PATH 構成ファイル変数にあるパスが、サポートされる最大バイト数 (4095) を超えています。
2005	システム機能の呼び出しが失敗しました。
2006	フラット・ファイル・インターフェース組み込み関数が指定ファイルにアクセスできません。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。

## 例

### 例 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "3000"  
}%  
@DTWF_INSERT(myFile, "Delimited", "|", myTable, myWait)
```

### 例 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	SCO	SUN	Win NT
X			X	X	X		X	X

## 目的

テキスト・ファイルをオープンします。

## 構文

@DTWF\_OPEN(filename, mode, retry)

@DTWF\_OPEN(filename, mode)

## 値

表 175. DTWF\_OPEN のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>filename</i>	INOUT	オープンするファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を戻します。
ストリング	<i>mode</i>	IN	要求されるアクセスの型。 <ul style="list-style-type: none"><li>• r - 読み取りのための既存のファイルをオープンします。</li><li>• w - 書き込みのためのファイルを作成します。(同じ名前の既存のファイルが存在する場合は、それを破棄します。)</li><li>• a - 付加のためのファイルをオープンします。そのファイルが見付からないと、Net.Data はそれを作成します。</li><li>• r+ - 読み取りと書き込みのための既存のファイルをオープンします。</li><li>• w+ - 読み取りと書き込みのための既存のファイルを作成します。(同じ名前の既存のファイルが存在する場合は、それを破棄します。)</li><li>• a+ - 読み取りと付加のためのファイルを付加モードでオープンします。そのファイルが見付からないと、Net.Data はそれを作成します。</li></ul>
整数	<i>retry</i>	IN	ファイルを即時にオープンできない場合に再試行する回数。デフォルトでは、再試行されません。

## 戻りコード

表 176. DTWF\_OPEN の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。

表 176. DTWF\_OPEN の戻りコード (続き)

戻りコード	説明
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
2000	フラット・ファイル・インターフェース組み込み関数が指定ファイルを検出できませんでした。
2001	フラット・ファイル・インターフェース組み込み関数が指定ファイルをオープンできませんでした。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。
2006	フラット・ファイル・インターフェース組み込み関数が指定ファイルにアクセスできません。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。

## 使用上の注意

1. ファイルが存在しない場合には、ファイル名として絶対パスを指定する必要があります。そのファイルが作成されるディレクトリーは、 FFI\_PATH で指定されたディレクトリーと一致しなければなりません。絶対パスを使用しないと、そのファイルは現行作業ディレクトリーにオープンされます。
2. DTWF\_OPEN はファイルをオープン状態にします。オープン状態が解けると、ファイルは各フラット・ファイル操作の後でクローズされます。
3. DTWF\_OPEN を使用して、ファイルをオープンする回数を減らしてください。DTWF\_OPEN を使用しないと、ファイルは各フラット・ファイル操作の後でクローズされます。ファイルは、DTWF\_CLOSE でクローズされるまで、またはマクロ処理が終了するまでオープン状態になっています。

## 例

### 例 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	SCO	SUN	Win NT
X			X	X	X		X	X

### 目的

テキスト・ファイルの行を Net.Data 表に取り込みます。

### 構文

@DTWF\_READ(filename, transform, delimiter, table, retry, rows, startrow, columns)

@DTWF\_READ(filename, transform, delimiter, table, retry, rows, startrow)

@DTWF\_READ(filename, transform, delimiter, table, retry, rows)

@DTWF\_READ(filename, transform, delimiter, table, retry)

@DTWF\_READ(filename, transform, delimiter, table)

### 値

表 177. DTWF\_READ のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>filename</i>	INOUT	レコードが表変数に読み取られるファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を返します。
ストリング	<i>transform</i>	IN	ファイルの形式。 <ul style="list-style-type: none"><li>• ASCIITEXT - 列値間に改行文字を入れて表をファイルに書き込み、<i>delimiter</i> パラメーターを無視します。</li><li>• DELIMITED - 区切り文字を <i>delimiter</i> パラメーターに指定して表をファイルに書き込みます。</li></ul> ASCIITEXT および DELIMITED 変形の場合、ファイル内の改行文字は、Net.Data マクロ表の行の終わりを示します。
ストリング	<i>delimiter</i>	IN	値の終わりを示す文字ストリング。このパラメーターでは、大文字小文字が区別されます。 <i>transform</i> が ASCIITEXT であれば、無視されます。
表	<i>table</i>	OUT	ファイル・レコードが読み取られる表変数。  <b>OS/400 以外のユーザー:</b> FFI 表の行の最大長は 16383 文字です。この長さには、Net.Data マクロ表の各列のヌル文字も含まれます。
整数	<i>retry</i>	IN	ファイルを即時に読み取れない場合に再試行する回数。デフォルトでは、再試行されません。

表 177. DTWF\_READ のパラメーター (続き)

データ型	パラメーター	用途	説明
整数	<i>rows</i>	INOUT	表へ読み取るファイル・レコードの最大数。デフォルトでは、すべてのレコードが読み取られるか、あるいは表がいっぱいになるまで読み取られます。0 の値は、ファイルの終わりまで読み取ることを意味します。結果表の行数が戻されます。
整数	<i>startrow</i>	IN	読み取りが開始されるファイル・レコード。デフォルトでは、最初のレコードから読み取りが開始されます。
整数	<i>columns</i>	OUT	表内の列数を戻します。

## 戻りコード

表 178. DTWF\_READ の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
1010	データは表がいっぱいになるまで書き込まれ、余分なデータは廃棄されます。
2000	フラット・ファイル・インターフェース組み込み関数が指定ファイルを検出できませんでした。
2001	フラット・ファイル・インターフェース組み込み関数が指定ファイルをオープンできませんでした。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。
2003	フラット・ファイル・インターフェース組み込み関数が、データ行を表変数に読み込めません。行内のバイト数が、サポートされる最大バイト数を超えています。
2004	フラット・ファイル・インターフェース組み込み関数がファイルを検出しようとしたますが、FFI_PATH 構成ファイル変数にあるパスが、サポートされる最大バイト数 (4095) を超えています。
2005	システム機能の呼び出しが失敗しました。

表 178. DTWF\_READ の戻りコード (続き)

戻りコード	説明
2006	フラット・ファイル・インターフェース組み込み関数が指定ファイルにアクセスできません。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。

## 例

### 例 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "1000"
}%
@DTWF_READ(myFile, "DELIMITED", ";", myTable, myWait)
```

### 例 2:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "0"
    myRows = "0"
    myStartrow = "1"
    myColumns = ""
}%
@DTWF_READ(myFile, "DELIMITED", ";", myTable, myWait, myRows,
            myStartrow, myColumns)
```

### 例 3:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_READ(myFile, "ASCIITEXT", ";", myTable)
@DTW_TB_TABLE(myTable, "BORDER", "")
```

## DTWF\_REMOVE

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

### 目的

ファイル全体を削除します。

### 構文

@DTWF\_REMOVE(filename, retry)

@DTWF\_REMOVE(filename)

### 値

表 179. DTWF\_REMOVE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>filename</i>	INOUT	削除ファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を戻します。
整数	<i>retry</i>	IN	ファイルを即時に削除できない場合に再試行する回数。デフォルトでは、再試行されません。

### 戻りコード

表 180. DTWF\_REMOVE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
2000	フラット・ファイル・インターフェース組み込み関数が指定ファイルを検出できませんでした。
2006	フラット・ファイル・インターフェース組み込み関数が指定ファイルにアクセスできません。このファイルは該または他のプロセスにより使用中で、指定モードでは共用できません。

### 例

例 1:

```
%DEFINE myFile = "c:/private/myfile"  
@DTWF_REMOVE(myFile)
```

**例 2:**

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myWait = "2000"  
%}  
@DTWF_REMOVE(myFile, myWait)
```

## DTWF\_SEARCH

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

### 目的

テキスト・ファイルのストリングを検索し、結果を Net.Data 表に戻します。

### 構文

@DTWF\_SEARCH(filename, transform, delimiter, table, searchFor, retry, rows, startrow)

@DTWF\_SEARCH(filename, transform, delimiter, table, searchFor, retry, rows)

@DTWF\_SEARCH(filename, transform, delimiter, table, searchFor, retry)

@DTWF\_SEARCH(filename, transform, delimiter, table, searchFor)

### 値

表 181. DTWF\_SEARCH のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>filename</i>	INOUT	検索ファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を返します。
ストリング	<i>transform</i>	IN	ファイルの形式。 <ul style="list-style-type: none"><li>• ASCIITEXT - 列値間に改行文字を入れて表をファイルに書き込み、<i>delimiter</i> パラメーターを無視します。</li><li>• DELIMITED - 区切り文字を <i>delimiter</i> パラメーターに指定して表をファイルに書き込みます。</li></ul> ASCIITEXT および DELIMITED 変形の場合、ファイル内の改行文字は、Net.Data マクロ表の行の終わりを示します。
ストリング	<i>delimiter</i>	IN	値の終わりを示す文字ストリング。このパラメーターでは、大文字小文字が区別されます。 <i>transform</i> が ASCIITEXT であれば、無視されます。
表	<i>table</i>	OUT	検索結果を入れる表変数。 <i>transform</i> が DELIMITED であれば、次の 3 つの列が戻されます。 <ul style="list-style-type: none"><li>• 一致が検出されなかった列。</li><li>• 一致が検出された列。</li><li>• ファイルで一致した列。</li></ul> <b>OS/400 以外のユーザー:</b> FFI 表の行の最大長は 16383 文字です。この長さには、Net.Data マクロ表の各列のヌル文字も含まれます。
ストリング	<i>searchFor</i>	IN	検索対象の文字ストリング。



表 181. DTWF\_SEARCH のパラメーター (続き)

データ型	パラメーター	用途	説明
整数	<i>retry</i>	IN	ファイルを即時に検索できない場合に再試行する回数。デフォルトでは、再試行されません。
整数	<i>rows</i>	INOUT	<i>table</i> に読み取る行の最大数。デフォルトでは、すべての行が読み取られるか、または <i>table</i> がいっぱいになるまで読み取られます。0 を指定すると、ファイルの終わりまで読み取られます。このパラメーターによって結果表の行数が戻されます。
整数	<i>startrow</i>	IN	検索が開始されるファイル・レコード。デフォルトは 1 です。つまり、検索は最初のレコードから開始されます。

## 戻りコード

表 182. DTWF\_SEARCH の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
1010	データは表がいっぱいになるまで書き込まれ、余分なデータは廃棄されます。
2000	フラット・ファイル・インターフェース組み込み関数が指定ファイルを検出できませんでした。
2001	フラット・ファイル・インターフェース組み込み関数が指定ファイルをオープンできませんでした。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。
2003	フラット・ファイル・インターフェース組み込み関数が、データ行を表変数に読み込めません。行内のバイト数が、サポートされる最大バイト数を超えています。
2004	フラット・ファイル・インターフェース組み込み関数がファイルを検出しようとしたますが、FFL_PATH 構成ファイル変数にあるパスが、サポートされる最大バイト数 (4095) を超えています。
2005	システム機能の呼び出しが失敗しました。

表 182. DTWF\_SEARCH の戻りコード (続き)

戻りコード	説明
2006	フラット・ファイル・インターフェース組み込み関数が指定ファイルにアクセスできません。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。

## 使用上の注意

1. DTWF\_SEARCH で戻される表には、3 つの列が含まれます。最初の 2 列には、一致した行と列の番号が入り、最後の列には、*SearchFor* パラメーターで指定された文字を含む列値が入ります。たとえば、ファイルの 4 番目の行の列 3 に、一致する文字が含まれている場合、戻される表の行の最初の列には番号 4 が入り、それらの文字があったファイルの行を示します。2 番目の列には番号 3 が入り、一致する文字がファイルのどの列に含まれているのかを示します。そして、3 番目の列には列値全体が入ります。
2. *SearchFor* パラメーターには、*delimiter* パラメーターの内容を含めることはできません。

## 例

### 例 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "1000"
    mySearch = "0123456789abcdef"
@DTWF_SEARCH(myFile, "DELIMITED", ";",
    myTable, mySearch, myWait)
```

