

Net.Data



## 解説書

バージョン 2 リリース 2



Net.Data



## 解説書

バージョン 2 リリース 2

**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、283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： VNDT-2REF-01 (この資料番号ではオーダーできません)  
Net.Data  
Reference  
Version 2 Release 2

発 行： 日本アイ・ビー・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 1999.5

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

© Copyright International Business Machines Corporation 1997, 1998. All rights reserved.

Translation: © Copyright IBM Japan 1999

# 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 . . . . .	73
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_MBMODE . . . . .	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_MP_PATH . . . . .	122
DTW_MP_VERSION . . . . .	123
DTW_PRINT_HEADER. . . . .	124
DTW_REMOVE_WS. . . . .	125
RETURN_CODE . . . . .	126
<b>第3章 Net.Data 組み込み関数 . . . . .</b>	<b>127</b>
関数名. . . . .	127
入出力パラメーター. . . . .	127
関数結果の形式化 . . . . .	128
関数パラメーターの規則 . . . . .	128
汎用関数. . . . .	129
DTW_ADDQUOTE . . . . .	130
DTW_CACHE_PAGE . . . . .	132
DTW_DATE. . . . .	135
DTW_EXIT . . . . .	136
DTW_GETCOOKIE . . . . .	137
DTW_GETENV . . . . .	139
DTW_GETINIDATA. . . . .	140
DTW_HTMLLENCODE . . . . .	141
DTW_QHTMLLENCODE . . . . .	143

DTW_SENDMAIL . . . . .	144
DTW_SETCOOKIE . . . . .	148
DTW_SETENV . . . . .	151
DTW_TIME . . . . .	152
DTW URLESCSEQ . . . . .	154
数学関数 . . . . .	156
DTW_ADD . . . . .	157
DTW_DIVIDE . . . . .	158
DTW_DIVREM . . . . .	159
DTW_FORMAT . . . . .	161
DTW_INTDIV . . . . .	164
DTW_MULTIPLY . . . . .	165
DTW_POWER . . . . .	166
DTW_SUBTRACT . . . . .	167
ストリング関数 . . . . .	169
DTW_ASSIGN . . . . .	170
DTW_CONCAT . . . . .	171
DTW_DELSTR . . . . .	172
DTW_INSERT . . . . .	173
DTW_LASTPOS . . . . .	175
DTW_LENGTH . . . . .	176
DTW_LOWERCASE . . . . .	177
DTW_POS . . . . .	178
DTW_REVERSE . . . . .	179
DTW_STRIP . . . . .	180
DTW_SUBSTR . . . . .	181
DTW_TRANSLATE . . . . .	183
DTW_UPPERCASE . . . . .	185
ワード関数 . . . . .	186
DTW_DELWORD . . . . .	187
DTW_SUBWORD . . . . .	188
DTW_WORD . . . . .	190
DTW_WORDINDEX . . . . .	191
DTW_WORDLENGTH . . . . .	192
DTW_WORDPOS . . . . .	193
DTW_WORDS . . . . .	195
表関数 . . . . .	196
DTW_TB_APPENDROW . . . . .	197
DTW_TB_COLS . . . . .	198
DTW_TB_DELETEROW . . . . .	199
DTW_TB_DELETECOL . . . . .	200
DTW_TB_DLIST . . . . .	201
DTW_TB_DUMPH . . . . .	203
DTW_TB_DUMPV . . . . .	204
DTW_TB_GETN . . . . .	205
DTW_TB_GETV . . . . .	206
DTW_TB_HTMLLENCODE . . . . .	207
DTW_TB_INPUT_CHECKBOX . . . . .	208
DTW_TB_INPUT_RADIO . . . . .	209
DTW_TB_INPUT_TEXT . . . . .	210
DTW_TB_INSERTCOL . . . . .	212
DTW_TB_INSERTROW . . . . .	213

DTW_TB_LIST . . . . .	214
DTW_TB_MAXROWS . . . . .	216
DTW_TB_QUERYCOLNONJ. . . . .	217
DTW_TB_ROWS . . . . .	218
DTW_TB_SELECT . . . . .	219
DTW_TB_SETCOLS. . . . .	221
DTW_TB_SETN . . . . .	222
DTW_TB_SETV . . . . .	223
DTW_TB_TABLE. . . . .	224
DTW_TB_TEXTAREA . . . . .	226
フラット・ファイル・インターフェース関数 . . . . .	227
フラット・ファイル・データ・ソースへのアクセス . . . . .	227
フラット・ファイル・インターフェース区切り文字 . . . . .	230
ファイルのロック . . . . .	231
DTWF_APPEND . . . . .	232
DTWF_CLOSE. . . . .	234
DTWF_DELETE . . . . .	235
DTWF_INSERT . . . . .	237
DTWF_OPEN . . . . .	239
DTWF_READ . . . . .	241
DTWF_REMOVE. . . . .	243
DTWF_SEARCH . . . . .	244
DTWF_UPDATE . . . . .	246
DTWF_WRITE. . . . .	248
Web 登録関数. . . . .	250
DTWR_ADDENTRY. . . . .	251
DTWR_CLEARREG . . . . .	252
DTWR_CLOSEREG . . . . .	253
DTWR_CREATEREG . . . . .	254
DTWR_DELENTY . . . . .	255
DTWR_DELREG . . . . .	256
DTWR_LISTREG . . . . .	257
DTWR_LISTSUB . . . . .	258
DTWR_OPENREG . . . . .	259
DTWR_RTVENTRY. . . . .	260
DTWR_UPDATEENTRY . . . . .	261
永続的なマクロ関数. . . . .	262
DTW_ACCEPT. . . . .	263
DTW_COMMIT . . . . .	265
DTW_ROLLBACK . . . . .	266
DTW_RTVHANDLE. . . . .	267
DTW_STATIC . . . . .	268
DTW_TERMINATE . . . . .	269
<b>付録A. Net.Data テクニカル・ライブラリー . . . . .</b>	<b>271</b>
<b>付録B. DB2 WWW 接続. . . . .</b>	<b>273</b>
EXEC_SQL . . . . .	273
HTML_INPUT . . . . .	273
HTML_REPORT . . . . .	273
SQL . . . . .	273
SQL_MESSAGE . . . . .	274



SQL_REPORT . . . . .	274
SQL_CODE . . . . .	275
付録C. <b>Net.Data</b> のオペレーティング・システムごとの参照 . . . . .	277
付録D. 特記事項 . . . . .	283
商標 . . . . .	284
用語集. . . . .	285
索引 . . . . .	287



---

## まえがき

動的 Web ページを作成するための IBM の開発ツールである 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の力を使用して、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 製品を使用すると、DB2、IMS、ODBC 使用可能データベース、DRDA を介してアクセスするデータベースなどの関係および非関係の両方のデータベース管理システム (DBMS) のデータを使用したり、Java、JavaScript、Perl、C、C++、および REXX などの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。製品の Net.Data ファミリーには、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と同様の機能が備えられています。

Net.Data は、Web サーバー・マシンでミドルウェアとして実行されるマクロ処理プログラムとして考えることができます。ユーザーは、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 はこれらのマクロを解釈して、ユーザーからの入力、データベースの現行状況、既存の業務論理、およびマクロ内に設計するその他の因数に基づいてカスタマイズした内容を使って、動的 Web ページを作成します。

URL 形式の要求は、実行するために、Netscape Navigator や Internet Explorer などのブラウザから、Net.Data に要求を転送する Web サーバーに送られます。Net.Data は、マクロを探して実行し、作成した関数に基づいてカスタマイズした Web ページを作成します。これらの関数では、以下のことが行えます。

- C、C++、RPG、COBOL、JAVA、Perl、または REXX 言語で作成されているが、それらに限定されないアプリケーション内部で業務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、この Web ページを Web サーバーに渡します。次に、Web サーバーはこのページをブラウザで表示するためにネットワークを介して転送します。

Net.Data は、HTTP やコモン・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ために構成されるサーバー環境で 사용할 ことができます。HTTP は、ブラウザと Web サーバー間で対話するための業界標準インターフェースであり、CGI は、Web サーバーが Net.Data のようなゲートウェイ・アプリケーションを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により、Net.Data での使用に応じて、好みのブラウザや Web サーバーを選択することができます。

改良されたパフォーマンスに応じて、Net.Data は、さまざまな 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。さらに、Net.Data は、Java サブレットとしても立ち上げることができます。

---

## 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。(英語版)

- <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。


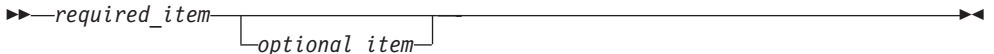
本書で説明している概念について理解するには、Web サーバー、単純な SQL ステートメント、および HTML (HTML 形式の使用法を含む) と、*Net.Data* 管理およびプログラミングの手引き の情報に精通している必要があります。

## 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方法を示しているわけではありません。例の中には、単独では実行できない断片的なコードもあります。

## 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。
- オプション項目はメインパスの下部に表示されます。

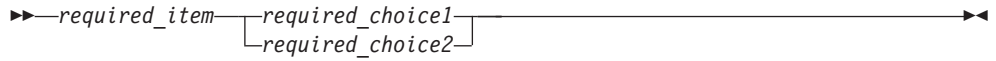
オプション項目がメインパスの上部に表示されている場合、その項目はステート

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。

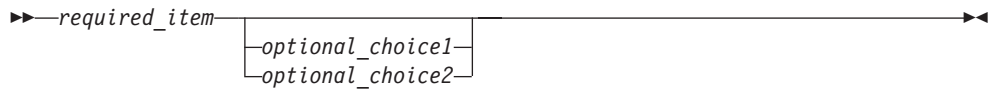


- 複数の項目からの選択が可能の場合、これらの項目は縦方向に重ねて表示されます。

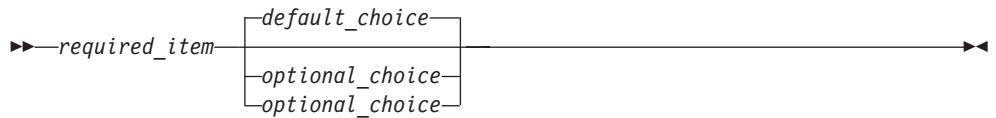
項目の中の 1 つを選択しなければならない 場合、スタックの 1 つの項目がメインパス上に表示されます。



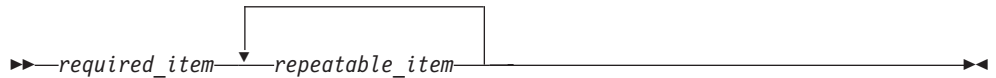
項目の中の 1 項目の選択がオプションの場合、すべてのスタックはメインパスの下部に表示されます。



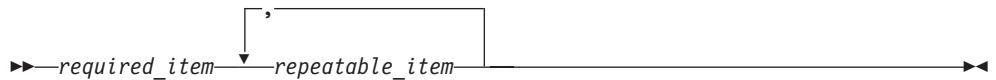
項目の 1 つがデフォルトの場合、その項目はメインパスの上部に表示され、残りの選択項目はメインパスの下部に表示されます。



- 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なければなりません。



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』

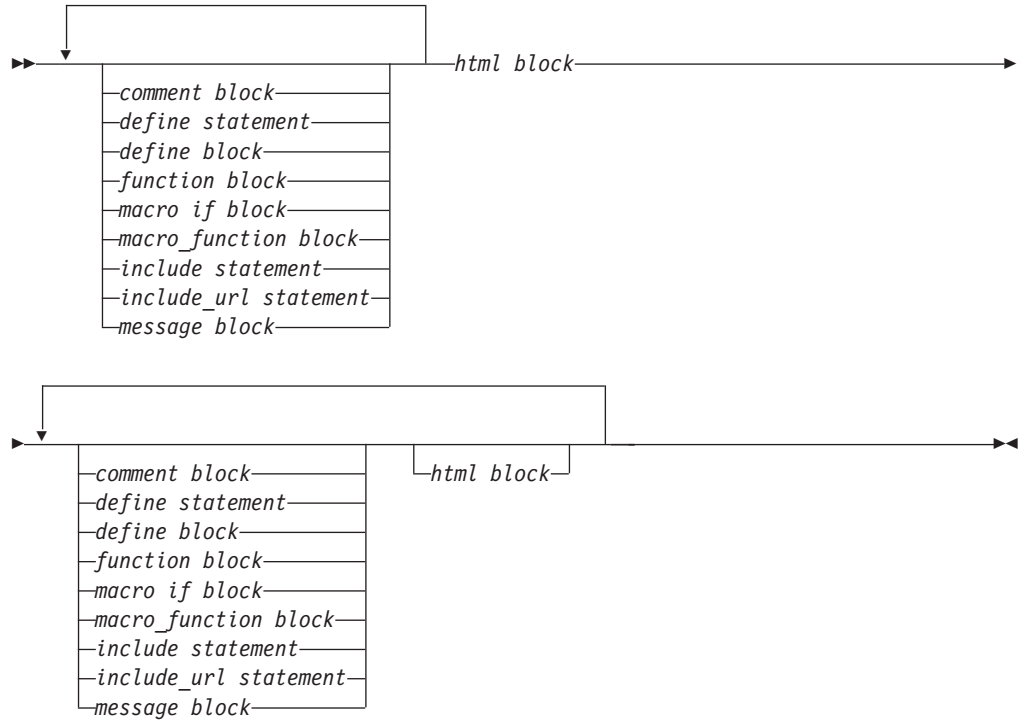
---

### 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
- HTML で処理出力を形式化して、Web ブラウザーにその出力を戻す

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。



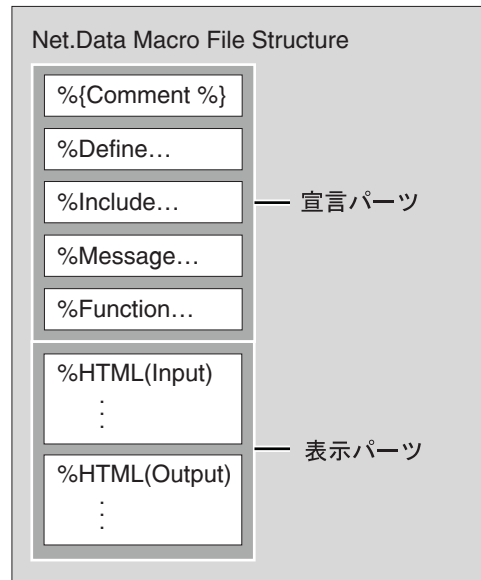


図 1. マクロの構造

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・テンプレートの形式

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 接続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 273ページの『付録B. DB2 WWW 接続』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。(英語版)

- <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：

1つまたは複数の名前を指定します。複数の名前の場合、2番目以降の名前はそれぞれピリオド (.) を使って連結します。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です。

引用符 (『』) 内のストリングには、改行文字以外のすべての文字を含めることができます。大括弧内のストリング ({ %}) 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。

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。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)。

構文：

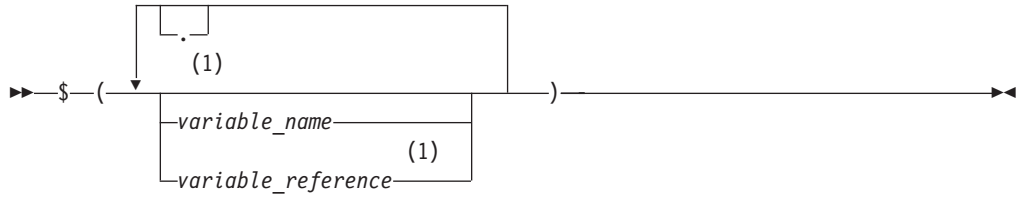


## 変数参照

目的：

変数の値を戻し、\$ と () を使って指定されます。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。