### 例 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	SCO	SUN	Win NT
X			X	X	X		X	X

### 目的

テキスト・ファイルの行を、Net.Data 表のデータで更新します。

### 構文

@DTWF\_UPDATE(filename, transform, delimiter, table, retry, rows, startrow)

@DTWF\_UPDATE(filename, transform, delimiter, table, retry, rows)

@DTWF\_UPDATE(filename, transform, delimiter, table, retry)

@DTWF\_UPDATE(filename, transform, delimiter, table)

### 値

表 183. DTWF\_UPDATE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>filename</i>	INOUT	レコードが表変数から更新されるファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を戻します。
ストリング	<i>transform</i>	IN	ファイルの形式。 <ul style="list-style-type: none"><li>• ASCII TEXT - 列値間に改行文字を入れて表をファイルに書き込み、<i>delimiter</i> パラメーターを無視します。</li><li>• DELIMITED - 区切り文字を <i>delimiter</i> パラメーターに指定して表をファイルに書き込みます。</li></ul> ASCII TEXT および DELIMITED 変形の場合、ファイル内の改行文字は、Net.Data マクロ表の行の終わりを示します。
ストリング	<i>delimiter</i>	IN	値の終わりを示す文字ストリング。このパラメーターでは、大文字小文字が区別されます。 <i>transform</i> が ASCII TEXT であれば、無視されます。
表	<i>table</i>	IN	ファイル・レコードが更新される表変数。  <b>OS/400 以外のユーザー:</b> FFI 表の行の最大長は 16383 文字です。この長さには、Net.Data マクロ表の各列のヌル文字も含まれます。
整数	<i>retry</i>	IN	ファイルを即時に書き込めない場合に再試行する回数。デフォルトでは、再試行されません。

表 183. DTWF\_UPDATE のパラメーター (続き)

データ型	パラメーター	用途	説明
整数	<i>rows</i>	IN	<i>table</i> から更新されるレコードの最大数。デフォルトでは、すべてのレコードが更新されます。0 の値は、ファイルのすべての行を更新することを意味します。
整数	<i>startrow</i>	INOUT	更新する最初のファイル・レコード。デフォルトは 1 です。つまり、更新はファイルの先頭から開始されます。この値がファイルのレコード数よりも大きい場合は、この値はファイルの最終レコードの番号を示すように変更され、エラーが戻されます。

## 戻りコード

表 184. DTWF\_UPDATE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
2000	フラット・ファイル・インターフェース組み込み関数が指定ファイルを検出できませんでした。
2001	フラット・ファイル・インターフェース組み込み関数が指定ファイルをオープンできませんでした。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。
2003	フラット・ファイル・インターフェース組み込み関数が、データ行を表変数に読み込めません。行内のバイト数が、サポートされる最大バイト数を越えています。
2004	フラット・ファイル・インターフェース組み込み関数がファイルを検出しようとしたますが、FFI_PATH 構成ファイル変数にあるパスが、サポートされる最大バイト数 (4095) を越えています。
2005	システム機能の呼び出しが失敗しました。
2006	フラット・ファイル・インターフェース組み込み関数が指定ファイルにアクセスできません。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。

## 使用上の注意

ファイルが存在しない場合には、ファイル名として絶対パスを指定する必要があります、そのファイルが作成されるディレクトリーは、 `FFI_PATH` で指定されたディレクトリーと一致しなければなりません。絶対パスを使用しないと、そのファイルは現行作業ディレクトリーにオープンされます。

### 例

#### 例 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
    myWait = "1500"  
    myRows = "2"  
}%  
@DTWF_UPDATE(myFile, "Delimited", "|", myTable, myWait, myRows)
```

#### 例 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	SCO	SUN	Win NT
X			X	X	X		X	X

### 目的

Net.Data 表の内容をテキスト・ファイルに書き込みます。

### 構文

@DTWF\_WRITE(filename, transform, delimiter, table, retry, rows, startrow)

@DTWF\_WRITE(filename, transform, delimiter, table, retry, rows)

@DTWF\_WRITE(filename, transform, delimiter, table, retry)

@DTWF\_WRITE(filename, transform, delimiter, table)

### 値

表 185. DTWF\_WRITE のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>filename</i>	INOUT	表変数のレコードが書き込まれるファイルの名前。呼び出しが正常終了すると、このパラメーターは完全修飾ファイル名を戻します。
ストリング	<i>transform</i>	IN	ファイルの形式。 <ul style="list-style-type: none"><li>• ASCIITEXT - 列値間に改行文字を入れて表をファイルに書き込み、<i>delimiter</i> パラメーターを無視します。</li><li>• DELIMITED - 区切り文字を <i>delimiter</i> パラメーターに指定して表をファイルに書き込みます。</li></ul> ASCIITEXT および DELIMITED 変形の場合、ファイル内の改行文字は、Net.Data マクロ表の行の終わりを示します。
ストリング	<i>delimiter</i>	IN	値の終わりを示す文字ストリング。このパラメーターでは、大文字小文字が区別されます。 <i>transform</i> が ASCIITEXT であれば、無視されます。
表	<i>table</i>	IN	行をファイルにエクスポートするために使用する表変数。  <b>OS/400 以外のユーザー:</b> FFI 表の行の最大長は 16383 文字です。この長さには、Net.Data マクロ表の各列のヌル文字も含まれます。
整数	<i>retry</i>	IN	ファイルを即時に書き込めない場合に再試行する回数。デフォルトでは、再試行は行われません。

表 185. DTWF\_WRITE のパラメーター (続き)

データ型	パラメーター	用途	説明
整数	<i>rows</i>	IN	書き込むファイル・レコードの最大数。デフォルトでは、表全体が書き込まれます。0 の値は、すべてのレコードをファイルの終わりまで書き込むことを意味します。
整数	<i>startrow</i>	INOUT	書き込みを開始する、ファイルのレコード番号。デフォルトは 1 です。つまり、最初のレコードから開始されます。ファイルの終わりを越えた値が指定されている場合は、ファイルの最終行が戻され、エラーが示されます。

## 戻りコード

表 186. DTWF\_WRITE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
2000	フラット・ファイル・インターフェース組み込み関数が指定ファイルを検出できませんでした。
2001	フラット・ファイル・インターフェース組み込み関数が指定ファイルをオープンできませんでした。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。
2003	フラット・ファイル・インターフェース組み込み関数が、データ行を表変数に読み込めません。行内のバイト数が、サポートされる最大バイト数を越えています。
2004	フラット・ファイル・インターフェース組み込み関数がファイルを検出しようとしたますが、FFI_PATH 構成ファイル変数にあるパスが、サポートされる最大バイト数 (4095) を越えています。
2005	システム機能の呼び出しが失敗しました。
2006	フラット・ファイル・インターフェース組み込み関数が指定ファイルにアクセスできません。このファイルは該当または他のプロセスにより使用中で、指定モードでは共用できません。

## 使用上の注意

ファイルが存在しない場合には、ファイル名として絶対パスを指定する必要があります、そのファイルが作成されるディレクトリーは、 `FFI_PATH` で指定されたディレクトリーと一致しなければなりません。絶対パスを使用しないと、そのファイルは現行作業ディレクトリーにオープンされます。

### 例

#### 例 1:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
%}  
@DTWF_WRITE(myFile, "DELIMITED", ";", myTable)
```

#### 例 2:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
%}  
@DTWF_WRITE(myFile, "ASCIITEXT", ";", myTable, "5000")
```

#### 例 3:

```
%DEFINE {  
    myFile = "c:/private/myfile"  
    myTable = %TABLE  
%}  
@DTWF_WRITE(myFile, "ASCIITEXT", ";", myTable, "5000", "10", "50")
```



---

## Web レジストリー関数

Web レジストリーは、Net.Data によって保守されるキーをもつファイルで、ユーザーはこれを使用して、項目の追加、検索、削除などを容易に行うことができます。単一のシステムで複数の Net.Data Web レジストリーを作成することができます。各レジストリーは名前をもっています。各レジストリーには複数の項目を含めることができます。Net.Data は、レジストリーおよびそれに含まれる項目を保守するための関数を提供します。

- 300ページの『DTWR\_ADDENTRY』
- 302ページの『DTWR\_CLEARREG』
- 303ページの『DTWR\_CLOSEREG』
- 304ページの『DTWR\_CREATEREG』
- 306ページの『DTWR\_DELENTY』
- 308ページの『DTWR\_DELREG』
- 309ページの『DTWR\_LISTREG』
- 311ページの『DTWR\_LISTSUB』
- 313ページの『DTWR\_OPENREG』
- 315ページの『DTWR\_RTVENTRY』
- 317ページの『DTWR\_UPDATEENTRY』

### 制約事項:

- OS/2 を使用するときは、*registry*、*registryVariable*、および *registryData* パラメーターにアスタリスク (\*) を使用しないでください。
- Web レジストリー関数に渡されるパラメーターは、それぞれ最大 2048 文字に制限されます。

## DTWR\_ADDENTRY

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

### 目的

項目を Web レジストリーに追加します。

### 構文

@DTWR\_ADDENTRY(registry, registryVariable, registryData, index)

@DTWR\_ADDENTRY(registry, registryVariable, registryData)

### 値

表 187. DTWR\_ADDENTRY のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	項目追加先のレジストリーの名前。
ストリング	registryVariable	IN	追加するレジストリー項目の <i>registryVariable</i> ストリング部分の値。
ストリング	registryData	IN	追加するレジストリー項目の <i>registryData</i> ストリング部分の値。
ストリング	index	IN	追加する索引付き項目の <i>registryVariable</i> ストリングの索引部分の値。このパラメーターはオプションです。これを指定すると、索引付き項目が指定レジストリーに追加されます。

### 戻りコード

表 188. DTWR\_ADDENTRY の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
3003	Web レジストリー組み込み関数が、指定したレジストリーに項目を追加できません。指定した項目は既に存在します。
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。
3006	Web レジストリー組み込み関数が、指定したレジストリーを作成できません。レジストリー名にあるパスが存在しません。

表 188. DTWR\_ADDENTRY の戻りコード (続き)

戻りコード	説明
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。

## 例

### 例 1:

```
@DTWR_ADDENTRY("Myregistry", "Jones", "http://Advantis.com/~Jones/webproj")
```

### 例 2:

```
@DTWR_ADDENTRY("URLLIST", "SMITH", "http://www.software.ibm.com/",  
"WORK_URL,")
```

## DTWR\_CLEARREG

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

### 目的

Web レジストリーから項目を消去します。

### 構文

```
@DTWR_CLEARREG(registry)
```

### 値

表 189. DTWR\_CLEARREG のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	消去するレジストリーの名前。

### 戻りコード

表 190. DTWR\_CLEARREG の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。
3006	Web レジストリー組み込み関数が、指定したレジストリーを作成できません。レジストリー名にあるパスが存在しません。
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。

### 例

#### 例 1:

```
@DTWR_CLEARREG("Myregistry")
```

## DTWR\_CLOSEREG

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

### 目的

Web レジストリーをクローズします。

### 構文

@DTWR\_CLOSEREG(registry)

### 値

表 191. DTWR\_CLOSEREG のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	クローズするレジストリーの名前。  制約事項：Web レジストリー名にアスタリスク (*) および円記号 (¥) などの特殊文字を使用してはなりません。

### 戻りコード

表 192. DTWR\_CLOSEREG の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。

### 例

例 1: レジストリーをクローズします。

```
@DTWR_CLOSEREG("/qsys.lib/mylib.lib/myreg.file")
```

## DTWR\_CREATEREG

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

### 目的

新規 Web レジストリーを作成します。

### 構文

@DTWR\_CREATEREG(registry, security)

@DTWR\_CREATEREG(registry)

### 値

表 193. DTWR\_CREATEREG のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	作成するレジストリーの名前。  制約事項：Web レジストリー名にアスタリスク (*) および円記号 (¥) などの特殊文字を使用してはなりません。
ストリング	security	IN	registry を作成する際の機密保護の型。 UNIX オペレーティング・システムの場合は、デフォルトの機密保護は、レジストリーが作成されるディレクトリーと同じです。3 つの機密保護グループ (ユーザー、グループ、およびパブリック) について機密保護を指定します。R は読み取り許可、W は書き込み許可、X は実行許可を与えます。たとえば、これらの 3 つのグループすべてに全権限を与えるには、このパラメーターに *RWX, *RWX, *RWX を指定します。このパラメーターはオプションです。

### 戻りコード

表 194. DTWR\_CREATEREG の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を越えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。

表 194. DTWR\_CREATEREG の戻りコード (続き)

戻りコード	説明
3002	Web レジストリー組み込み関数が、指定したレジストリーを削除できません。
3006	Web レジストリー組み込み関数が、指定したレジストリーを作成できません。レジストリー名にあるパスが存在しません。
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。
3008	Web レジストリー組み込み関数が、指定したレジストリーを作成できません。原因は不明です。

## 例

### 例 1:

```
@DTWR_CREATEREG("myRegistry")
```

### 例 2:

```
@DTWR_CREATEREG("URLLIST", "*RWX, *RWX, *R")
```

## DTWR\_DELENTY

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

### 目的

項目を Web レジストリーから削除します。

### 構文

@DTWR\_DELENTY(registry, registryVariable, index)

@DTWR\_DELENTY(registry, registryVariable)

### 値

表 195. DTWR\_DELENTY のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	項目が削除されるレジストリーの名前。
ストリング	registryVariable	IN	除去する項目の registryVariable ストリング部分の値。
ストリング	index	IN	索引付き項目の registryVariable ストリングの索引部分の値。これはオプション・パラメーターです。これを指定すると、索引付き項目がレジストリーから除去されます。

### 戻りコード

表 196. DTWF\_DELENTY の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
3004	Web レジストリー組み込み関数が、指定したレジストリーから項目を削除または検索することができません。指定した項目が存在しません。
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。



## 例

### 例 1:

```
@DTWR_DELENTY("Myregistry", "Jones")
```

### 例 2:

```
@DTWR_DELENTY("URLLIST", "SMITH", "WORK_URL")
```

## DTWR\_DELREG

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

### 目的

Web レジストリーを削除します。

### 構文

@DTWR\_DELREG(registry)

### 値

表 197. DTWR\_DELREG のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	削除するレジストリーの名前。

### 戻りコード

表 198. DTWR\_DELREG の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。

### 例

#### 例 1:

```
@DTWR_DELREG("Myregistry")
```

## DTWR\_LISTREG

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

### 目的

Web レジストリーの内容をリストします。

### 構文

@DTWR\_LISTREG(registry, registryTable)

### 値

表 199. DTWR\_LISTREG のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	リストするレジストリーの名前。
表	registryTable	OUT	レジストリー項目を入れる表変数の名前。

### 戻りコード

表 200. DTWR\_LISTREG の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1004	関数呼び出しで渡されたパラメーター (Net.Data マクロ表変数とするために必須) が異なる変数型です。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。

### 使用上の注意

DTWR\_LISTREG は、ユーザーが渡した OUT 表変数のレジストリー項目に関する情報を戻します。表変数は、ユーザー・マクロに定義された後で、LISTREG レジストリー操作のために FUNCTION ブロックにパラメーターとして渡されます。

ユーザーが表の最大行数について ALL オプションを使用して表変数を定義すると、この操作では、表内の使用可能なすべてのレジストリー項目が、表の各行ごとに 1 つずつリストされます。一方、ユーザーが表の最大行数について X の値を指定した場合に、指定されたレジストリーに X よりも多くの項目が含まれている場合は、最初の X 項目のみがリストされ、エラー・コードが戻され、追加の項目をリストできるだけの十分な数の表行がなかったために部分リストしか作成できなかったことが示されます。X の値が指定レジストリーの使用可能項目数よりも大きい場合は、すべてのレジストリー項目がリストされます。

表には、常に、2 つの列があります。表の列見出しは、REGISTRY\_VARIABLE および REGISTRY\_DATA に設定されます。

## 例

### 例 1:

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

@DTWR_LISTREG("URLLIST", RegistryTable)
```

## DTWR\_LISTSUB

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

### 目的

Web レジストリーの即時サブキー項目をリストします。

### 構文

@DTWR\_LISTSUB(registry, registryTable)

### 値

表 201. DTWR\_LISTSUB のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	リストするレジストリーの名前。
表	registryTable	OUT	レジストリー項目を入れる表変数の名前。

### 戻りコード

表 202. DTWR\_LISTSUB の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。

### 使用上の注意

- DTWR\_LISTSUB は、ユーザーが渡した OUT 表パラメーターのレジストリー項目に関する情報を戻します。表変数は、マクロに定義されてから、パラメーターとして LISTSUB レジストリー操作に渡されます。

ユーザーが表の最大行数について ALL オプションを使用して表変数を定義すると、この操作では、表内の使用可能なすべてのレジストリー項目が、表の各行ごとに 1 つずつリストされます。一方、ユーザーが表の最大行数について X の値を指定した場合に、指定されたレジストリーに X よりも多くの項目が含まれてい

る場合は、最初の X 項目のみがリストされ、エラー・コードが戻され、追加の項目をリストできるだけの十分な数の表行がなかったために、部分リストしか作成できなかったことが示されます。X の値が指定レジストリーの使用可能項目数よりも大きい場合は、すべてのレジストリー項目がリストされます。表内の列の数は、常に 1 です。

表の列ヘッダーは、"REGISTRY\_SUBKEY" に設定されます。

2. この関数は、Windows 95 システム登録と互換性のあるオペレーティング・システムでのみ有効です。

## 例

### 例 1:

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

@DTWR_LISTSUB("URLLIST", RegistryTable)
```

## DTWR\_OPENREG

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

### 目的

Web レジストリーをオープンします。

### 構文

@DTWR\_OPENREG(registry, commit)

@DTWR\_OPENREG(registry)

### 値

表 203. DTWR\_OPENREG のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	オープンするレジストリーの名前。
ストリング	commit	IN	そのレジストリーがコミットメント制御のもとでオープンされるのかどうかを指定する、単一のシンボルまたはリテラル・ストリング。指定できる値は、次のとおりです。  <b>Y</b> コミットメント制御されているレジストリーをオープンします。  <b>N</b> コミットメント制御されているレジストリーをオープンしません。 デフォルトは N です。

### 戻りコード

表 204. DTWR\_OPENREG の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。

## 例

例 1: コミットメント制御のもとでレジストリーをオープンします。

```
@DTWR_OPENREG("/qsys.lib/mylib.lib/myreg.file", "Y")
```



## DTWR\_RTVENTRY

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

### 目的

Web レジストリーからレジストリー・ストリング値を検索します。

### 構文

@DTWR\_RTVENTRY(registry, registryVariable, registryData, index)

@DTWR\_RTVENTRY(registry, registryVariable, registryData)

@DTWR\_rRTVENTRY(registry, registryVariable, index)

@DTWR\_rRTVENTRY(registry, registryVariable)

### 値

表 205. DTWR\_RTVENTRY のパラメーター

データ型	パラメーター	用途	説明
ストリング	<i>registry</i>	IN	検索対象の項目が含まれているレジストリーの名前。
ストリング	<i>registryVariable</i>	IN	取り出す registryData ストリングが含まれているレジストリー項目の registryVariable ストリング部分の値。
ストリング	<i>registryData</i>	OUT	registryVariable と一致するレジストリー項目の registryData ストリング部分の値を返します。
ストリング	<i>index</i>	IN	戻される registryData ストリングが含まれている索引付き項目の registryVariable ストリングの索引部分の値。これはオプション・パラメーターです。これが指定されていると、索引付き項目の registryData ストリングが戻されます。

### 戻りコード

表 206. DTWR\_RTVENTRY の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。

表 206. DTWR\_RTVENTRY の戻りコード (続き)

戻りコード	説明
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。
1007	パラメーターに無効な値が含まれています。
3004	Web レジストリー組み込み関数が、指定したレジストリーから項目を削除または検索することができません。指定した項目が存在しません。
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。

## 例

### 例 1:

```
%DEFINE RegistryData = ""
@DTWR_RTVENTRY("Myregistry", "Jones", RegistryData)
```

### 例 2:

```
@DTWR_RTVENTRY("URLLIST", "SMITH", RegistryData, "WORK_URL")
```

### 例 3:

```
@DTWR_rRTVENTRY("Myregistry", "Jones")
```

### 例 4:

```
@DTWR_rRTVENTRY("URLLIST", "SMITH", "WORK_URL")
```

## DTWR\_UPDATEENTRY

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

### 目的

Web レジストリーのレジストリー・ストリング値を更新します。

### 構文

@DTWR\_UPDATEENTRY(registry, registryVariable, newData, index)

@DTWR\_UPDATEENTRY(registry, registryVariable, newData)

### 値

表 207. DTWR\_UPDATEENTRY のパラメーター

データ型	パラメーター	用途	説明
ストリング	registry	IN	更新する項目が含まれているレジストリーの名前。
ストリング	registryVariable	IN	更新するレジストリー項目の registryVariable ストリング部分の値。
ストリング	newData	IN	更新するレジストリー項目の registryData ストリング部分の新規値。
ストリング	index	IN	更新する索引付き項目の registryVariable ストリングの索引部分の値。これはオプション・パラメーターです。これを指定すると、索引付き項目が更新されます。

### 戻りコード

表 208. DTWR\_UPDATEENTRY の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当ててするための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1002	入力パラメーターに、ヌルで終わる文字からなるストリング値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター (ストリング変数とするために必須) が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
3003	Web レジストリー組み込み関数が、指定したレジストリーに項目を追加できません。指定した項目は既に存在します。
3004	Web レジストリー組み込み関数が、指定したレジストリーから項目を削除または検索することができません。指定した項目が存在しません。

表 208. DTWR\_UPDATEENTRY の戻りコード (続き)

戻りコード	説明
3005	Web レジストリー組み込み関数は、指定したレジストリーを使用できません。レジストリーが検出できません。
3007	Web レジストリー組み込み関数が、指定したオペレーションを完了できません。指定されたレジストリーに対し、要求発行者が適切な権限を持っていません。

## 使用上の注意

値に対応するレジストリー項目名は変更することができません。

## 例

### 例 1:

```
@DTWR_UPDATEENTRY("Myregistry", "Jones", "http://advantis.com/~Jones/personal")
```

### 例 2:

```
@DTWR_UPDATEENTRY("URLLIST", "SMITH", "http://www.software.ibm.com/personal", "WORK_URL")
```

---

## 永続的なマクロ関数

永続的なマクロ関数は、単一トランザクション内でどのマクロ・ブロックが永続的であるのかを定義できるようにして、`Net.Data` におけるトランザクション処理をサポートします。これらの関数を使用して、トランザクションの開始と終了、そのトランザクションを通じて永続的な HTML ブロック、そのトランザクション内の変数の効力範囲、およびトランザクションの変更内容をコミットまたはロールバックするかどうかを定義してください。

- 320ページの『DTW\_ACCEPT』
- 322ページの『DTW\_COMMIT』
- 323ページの『DTW\_ROLLBACK』
- 324ページの『DTW\_RTVHANDLE』
- 325ページの『DTW\_STATIC』
- 327ページの『DTW\_TERMINATE』

## DTW\_ACCEPT

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

### 目的

永続的なマクロを起動するために使用されるトランザクション・ハンドルを定義します。

### 構文

@DTW\_ACCEPT(handle, timeout)

@DTW\_ACCEPT(handle)

### 値

表 209. DTW\_ACCEPT のパラメーター

データ型	パラメーター	用途	説明
ストリング	handle	IN	この永続トランザクションで後続のマクロを起動するために URL で使用されるトランザクション・ハンドルを指定する、変数またはリテラル・ストリング。
整数	timeout	IN	このポートをサービスするジョブが応答を待つ時間を秒単位で指定する、変数またはリテラル・ストリング。この値は、DTW_STATIC() 関数で指定されたタイムアウト値をオーバーライドします。

### 戻りコード

表 210. DTW\_ACCEPT の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
8200	マクロ・パーシスタンスが使用可能ではありません。
8201	永続的組み込み関数がシーケンス外で呼び出されました。

### 使用上の注意

1. Net.Data では、Web ブラウザーからの応答としてマクロを起動する URL に、トランザクション・ハンドルがインクルードされている必要があります。Web サーバーに要求が届くと、サーバーはトランザクション・ハンドルを使用して、トランザクションを処理している CGI プロセスにその要求を送送します。

トランザクション・ハンドルは、マクロ内の各 HTML ブロックの先頭で呼び出され、最後の論理ブロックまで実行されなければなりません。最後の論理ブロックには、DTW\_TERMINATE() への呼び出しが含まれています。なんらかのテキストがブラウザーに出力されるまでに DTW\_ACCEPT() または DTW\_TERMINATE() への呼び出しが見つからない場合には、Net.Data エラーが発生します。

2. このページのタイムアウト値として、@DTW\_STATIC() 関数で指定されたタイムアウト値をオーバーライドする値を指定することができます。Web サーバーは、ユーザーがこの要求に応答するのを、指定された時間 (秒単位) だけ待ちます。
3. マクロが永続的な状態になっていないときにこの関数が呼び出されると、Net.Data エラーが発生します。
4. トランザクション・ハンドルを含む URL は、押しボタンからのアクションとしてコーディングすることも、ブラウザーに表示されたページ上のハイパーテキスト・リンクとしてコーディングすることもできます。

## 例

### 例 1:

```
%DEFINE handle = ""
@DTW_RTVHANLDE(handle)

%HTML(REPORT) {
@DTW_ACCEPT(handle)
...
%}
```

## DTW\_COMMIT

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

### 目的

コミットメント制御されているリソースに対して最後のコミットメント境界以降に行われた、保留中の変更を永続的なものにし、新規コミットメント境界を設定します。

### 構文

```
@DTW_COMMIT()
```

### 値

なし。

### 戻りコード

表 211. *DTW\_COMMIT* の戻りコード

戻りコード	説明
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。

### 例

例 1: コミットを指定します。

```
@DTW_COMMIT()  
%HTML(report){  
%}
```



## DTW\_ROLLBACK

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

### 目的

最後のコミットメント境界を現行コミットメント境界として再設定します。 Net.Data が実行されているプロセスに関連して、コミットメント制御されているリソースに対して最後のコミットメント境界以降に行われたすべての変更が、バックアウトされます。

### 構文

```
@DTW_ROLLBACK()
```

### 値

なし。

### 戻りコード

表 212. DTW\_ROLLBACK の戻りコード

戻りコード	説明
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。

### 例

例 1: ロールバックを指定します。

```
@DTW_ROLLBACK()  
%HTML(report){  
%}
```

## DTW\_RTVHANDLE

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

### 目的

スレッド情報、タイム・スタンプ、および現行ユーザーの組み合わせに基づいて計算される、複数の別個の起動を通じてこのマクロに固有なトランザクション・ハンドルを生成して戻します。

### 構文

@DTW\_RTVHANDLE(handle)

### 値

表 213. DTW\_RTVHANDLE のパラメーター

データ型	パラメーター	用途	説明
ストリング	handle	OUT	現行の永続マクロ用の固有なトランザクション・ハンドルが入る変数。

### 戻りコード

表 214. DTW\_RTVHANDLE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1006	関数呼び出しにおいて、出力パラメーターとするために必須であるパラメーターにリテラル・ストリングが渡されました。

### 使用上の注意

トランザクション・ハンドルを使用すると、永続的なトランザクションの一部として指定された URL が HTTP サーバーに固有なものになり、有効な要求として確実に識別されるようになります。

### 例

**例 1:** トランザクション・ハンドルを検索するために使用される handle 変数を定義します。

```
%DEFINE handle = ""
@DTW_RTVHANDLE(handle)
```

## DTW\_STATIC

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

### 目的

マクロ全体が永続的であることを示します。

### 構文

@DTW\_STATIC(timeout)

@DTW\_STATIC()

### 値

表 215. DTW\_STATIC のパラメーター

データ型	パラメーター	用途	説明
整数	<i>timeout</i>	IN	このトランザクションを処理するプロセスが応答を待つ時間を秒単位で指定する、変数またはリテラル・ストリング。

### 戻りコード

表 216. DTW\_STATIC の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1001	入力パラメーターにヌル値が含まれています。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
1005	関数呼び出しで渡されたパラメーター（ストリング変数とするために必須）が異なる変数型です。
1007	パラメーターに無効な値が含まれています。
8202	パーシスタンスを使用可能にできません。

### 使用上の注意

- DTW\_STATIC は、マクロ内の最初のステートメントでなければなりません。マクロ内でこの関数呼び出し以降に定義されたすべての変数は、明示的に別の指定が行われていない限り、DTW\_TERMINATE() が呼び出されるかあるいはプロセスが終了するまで、複数のマクロ起動を通じて永続します。
- 関数呼び出しで秒単位のタイムアウト値を指定して、Net.Data が実行されているプロセスがブラウザーからの応答を待つ時間を指示することができます。タイムアウト値が満了するとプロセスは終了し、コミットメント制御されているリソースに対して、最後のコミットメント境界以降に行われたすべての変更がロールバックされます。
- 後続の @DTW\_ACCEPT() 呼び出しでタイムアウト値が指定されていると、Net.Data はこの値を後続の呼び出しで指定された値によってオーバーライドしま

す。この呼び出しでも後続の @DTW\_ACCEPT() 呼び出しでもタイムアウト値が指定されていない場合、 Web サーバーのデフォルト・タイムアウト値が使用されます。

## 例

**例 1:** タイムアウト値を 60 に指定する DTW\_STATIC() への呼び出し  
`@DTW_STATIC("60")`

# DTW\_TERMINATE

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

## 目的

永続的なトランザクションを終了します。コミットメント制御されているリソースに対して最後のコミットメント境界以降に行われた、すべての変更を永続的なものにします。

## 構文

@DTW\_TERMINATE()

## 値

なし。

## 戻りコード

表 217. DTW\_TERMINATE の戻りコード

戻りコード	説明
-1001	サーバーがメモリーを割り当てるための Net.Data 要求を処理できませんでした。
1003	関数呼び出しで渡されたパラメーターの数が、許可される最大数を超えた、または関数が必要とする最小数より少ない。
8200	マクロ・パーシスタンスが使用可能ではありません。
8201	永続的組み込み関数がシーケンス外で呼び出されました。

## 使用上の注意

- DTW\_TERMINATE 関数は、永続的なトランザクション内で、何らかのテキストがブラウザーに出力される前に、論理的な最終 HTML ブロックの先頭で呼び出されます。この関数の前に何らかのテキスト出力がブロック内で行われると、Net.Data エラーが発生します。アプリケーションの書き方によっては、論理的な最終 HTML ブロックが複数存在する場合がありますので、注意してください。
- マクロが永続的な状態になっていないときにこの関数が呼び出されると、Net.Data エラーが発生します。

## 例

例 1: 永続的なトランザクションを終了します。

```
%HTML(QUIT){
@DTW_TERMINATE()
...
%}
```



## 付録A. Net.Data 技術ライブラリー

Net.Data 技術ライブラリーは、以下の Net.Data Web サイトでアクセスできます。

<http://www.software.ibm.com/data/net.data/library.html>

文書	説明
<ul style="list-style-type: none"><li>• <i>Net.Data</i> 管理およびプログラミングの手引き OS/390</li><li>• <i>Net.Data</i> 管理およびプログラミングの手引き OS/2、Windows NT、および UNIX 版</li><li>• <i>Net.Data</i> 管理およびプログラミングの手引き OS/400</li></ul>	Net.Data のインストール、構成および起動するに関する概説およびタスク情報が含まれています。また、Net.Data マクロの作成、Net.Data パフォーマンス向上のための手法、Net.Data 言語環境の使用、接続の管理、問題解決およびパフォーマンス・チューニングのための Net.Data ログおよびトレースに関する方法を説明しています。
<i>Net.Data</i> 解説書	Net.Data マクロ言語、変数、および組み込み関数を説明しています。
<i>Net.Data</i> 言語環境解説書	Net.Data 言語環境インターフェースを説明しています。
<i>Net.Data</i> メッセージおよびコード解説書	Net.Data エラー・メッセージおよび戻りコードを説明しています。





---

## 付録B. DB2 WWW Connection

DB2 WWW Connection を使用している場合は、既存のアプリケーションを Net.Data で実行することができます。Net.Data バージョン 2 の機能を活用するためには、アプリケーションを更新することをお勧めします。

DB2 WWW 言語構成要素は、以下のとおりです。

- 『EXEC\_SQL』
- 『HTML\_INPUT』
- 『HTML\_REPORT』
- 『SQL』
- 332ページの『SQL\_MESSAGE』
- 332ページの『SQL\_REPORT』
- 333ページの『SQL\_CODE』

---

### EXEC\_SQL

この言語構成要素は、SQL ブロックを呼び出します。この方法ではなく、SQL ステートメントを関数として呼び出すことをお勧めします。詳しくは、19ページの『FUNCTION ブロック』を参照してください。

---

### HTML\_INPUT

この言語構成要素は、INPUT という名前の HTML ブロックと同じです。詳しくは、30ページの『HTML ブロック』を参照してください。

---

### HTML\_REPORT

この言語構成要素は、REPORT という名前の HTML ブロックと同じです。詳しくは、30ページの『HTML ブロック』を参照してください。

---

### SQL

この言語構成要素は、Net.Data において FUNCTION(DTW\_SQL) で呼び出される関数と同等です。

これには SQL\_REPORT および SQL\_MESSAGE ステートメントが含まれ、これらのステートメントは DB2 WWW Connection から呼び出されます。DB2 WWW Connection は名前付き %SQL ブロックをサポートしません。

例:

**例 1:** DB2 WWW Connection マクロ

```
%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>
%}
```

**例 1:** 同等な Net.Data マクロ

```
%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

この言語構成要素は、Net.Data の MESSAGE ステートメントと同等です。例については、52ページの『MESSAGE ブロック』を参照してください。

---

## SQL\_REPORT

この言語構成要素は、Net.Data の REPORT ステートメントと同等です。例については、57ページの『REPORT ブロック』を参照してください。

---

## SQL\_CODE

この言語構成要素は DB2 WWW Connection から呼び出されるもので、互換性のために Net.Data でサポートされています。これは、128ページの『RETURN\_CODE』と同等です。



## 付録C. Net.Data のオペレーティング・システムごとの参照

Net.Data のすべての機能が各オペレーティング・システムでサポートされるわけではありません。このセクションでは、ご使用のオペレーティング・システムでどの機能がサポートされるかを示します。 **X** は、その機能がサポートされることを示します。

表 218. Net.Data の言語環境

言語環境	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
直接呼び出し					X				
フラット・ファイル・インターフェース	X			X	X	X	X	X	X
IMS Web	X			X				X	X
Java アプレット	X	X	X	X	X	X	X	X	X
Java アプリケーション	X			X		X		X	X
ODBC	X	X		X	X		X	X	X
Oracle	X								X
Perl	X	X	X	X	X			X	X
REXX	X		X	X	X	X	X	X	X
SQL	X	X	X	X	X	X	X	X	X
Sybase	X								X
システム	X	X	X	X	X	X	X	X	X
Web レジストリー	X			X		X			X

表 219. Net.Data の構成変数

構成変数	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	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_PORT	X								X
DTW_CM_PORT	X	X		X					X
DTW_DIRECT_REQUEST	X	X		X	X		X	X	X
DTW_DO_NOT_CACHE_MACRO					X				
DTW_INST_DIR	X		X	X			X	X	X
DTW_LOG_DIR	X	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				

表 219. Net.Data の構成変数 (続き)

構成変数	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	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	X
DTW_SQL_ISOLATION						X			
DTW_SQL_NAMING_MODE						X			
DTW_UNICODE	X	X		X			X	X	X
DTW_VARIABLE_SCOPE	X	X		X			X	X	X

表 220. Net.Data の変数

変数	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	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
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				
LOGIN	X	X	X	X		X	X	X	X
Nn	X	X	X	X	X	X	X	X	X
NLIST	X	X	X	X	X	X	X	X	X
NULL_RPT_FIELD						X			
NUM_COLUMNS	X	X	X	X	X	X	X	X	X
NUM_ROWS						X			
PASSWORD	X	X	X	X		X	X	X	X

表 220. Net.Data の変数 (続き)

変数	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
RETURN_CODE	X	X	X	X	X	X	X	X	X
ROW_NUM	X	X	X	X	X	X	X	X	X
RPT_MAX_ROWS	X	X	X	X	X	X	X	X	X
SHOWSQL	X	X	X	X	X	X	X	X	X
SQL_CODE	X	X	X	X	X	X	X	X	X
SQL_STATE	X	X	X	X	X	X	X	X	X
START_ROW_NUM	X	X	X	X	X	X	X	X	X
TOTAL_ROWS	X	X	X	X	X	X	X	X	X
TRANSACTION_SCOPE	X	X	X	X	X	X	X	X	X
V_columnName	X	X	X	X	X	X	X	X	X
VLIST	X	X	X	X	X	X	X	X	X
Vn	X	X	X	X	X	X	X	X	X

表 221. Net.Data の関数

関数	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	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_COMMIT						X			
DTW_CHARTOHEX						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			
DTW_HTMLENCODE	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
DTW_LOWERCASE	X	X	X	X	X	X	X	X	X
DTW_MULTIPLY	X	X	X	X	X	X	X	X	X

表 221. Net.Data の関数 (続き)

関数	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_POS	X	X	X	X	X	X	X	X	X
DTW_POWER	X	X	X	X	X	X	X	X	X
DTW_QHTMLENCODE	X	X	X	X	X	X	X	X	X
DTW_REVERSE	X	X	X	X	X	X	X	X	X
DTW_ROLLBACK						X			
DTW_RVTHANDLE						X			
DTW_SENMAIL	X	X	X	X	X	X	X	X	X
DTW_SETCOOKIE	X	X	X	X	X	X	X	X	X
DTW_SETENV	X	X	X	X	X	X	X	X	X
DTW_STATIC						X			
DTW_STRIP	X	X	X	X	X	X	X	X	X
DTW_SUBSTR	X	X	X	X	X	X	X	X	X
DTW_SUBTRACT	X	X	X	X	X	X	X	X	X
DTW_SUBWORD	X	X	X	X	X	X	X	X	X
DTW_TB_APPENDROW	X	X		X	X	X	X	X	X
DTW_TB_COLS	X	X	X	X	X	X	X	X	X
DTW_TB_DELETEROW	X	X		X	X	X	X	X	X
DTW_TB_DELETECOL	X	X		X	X	X	X	X	X
DTW_TB_DLIST	X	X	X	X	X	X	X	X	X
DTW_TB_DUMP	X	X	X	X	X	X	X	X	X
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_HTMLNENCODE	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	X
DTW_TB_INSERTROW	X	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	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	X
DTW_TB_SETN	X	X		X	X	X	X	X	X
DTW_TB_SETV	X	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			



表 221. Net.Data の関数 (続き)

関数	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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
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
DTWF_CLOSE	X			X	X	X		X	X
DTWF_DELETE	X			X	X	X		X	X
DTWF_INSERT	X			X	X	X		X	X
DTWF_OPEN	X			X	X	X		X	X
DTWF_READ	X			X	X	X		X	X
DTWF_REMOVE	X			X	X	X		X	X
DTWF_SEARCH	X			X	X	X		X	X
DTWF_UPDATE	X			X	X	X		X	X
DTWF_WRITE	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
DTWR_OPENREG						X			
DTWR_RTVENTRY	X			X		X			X
DTWR_UPDATEENTRY	X			X		X			X

表 222. Net.Data のインターフェース

インターフェース・タイプ	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
FastCGI	X							X	
CGI	X	X	X	X	X	X	X	X	X
Java Beans									X
Internet Connection API (ICAPI)	X			X	X				X
Internet Server API (ISAPI)									X
Live コネクション	X			X				X	X
Lotus Domino Go Web Server (GWAPI)	X			X	X				X

表 222. *Net.Data* のインターフェース (続き)

インターフェース・タイプ	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
Netscape API (NSAPI)	X							X	X
Servlets	X				X				X

表 223. *Net.Data* のツール

ツール	AIX	HP	LINUX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
管理ツール	X			X					X
NetObjects Fusion Plug-ins									X
Wizards	X			X			X	X	X