構文：



注:

1. ネストされた変数参照は、OS/2、Windows NT、および UNIX 上の Net.Data でサポートされます。

#### 例 1: 変数参照

変数 homeURL を定義した場合 :

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

`$(homeURL)` としてホーム・ページを参照し、リンクを作成することができます。

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

#### 例 2: ネストされた変数参照

ネストされた変数参照を使用して、データベースから戻された行のフィールド値を動的に参照することができます。

```
DTW_ASSIGN(INDEX, "2")
%ROW{
  $(V$(INDEX))
```

その後、ネストされた変数参照は、V2 のフィールド値である行の 2 番目のフィールドを派生させます。'V2' は定義済みの Net.Data 変数であり、データベースから戻された行の 2 番目のフィールドを表すことに注意してください。

## ストリング

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。言語構成要素で使用する際の制約事項については、それぞれの言語構成要素のストリング・パラメーター説明を参照してください。

引用符付き文字列を強調する二重引用符を指定するには、2 組の二重引用符を使用します。関数実引き数として、または比較式の用語として使用するストリングには、二重引用符を含むことができます。たとえば、ストリング値を以下のように定義する場合です。

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

`result` の値は以下ようになります。

```
"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、可変値として使用するストリングには、変数参照と関数呼び出しを含むことができます。以下の例では、関数呼び出し `myfunc2` には、変数参照と関数呼び出しを含むストリング・パラメーターがあります。

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

`Net.Data` は、ストリングを関数 `myfunc2` に渡す前に、変数参照 `$(var1)` と関数呼び出し `@myfunc()` を、ストリングのパーツとして文字どおりに解釈するのではなく解決します。

---

## 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、ほか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することができます。

### 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。

### 情況

コメントは、Net.Data マクロまたは Net.Data 初期設定ファイルの Net.Data 言語構成要素間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           %}
%}
```

#### 例 5: Net.Data 初期設定ファイル内のコメント

```
%{ changes: removed RETURN_CODE parm and DTW_DEFAULT ENVIRONMENT statement %}
...
ENVIRONMENT (DTW_SQL) dtwsq1 (IN LOCATION, DB2SSID, DB2PLAN, TRANSACTION_SCOPE)
ENVIRONMENT (DTW_ODBC) odbcd11 (IN LOCATION, TRANSACTION_SCOPE)
ENVIRONMENT (DTW_PERL) perl111 ()
ENVIRONMENT (DTW_REXX) rexxd11 ()
ENVIRONMENT (DTW_FILE) filed11 ()
ENVIRONMENT (DTW_APPLET) appld11 ()
ENVIRONMENT (DTW_SYSTEM) sysd11 ()
```



## DEFINE ブロックまたはステートメント

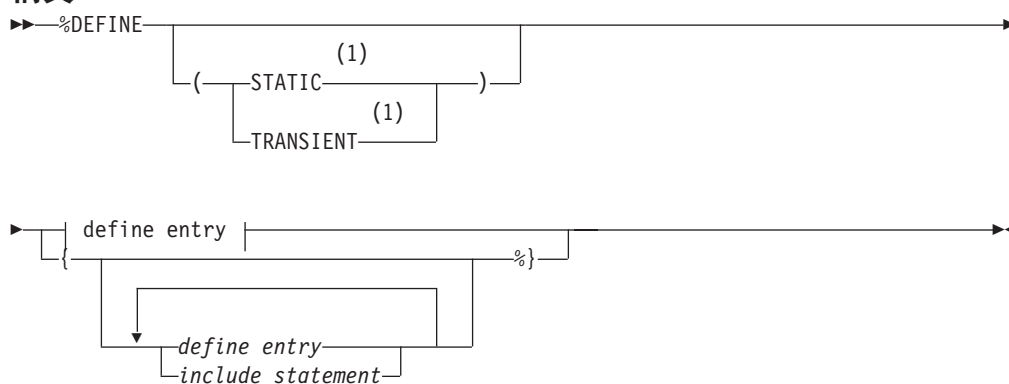
## 目的

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。

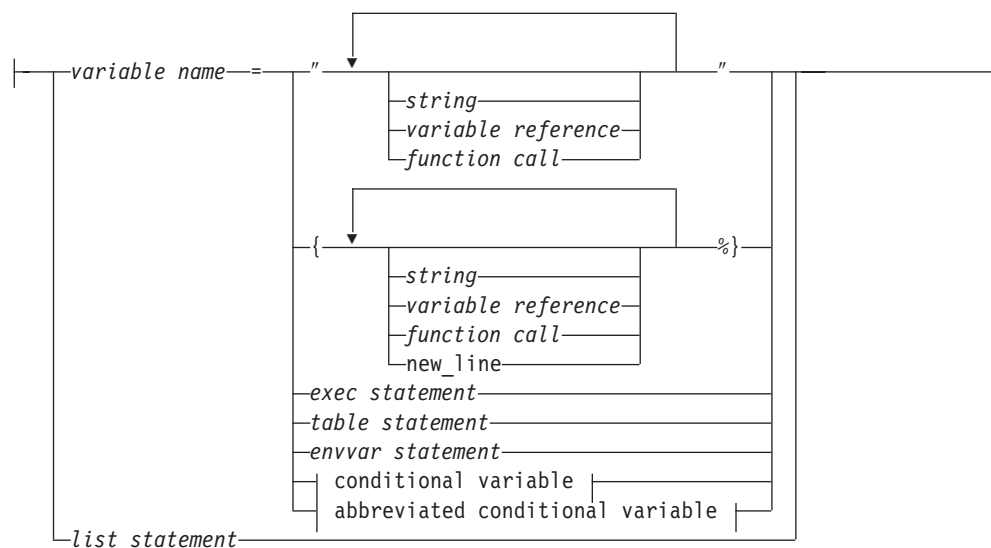
- 一度に 1 つの変数を定義するときにはステートメントを使用
- 数個の変数を定義するときはブロックを使用

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。

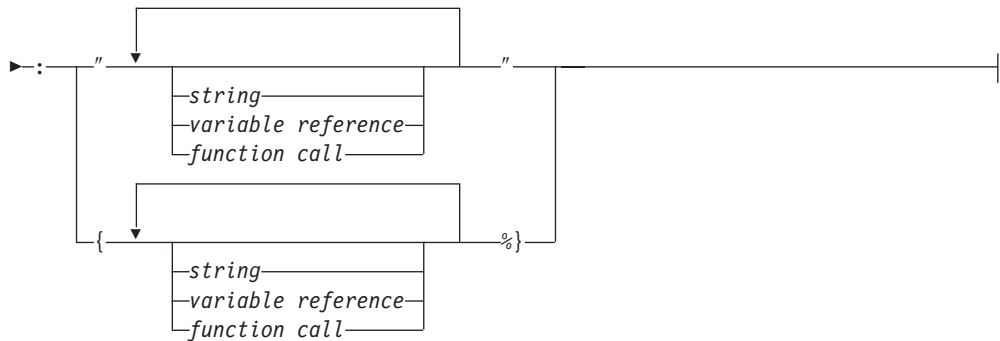
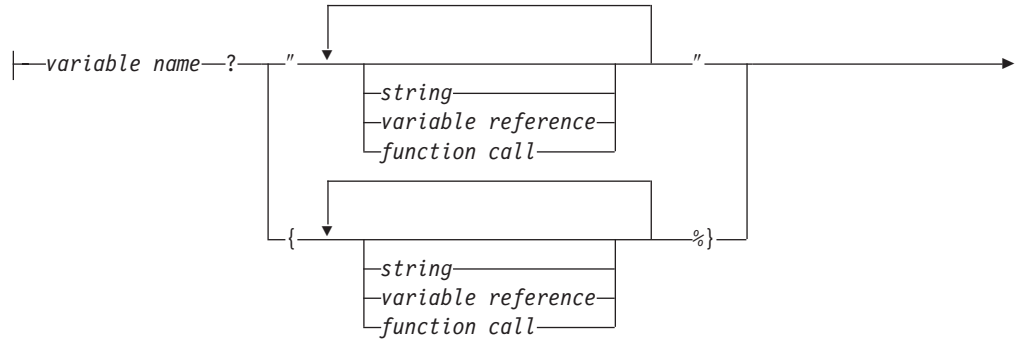
## 構文



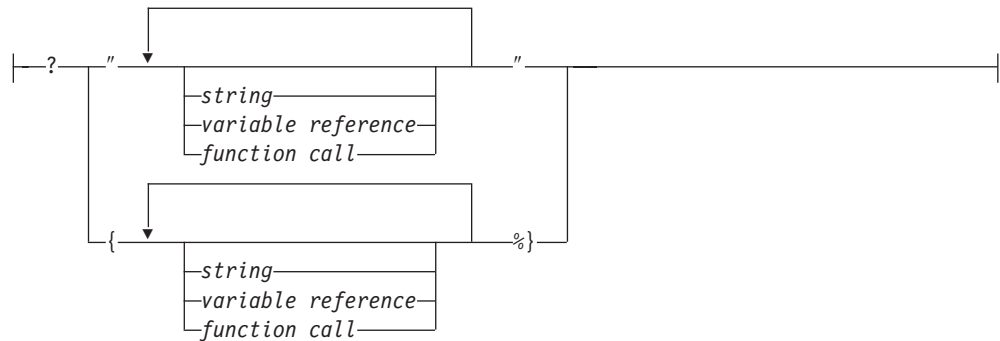
**define entry:**



**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あり、現在は、OS/400 オペレーティング・システムでのみ使用可能です。

#### 値

##### **%DEFINE**

変数を定義するキーワードです。

##### **STATIC**

永続的マクロ内でマクロを起動することによって、変数とその値を保存することを指定するキーワードです。これは、永続的マクロのデフォルトです。

## TRANSIENT

マクロを起動することによって、この変数が値を保存しないことを指定するキーワードです。これは、永続的でないマクロのデフォルトです。

### define entry:

#### variable name

1つまたは複数の名前で、2番目以降の名前はそれぞれピリオド (.) を使って連結します。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

変数の値を戻し、\$ と () を使って指定されます。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。条件変数の簡易書式です。

#### 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。

## 情況

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—▶▶

### 状況

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。

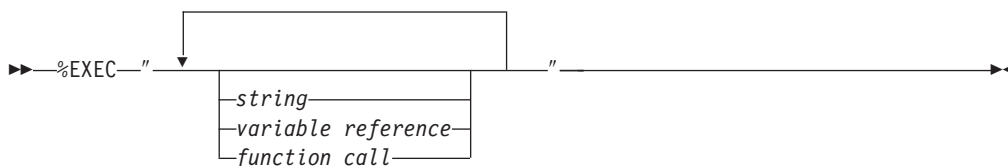
1. Net.Data 初期設定ファイルの中で EXEC\_PATH を探します。EXEC\_PATH についての詳細は、*Net.Data 管理およびプログラミングの手引き* (各オペレーティング・システム用) の構成の章を参照してください。
2. プログラムが見つからない場合、Net.Data はシステムで定義されているディレクトリーを探します。実行可能プログラムが見つかったら、Net.Data はそのプログラムを実行します。

**許可のヒント：**EXEC ステートメントまたはブロックで参照されるすべてのファイルへのアクセス権限を、Net.Data が実行するユーザー ID に必ずもたせてください。詳しくは、*Net.Data 管理およびプログラミングの手引き* (各オペレーティング・システム用) 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。

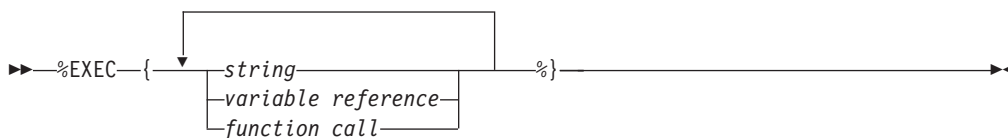
EXEC ステートメントおよびブロックは、使用する場所に応じて 2 つの異なるコンテキストで使用し、その構文も異なります。DEFINE ブロックでは EXEC ステートメントを使用し、FUNCTION ブロックでは EXEC ブロックを使用します。

### 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

変数の値を戻し、\$ と () を使って指定されます。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。

## 情況

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

Net.Data が提供する以下の言語環境では、EXEC ステートメントをサポートします。

- 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。

**関数ブロック内の 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ために使用するメソッドは、特定の言語環境によって異なります。REXX、System、および Perl の Net.Data 提供言語環境が、EXEC ブロックをサポートします。

**言語環境における特殊文字の使用:** Net.Data の言語構成要素構文と一致する文字が、関数ブロックの言語ステートメント・セクションで、構文的に無効な組み込みプログラム・コード (REXX または Perl など) のパーツとして使用されると、Net.Data 言語構成要素として間違って解釈される可能性があります。これは、マクロ内でエラーまたは予測不能の結果が発生する原因となります。

たとえば、Perl の関数が COMMENT ブロックの区切り文字である %{ を使用する可能性があります。マクロを実行すると、%{ 文字は COMMENT ブロックの始りと解釈されます。その後、Net.Data は COMMENT ブロックの終わりを捜します。関数ブロックの終わりを読みとったときに、Net.Data は COMMENT ブロックの終わりを見つけたと思います。その後、Net.Data は続けて関数の終わりを捜し、見つけることができない場合には、エラーが発生します。

以下のメソッドのいずれかで、Net.Data によって特殊文字として解釈されることなく、Net.Data の特殊文字を組み込みプログラム・コードのパーツとして使用します。

- コードをインラインに書き込むのではなく、EXEC ステートメントを使用してプログラム・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、以下のような Perl の関数には、COMMENT ブロックの区切り文字を表す文字 %{ が、Perl の言語ステートメントとして含まれています。

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

Net.Data に、%{ 文字を Net.Data の COMMENT ブロック区切り文字としてではなく、Perl のソース・コードとして確実に解釈させるには、以下の方法のいずれかで関数を再書き込みします。

- %EXEC ステートメントを使用する。

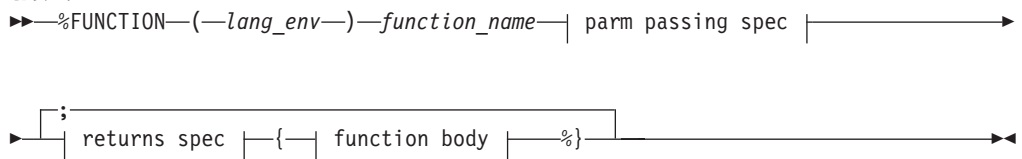
```
%function(DTW_PERL) func() {
    %EXEC{ func.pr1 %}
%}
```

- 変数参照を使用して、%{ 文字を指定する。

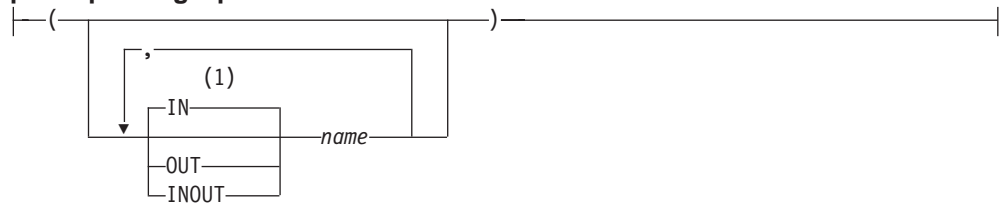
```
%define percent_openbrace = "%{"

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

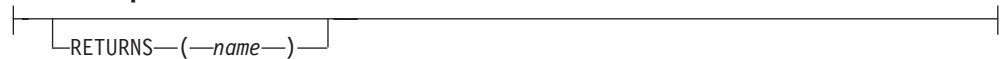
## 構文



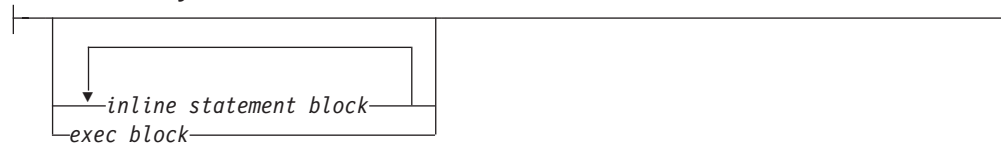
### parm passing spec:

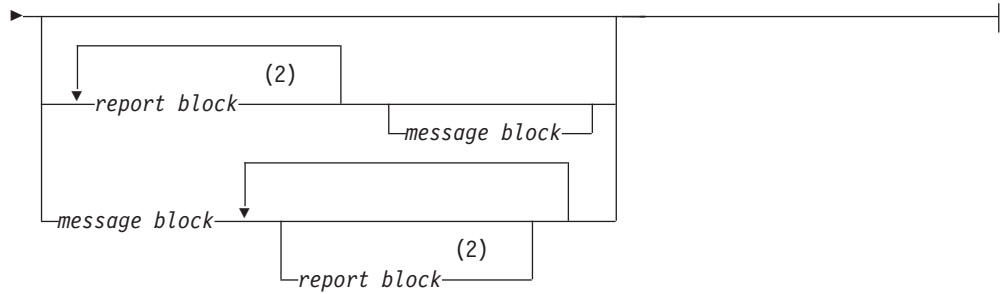


### returns spec:



### 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. 繰り返されたレポート・ブロックは、以下に対して有効です。
  - OS/390 オペレーティング・システムに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中の場合は、SQL 言語環境および ODBC 言語環境。
  - OS/400、OS/2、Windows NT、および UNIX オペレーティング・システムのすべての言語環境を呼び出す関数。

**値**

**%FUNCTION**

Net.Data がマクロ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stringにすることができます。

**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string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。

**returns spec:**

## RETURNS

関数が完了したあとで、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。

## 情況

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 および Windows NT の場合 : 64KB
- AIX の場合 : 256KB
- OS/390 の場合 : 256KB
- OS/400 の場合 : 256KB
- インライン・ステートメント・ブロック内の SQL ステートメントの長さは、以下のようになることができます。データベースの制限が異なる可能性があります。使用しているデータベースのドキュメンテーションを参照して、制約事項が短いかどうかを決定してください。Net.Data の制限と異なるかどうか、IBM の DB2 データベースの制限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 Universal Database V6 またはそれ以降 : 64 KB
      - DB2 Universal Database V5.2 またはそれ以前 : 32 KB
  - OS/390 の場合 : 32 KB
  - OS/400 の場合 : 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.prl" %}  
}%  
%HTML(INPUT) {  
    @today()  
}%
```

- Perl プログラム today.prl:

```
$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。

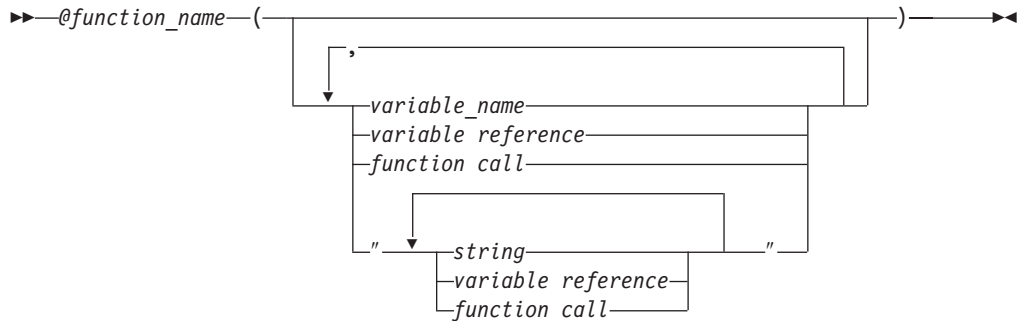


## 関数呼び出し (@)

### 目的

指定した引き数で、FUNCTION ブロック、MACRO\_FUNCTION ブロック、または組み込み関数を起動します。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字符串です。

#### variable name

1 つまたは複数の名前で、2 番目以降の名前はそれぞれピリオド (.) を使って連結します。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

変数の値を戻し、\$ と () を使って指定されます。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。

### 状況

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 パラメーター値または 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を示すマクロ

```
%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>
%}
```

結果の出力 :

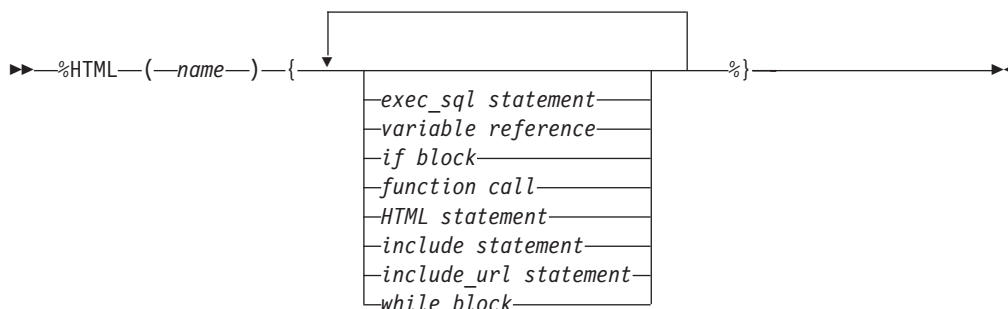
```
initial values:
a = initial value of a
value at start of function:
x = initial value of a
value at start of report block:
x = new value of a
value at end of report block:
x = newest value of a
value after function call:
a = newest value of a
```

## HTML ブロック

### 目的

Web ページを表示する方法を定義します。Net.Data を起動すると、実行する HTML ブロックの名前は URL で指定されます。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と、HTML および JavaScript などのすべて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、ピリオドを含めて、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文字列です。

#### 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。273ページの『付録B. DB2 WWW 接続』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

変数の値を返し、`$` と `()` を使って指定されます。たとえば、`VAR='abc'` の場合、`$(VAR)` は値 `'abc'` を返します。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。条件付き文字列処理を行います。条件リストの文字列値が整数を表す文字列であり、しかもこれらの文字列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。

## 情況

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:** 名前にピリオドを含む 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、非整数値 (たとえば、浮動小数点数値) の数値比較をサポートしません。

**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。

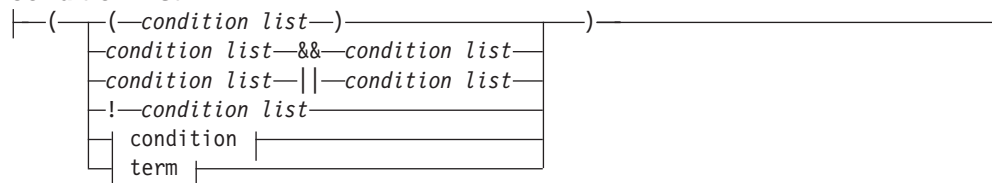
```
%IF block
...
%HTML block
...
%IF block
```

最高 1024 の IF ブロックをネストすることができます。

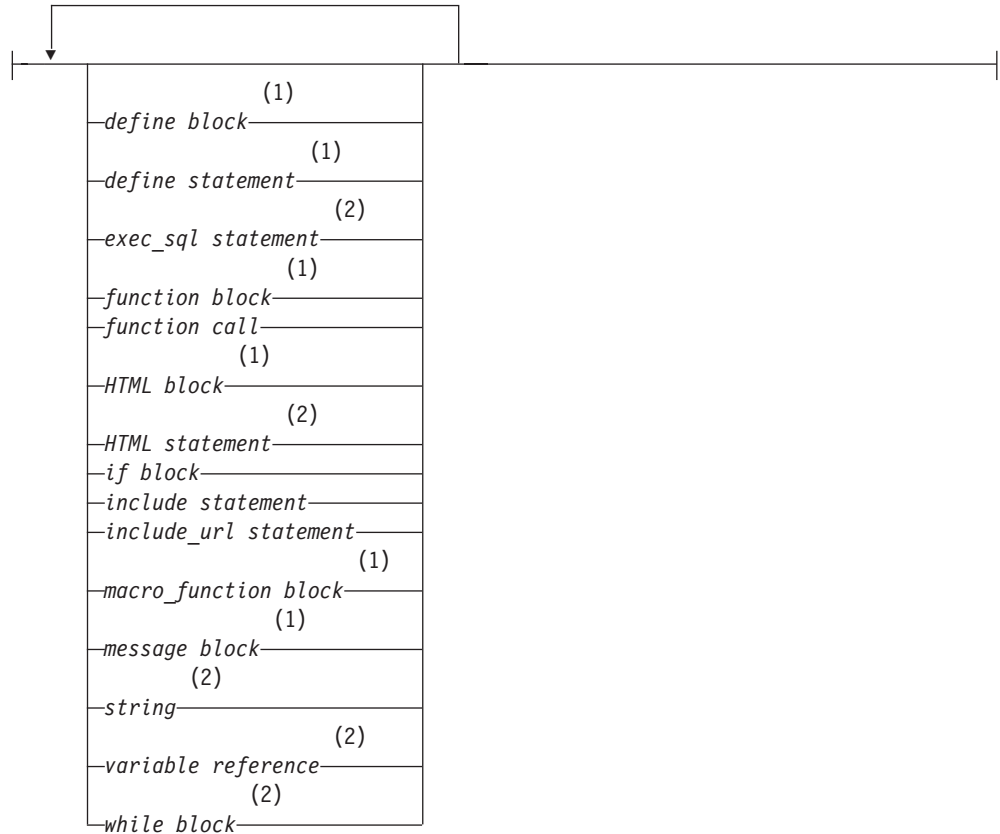
### 構文

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

#### condition list:



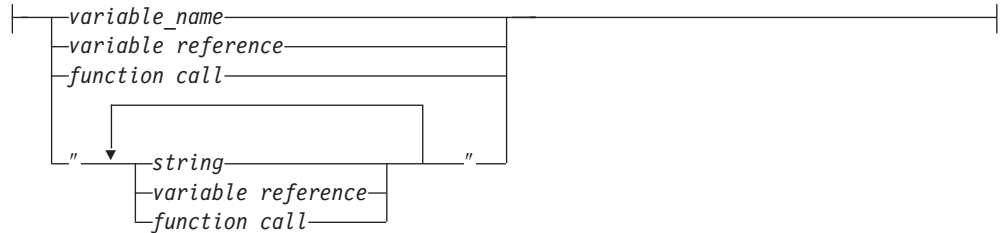
#### statement\_block:



#### condition:

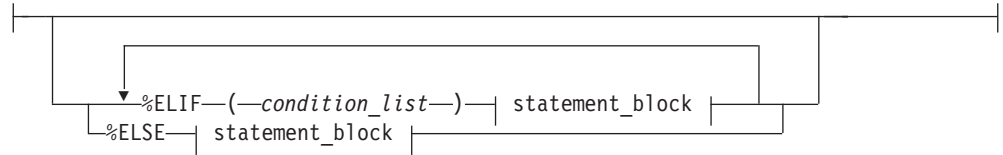


#### term:



#### 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

次の有効な Net.Data マクロ構成要素です。ダイアグラム・ノーツおよび制約事項を参照して、マクロ構成要素が有効なコンテキストを決定してください。

#### 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。273ページの『付録B. DB2 WWW 接続』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

変数の値を戻し、\$ と () を使って指定されます。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。

### **情況**

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

最高 1024 の IF ブロックをネストすることができます。

## 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。

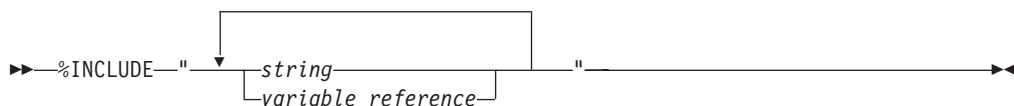
Net.Data は、マクロの処理時に一度だけ INCLUDE ステートメントを実行し、マクロ内の INCLUDE ステートメントのロケーションで組み込みファイルの内容を挿入します。組み込みファイルの名前の中のあらゆる変数参照は、組み込みファイルの内容を実行するときではなく、INCLUDE ステートメントを最初に実行するときに解決されます。

INCLUDE ステートメントが ROW ブロックまたは WHILE ブロック内にある場合は、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、ROW ブロックまたは WHILE ブロックを初めて実行するときに 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：**INCLUDE ステートメントで参照されるすべてのファイルへのアクセス権限を、Net.Data が実行するユーザー ID に必ずもたせてください。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

変数の値を戻し、\$ と () を使って指定されます。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。

## 情況

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。

最高 10 の INCLUDE ステートメントをネストすることができます。

## 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、マクロの処理時に一度だけ 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で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中のあらゆる変数参照は、組み込みファイルの内容を実行するときではなく、INCLUDE\_URL ステートメントを最初に実行するときに解決されます。

INCLUDE\_URL ステートメントが ROW ブロックまたは WHILE ブロック内にある場合は、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、ROW ブロックまたは WHILE ブロックを初めて実行するときに 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

変数の値を戻し、\$ と () を使って指定されます。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。

### 状況

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
- ストリング
- 変数参照

INCLUDE\_URL ファイルには、以下のファイル・サイズの制限があります。

- OS/2 と Windows NT の場合 : 64 KB
- AIX の場合 : 256 KB
- OS/390 の場合 : 256 KB

最高 10 の INCLUDE\_URL ステートメントをネストすることができます。

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、ストリング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)"
}
```

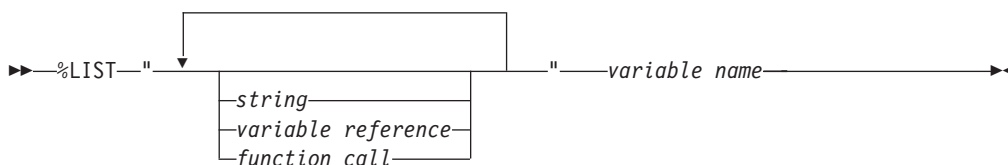
この例では、マクロの `hotpic.mac` が呼び出され、`custno` が変数として送られます。  
`string` がスラッシュで始まっている場合、`Net.Data` は、ローカル Web サーバーから  
`INCLUDE` ファイルを取り出します。

# 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

変数の値を戻し、\$ と () を使って指定されます。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

1 つまたは複数の名前で、2 番目以降の名前はそれぞれピリオド (.) を使って連結します。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。