## 付録D. 特記事項

本書において、日本では発表されていない IBM 製品 (機械およびプログラム)、プログラミング、またはサービスについて言及または説明する場合があります。しかし、このことは、弊社がこのような IBM 製品、プログラミングまたはサービスを、日本で発表する意図があることを必ずしも示すものではありません。本書で、IBM ライセンス・プログラムまたは他の IBM 製品に言及している部分があっても、このことは当該プログラムまたは製品のみが使用可能であることを意味するものではありません。これらのプログラムまたは製品に代えて、IBM の知的所有権を侵害することのない機能的に同等のプログラムまたは製品を使用することができます。ただし、IBM によって明示的に指示されたものを除き、これらのプログラムまたは製品に関連する動作の評価および検査はお客様の責任で行っていただきます。

IBM は、本書で説明する主題に関する特許権 (特許出願を含む) 商標権、または著作権を所有している場合があります。本書は、これらの特許権、商標権、および著作権について、本書で明示されている場合を除き、実施権、使用権等を許諾することを意味するものではありません。実施権、使用権等の許諾については、下記の宛先に、書面にてご照会ください。

〒106-0032 東京都港区六本木3丁目2-31

AP事業所

IBM World Trade Asia Corporation

Intellectual Property Law & Licensing

IBM は、本書を “現状のまま” 提供するものであり、明示的または黙示にかかわらず、非侵害性、商業的な使用可能性、または特定の目的に対する適合性に関する黙示の保証を含み、かつそれには限定されない、いかなる保証も行いません。一部の州では特定の商取引における明示的または黙示の保証の放棄は認められていません。したがってこの記述はお客様には適用されない場合もあります。

本書の情報は定期的に変更されており、それらの変更は本書の新しい版で取り入れられる予定です。IBM は本書に記載された製品およびプログラムを予告なく改良または変更する場合があります。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation

555 Bailey Avenue, W92/H3

P.O. Box 49023

San Jose, CA 95161-9023

本プログラムに関する上記の情報は、適切な条件の下で、使用することができますが、有償の場合もあります。

本書において解説されているライセンス・プログラムおよびそのライセンス・プログラム資料は、「IBM 使用契約書」の契約条件に基づいて弊社から提供されるものです。

IBM 以外の製品に関する情報は、それぞれの製品の提供元、それに関する印刷物、その他の公に使用可能な情報源から得たものです。IBM はそれらの製品をテストしておらず、それら IBM 以外の製品に関して、パフォーマンスの正確さ、互換性、またはその他についての苦情を受け付けることはできません。IBM 以外の製品の機能に関しては、それぞれの製品の提供元にお問い合わせください。

著作権の使用許諾:

本書には、IBM が説明するための一例として提供している簡単なプログラムが含まれています。これらの例は必ずしもすべての場合について完全にテストされたものではありません。IBM はこれらのプログラムの信頼性、可用性、および機能について法律上の瑕疵担保責任を含むいかなる明示または暗示の保証責任も負いません。本書中に含まれているすべてのプログラムは "現存するままの状態" で提供されます。IBM はプログラムの商業的な使用可能性および特定の目的に対する適合性については、いかなる保証も行いません。

---

## 商標

以下の用語は、米国またはその他の国における IBM Corporation の商標です。

AIX	IMS
AS/400	Language Environment
CBIDO	MVS/ESA
CBPDO	Net.Data
CICS	OpenEdition
CustomPac	Operating System/400
DB2	OS/2
DB2 Universal Databas	OS/390
DataJoiner	OS/400
Distributed Relational Database Architecture	RACF
DRDA	SystemPac
IBM	

以下の用語は、米国またはその他の国における IBM Corporation の商標です。

Java およびすべての Java ベースの商標およびロゴは、Sun Microsystems, Inc. の米国、その他の国における商標または登録商標です。

Lotus および Domino Go Webserver は、Lotus Development Corporation の米国、その他の国における商標または登録商標です。

Microsoft、Windows、Windows NT、および Windows ロゴは、Microsoft Corporation の米国、その他の国における商標または登録商標です。

その他の会社名、製品名、およびサービス名 (2 重アスタリスク \*\*) で示されます) は、他社の商標またはサービス・マークです。

## 用語集

**絶対パス (absolute path).** オブジェクトのフルパス名。絶対パス名は最上位ディレクトリー、すなわちルート・ディレクトリー (スラッシュ (/) または円記号 (¥) 文字によって識別されます) から始まる。

**API.** アプリケーション・プログラミング・インターフェース (Application programming interface)。Net.Data は、CGI プロセスのパフォーマンスを向上させるための 3 つの Web サーバー API をサポートする。

**アプレット (applet).** HTML ページに組み込まれる Java プログラム。アプレットは、Netscape Navigator などの Java 対応のブラウザを処理し、HTML ページの処理の際にロードされる。

**アプリケーション・プログラミング・インターフェース (application programming interface (API)).** オペレーティング・システムまたは別途発注可能なライセンス・プログラムによって提供される機能インターフェース。これにより、高水準言語で書かれたアプリケーション・プログラムが、特定のデータもしくは、オペレーティング・システムまたはライセンス・プログラムを使用できるようになる。Net.Data は、所有権を主張できる Web サーバー API (ICAPI, GWAPI, ISAPI, および NSAPI) をサポートし、CGI 処理のパフォーマンスを向上させる。

**キャッシュ (cache).** 最近アクセスされたデータが入る、メモリーまたはディスク・スペースの一部。同一データに続けてアクセスする場合の速度を上げることが目的。キャッシュは、ネットワークを介してアクセス可能な、使用頻度の高いデータのローカル・コピーを保留するのに使用される場合が多い。

**キャッシング (caching).** ローカルに Web サーバーに要求した結果得られる使用頻度の高いデータを高速で取り出すために、情報を最新表示するときまで保管するプロセス。

**キャッシュ管理プログラム (Cache Manager).** 1 つのマシン用のキャッシュを管理するプログラム。複数のキャッシュを管理できる。

**CGI.** コモン・ゲートウェイ・インターフェース (Common Gateway Interface)。

**CLIETTE.** Web サーバーからの要求に長期にわたって対応する、Net.Data Live コネクションのプロセス。この接続管理プログラムは、これらの要求に対応する CLIETTE プロセスのスケジュールを行う。

**コミットメント制御 (commitment control).** Net.Data が実行されているプロセス内で、リソースの操作が作業単位の一部となる境界を設定すること。

**コモン・ゲートウェイ・インターフェース (Common Gateway Interface (CGI)).** Web サーバーがアプリケーション・プログラムに制御権を渡し、反対にデータを受け取る、標準的な方法。

**接続管理プログラム (Connection Manager).** Live コネクションのサポートに必要とされる Net.Data 内の実行可能ファイル dtwcm。

**cookie.** HTTP サーバーによって Web ブラウザーに送信され、次にブラウザーがそのサーバーにアクセスするたびに送り返される、情報のパケット。cookie には、サーバーが選択する任意の情報を含めることができ、特に国籍をうたわない HTTP トランザクション間の状態を保持するのに使用される。Free Online Dictionary of Computing より。

**現行作業ディレクトリー (current working directory).** すべての相対パス名を解決する基準となる、プロセスのデフォルト・ディレクトリー。

**データベース (database).** 表の集合、もしくは表スペースおよび索引スペースの集合。

**データベース管理システム (database management system (DBMS)).** データベースの作成、編成、および修正を制御し、データベース内に格納されたデータにアクセスする、ソフトウェア・システム。

**データ型 (data type).** 列およびリテラルの属性。

**DBMS.** データベース管理システム (Database management system)。

**Domino Go Web サーバー (Domino Go Web server).** Lotus Corp. および IBM が提供する Web サーバーで、正規およびセキュア接続の両方を提供する。このサーバーには、ICAPI および GWAPI のインターフェースが提供される。

**ファイアウォール (firewall).** 内部ネットワークを無許可の外部アクセスから保護するソフトウェアを備えたコンピュータ。

**フラット・ファイル・インターフェース (flat file interface).** 平文ファイルのデータを読み書きすることができる、一連の Net.Data 組み込み関数。

**GWAPI.** Go Web サーバー API。

**HTML.** ハイパーテキスト・マークアップ言語 (Hypertext markup language)。

**HTTP.** Hypertext transfer protocol (HTTP)。

**ハイパーテキスト・マークアップ言語 (hypertext markup language).** Web 文書の作成に使用するタグ言語。

**hypertext transfer protocol (HTTP).** Web サーバーとブラウザ間で使用する通信プロトコル。

**ICAPI.** インターネット接続 API (Internet Connection API)。Domino Go Web サーバー (Domino Go Web server) を参照。

**インターネット (Internet).** 国際パブリック TCP/IP コンピューター・ネットワーク。

**イントラネット (Intranet).** 企業ファイアウォール内の TCP/IP ネットワーク。

**ISAPI.** Microsoft のインターネット・サーバー API。

**Java.** 特にインターネット・アプリケーションに役立つ、オペレーティング・システムに影響されないオブジェクト指向プログラミング言語。

**言語環境 (language environment).** Net.Data マクロから、DB2 などの外部データ・ソースや Perl などのプログラミング言語へのアクセスを行うモジュール。

**Live コネクション (Live Connection).** 1 つの Connection Manager と複数の CLIETTE からなる Net.Data コンポーネント。Live コネクションは、データベースと Java 仮想マシンの接続の再使用を管理する。

**LOB.** ラージ・オブジェクト (Large object)。

**ミドルウェア (middleware).** アプリケーション・プログラムとネットワークの中間にあるソフトウェア。ネットワークを介したクライアント・アプリケーション・プログラムとサーバーの間の対話を管理する。

**NSAPI.** Netscape API。

**ヌル (null).** 情報が無いことを示す特殊な値。

**パス (path).** ファイルを探すのに使用する検索経路。

**パス名 (path name).** オブジェクトの見付け方をシステムに知らせる。パス名は、一連のディレクトリー名の後にオブジェクトの名前を続けた形で表現される。個々のディレクトリーおよびオブジェクト名は、スラッシュ (/) または円記号 (¥) によって区切られる。

**Perl.** インタープリター・プログラミング言語。

**永続 (persistence).** 割り当てられた値をトランザクション全体で保持する状態。この状態においては、1 つのトランザクションが複数の Net.Data 起動にわたって存続する。永続的にできるのは、変数だけである。また、コミットメント制御の影響を受けるリソースの操作は、明示的なコミットまたはロールバックが行われるか、あるいはトランザクションが完了するまで、活動状態のままになる。