## 状況

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
- ストリング

## 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)" : ""  
%}
```

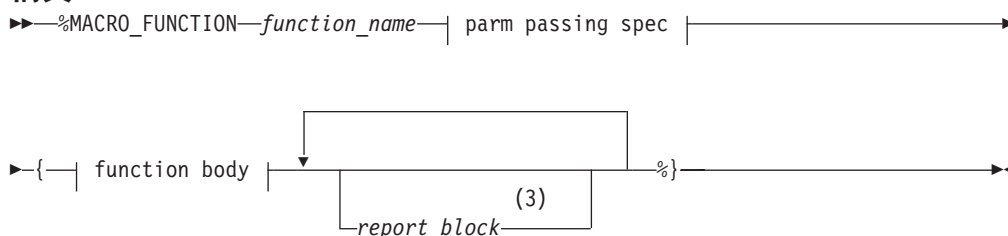
## MACRO FUNCTION ブロック

## 目的

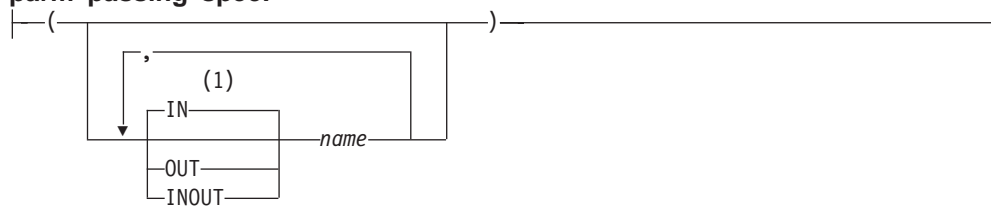
Net.Data マクロから呼び出すことのできるサブルーチンを定義します。

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。

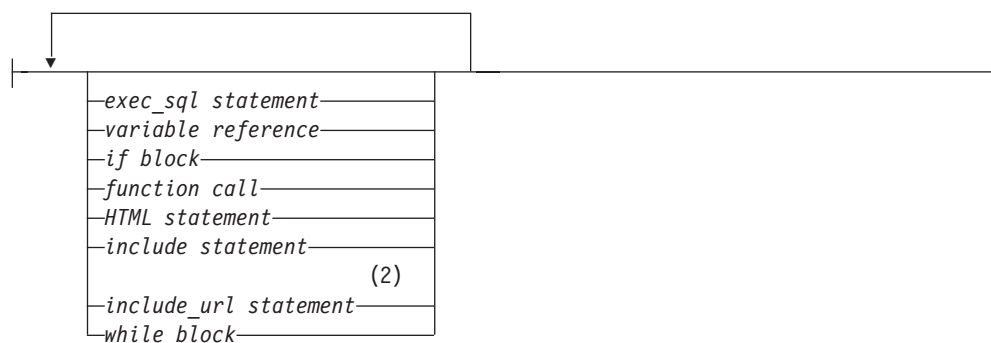
## 構文



**parm passing 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. INCLUDE\_URL ステートメントは、OS/400 ではサポートされません。
3. MACRO\_FUNCTION ブロック内の複数の 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。繰り返されたレポート・ブロックは、すべての言語環境を呼び出す関数に対して有効です。

## 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 の表または結果セットを表すことができます。

### 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。273ページの『付録B. DB2 WWW 接続』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

変数の値を返し、\$ と () を使って指定されます。たとえば、VAR='abc' の場合、\$(VAR) は値 'abc' を返します。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。

### **情況**

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 ではサポートされません。
- 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

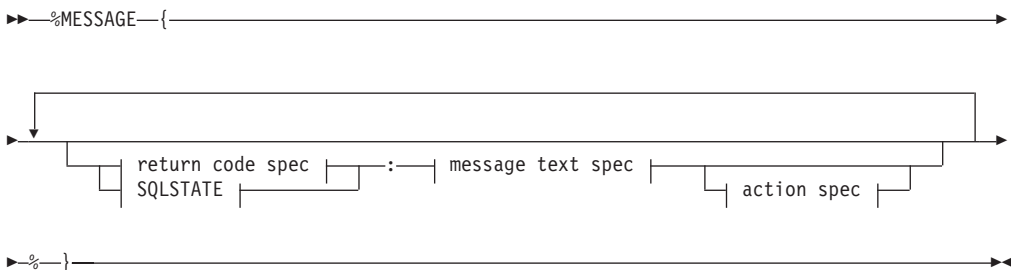
### 目的

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



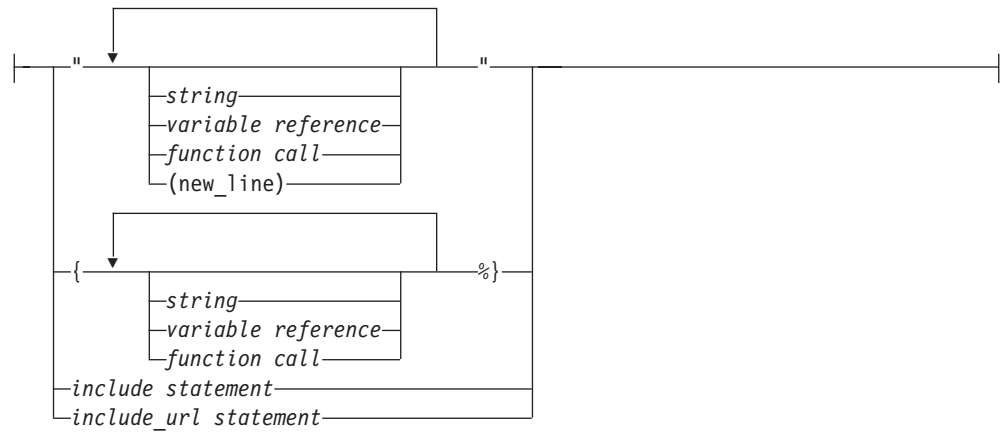
#### return code spec:



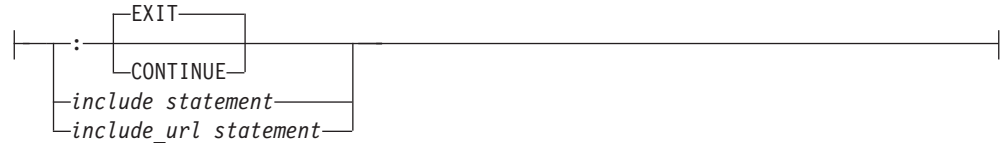
#### SQLSTATE:



### 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 の形式をもつ 5 文字 (バイト) の英数字です。ただし、cc はクラスを示し、sss はサブクラスを示します。

#### **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**

変数の値を戻し、\$ と () を使って指定されます。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。

## 情況

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"
%}

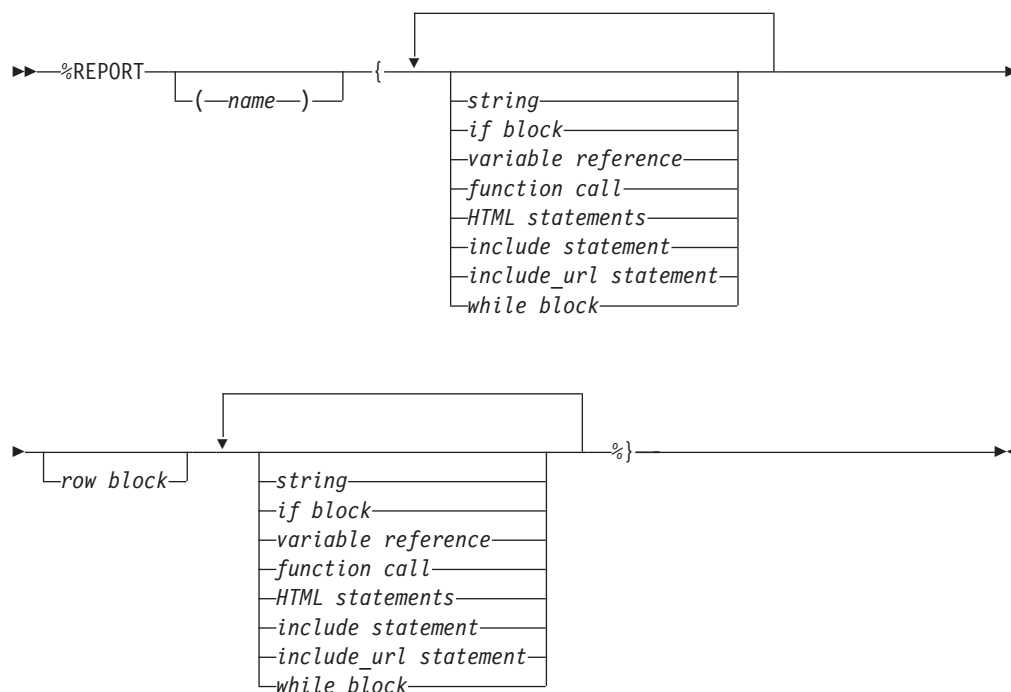
```

## 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` の表または結果セットを表します。詳しくは、*Net.Data* 管理およびプログラミングの手引き の 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**

変数の値を戻し、\$ と () を使って指定されます。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。

### **情況**

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
- MACRO\_FUNCTION 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 ブロック内では許可されません。

## 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>
    %}
```

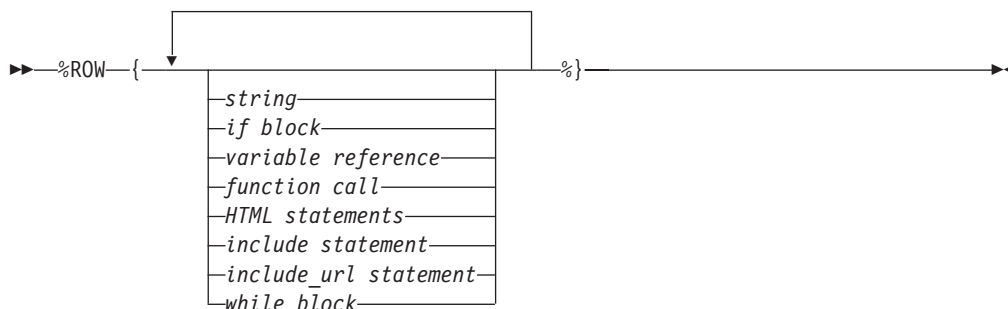
表の名前を選択すると、*name.mac* Net.Data マクロの *details* HTML ブロックが呼び出され、URL のパーツとして 2 つの値が送信されます。この例では、*name.mac* で値を使用すると、名前に関する詳細をルックアップすることができます。

## 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

変数の値を戻し、\$ と () を使って指定されます。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。

### 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。

## 情況

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。

## TABLE ステートメント

### 目的

関連データの集合である変数を定義します。この変数には、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。

### 構文

▶▶%TABLE | upper limit |

#### upper limit:

|  
| ( number )  
| ALL

### 値

#### %TABLE

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の定義を指定するキーワードです。

#### upper limit

表に入れることのできる行数。上限値を指定しなければ、表に行を無制限に含むことはできます。

#### number

0 から 9 までの値の数字のストリングです。0 の値により、表の行数を無制限にすることができます。

#### ALL

表の行数を無制限にすることができるキーワードです。

### 情況

TABLE 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

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。

- コメント・ブロック
- 数字

### 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

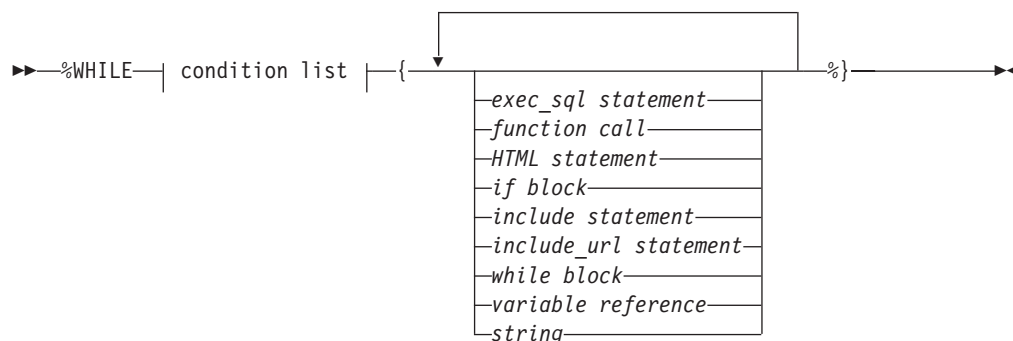
```
%DEFINE myTable3=%TABLE(ALL)
```

# WHILE ブロック

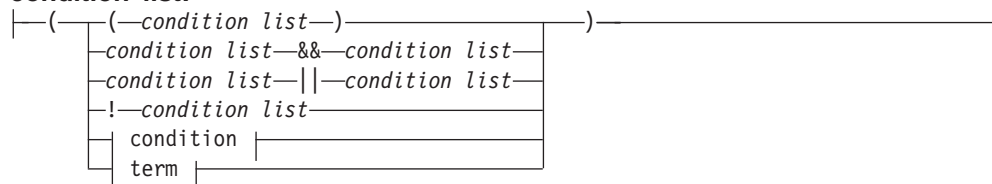
## 目的

条件付きストリング処理に基づいて、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



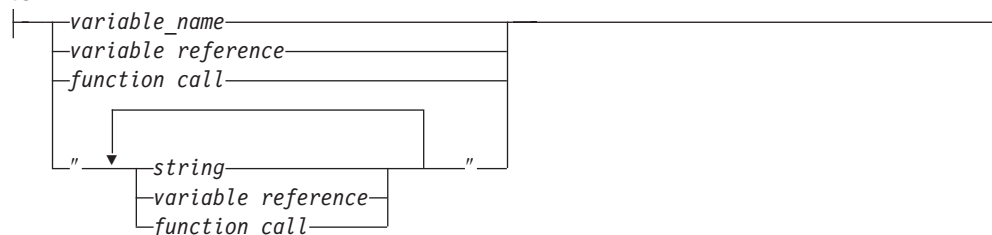
### condition list:



### 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。273ページの『付録B. DB2 WWW 接続』 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 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

変数の値を戻し、\$ と () を使って指定されます。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

1つまたは複数の名前で、2番目以降の名前はそれぞれピリオド (.) を使って連結します。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。

## 情況

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>$(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。

- 73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 ブロックまたは ROW ブロック内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 Operation 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の中で定義します。

## 条件変数

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

テスト結果が真だった場合に使用する値です。

**falseValue**

テスト結果が偽だ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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。

**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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
-----	-------	------	--------	--------	-----	-----	--------

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 照会は以下のようになります。

```
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 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

表変数には、値のセットおよび関連した列名が含まれます。配列内の各要素が行です。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** ブロックの中で、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。

### 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。

**制約事項:** これらの変数の値は、DEFINE セクションでは定義しないでください。

- 77ページの『*N<sub>n</sub>*』
- 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<sub>columnName</sub>*』
- 85ページの『*V<sub>n</sub>*』
- 84ページの『VLIST』

## Nn

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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` ブロックおよび `ROW` ブロックで参照できます。

これらの変数は定義済みの変数で、値を変更することはできません。これらの変数を変数参照として使用し、参照したい列の番号を指定します。

### 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、170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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ブロックおよび ROW ブロックで参照できます。

この変数は事前定義変数であり、その値は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:** 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 {
...
%}
%}
%}
```

ALIGN が YES に設定され、列名と列名の間にスペースを使った、列名のリスト。

**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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ブロックおよび ROW ブロックで参照できます。

この変数は事前定義変数であり、その値は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:** 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	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 ブロックおよび ROW ブロックで参照できます。

この変数は事前定義変数であり、その値は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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ブロックでしか参照できません。

この変数は事前定義変数であり、その値は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:** 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

照会が戻る行数の合計。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。

この変数は事前定義変数であり、その値は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。

### 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：**この変数を使用するのは、以下のデータベース言語環境でのみです。

- 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 displayed: $(NUM_ROWS)<BR>
Names found: $(TOTAL_ROWS)
%}
```



## V\_columnName

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ブロック内でのみ参照できます。

これらの変数は定義済みの変数で、値を変更することはできません。これらの変数を変数参照として使用し、参照したい列の名前を指定します。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ブロック内でのみ参照できます。デフォルトの区切り記号はスペースです。

この変数は事前定義変数であり、その値は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:** 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

現在の行の、指定された列番号  $n$  の値。

V<sub>n</sub> は、ROW ブロック内でのみ参照できます。

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>
Found $(NUM_ROWS) models matching your description.
%}
```

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。それぞれの変数には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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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** : `Net.Data` が埋め込み要件を計算するために、表内のすべての列の最大長を決定する必要があるときのように、必要な場合にのみ `ALIGN` を使用します。このプロセスは、パフォーマンスに悪い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` ステートメントを使用して、または `@DTW_ASSIGN()` 関数で、この変数の値を指定します。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

Net.Data が、REPORT ブロックがない関数のために、デフォルト・レポートを生成するかどうかを決定します。この変数を YES に設定すると、Net.Data はデフォルト・レポートを作成します。NO に設定すると、Net.Data はデフォルト・レポートの作成を抑制します。デフォルト・レポートの抑制は、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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 値

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

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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

関数 REPORT ブロック内で処理する表の行数、あるいは、REPORT ブロックが指定されていない場合には、デフォルト・レポートの生成中の表の行数を指定します。

データベース言語環境では、この変数を使用して、戻される行数を制限します。これによって、大きな結果セットのパフォーマンスを十分に改善することができます。この変数は START\_ROW\_NUM と一緒に使用して、大きな結果セットをもつ照会を、より小さい表である、独自の HTML ページの各表に分割します。

**OS/400、Windows NT、OS/2、および UNIX のユーザー**：この変数を言語環境に渡すためには、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含めます。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* (各オペレーティング・システム用) の構成の章を参照してください。

**OS/390 ユーザー**：RPT\_MAX\_ROWS は、マクロ内で定義されると、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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

関数 REPORT ブロック内で処理される表の開始行番号、あるいは、REPORT ブロックが指定されていない場合には、デフォルト・レポートの生成中の表の開始行番号を指定します。

データベース言語環境ではこの変数を使用して、結果セット内の開始行を決定し、処理を開始します。大きい結果セットのパフォーマンスを十分に改善するには、この変数を RPT\_MAX\_ROWS と一緒に使用して、大きな結果セットをも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 のユーザー**：この変数を言語環境に渡すためには、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含めます。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* (各オペレーティング・システム用) の構成の章を参照してください。

**OS/390 ユーザー**：START\_ROW\_NUM は、マクロ内で定義されると、暗黙的にデータベース言語環境に渡されます。

DEFINE ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 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
%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。それぞれの変数には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\_MBMODE』
- 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 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	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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

**要件** : マクロ内のこの変数の値が効力を発するには、SQL 言語環境の ENVIRONMENT ステートメントにリストされなければなりません。

### 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	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** : マクロ内のこの変数の値が効力を発するには、SQL 言語環境の ENVIRONMENT ステートメントにリストされなければなりません。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 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	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。その列に対して変数 DTW\_EDIT\_CODES で編集コード 'Y' を指定すると、結果として生じる表で対応する列が、日付 '11/26/98' を表す文字ストリングとして表示されます。

**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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 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\_MBMODE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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、この変数をマクロの中で使用しても、設定および現在の設定のオーバーライドができます。

DEFINE ステートメントを使用して、または `@DTW_ASSIGN()` 関数で、この変数の値を指定します。

**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の中のこの変数を無視します。

### 値

`DTW_MBMODE="YES"|"NO"`

表 13. `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\_SAVE\_TABLE\_IN

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

**OS/400、OS/2、Windows NT、および UNIX のユーザー**：この変数を言語環境に渡すためには、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含めます。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**：DTW\_SET\_TOTAL\_ROWS は、マクロ内で定義されると、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はすべての行を検索しなければならないので、パフォーマンスに影響を与えます。

## 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>$(NUM_ROWS) returned. Your query is limited to $(TOTAL_ROWS) rows.  
%}  
%}
```



# LOCATION

AIX	HP-UX	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** : マクロ内のこの変数の値が効力を発するには、SQL 言語環境の ENVIRONMENT ステートメントにリストされなければなりません。

DEFINE ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

## 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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** : OS/400 では、DATABASE 変数が定義されていないか、あるいは、値 `"*LOCAL"` に設定されている場合は、LOGIN と PASSWORD の両方を無視します。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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

## 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	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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 値

NULL\_RPT\_FIELD="null\_char"

表 18. NULL\_RPT\_FIELD 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 :** OS/400 では、DATABASE 変数が定義されていないか、あるいは、値 `"*LOCAL"` に設定されている場合は、LOGIN と PASSWORD の両方を無視します。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 ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。DTW\_SHOWSQL が Net.Data 構成ファイルで YES に設定されている場合にのみ、SHOWSQL を使用することができます。DTW\_SHOWSQL 構成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 (各オペレーティング・システム用) の構成の章を参照してください。

DEFINE ステートメントを使用して、または @DTW\_ASSIGN() 関数で、この変数の値を指定します。

## 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。

構成ファイルでは :  
DTW\_SHOWSQL YES

マクロでは :  
%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

構成ファイルでは :  
DTW\_SHOWSQL YES

マクロでは :  
SHOWSQL: <INPUT TYPE="radio" NAME="SHOWSQL" VALUE="YES"> Yes  
          <INPUT TYPE="radio" NAME="SHOWSQL" VALUE="" CHECKED> No

## SQL\_STATE

AIX	HP-UX	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、その値は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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 変数が効力を発するためには、Net.Data 構成ファイルの ENVIRONMENT ステートメントにその変数を含めます。その後、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 上の Net.Data から、または IBM の DataJoiner を使って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出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 マクロから出され、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\_MP\_PATH』
- 123ページの『DTW\_MP\_VERSION』
- 124ページの『DTW\_PRINT\_HEADER』
- 125ページの『DTW\_REMOVE\_WS』
- 126ページの『RETURN\_CODE』

## DTW\_CURRENT\_FILENAME

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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、その値は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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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、その値は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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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:** 関数が非ゼロの戻り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 以外の戻り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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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、その値は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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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、その値は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\_MP\_PATH

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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、その値は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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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、その値は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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 (`DTW_PRINT_HEADER = "NO"`) を生成するために、`DTW_PRINT_HEADER` を使用している場合は、`DTW_REMOVE_WS` を `"NO"` に設定しなければなりません。

`DEFINE` ステートメントを使用して、または `@DTW_ASSIGN()` 関数で、この変数の値を指定します。

### 値

`DTW_PRINT_HEADER="YES"|"NO"`

表 23. `DTW_PRINT_HEADER` の値

値	説明
YES	<code>Net.Data</code> は、HTTP ヘッダーとしてテキスト <code>Content-type: text/html</code> を印刷する。YES はデフォルト。
NO	<code>Net.Data</code> 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` マクロがクッキーを送信できるようにすることです。クッキー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` がクッキー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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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 (DTW\_PRINT\_HEADER = "NO") を生成するために、DTW\_PRINT\_HEADER を使用している場合は、DTW\_REMOVE\_WS を "NO" に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。マクロの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 24. 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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戻りコード。 `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:** 戻りコードが 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 以外の戻り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。129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。156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。169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。186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。196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。227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 登録関数**は、Web 登録の操作を行います。250ページの『Web 登録関数』を参照してください。
- **永続的なマクロ関数**は、Net.Data のトランザクション処理をサポートします。262ページの『永続的なマクロ関数』を参照してください。

一部の関数パラメーターは、整数 または浮動 の型をもっていると説明されていますが、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 組み込み関数でない関数に DTW 接頭部を使用すると、予測不能な振る舞いが生じる場合があります。

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か、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、他のカテゴリーには適合しません。汎用関数は次のとおりです。

- 130ページの『DTW\_ADDQUOTE』
- 132ページの『DTW\_CACHE\_PAGE』
- 135ページの『DTW\_DATE』
- 136ページの『DTW\_EXIT』
- 137ページの『DTW\_GETCOOKIE』
- 139ページの『DTW\_GETENV』
- 140ページの『DTW\_GETINIDATA』
- 141ページの『DTW\_HTMLENCODER』
- 143ページの『DTW\_QHTMLENCODER』
- 144ページの『DTW\_SENDMAIL』
- 148ページの『DTW\_SETCOOKIE』
- 151ページの『DTW\_SETENV』
- 152ページの『DTW\_TIME』
- 154ページの『DTW\_URLESCSEQ』

## DTW\_ADDQUOTE

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。この置き換えは、ストリングに単一引用符が含まれている場合に、SQL ステートメントを正しく処理できるようにするために必要です。

この関数をすべての SQL INPUT ステートメントに使用する場合を考えてみます。たとえば、以下の例のように O'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 ('O'Brien', 'Patrick')
```

DTW\_ADDQUOTE 関数を使用すれば、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')
```

### 形式

@DTW\_ADDQUOTE(stringIn, stringOut)

@DTW\_rADDQUOTE(stringIn)

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

### 値

表 25. DTW\_ADDQUOTE 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>

### 例

**例 1:** エクストラ単一引用符を OUT パラメーターに追加する。

```
@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する。



```
@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する。

```
@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する。

```
%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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X							

### 目的

マクロの関数位置の後に含まれているすべての HTML 出力のキャッシングを開始します。この関数を起動すると、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ています。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。

### マクロの DTW\_CACHE\_PAGE 関数の位置の決定

- 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 を指定してすべての HTML 出力をキャッシュします。この技法を使用すれば、新規レポート・ブロックを追加するときに、マクロの保守操作がやりやすくなります。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、この方法を使用すれば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 HTML 出力セクションに組み込むことができます。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。

### 形式

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

### 値

表 26. DTW\_CACHE\_PAGE 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。

表 26. DTW\_CACHE\_PAGE パラメーター (続き)

パラメーター	用途	説明
<i>age</i>	IN	<p>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> よりも古ければ、このページはブラウザに送信されません。</p> <p><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れば、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。</p>
<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>

## 例

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

```
%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 ブロックに入れる。

```
%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 を動的に検索する。

```
%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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

指定形式の現行システム日付を戻します。

### 形式

@DTW\_DATE(format, stringOut)

@DTW\_DATE(stringOut)

@DTW\_rDATE(format)

@DTW\_rDATE()

### 値

表 27. DTW\_DATE 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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、これまでにマクロが作成したページがブラウザに送信されるようにします。

**パフォーマンス上のヒント:** 出力が生成されたときに、DTW\_EXIT を使用してマクロの処理を停止します。こうすることで、Net.Data がファイル全体を処理するための時間が節約できます。

**重要!** マクロ全体が構文上正しいことを確認してから、DTW\_EXIT 関数を追加してください。DTW\_EXIT() を使用すると、Net.Data がこの関数に対する呼び出しに出会ったときに、マクロの処理を停止します。これによって、DTW\_EXIT() 関数が処理された後に発生するエラーをキャッチすることを回避することができます。

### 形式

@DTW\_EXIT()

### 例

**例 1:** マクロを終了する。

```
%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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

読みとるクッキーの名前を指定し、クッキーの値を戻します。

**ヒント**：2 つの別な HTTP 要求でクッキーを定義し検索します。クッキーが見えるのは、クライアントに送信された後だけであるため、マクロが同じ HTTP 要求で定義されたクッキーを入手しようとする、予測不能の結果を受け取る場合があります。

**OS/400 および OS/390 のユーザー**：クッキー値が URL スタイルのエンコード (たとえば、"%20") である場合は、クッキー値が戻される前にデコードされます。

『Persistent Client State HTTP Cookies』の最新の情報については、Netscape の指定を参照してください。

### 形式

@DTW\_GETCOOKIE(IN cookie\_name, OUT cookie\_value)

@DTW\_rGETCOOKIE(IN cookie\_name)

### 値

表 28. DTW\_GETCOOKIE 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	クッキー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	関数によって検索されたクッキーの値を含む変数。ユーザー状態情報など。

### 例

**例 1**：ユーザー ID およびパスワード情報を含むクッキーを検索する。

```
@DTW_GETCOOKIE("mycookie_name_for_userID", userID)
@DTW_GETCOOKIE("mycookie_name_for_password", password)
```

**例 2**：ユーザー情報を収集する前に、ユーザーのためのクッキーが存在するかどうかを決定する。

```
%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、クッキー NDC\_name が存在するかどうかを検査します。そのクッキーが存在する場合は、ブラウザーに個人情報設定のあいさつが表示されます。クッキーが存在しない場合は、ユーザー名を求めるプロンプトが表示され、そのユーザー名が HTML remember セクションに通知されます。このセクションでは、以下のように、クッキー NDC\_name にユーザー名を設定します。

```

%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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

指定された環境変数の値を戻します。 ENVVAR 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。

### 形式

@DTW\_GETENV(envVarName, envVarValue)

@DTW\_rGETENV(envVarName)

### 値

表 29. DTW\_GETENV 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>envVarName</i>	IN	変数またはリテラル・ストリング。
ストリング	<i>envVarValue</i>	OUT	<i>envVarName</i> 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・ストリングが戻されます。

### 例

**例 1:** OUT パラメーター上の PATH ステートメントの値を戻す。

```
@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戻す。

```
@DTW_rGETENV(myPath)
```

- 入力: myPath = "PATH"
- 戻り : "/usr/bin"

**例 3:** サーバーのプロトコルに値を戻す。

```
The server is @DTW_rGETENV("SERVER_PROTOCOL").
```

- 戻り : "HTTP/1.0"

## DTW\_GETINIDATA

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。値が見付からないと、ヌル・ストリングが返されます。

**制約事項:** 非 OS/400 オペレーティング・システムの場合は、ENVIRONMENT ステートメントと同様、構成パス変数 (MACRO\_PATH, EXEC\_PATH, and INCLUDE\_PATH) も、この呼び出しで検索することはできません。OS/400 オペレーティング・システムの場合、この制約事項は ENVIRONMENT ステートメントにのみ適用されます。

### 形式

@DTW\_GETINIDATA(iniVarName, iniVarValue)

@DTW\_rGETINIDATA(iniVarName)

### 値

表 30. DTW\_GETINIDATA 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。

### 例

**例 1:** Net.Data パス変数値を返す。

@DTW\_GETINIDATA(myEnvVarName, myEnvVarValue)

- 入力: myEnvVarName = "FFI\_PATH"
- 戻り: myEnvVarValue = "D:¥FFI"

## DTW\_HTMLENCODE

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

### 目的

HTML 文字拡張コードを使用し、選択済みの文字をエンコードします。この関数を使用すれば、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で、より小 (<) およびより大 (>) などの文字を Web ページ内に表示することができます。そうでない場合は、このような文字は、ブラウザーによって、HTML タグのコンポーネントとして解釈されます。

表31 は、DTW\_HTMLENCODE 関数によってエンコードされた文字を示しています。

表 31. 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;
#	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;
:	コロンの	&#58;
;	セミコロンの	&#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;
\	逆スラッシュ	&#92;
^	caret	&#94;
{	左中括弧	&#123;
	縦線	&#124;
}	右中括弧	&#125;
~	ティルド	&#126;

### 形式

@DTW\_HTMLENCODE(stringIn, stringOut)

@DTW\_rHTMLENCODE(stringIn)

## 値

表 32. DTW\_HTMLENCODE 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入力ストリングが含まれている変数。

## 例

例 1: スペース文字をエンコードする。

```
@DTW_HTMLENCODE(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する。

```
@DTW_rHTMLENCODE("X <= 10")
```

- 戻り: "X&#32;&#60;&#61;&#32;10"

## DTW\_QHTMLENCODE

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

### 目的

@DTW\_HTMLENCODE と同じ機能を実行しますが、単一引用符文字 (') から &#39; へのエンコードも行います。DTW\_QHTMLENCODE が使用する HTML 文字エスケープ・コードが、141ページの表31 に示されています。

### 形式

@DTW\_QHTMLENCODE(stringIn, stringOut)