**ポート (port).** TCP/IP と高水準プロトコルもしくはアプリケーション間の通信に使用する 16 ビット数。

**レジストリー (registry).** スtringを保管および検索することのできるリポジトリ。

**相対パス名 (relative path name).** 最上位、すなわちルート・ディレクトリーから始まらないパス名。システムは、パス名がプロセスの現行作業ディレクトリーから始まることを前提としている。

**TCP/IP.** 伝送制御プロトコル/インターネット・プロトコル (Transmission Control Protocol / Internet Protocol)。

**トランザクション (transaction).** 1 つの Net.Data 起動。永続的な Net.Data が使用される場合、トランザクションは複数の Net.Data 起動にわたって存続することがある。

**伝送制御プロトコル/インターネット・プロトコル (Transmission Control Protocol / Internet Protocol).** ローカル・エリア・ネットワークと広域ネットワークの両方に使用する、対等接続機能をサポートする一連の通信プロトコル。

**URL.** Uniform resource locator (URL)。

**uniform resource locator (URL).** HTTP サーバー、および任意選択でディレクトリーとファイル名を指名するアドレス。たとえば、  
<http://www.software.ibm.com/data/net.data/index.html>。

**作業単位 (unit of work).** 1 つのアトミック操作として取り扱われる、回復可能な一連の操作。作業単位内のすべての操作は、その操作が単一操作である場合と同じように完了 (コミット) または取り消し (ロールバック) することができる。コミットまたはロールバックできるのは、コミットメント制御の影響を受けるリソースの操作だけである。

**Web サーバー (Web server).** インターネット接続 (Internet Connection) などの HTTP サーバー・ソフトウェアを実行しているコンピューター。



# 索引

日本語, 英字, 数字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

## [ア行]

アクセス、フラット・ファイル 269

値のグループの受け渡し 75

位置、フラット・ファイル 270

隠蔽変数

    ステップ 73

    説明 73

    例、HTML 形式での 73

永続的なマクロ関数

    DTW\_ACCEPT 320

    DTW\_COMMIT 322

    DTW\_ROLLBACK 323

    DTW\_RTVHANDLE 324

    DTW\_STATIC 325

    DTW\_TERMINATE 327

エラー処理 52

大文字、指定 98

大文字小文字、SQL コマンドのための指定 98

オペレーティング・システムの参照 333

## [カ行]

解放、ファイルの、FFI 関数 273

各種変数

    説明 115

    DTW\_CURRENT\_FILENAME 116

    DTW\_CURRENT\_LAST\_MODIFIED 117

    DTW\_DEFAULT\_MESSAGE 118

    DTW\_MACRO\_LAST\_MODIFIED 121

    DTW\_MP\_PATH 124

    DTW\_MP\_VERSION 125

    DTW\_PRINT\_HEADER 126

    DTW\_REMOVE\_WS 127

    RETURN\_CODE 128

環境変数

    説明 71

    例 71

    ENVVAR ステートメント 15

関数

    値のグループの受け渡し 75

    永続的な 319

    数学 164

    ストリング 183

    説明 129

関数 (続き)

    汎用 75

    表 225

    フラット・ファイル・インターフェース (FFI) 269

    命名規則 129

    ワード 212

    Web レジストリー 299

関数呼び出し

    構文 27

    出力の形式化 57

    説明 27

    表行の処理 60

    INOUT 変数の使用 28

機密保護

    パスワード 110

    ログイン ID 108

機密保護の推奨事項、FFI\_PATH 271

行の長さの制限、マクロ 4

許可要件、FFI\_PATH 272

区切り文字、FFI 言語環境

    ASCITEXT 272

    DELIMITED 272

区切られた値のストリング 74

組み込み関数 129

組み込みファイル 40

現行ディレクトリー、フラット・ファイルの、判別  
270

言語環境変数

    説明 95

    DATABASE 96

    DB2PLAN 99

    DB2SSID 100

    DB\_CASE 98

    DTW\_APPLET\_ALTTEXT 101

    DTW\_EDIT\_CODES 102

    DTW\_MBMODE 122

    DTW\_PAD\_PGM\_PARMS 103

    DTW\_SAVE\_TABLE\_IN 104

    DTW\_SET\_TOTAL\_ROWS 105

    LOCATION 107

    LOGIN 108

    NULL\_RPT\_FIELD 109

    PASSWORD 110

    SHOWSQL 111

    SQL\_STATE 112

    TRANSACTION\_SCOPE 113

言語構成要素

    関数呼び出し 27

    共通の構文要素 5

言語構成要素 (続き)

  ストリング 27

  変数参照 5

  変数名 5

  マクロ

    構文 1

    説明 7

  COMMENT ブロック 9

  DB2 WWW Connection 331

  DEFINE ブロックまたはステートメント 11

  ENVVAR ステートメント 15

  EXEC ブロックまたはステートメント 16

  FUNCTION ブロック 19

  HTML ブロック 30

  IF ブロック 33

  INCLUDE ステートメント 40

  INCLUDE\_URL ステートメント 43

  LIST ステートメント 46

  MACRO\_FUNCTION ブロック 48

  MESSAGE ブロック 52

  REPORT ブロック 57

  ROW ブロック 60

  TABLE ステートメント 63

  WHILE ブロック 65

構成、FFI 言語環境 271

小文字、指定 98

## [サ行]

サブシステム ID、DB2 サブシステムへの接続 100

サポートされる機能の表 333

実行可能変数

  説明 72

  パラメーターを指定した 73

  変数参照として 72

  例 72

上限 63

条件付きストリング処理 33, 65

条件変数

  説明 70

  変数参照を指定した 70

  例 74

  LIST ステートメントを指定した 71

数学関数

  DTW\_ADD 165

  DTW\_DIVIDE 167

  DTW\_DIVREM 169

  DTW\_FORMAT 171

  DTW\_INTDIV 175

  DTW\_MULTIPLY 177

  DTW\_POWER 179

  DTW\_SUBTRACT 181

数値比較、ストリングの 33, 65

スクロール、Next および Previous ボタンでの 92

ストリング

  値、区切られた 74

  条件付き処理 33, 65

  数値比較 33, 65

  説明 6

ストリング関数

  DTW\_ASSIGN 184

  DTW\_CHARTOHEX 185

  DTW\_CONCAT 186

  DTW\_DELSTR 188

  DTW\_HEXTOCHAR

    DTW\_HEXTOCHAR 190

  DTW\_INSERT

    DTW\_INSERT 192

  DTW\_LASTPOS 194

  DTW\_LENGTH 196

  DTW\_LOWERCASE 197

  DTW\_POS 199

  DTW\_REPLACE 201

  DTW\_REVERSE 203

  DTW\_STRIP 204

  DTW\_SUBSTR 206

  DTW\_TRANSLATE 208

  DTW\_UPPERCASE 210

  MBCS サポート 183

制限、データベース・アクセスの 108, 110

接続、DB2 サブシステムへの

  サブシステム ID 100

  ロケーション 107

  DB2 プラン 99

絶対パス、フラット・ファイルの 270

宣言パーツ、マクロ 2

## [タ行]

代替テキスト、Web ブラウザー 101

データベース整合性、トランザクション効力範囲 113

データベースへの接続、DATABASE 変数 96

特記事項 341

## [ハ行]

パフォーマンス、DTW\_EXIT 140

パラメーター、引き渡し 25

汎用関数 131

  DTW\_ADDQUOTE 132

  DTW\_CACHE\_PAGE 134

  DTW\_DATE 138

  DTW\_EXIT 140

  DTW\_GETCOOKIE 141

  DTW\_GETENV 143

  DTW\_GETINIDATA 145



汎用関数 132 (続き)

DTW\_HTMLENCODER 132  
DTW\_QHTMLENCODER 148  
DTW\_SENDMAIL 149  
DTW\_SETCOOKIE 154  
DTW\_SETENV 158  
DTW\_TIME 160  
DTW\_URLESCSEQ 162

引き渡し、パラメーター、System 言語環境 25

日付形式、UTF-8 138

日付変数 115

表

HTML での結果 89  
Net.Data、行数の指定 90

表関数

DTW\_TB\_APPENDROW 226  
DTW\_TB\_COLS 228  
DTW\_TB\_DELETECOL 229  
DTW\_TB\_DELETEROW 230  
DTW\_TB\_DLIST 232  
DTW\_TB\_DUMPH 235  
DTW\_TB\_DUMPV 236  
DTW\_TB\_GETN 238  
DTW\_TB\_GETV 240  
DTW\_TB\_HTMLENCODER 242  
DTW\_TB\_INPUT\_CHECKBOX 244  
DTW\_TB\_INPUT\_RADIO 246  
DTW\_TB\_INPUT\_TEXT 248  
DTW\_TB\_INSERTCOL 250  
DTW\_TB\_INSERTROW 251  
DTW\_TB\_LIST 253  
DTW\_TB\_QUERYCOLNONJ 255  
DTW\_TB\_ROWS 257  
DTW\_TB\_SELECT 258  
DTW\_TB\_SETCOLS 260  
DTW\_TB\_SETN 261  
DTW\_TB\_SETV 263  
DTW\_TB\_TABLE 265  
DTW\_TB\_TEXTAREA 267

表示パーツ、マクロ 2

表処理変数

説明 76  
NLIST 78  
NUM\_COLUMNS 79  
NUM\_ROWS 80  
Nn 77  
ROW\_NUM 81  
SQL 言語環境のための指定 104  
TOTAL\_ROWS 82  
VLIST 84  
Vn 85  
V\_columnName 83