@DTW\_rQHTMLENCODE(stringIn)

### 値

表 33. DTW\_QHTMLENCODE 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	特定の文字がエンコード済み HTML 文字エスケープ・コードによって置き換えられた <i>stringIn</i> の変更形式が含まれている変数。

### 例

**例 1:** アポストロフィとスペースをエンコードする。

```
@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する。

```
@DTW_rQHTMLENCODE("John's & Jane's")
```

- 戻り: "John&#39;s&#32;&#38;&#32;Jane&#39;s"

## DTW\_SENDMAIL

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

この機能は、任意選択の構成変数 `DTW_SMTP_SERVER` を処理します。この変数で SMTP サーバーを指定し、電子メール・メッセージを送信するために使用します。このパラメーターの値は、ホスト名、IP アドレス、OS/390 サーバーを指定している場合には、そのノードおよび名前のいずれでも構いません。この変数が定義されない場合は、`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。

各国語は、**OS/400、OS/2、Windows NT、UNIX** 用に発行されます：SMTP サーバーは、米国 ASCII 文字などの 7 ビット・データしか受け入れません。ユーザーのメッセージが 8 ビット文字である場合は、ESMTP サーバーを指定することをお勧めします。ESMTP サーバーは、8 ビット文字を受け入れます。`Net.Data` はユーザーの 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、すべての 8 ビット文字を電子メール・メッセージから除去します。

### 形式

@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 34. DTW\_SENDMAIL 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・ストリングです。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>• Name &lt;user@domain&gt;</li><li>• &lt;user@domain&gt;</li><li>• user@domain</li></ul>

表 34. DTW\_SENDMAIL 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>recipient</i>	IN	<p>このメッセージを送信する電子メール・アドレスを指定する変数またはリテラル・ストリングです。この値には、複数の受信側を、コンマ (,) で区切って入れることができます。このパラメーターは必須です。有効な <i>recipient</i> の形式は、以下のとおりです。</p> <ul style="list-style-type: none"> <li>• Name &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	<p>電子メール・メッセージのテキストが入る変数またはリテラル・ストリングです。このパラメーターは必須です。</p>
ストリング	<i>subject</i>	IN	<p>件名行のテキストが入る変数またはリテラル・ストリングです。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。</p>
ストリング	<i>CarbonCopy</i>	IN	<p>電子メール・アドレス、すなわち、追加の受信側の名前と電子メール・アドレスが入る変数またはリテラル・ストリングです。この値には、複数の追加の受信側を、コンマ (,) で区切って入れることができます。有効な受信側の形式については、<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。</p>
ストリング	<i>BlindCarbonCopy</i>	IN	<p>電子メール・アドレス、すなわち、追加の受信側の名前とアドレスが入る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、複数の追加の受信側を、コンマ (,) で区切って入れることができます。有効な受信側の形式については、<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。</p>

表 34. DTW\_SENDMAIL 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>ReplyTo</i>	IN	このメッセージに対する応答を受け取るべき電子メール・アドレスが入る変数またはリテラル・ストリングです。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、以下のとおりです。 <ul style="list-style-type: none"> <li>• Name &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> の組織名が入る変数またはリテラル・ストリングです。これはオプション・パラメーターです。

## 例

### 例 1: 簡単な電子メール・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、以下の情報をもつ電子メール・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 に関する情報は、システムの日付と時刻関数を使用して構成され、SMTP 固有の日付形式で形式化されます。

### 例 2: 複数の受信側、カーボン・コピーとブラインド・カーボン・コピーの受信側、および会社名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、以下の情報をもつ電子メール・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 形式のインターフェースを介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 形式のインターフェースを介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、件名、およびメッセージを入力する形式が表示されます。ユーザーが「**電子メールの送信 (Send E-mail)**」ボタンをクリックすると、メッセージが、HTML(sendemail) セクションで指定された受信側に送信されます。このセクションは DTW\_SENDMAIL を呼び出し、Web フォーム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・メッセージの内容も決定します。電子メール・メッセージ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決定するマクロ

```

%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、自動化された電子メール・メッセージを指定されたカスタマーのグループに送信します。このカスタマーのグループは、カスタマー・データベースからの SQL 照会の結果によって決定されます。SQL 照会は、カスタマーの電子メール・アドレスも検索します。電子メールの内容は *message* の値で決定され、静的でも動的でも構いません (たとえば、別の 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

クッキー名、値、および有効期限や機密保護要件などのオプションを定義します。

クッキーを検索するには、関数 DTW\_GETCOOKIE() を使用します。クッキーの定義方法については、137ページの『DTW\_GETCOOKIE』を参照してください。

### ヒント：

- 2つの別な HTTP 要求でクッキーを定義し検索します。クッキーが見えるのは、クライアントに送信された後だけであるため、マクロが同じ HTTP 要求で定義されたクッキーを入手しようとすると、予測不能の結果を受け取る場合があります。
- 分かりやすくするために、セミコロン、コンマ、およびスペースをクッキー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には、Net.Data の関数 DTW\_rURLESCSEQ を使用して、DTW\_SETCOOKIE に渡す前に特殊文字が入っているストリングを処理してください。たとえば、

```
@DTW_SETCOOKIE("my_cookie_name", @DTW_rURLESCSEQ("my cookie value"))
```

### 制約事項：

- クライアント Web ブラウザーが Java スクリプトをサポートしない場合には、ブラウザはクッキーを設定しません。
- DTW\_SETCOOKIE は Java スクリプト・コードを生成するため、<SCRIPT> HTML 要素または <NOSCRIPT> HTML 要素内で DTW\_SETCOOKIE を呼び出してはいけません。

### 形式

```
@DTW_SETCOOKIE(IN cookie_name, IN cookie_value, IN advanced_options)
```

```
@DTW_SETCOOKIE(IN cookie_name, IN cookie_value)
```

### 値

表 35. DTW\_SETCOOKIE 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	クッキーの名前を指定する変数またはリテラル・ストリングです。
ストリング	<i>cookie_value</i>	IN	クッキーの値を指定する変数またはリテラル・ストリングです。

表 35. DTW\_SETCOOKIE 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す。クッキーを定義するために使用します。任意選択の属性には、以下のようなものがあります。</p> <p><b>expires = date</b> クッキー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。日付の有効期限が切れた後、クッキーを保管したり検索することはできません。構文： <i>weekday,%20DD-month-YYYY%20HH:MM:SS%20GMT</i></p> <p>ここで、</p> <p><i>weekday</i> 曜日のフルネームを指定します。</p> <p><i>DD</i> 月の日付を数値で指定します。</p> <p><i>month</i> 月を 3 文字の省略形で指定します。</p> <p><i>YYYY</i> 年を 4 文字の数字で指定します。</p> <p><i>HH:MM:SS</i> タイム・スタンプを時間、分、および秒で指定します。</p> <p><b>domain = domain_name</b> クッキーのドメイン属性を指定します。ドメイン属性の突き合わせで使用します。</p> <p><b>path = path</b> クッキーが有効なドメインで URL のサブセットを指定します。</p> <p><b>secure</b> クッキーが、保護されたチャネルを介してのみ HTTPS サーバーに送信されることを指定します。</p> <p><b>secure</b> オプションを指定しないと、クッキーは保護されていないチャネルを介して送信される可能性があります。 <b>secure</b> オプションでは、ブラウザがクッキーを暗号化する必要も、 <b>DTW_SETCOOKIE</b> ステートメントが入っているページを確実に SSL を介して送信するようにする必要もありません。</p> <p>拡張機能オプションに関する追加情報については、<a href="http://home.netscape.com">http://home.netscape.com</a> (英語版) で Netscape のクッキーの指定を参照してください。</p>

## 例

**例 1:** 保護拡張機能オプションを使用して、ユーザー ID およびパスワード情報が入っているクッキーを定義する。

```
@DTW_SETCOOKIE("mycookie_name_for_userID", "User1")
@DTW_SETCOOKIE("mycookie_name_for_password", "sd3dT", "secure")
```

**例 2:** 有効期限拡張機能オプションが入っているクッキーを定義する。

```
@DTW_SETCOOKIE("mycookie_name_for_userID", "User1",  
    "expires=Wednesday%2001-Dec-2010%2000:00:00")  
@DTW_SETCOOKIE("mycookie_name_for_password", "sd3dT",  
    "expires=Wednesday,%2001-Dec-2010%2000:00:00;secure")
```

関数呼び出しは 1 つの行になければなりません。この例では、紙面上の都合で行を分割しています。

**例 3:** ユーザー情報を収集する前に、ユーザーのためのクッキーが存在するかどうかを判別する。

```
%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、クッキー NDC\_name が存在するかどうかを検査します。そのクッキーが存在する場合は、ブラウザーに個人情報設定のあいさつが表示されます。クッキーが存在しない場合は、ブラウザーがユーザー名を求めるプロンプトを出し、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、ユーザーの名前をクッキー NDC\_name に記録します。

```
%HTML(remember) {  
    <html>  
    <body>  
    <H1>Net.Data Club>  
    @DTW_SETCOOKIE("NDC_name", name, "expires=Wednesday,%2001-Dec-2010%2000:00:00;path=/")  
    <p>Thank you.  
    <p><a href="welcome">Come back</a>  
    </body>  
    </html>  
    %}
```

## DTW\_SETENV

AIX	HP-UX	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。前の値が見付からないと、ヌル・ストリングを戻します。

### 形式

@DTW\_SETENV(envVarName, envVarValue, prevValue)

@DTW\_rSETENV(envVarName, envVarValue)

### 値

表 36. DTW\_SETENV 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>envVarName</i>	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	<i>envVarValue</i>	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	<i>prevValue</i>	OUT	前の環境変数値が含まれている変数。

### 例

**例 1:** 前のパスの値を戻す。

```
@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 の値を割り当てる。

```
@DTW_rSETENV("PATH",  
"myPath")
```

- 入力 : envVarName = "PATH", envVarValue = "myPath"
- 戻り: "myPreviousPath", PATH = "myPath"

## DTW\_TIME

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

### 目的

現行システム時刻を指定形式で戻します。

### 形式

@DTW\_TIME(stringIn, stringOut)

@DTW\_TIME(stringOut)

@DTW\_rTIME(stringIn)

@DTW\_rTIME()

### 値

表 37. DTW\_TIME 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 - 拡張時刻形式 (24 時間クロックを使用する hh:mm:ss.ccc。ただし、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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

URL で選択済みの文字をエスケープ・コードで置き換えます。他のマクロまたは HTML ブロックを参照するためにユーザーのマクロが使用する URL にこれらのコードが表示されるときには、この関数を使用して、表38 にリストされた文字をエンコードします。

表 38. URL の文字エスケープ・コード

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
\	逆スラッシュ	%2F
:	コロンの	%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

### 形式

@DTW\_URLESCSEQ(stringIn, stringOut)

@DTW\_rURLESCSEQ(stringIn)



## 値

表 39. DTW\_URLESCSEQ 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。

## 例

**例 1:** *string1* 内のスペースとアンパーサンド文字をエスケープ・コードで置き換え、その結果を *string2* に割り当てる。

```
@DTW_URLESCSEQ(string1,string2)
```

- 入力: *string1* = "Guys & Dolls"
- 戻り: *string2* = "Guys%20%26%20Dolls"

**例 2:** スペースとアンパーサンド文字をエスケープ・コードで置き換える。

```
@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 'at' 文字をエスケープ・コードで置き換える。

```
%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%20O'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、その名前と電子メール・アドレスに対して指定された値が、URL の照会ストリング内を流れ、Net.Data に fullrpt.mac マクロの入力セクションを実行させます。

---

## 数学関数

これらの関数を使用すれば、数値計算を行うことができます。

**数学関数に対する NLS の考慮事項：** Net.Data は、Net.Data が実行されている Web サーバーで指定された地域の設定に基づいて、数値に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 (,) に指定されている場合は、コンマを使用して 10 進数データをフォーマットします。Net.Data は以下の設定を使用して、小数点を指定するためにどの文字を使用するかを決定します。

**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 サーバーが実行する LOCALE

**OS/400 オペレーティング・システムの場合：**

- V4R2 または以降のリリース：プロセスが実行されているユーザー・プロファイルによって指定されます。
- V4R1 または以前のリリース：QDECFMT システム値から検索できます。

以下の関数は、数値計算に使用できます。

- 157ページの『DTW\_ADD』
- 158ページの『DTW\_DIVIDE』
- 159ページの『DTW\_DIVREM』
- 161ページの『DTW\_FORMAT』
- 164ページの『DTW\_INTDIV』
- 165ページの『DTW\_MULTIPLY』
- 166ページの『DTW\_POWER』
- 167ページの『DTW\_SUBTRACT』

## DTW\_ADD

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

### 目的

2 つのパラメーターの値を追加します。

### 形式

@DTW\_ADD(number1, number2, precision, result)

@DTW\_ADD(number1, number2, result)

@DTW\_rADD(number1, number2, precision)

@DTW\_rADD(number1, number2)

### 値

表 40. DTW\_ADD 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

最初のパラメーターの値を 2 番目のパラメーターの値で割ります。

### 形式

@DTW\_DIVIDE(number1, number2, precision, result)

@DTW\_DIVIDE(number1, number2, result)

@DTW\_rDIVIDE(number1, number2, precision)

@DTW\_rDIVIDE(number1, number2)

### 値

表 41. DTW\_DIVIDE 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> を <i>number2</i> で割った結果が含まれている変数。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

最初のパラメーターを 2 番目のパラメーターで割り、剰余を戻します。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。

### 形式

@DTW\_DIVREM(number1, number2, precision, result)

@DTW\_DIVREM(number1, number2, result)

@DTW\_rDIVREM(number1, number2, precision)

@DTW\_rDIVREM(number1, number2)

### 値

表 42. DTW\_DIVREM 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> を <i>number2</i> で割ったときの剰余が含まれている変数。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。*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。その他の任意のオプションを指定すると、数値は次のような規則に従って形式化されます。

- *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。
- *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超えている場合は、*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。
- また、*expp* および *expt* パラメーターも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。
- *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。
- 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。

### 形式

@DTW\_FORMAT(*number*, *before*, *after*, *expp*, *expt*, *precision*, *result*)

@DTW\_FORMAT(*number*, *before*, *after*, *expp*, *expt*, *result*)

@DTW\_FORMAT(*number*, *before*, *after*, *expp*, *result*)

@DTW\_FORMAT(*number*, *before*, *after*, *result*)

@DTW\_FORMAT(*number*, *before*, *result*)

@DTW\_FORMAT(*number*, *result*)

@DTW\_rFORMAT(*number*, *before*, *after*, *expp*, *expt*, *precision*)

@DTW\_rFORMAT(*number*, *before*, *after*, *expp*, *expt*)

@DTW\_rFORMAT(*number*, *before*, *after*, *expp*)

@DTW\_rFORMAT(number, before, after)

@DTW\_rFORMAT(number, before)

@DTW\_rFORMAT(number)

## 値

表 43. DTW\_FORMAT 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>expp</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。
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。

## 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

最初のパラメーターを 2 番目のパラメーターで割り、結果の整数部分を返します。

### 形式

@DTW\_INTDIV(number1, number2, precision, result)

@DTW\_INTDIV(number1, number2, result)

@DTW\_rINTDIV(number1, number2, precision)

@DTW\_rINTDIV(number1, number2)

### 値

表 44. DTW\_INTDIV 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> を <i>number2</i> で割った結果の整数部分が含まれている変数。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

2 つのパラメーターを乗算し、結果を返します。

### 形式

@DTW\_MULTIPLY(number1, number2, precision, result)

@DTW\_MULTIPLY(number1, number2, result)

@DTW\_rMULTIPLY(number1, number2, precision)

@DTW\_rMULTIPLY(number1, number2)

### 値

表 45. DTW\_MULTIPLY 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

最初のパラメーターに 2 番目のパラメーターを累乗し、結果を戻します。

### 形式

@DTW\_POWER(number1, number2, precision, result)

@DTW\_POWER(number1, number2, result)

@DTW\_rPOWER(number1, number2, precision)

@DTW\_rPOWER(number1, number2)

### 値

表 46. DTW\_POWER 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> を <i>number2</i> 乗した結果が含まれている変数。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

最初のパラメーターの値から 2 番目のパラメーターの値を引き、結果を返します。

### 形式

@DTW\_SUBTRACT(number1, number2, precision, result)

@DTW\_SUBTRACT(number1, number2, result)

@DTW\_rSUBTRACT(number1, number2, precision)

@DTW\_rSUBTRACT(number1, number2)

### 値

表 47. DTW\_SUBTRACT 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> の差が含まれている変数。

### 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。

- 170ページの『DTW\_ASSIGN』
- 171ページの『DTW\_CONCAT』
- 172ページの『DTW\_DELSTR』
- 173ページの『DTW\_INSERT』
- 175ページの『DTW\_LASTPOS』
- 176ページの『DTW\_LENGTH』
- 177ページの『DTW\_LOWERCASE』
- 178ページの『DTW\_POS』
- 179ページの『DTW\_REVERSE』
- 180ページの『DTW\_STRIP』
- 181ページの『DTW\_SUBSTR』
- 183ページの『DTW\_TRANSLATE』
- 185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 103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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

入力変数の値を出力変数に割り当てます。この関数を使用してマクロの変数を変更します。

### 形式

@DTW\_ASSIGN(stringOut, stringIn)

### 値

表 48. DTW\_ASSIGN 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。

### 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\_CONCAT

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

### 目的

2 つのストリングを連結します。

### 形式

@DTW\_CONCAT(stringIn1, stringIn2, stringOut)

@DTW\_rCONCAT(stringIn1, stringIn2)

### 値

表 49. DTW\_CONCAT 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	' <i>stringIn1stringIn2</i> ' ストリングが含まれている変数。ただし、 <i>string1</i> は <i>string2</i> と連結されています。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

長さ文字の  $n$  番目の文字から、指定されたストリングのサブストリングを削除します。

### 形式

@DTW\_DELSTR(stringIn, n, length, stringOut)

@DTW\_DELSTR(stringIn, n, stringOut)

@DTW\_rDELSTR(stringIn, n, length)

@DTW\_rDELSTR(stringIn, n)

### 値

表 50. DTW\_DELSTR 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	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。

### 例

#### 例 1:

```
@DTW_DELSTR("abcde", "3", "2", result)
```

- 戻り: result = "abe"

#### 例 2:

```
@DTW_rDELSTR("abcde", "4", "1")
```

- 戻り: "abce"

## DTW\_INSERT

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

### 形式

```
@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 51. DTW\_INSERT 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。

### 形式

@DTW\_LASTPOS(stringIn1, stringIn2, n, position)

@DTW\_LASTPOS(stringIn1, stringIn2, position)

@DTW\_rLASTPOS(stringIn1, stringIn2, n)

@DTW\_rLASTPOS(stringIn1, stringIn2)

### 値

表 52. DTW\_LASTPOS 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

ストリングの長さを戻します。

### 形式

@DTW\_LENGTH(stringIn, length)

@DTW\_rLENGTH(stringIn)

### 値

表 53. DTW\_LENGTH 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。

### 例

#### 例 1:

```
@DTW_LENGTH("abcdefgh", result)
```

- 戻り: result = "8"

#### 例 2:

```
@DTW_rLENGTH("")
```

- 戻り: "0"

## DTW\_LOWERCASE

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

### 形式

@DTW\_LOWERCASE(stringIn, stringOut)

@DTW\_rLOWERCASE(stringIn)

@DTW\_mLOWERCASE(stringMult1, stringMult2, ..., stringMultn)

### 値

表 54. DTW\_LOWERCASE 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>

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。

### 形式

@DTW\_POS(stringIn1, stringIn2, n, nOut)

@DTW\_POS(stringIn1, stringIn2, nOut)

@DTW\_rPOS(stringIn1, stringIn2, n)

@DTW\_rPOS(stringIn1, stringIn2)

### 値

表 55. DTW\_POS 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。

### 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\_REVERSE

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

### 目的

入力ストリングを反転します。

### 形式

@DTW\_REVERSE(stringIn, stringOut)

@DTW\_rREVERSE(stringIn)

### 値

表 56. DTW\_REVERSE 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	反転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	<i>stringIn</i> の反転形式が含まれている変数。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

先行ブランク、後書きブランク、またはその両方を入力ストリングから除去します。

### 形式

@DTW\_STRIP(stringIn, option, stringOut)

@DTW\_STRIP(stringIn, stringOut)

@DTW\_rSTRIP(stringIn, option)

@DTW\_rSTRIP(stringIn)

### 値

表 57. DTW\_STRIP 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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、入力ストリングのサブストリングを戻します。

### 形式

@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 58. DTW\_SUBSTR 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。デフォルトは空白です。
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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

入出力変換表 (*tableI* および *tableO*) を使用して、入力ストリングの文字を変換します。 *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ければ、*stringIn* パラメーターが大文字に変換されます。 *tableI* と *tableO* がリストに入っていれば、*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。

### 形式

```
@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 59. DTW\_TRANSLATE 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

文字列を大文字で戻します。

### 形式

@DTW\_UPPERCASE(stringIn, stringOut)

@DTW\_rUPPERCASE(stringIn)

@DTW\_mUPPERCASE(stringMult1, stringMult2, ..., stringMultn)

### 値

表 60. DTW\_UPPERCASE 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>

### 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 103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。

- 187ページの『DTW\_DELWORD』
- 188ページの『DTW\_SUBWORD』
- 190ページの『DTW\_WORD』
- 191ページの『DTW\_WORDINDEX』
- 192ページの『DTW\_WORDLENGTH』
- 193ページの『DTW\_WORDPOS』
- 195ページの『DTW\_WORDS』



## DTW\_DELWORD

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

### 目的

入力ストリングのサブストリングを戻します。length で指定されたワード数だけ、ワード *n* からワードが削除されます。

### 形式

@DTW\_DELWORD(stringIn, n, length, stringOut)

@DTW\_DELWORD(stringIn, n, stringOut)

@DTW\_rDELWORD(stringIn, n, length)

@DTW\_rDELWORD(stringIn, n)

### 値

表 61. DTW\_DELWORD 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

入力ストリングのサブストリングを戻します。サブストリングはワード *n* から始まり、*length* で指定されたワード数だけ続きます。

### 形式

@DTW\_SUBWORD(stringIn, n, length, stringOut)

@DTW\_SUBWORD(stringIn, n, stringOut)

@DTW\_rSUBWORD(stringIn, n, length)

@DTW\_rSUBWORD(stringIn, n)

### 値

表 62. DTW\_SUBWORD 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

入力ストリングの指定位置から単一のワードを戻します。

### 形式

@DTW\_WORD(stringIn, n, stringOut)

@DTW\_rWORD(stringIn, n)

### 値

表 63. DTW\_WORD 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	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。
ストリング	<i>stringOut</i>	OUT	ワード位置 <i>n</i> のワードが含まれている変数。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

入力ストリングの  $n$  番目のワードの先頭文字の文字位置を戻します。

### 形式

@DTW\_WORDINDEX(stringIn, n, stringOut)

@DTW\_rWORDINDEX(stringIn, n)

### 値

表 64. DTW\_WORDINDEX 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	索引に入れるワードのワード位置。この値が入力ストリングのワードの数よりも大きい場合は、0 が戻されます。
ストリング	<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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

入力ストリングの  $n$  番目のワードの長さを戻します。

### 形式

@DTW\_WORDLENGTH(stringIn, n, stringOut)

@DTW\_rWORDLENGTH(stringIn, n)

### 値

表 65. DTW\_WORDLENGTH 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	長さを知りたいワードのワード位置。この値が入力ストリングのワードの数よりも大きい場合は、0 が戻されます。
ストリング	<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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。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。比較では、大文字小文字が区別されます。

### 形式

@DTW\_WORDPOS(stringIn1, stringIn2, n, stringOut)

@DTW\_WORDPOS(stringIn1, stringIn2, stringOut)

@DTW\_rWORDPOS(stringIn1, stringIn2, n)

@DTW\_rWORDPOS(stringIn1, stringIn2)

### 値

表 66. DTW\_WORDPOS 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

ストリングのワード数を返します。

### 形式

@DTW\_WORDS(stringIn, stringOut)

@DTW\_rWORDS(stringIn)

### 値

表 67. DTW\_WORDS 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。

### 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。

- 197ページの『DTW\_TB\_APPENDROW』
- 198ページの『DTW\_TB\_COLS』
- 199ページの『DTW\_TB\_DELETEROW』
- 200ページの『DTW\_TB\_DELETECOL』
- 201ページの『DTW\_TB\_DLIST』
- 203ページの『DTW\_TB\_DUMP』
- 204ページの『DTW\_TB\_DUMPV』
- 205ページの『DTW\_TB\_GETN』
- 206ページの『DTW\_TB\_GETV』
- 207ページの『DTW\_TB\_HTMLENCODING』
- 208ページの『DTW\_TB\_INPUT\_CHECKBOX』
- 209ページの『DTW\_TB\_INPUT\_RADIO』
- 210ページの『DTW\_TB\_INPUT\_TEXT』
- 212ページの『DTW\_TB\_INSERTCOL』
- 213ページの『DTW\_TB\_INSERTROW』
- 214ページの『DTW\_TB\_LIST』
- 216ページの『DTW\_TB\_MAXROWS』
- 217ページの『DTW\_TB\_QUERYCOLNONJ』
- 218ページの『DTW\_TB\_ROWS』
- 219ページの『DTW\_TB\_SELECT』
- 221ページの『DTW\_TB\_SETCOLS』
- 222ページの『DTW\_TB\_SETN』
- 223ページの『DTW\_TB\_SETV』
- 224ページの『DTW\_TB\_TABLE』
- 226ページの『DTW\_TB\_TEXTAREA』

# DTW\_TB\_APPENDROW

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

## 目的

1 つ以上の行を表の終わりに追加します。行が表に付加された後、すなわち、処理するために表を言語環境に渡した後、 DTW\_TB\_SETV() 関数を使用して新規の行に値を割り当てることができます。

表内の列数は、DTW\_TB\_APPENDROW() を呼び出す前に設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使用して、あるいは、設定する言語環境に表を渡すことによって、列数を設定することができます。

表内で許可される行数に制限があり、付加される行数が制限を超えるような場合には、呼び出し側にエラーが戻されます。

## 形式

@DTW\_TB\_APPENDROW(table, rows)

## 値

表 68. DTW\_TB\_APPENDROW パラメーター

データ型	パラメーター	用途	説明
表	table	INOUT	付加される行のマクロ表変数。
整数	rows	IN	table に付加する行数。

## 例

例 1: 表に 10 行を付加する。

```
%DEFINE myTable = %TABLE
@DTW_TB_APPENDROW(myTable, "10")
```

## DTW\_TB\_COLS

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

### 目的

表内の現行の列番号を戻します。

### 形式

@DTW\_TB\_COLS(table, cols)

@DTW\_TB\_rCOLS(table)

### 値

表 69. DTW\_TB\_COLS 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	戻される列数のマクロ表変数。
整数	<i>cols</i>	OUT	<i>table</i> の列数が含まれている変数。

### 例

**例 1:** 列数を検索し、*cols* に値を割り当てる。

```
%DEFINE myTable = %TABLE
%DEFINE cols = ""
...
@FillTable(myTable)
...
@DTW_TB_COLS(myTable, cols)
```

**例 2:** 表内の現行の列番号を検索し表示する。

```
%DEFINE myTable = %TABLE
...
@FillTable(myTable)
...
<P>My table contains @DTW_TB_rCOLS(myTable) columns.
```

## DTW\_TB\_DELETEROW

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

### 目的

`start_row` で指定された行で始まる 1 つ以上の行を削除します。

表内の列数は、`DTW_TB_DELETEROW()` を呼び出す前に設定しなければなりません。`DTW_TB_SETCOLS()` 関数または `DTW_TB_INSERTCOL()` 関数を使用して、あるいは、設定する言語環境に表を渡すことによって、列数を設定することができます。

### 形式

`@DTW_TB_DELETEROW(table, start_row, rows)`

### 値

表 70. `DTW_TB_DELETEROW` パラメーター

データ型	パラメーター	用途	説明
表	<code>table</code>	INOUT	行が削除されるマクロ表変数。
整数	<code>start_row</code>	IN	削除する <code>table</code> 内の先頭行の行番号。
整数	<code>rows</code>	IN	<code>table</code> から削除する行数。

### 例

**例 1:** 表の 10 行で始まる 5 行を削除する。

```
%DEFINE myTable = %TABLE
@DTW_TB_DELETEROW(myTable, "10", "5")
```

**例 2:** 表のすべての行を削除する。

```
%DEFINE myTable = %TABLE
@DTW_TB_DELETEROW(myTable, "1", @DTW_TB_rROWS(myTable))
```

## DTW\_TB\_DELETECOL

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

### 目的

*table* 内で指定された列の後から始まる、1 つ以上の列を表から削除します。

### 形式

@DTW\_TB\_DELETECOL(*table*, *after\_col*, *cols*)

### 値

表 71. DTW\_TB\_DELETECOL 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	列が削除されるマクロ表変数。
整数	<i>after_col</i>	IN	後続の列が削除された後の列数。最初の列を削除するには、0 を指定します。
整数	<i>cols</i>	IN	<i>table</i> から削除する列数。

### 例

**例 1:** 3 番目と 4 番目の列を表から削除する。

```
%DEFINE myTable = %TABLE  
@DTW_TB_DELETECOL(myTable, "3", "2")
```

**例 2:** 最初の列を表から削除する。

```
%DEFINE myTable = %TABLE  
@DTW_TB_DELETECOL(myTable, "0", "1")
```

## DTW\_TB\_DLIST

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

### 目的

マクロ表から HTML 定義リストを戻します。

### 形式

| @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 72. DTW\_TB\_DLIST 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 リンクを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 72. DTW\_TB\_DLIST 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 参照のための URL が含まれている <i>table</i> 内の列数。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。

## 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する。

```
@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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

マクロ変数の内容を戻します。表の各行は別々の行に表示されます。表全体は、`<PRE></PRE>` タグで囲まれています。

### 形式

`@DTW_TB_DUMP(table)`

### 値

表 73. *DTW\_TB\_DUMP Parameters*

データ型	パラメーター	用途	説明
表	<i>table</i>	IN	表示するマクロ変数を指定するシンボル。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

マクロ表変数の内容を戻します。各表値は別々の行に表示されます。表全体は、`<PRE></PRE>` タグで囲まれています。

### 形式

`@DTW_TB_DUMPV(table)`

### 値

表 74. *DTW\_TB\_DUMPV* 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	表示するマクロ表変数を指定するシンボル。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

## 目的

*col* で指定される列数の列見出しを検索します。

表内の列数は、DTW\_TB\_GETN() を呼び出す前に設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使用して、あるいは、設定する言語環境に表を渡すことによって、列数を設定することができます。

## 形式

@DTW\_TB\_GETN(table, col, name)

@DTW\_TB\_rGETN(table, col)

## 値

表 75. DTW\_TB\_GETN 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	列名が入っているマクロ表変数。
整数	<i>col</i>	IN	名前が戻される列の列番号。
ストリング	<i>name</i>	OUT	<i>col</i> で指定された列の名前が入っている変数。

## 例

**例 1:** 列 4 の列名を検索する。