表変数

説明 75

例 75

ファイル場所変数 115

フッター 40

プラットフォーム・サポートの参照 333

フラット・ファイル

アクセス 269

アクセスの推奨事項 270

位置

現行ディレクトリー 270

FFI\_PATH 270

機密保護の推奨事項 271

許可要件 272

区切り文字 272

現行ディレクトリーでの作成 270

構成規則 271

絶対パス 270

データ・ソース 269

定義 269

ファイルのロック 273

FFI\_PATH の突き合わせ 270

プラン、DB2 サブシステムへの接続 99

ヘッダー 40

変数

隠蔽 73

各種 115

環境 71

言語環境 95

実行可能 72

条件 70

表 75, 76

リスト 74

レポート 86

Net.Data、概要 69

変数参照 5

変数名 5

変数名の隠蔽 73

## [マ行]

マクロ

共通の構文要素 5

行の長さの制限 4

グローバル構文 1

形式 3

言語構成要素 1

サンプル 3

処理の停止 140

宣言パーツ 2

表示パーツ 2

マクロから電子メールを送信 149

メッセージ、デフォルト・テキスト 118

## [ヤ行]

用語集 342

呼び出し

外部プログラム 16

関数 27

呼び出し、FFI 言語環境の 269

## [ラ行]

リスト、区切られたストリングの 74

リスト変数

値区切り記号 74

説明 74

例 74

リモート DB2 サブシステム、ロケーション 107

ループ 65

レポート

形式 57

Net.Data デフォルトのオーバーライド 88

レポート変数

説明 86

ALIGN 87

DTW\_DEFAULT\_REPORT 88

DTW\_HTML\_TABLE 89

RPT\_MAX\_ROWS 90

START\_ROW\_NUM 92

ローカル DB2 サブシステム、ID 100

ロケーション、DB2 サブシステムへの接続 107

ロック、ファイルの、FFI 関数 273

## [ワ行]

ワード関数

DTW\_DELWORD 213

DTW\_SUBWORD 215

DTW\_WORD 217

DTW\_WORDINDEX 219

DTW\_WORDLENGTH 221

DTW\_WORDPOS 222

DTW\_WORDS 224

MBCS サポート 212

## A

ALIGN 87

APPLET タグ、代替テキスト 101

## C

COMMENT ブロック

構文 9

説明 9

cookies

送信 126

DTW\_GETCOOKIE 141

DTW\_PRINT\_HEADER 126

DTW\_SETCOOKIE 154

## D

DATABASE 96

DB2 WWW Connection、言語構成要素 331

DB2PLAN 99

DB2SSID 100

DB\_CASE 98

DEFINE ステートメント

構文 11

説明 11

DEFINE ブロック

構文 11

説明 11

DTWF\_APPEND 273

DTWF\_CLOSE 273, 277

DTWF\_DELETE 278

DTWF\_INSERT 280

DTWF\_OPEN 273, 283

DTWF\_READ 285

DTWF\_REMOVE 288

DTWF\_SEARCH 290

DTWF\_UPDATE 293

DTWF\_WRITE 296

DTWR\_ADDENTRY 299

DTWR\_CLEARREG 302

DTWR\_CLOSEREG 303

DTWR\_CREATEREG 304

DTWR\_DELENTY 306

DTWR\_DELREG 308

DTWR\_LISTREG 309

DTWR\_LISTSUB 311

DTWR\_OPENREG 313

DTWR\_RTVENTRY 315

DTWR\_UPDATEENTRY 317

DTW\_ACCEPT 320

DTW\_ADD 164

DTW\_ADDQUOTE 132

DTW\_APPLET\_ALTTEXT 101

DTW\_ASSIGN 77, 183, 184

DTW\_CACHE\_PAGE 134

DTW\_CHARTOHEX 185

DTW\_COMMIT 322

DTW\_CONCAT 186

DTW\_CURRENT\_FILENAME 116

DTW\_CURRENT\_LAST\_MODIFIED 117

DTW\_DATE 138

DTW\_DEFAULT\_MESSAGE 118  
DTW\_DEFAULT\_REPORT 88  
DTW\_DELSTR 188  
DTW\_DELWORD 212  
DTW\_DIVIDE 167  
DTW\_DIVREM 169  
DTW\_EDIT\_CODES 102  
DTW\_FORMAT 171  
DTW\_GETCOOKIE 141  
DTW\_GETENV 143  
DTW\_GETINIDATA 145  
DTW\_HEXTOCHAR 190  
DTW\_HTMLENCODER 146  
DTW\_HTML\_TABLE 89  
DTW\_INSERT 192  
DTW\_INTDIV 175  
DTW\_LASTPOS 194  
DTW\_LENGTH 196  
DTW\_LOG\_LEVEL 119  
DTW\_LOWERCASE 197  
DTW\_MACRO\_FILENAME 120  
DTW\_MACRO\_LAST\_MODIFIED 121  
DTW\_MBMODE 122  
DTW\_MP\_PATH 124  
DTW\_MP\_VERSION 125  
DTW\_MULTIPLY 177  
DTW\_PAD\_PGM\_PARMS 103  
DTW\_POS 199  
DTW\_POWER 179  
DTW\_PRINT\_HEADER 126  
DTW\_QHTMLENCODER 148  
DTW\_REMOVE\_WS 127  
DTW\_REPLACE 201  
DTW\_REVERSE 203  
DTW\_ROLLBACK 323  
DTW\_RTVHANDLE 324  
DTW\_SAVE\_TABLE\_IN 104  
DTW\_SENDEMIL 149  
DTW\_SETCOOKIE 154  
DTW\_SETENV 158  
DTW\_SET\_TOTAL\_ROWS 105  
DTW\_STATIC 325  
DTW\_STRIP 204  
DTW\_SUBSTR 206  
DTW\_SUBTRACT 181  
DTW\_SUBWORD 215  
DTW\_TB\_APPENDROW 226  
DTW\_TB\_COLS 228  
DTW\_TB\_deleteCOL 229  
DTW\_TB\_DELETEROW 230  
DTW\_TB\_DLIST 232  
DTW\_TB\_DUMPV 236

DTW\_TB\_GETN 238  
DTW\_TB\_GETV 240  
DTW\_TB\_HTMLENCODER 242  
DTW\_TB\_INPUT\_CHECKBOX 244  
DTW\_TB\_INPUT\_RADIO 246  
DTW\_TB\_INPUT\_TEXT 248  
DTW\_TB\_INSERTCOL 250  
DTW\_TB\_INSERTROW 251  
DTW\_TB\_LIST 250  
DTW\_TB\_QUERYCOLNONJ 255  
DTW\_TB\_ROWS 257  
DTW\_TB\_SELECT 258  
DTW\_TB\_SETCOLS 260  
DTW\_TB\_SETN 261  
DTW\_TB\_SETV 263  
DTW\_TB\_TABLE 265  
DTW\_TB\_TEXTAREA 267  
DTW\_TERMINATE 327  
DTW\_TIME 160  
DTW\_TRANSLATE 208  
DTW\_UPPERCASE 210  
DTW\_URLESCSEQ 162  
DTW\_WORD 217  
DTW\_WORDINDEX 219  
DTW\_WORDLENGTH 221  
DTW\_WORDPOS 222  
DTW\_WORDS 224

## E

ENVVAR ステートメント 71  
    構文 15  
    説明 15  
EXEC ステートメント 72  
    構文 16  
    説明 16  
EXEC ブロック  
    構文 16  
    説明 16  
EXEC\_PATH 16  
EXEC\_SQL 331

## F

FFI 関数  
    ファイルの解放 273  
    ファイルのロック 273  
DTWF\_APPEND 274  
DTWF\_CLOSE 277  
DTWF\_DELETE 278  
DTWF\_INSERT 280  
DTWF\_OPEN 283

FFI 関数 (続き)

DTWF\_READ 273

DTWF\_REMOVE 288

DTWF\_SEARCH 290

DTWF\_UPDATE 293

DTWF\_WRITE 296

FFI 言語環境

機密保護の推奨事項 271

許可要件 272

区切り文字 272

現行ディレクトリー 270

構成規則 271

ファイル位置 270

ファイルへのアクセス 269

FFI\_PATH

アクセス、フラット・ファイル 269

機密保護の推奨事項 271

構成規則 271

構文 269

パスと *filename* パラメーターの突き合わせ 270

フラット・ファイル位置 270

FUNCTION ブロック

構文 20

説明 19

## H

HTML

形式、パスワードの入力 110

形式、ユーザー ID の入力 108

表結果の表示 89

変数名の隠蔽 73

HTML ブロック

構文 30

説明 30

HTML\_INPUT ブロック 331

HTML\_REPORT ブロック 331

## I

IF ブロック

構文 33

説明 33

IN キーワード 21, 49, 129

INCLUDE ステートメント

構文 40

説明 40

INCLUDE\_PATH 40

INCLUDE\_URL ステートメント

構文 43

説明 43

INOUT キーワード 21, 49, 129

INOUT 変数

例 28

## L

LIST ステートメント

構文 46

説明 46

LOCATION 107

LOGIN 108

## M

MACRO\_FUNCTION ブロック

構文 48

説明 48

MBCS サポート、関数の

ストリング関数 183

ワード関数 212

MESSAGE ブロック

構文 52

説明 52

## N

Net.Data 表

上限 63

定義 63

Next ボタン、RPT\_MAX\_ROWS 92

NLIST 78

NULL\_RPT\_FIELD 109

NUM\_COLUMNS 79

NUM\_ROWS 80

Nn 77

## O

OUT キーワード 21, 49, 129

## P

PASSWORD 110

Previous ボタン、RPT\_MAX\_ROWS 92

## R

REPORT ブロック

構文 57

説明 57

表変数 75

ALIGN 87

DTW\_DEFAULT\_REPORT 88

REPORT ブロック (続き)  
DTW\_HTML\_TABLE 57  
NLIST 78  
NUM\_COLUMNS 79  
NUM\_ROWS 80  
Nn 77  
RPT\_MAX\_ROWS 90  
START\_ROW\_NUM 92  
TOTAL\_ROWS 82  
RETURNS キーワード 22, 49  
RETURN\_CODE 128  
ROW ブロック  
構文 60  
説明 60  
NLIST 78  
NUM\_COLUMNS 79  
NUM\_ROWS 80  
Nn 77  
ROW\_NUM 81  
TOTAL\_ROWS 82  
Vn 84, 85  
V\_columnName 83  
ROW\_NUM 81  
RPT\_MAX\_ROWS 90

## S

SHOWSQL 111  
SQL  
隠蔽または表示 111  
コマンド、大文字小文字の指定 98  
SQL 状態の表示 112  
SQL ブロック 331  
SQL\_CODE 333  
SQL\_MESSAGE ブロック 332  
SQL\_REPORT ブロック 332  
SQL\_STATE 112  
START\_ROW\_NUM 92  
System 言語環境、パラメーターの引き渡し 25

## T

TABLE ステートメント 75  
構文 63  
説明 63  
TOTAL\_ROWS 82  
TRANSACTION\_SCOPE 113

## U

UTF-8 形式  
日付 138

## V

VLIST 84  
Vn 85  
V\_columnName 83

## W

Web レジストリー関数  
DTWR\_ADDENTRY 300  
DTWR\_CLEARREG 302  
DTWR\_CLOSEREG 303  
DTWR\_CREATEREG 304  
DTWR\_DELENTY 306  
DTWR\_DELREG 308  
DTWR\_LISTREG 309  
DTWR\_LISTSUB 311  
DTWR\_OPENREG 313  
DTWR\_RTVENTRY 315  
DTWR\_UPDATEENTRY 317  
WHILE ブロック  
構文 65  
説明 65







Printed in Japan

SB88-7411-00



日本アイ・ビー・エム株式会社  
〒106-8711 東京都港区六本木3-2-12