```
%DEFINE myTable = %TABLE
%DEFINE name = ""
...
@FillTable(myTable)
...
@DTW_TB_GETN(myTable, "4", name)
```

**例 2:** 表内の最後の列名を検索する。

```
%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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

*row* および *col* で指定された行数と列数の表値を検索します。

表内の列数は、DTW\_TB\_GETV() を呼び出す前に設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使用して、あるいは、設定する言語環境に表を渡すことによって、列数を設定することができます。

### 形式

@DTW\_TB\_GETV(table, row, col, value)

@DTW\_TB\_rGETV(table, row, col)

### 値

表 76. DTW\_TB\_GETV 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。

### 例

**例 1:** 6 行、3 列で表値を検索する。

```
%DEFINE myTable = %TABLE
%DEFINE value = ""
...
@FillTable(myTable)
...
@DTW_TB_GETV(myTable, "6", "3", value)
```

**例 2:** 1 行、1 列で表値を検索する。

```
%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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

以下のエンコード HTML 文字をもつ入力マクロ表を戻します。

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;

### 形式

@DTW\_TB\_HTMLENCODE(table, collist)

### 値

表 77. DTW\_TB\_HTMLENCODE 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

マクロ表変数から 1 つまたは複数の HTML チェックボックス入力タグを戻します。

### 形式

@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 78. DTW\_TB\_INPUT\_CHECKBOX 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	入力フィールド値を含む <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。

### 例

例 1: 3 つのチェックボックス入力タグのための HTML を生成する。

```
@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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

マクロ表変数から 1 つまたは複数の HTML ラジオ・ボタン入力タグを戻します。

### 形式

@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 79. DTW\_TB\_INPUT\_RADIO 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	入力フィールド値を含む <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。

### 例

例 1: 3 つのラジオ・ボタン入力タグのための HTML を生成する。

```
@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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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。たとえば、Net.Data は、VALUE、SIZE、および MAXLENGTH 属性をもつ HTML <INPUT> タグを生成することができます。

```
<INPUT TYPE="TEXT" NAME="someName" VALUE="someValue" SIZE="20" MAXLENGTH="40">
```

### 形式

@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 80. DTW\_TB\_INPUT\_TEXT 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	入力フィールド名を含む <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。



## 例

例 1: 3 つの HTML <INPUT> タグを戻す。

```
@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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

*after\_col* 内で指定された列の後から始まる、1 つ以上の列を挿入します。

### 形式

@DTW\_TB\_INSERTCOL(table, after\_col, cols)

### 値

表 81. DTW\_TB\_INSERTCOL 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	新規列が挿入された後の列の列番号。表の始めで列を挿入するには、0 を指定します。
整数	<i>cols</i>	IN	<i>table</i> に挿入する列数。

### 例

**例 1:** 表の終わりに 5 つの列を挿入する。

```
%DEFINE myTable = %TABLE
@DTW_TB_INSERTCOL(myTable, @DTW_TB_rCOLS(myTable), "5")
```

**例 2:** 表の始めに 1 つの列を挿入する。

```
%DEFINE myTable = %TABLE
@DTW_TB_INSERTCOL(myTable, "0", "1")
```

## DTW\_TB\_INSERTROW

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

### 目的

*after\_row* で指定された行の後から始まる 1 つ以上の行を挿入します。

表内の列数は、DTW\_TB\_INSERTROW() を呼び出す前に設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使用して、あるいは、設定する言語環境に表を渡すことによって、列数を設定することができます。

### 形式

@DTW\_TB\_INSERTROW(table, after\_row, rows)

### 値

表 82. DTW\_TB\_INSERTROW 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	行を挿入するマクロひょう変数。
整数	<i>after_row</i>	IN	新規行が挿入された後の行の番号。表の始めで行を挿入するには、0 を指定します。
整数	<i>rows</i>	IN	<i>table</i> に挿入する行数。

### 例

**例 1:** 表の 5 行目の後に 1 行を挿入する。

```
%DEFINE myTable = %TABLE
@DTW_TB_INSERTROW(myTable, "5", "1")
```

**例 2:** 表の始めに 3 行を挿入する。

```
%DEFINE myTable = %TABLE
@DTW_TB_INSERTROW(myTable, "0", "3")
```

## DTW\_TB\_LIST

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

### 目的

HTML リストを戻します。

### 形式

@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 83. DTW\_TB\_LIST 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 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。
整数	<i>link_u</i>	IN	HTML リンクのための URL が含まれている <i>table</i> 内の列番号。この値を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。

### 例

例 1: 順序付きリストのための HTML タグを生成する。

```
@DTW_TB_LIST(Mytable,"OL","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\_MAXROWS

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

### 目的

表内で許可される行の最大数を返します。

### 形式

@DTW\_TB\_MAXROWS(table, maxrows)

@DTW\_TB\_rMAXROWS(table)

### 値

表 84. DTW\_TB\_MAXROWS 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	行の最大数が戻されるマクロ表変数。
整数	<i>maxrows</i>	OUT	<i>table</i> で許可される行の最大数が含まれている変数。行数の制限なしで表を定義した場合は、0 が戻されます。

### 例

**例 1:** 表内で許可される行の最大数を検索し、値を返す。

```
%DEFINE myTable = %TABLE
%DEFINE maxrows = ""
@DTW_TB_MAXROWS(myTable, maxrows)
%IF (maxrows != "0")
    The maximum number of rows allowed is $(maxrows).
%ELSE
    There is no limit on the number of rows allowed.
%ENDIF
```

**例 2:** 行の最大数の値を検索し表示する。

```
%DEFINE myTable = %TABLE
%IF (@DTW_TB_rMAXROWS(myTable) != "0")
    The maximum number of rows allowed is @DTW_TB_rMAXROWS(myTable).
%ELSE
    There is no limit on the number of rows allowed.
%ENDIF
```

## DTW\_TB\_QUERYCOLNONJ

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

### 目的

列見出しに関連した列番号を戻します。列見出しが表内に存在しない場合は、0 が戻されます。

表内の列数は、DTW\_TB\_QUERYCOLNONJ() を呼び出す前に設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使用して、あるいは、設定する言語環境に表を渡すことによって、列数を設定することができます。

### 形式

@DTW\_TB\_QUERYCOLNONJ(table, name, col)

@DTW\_TB\_rQUERYCOLNONJ(table, name)

### 値

表 85. DTW\_TB\_QUERYCOLNONJ 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	列番号が入っているマクロ表変数。
ストリング	<i>name</i>	IN	列番号が戻される列見出しの名前。列見出しが表内に存在しない場合は、0 が戻されます。
整数	<i>col</i>	OUT	名前が <i>name</i> で指定される列の列番号が含まれている変数。

### 例

**例 1:** 名前が SERIAL\_NUMBER である列の列番号を検索する。

```
%DEFINE myTable = %TABLE
%DEFINE col = ""
@DTW_TB_QUERYCOLNONJ(myTable, "SERIAL_NUMBER", col)
```

**例 2:** 名前が SERIAL\_NUMBER である列の列番号を検索する。

```
%DEFINE myTable = %TABLE
<P>The "SERIAL_NUMBER" column is column number @DTW_TB_rQUERYCOLNONJ(myTable, "SERIAL_NUMBER")
```

## DTW\_TB\_ROWS

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

### 目的

表内の現行の行番号を戻します。

### 形式

@DTW\_TB\_ROWS(table, rows)

@DTW\_TB\_rROWS(table)

### 値

表 86. DTW\_TB\_ROWS 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	現行の行番号が戻されるマクロ表変数。
整数	<i>rows</i>	OUT	<i>table</i> の現行の行番号が含まれている変数。

### 例

**例 1:** 表内の現行の行番号を検索し、 *rows* に値を割り当てる。

```
%DEFINE myTable = %TABLE
%DEFINE rows = ""
...
@FillTable(myTable)
...
@DTW_TB_ROWS(myTable, rows)
```



## DTW\_TB\_SELECT

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

### 目的

HTML SELECT メニューを戻します。

### 形式

@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 87. DTW\_TB\_SELECT 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。

### 例

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>
```

# DTW\_TB\_SETCOLS

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

## 目的

表内の列数を設定します。 DTW\_TB\_SETN() 関数を使用して、列見出しを指定します。表のために DTW\_TB\_SETCOLS() 関数を一度だけ使用することができます。その後は、DTW\_TB\_DELETECOL() 関数または DTW\_TB\_INSERTCOL() 関数を使用して、表内の列数を変更します。

## 形式

@DTW\_TB\_SETCOLS(table, cols)

## 値

表 88. DTW\_TB\_SETCOLS パラメーター

データ型	パラメーター	用途	説明
表	table	INOUT	列番号が設定されるマクロ表変数。
整数	cols	IN	table 内で割り当てられる最初の列番号。

## 例

例 1: 表に 3 つの列を割り当て、その列に名前を割り当てる。

```
%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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

列見出しに名前を割り当てます。列見出しを削除するには、列見出し値に NULL を割り当てます。

表内の列数は、DTW\_TB\_SETN() を呼び出す前に設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使用して、あるいは、設定する言語環境に表を渡すことによって、列数を設定することができます。

### 形式

@DTW\_TB\_SETN(table, name, col)

### 値

表 89. DTW\_TB\_SETN 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に割り当てられる文字ストリング。
整数	<i>col</i>	IN	列見出しが設定されている列の列番号。

### 例

**例 1:** 列見出し 1 から 3 に名前を割り当てる。

```
%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 の列見出しを削除する。これは、定義された関数呼び出しの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 です。

```
%DEFINE myTable = %TABLE
@DTW_TB_SETN(myTable, nullVar, "2")
```

## DTW\_TB\_SETV

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

### 目的

表内で値を割り当てます。表値を削除するには、値に NULL を割り当てます。

表内の列数は、DTW\_TB\_SETV() を呼び出す前に設定しなければなりません。DTW\_TB\_SETCOLS() 関数または DTW\_TB\_INSERTCOL() 関数を使用して、あるいは、設定する言語環境に表を渡すことによって、列数を設定することができます。

### 形式

@DTW\_TB\_SETV(table, value, row, col)

### 値

表 90. DTW\_TB\_SETV 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	その中で表値を設定するマクロ表変数。
文字列	<i>value</i>	IN	<i>row</i> および <i>col</i> で指定された行および列の表値に割り当てられる文字列。
整数	<i>row</i>	IN	設定される値の行番号。
整数	<i>col</i>	IN	設定される値の列番号。

### 例

**例 1:** 3 行目の 3 列目に値を割り当てる。

```
%DEFINE myTable = %TABLE
@DTW_TB_SETV(myTable, "value3.3", "3", "3")
```

**例 2:** 4 行目の 2 列目で表値を削除する。これは、定義されなかった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 です。

```
%DEFINE myTable = %TABLE
@DTW_TB_SETV(myTable, nullVar, "4", "2")
```

## DTW\_TB\_TABLE

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

### 目的

マクロ表変数から HTML 表を戻します。

### 形式

```
@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 91. DTW\_TB\_TABLE 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 91. DTW\_TB\_TABLE 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・イメージのため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。

## 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 タグを生成する。

```
@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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X	X	X	X	X	X	X	X

### 目的

マクロ表変数から HTML TEXTAREA タグを戻します。

### 形式

@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 92. DTW\_TB\_TEXTAREA 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。

### 例

例 1: HTML TEXTAREA タグを生成し、組み込む行を指定する。

```
@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・インターフェースの組み込み関数を使用可能です。

- 232ページの『DTWF\_APPEND』
- 234ページの『DTWF\_CLOSE』
- 235ページの『DTWF\_DELETE』
- 239ページの『DTWF\_OPEN』
- 241ページの『DTWF\_READ』
- 243ページの『DTWF\_REMOVE』
- 244ページの『DTWF\_SEARCH』
- 246ページの『DTWF\_UPDATE』
- 248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ふらっと・ファイル・ソースへのアクセス方法を説明します。

- 『フラット・ファイル・データ・ソースへのアクセス』
- 230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』
- 231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 初期設定ファイルの 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とサブディレクトリーをリストし、パス・ステートメントに組み込まれるディレクトリーにないファイルに機密保護を提供します。 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...
```

マクロ関数で FFI 言語環境を呼び出すときには、 FFI 関数の *filename* パラメーターを使用して、 FFI 関数が処理しているフラット・ファイルにパスを指定します。たとえば、以下ようになります。

```
%DEFINE myfile = "/macros/myfile.txt" @DTWF_READ(myfile, ...)
```

以下のセクションで説明しています。

- 228ページの『Net.Data がフラット・ファイル・ロケーションを決定する方法』
- 229ページの『フラット・ファイルの構成規則』
- 229ページの『機密保護の勧告』

- 230ページの『許可要件』

## Net.Data がフラット・ファイル・ロケーションを決定する方法

Net.Data は、FFI 関数の *filename* パラメーターの情報を使用して、Net.Data 初期設定ファイルで 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か、または現行ディレクトリーを使用するかを決定します。

FFI 関数でファイル名が指定されると、Net.Data は、FFI\_PATH にリストされる各パスを検索することによって、ファイルを位置指定しようとして、指定されている最初のパスから開始します。Net.Data は、見つける最初のコピーを使用します。ファイルが見つからない場合は、次に、Net.Data は、プロセスまたは Net.Data が実行されているスレッドの現行作業ディレクトリー内のファイルを見つけようとします。

**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位置指定する。

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
```

さらに、ファイルは現行ディレクトリーと /macros/org1 の両方で位置指定されます。関数呼び出しの場合は、以下のようになります。

```
DTWF_READ("myfile.txt")
```

Net.Data は /macros/org1/myfile.txt を使用します。

DTWF\_READ 関数が既存のファイルを読みとるために使用されていて、myfile.txt のファイル名が指定されていると、Net.Data は、FFI\_PATH には上記で指定されたパスのリストが含まれていると想定して、次にファイルのディレクトリー /macros、/macros/org1 および /macros/org2 を検索します。

### 現行ディレクトリーの決定

Net.Data の現行ディレクトリーは、使用している 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は、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は、現行ディレクトリーが変化する可能性があります。サーバーのデフォルト要求経路指定または資源マッピングが変わると、現行ディレクトリーも変わる可能性があります。

### フラット・ファイル・アクセスを指定するための勧告

以下の勧告を使用して、Net.Data が確実にフラット・ファイルのデータ・ソースにアクセスできるようにします。

- DTWF\_OPEN 関数を使用してフラット・ファイルを作成する場合は、FFI\_PATH にあるディレクトリー・パス、または現行ディレクトリーの内容を知っているディレクトリー・パスを確実に指定する。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で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には、FFI\_PATH のパスの 1 つと一致する全パスを指定する。これは、Net.Data は、FFI\_PATH で指定されたディレクトリー内ではサブディレクトリーを検索しないからです。
- 特に Web サーバー API を使用している場合には、*filename* パラメーターの絶対パスを使用する。

## フラット・ファイルの構成規則

Net.Data 初期設定ファイルで 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ます。

- FFI\_PATH のパス・ステートメントには、有効な印刷可能文字が含まれていなければならない。FFI では、疑問符 (?) または二重引用符 (") を組み込んだパスは許可されません。
- マクロ内で *filename* パラメーターと一緒に使用されるすべてのディレクトリーおよびサブディレクトリーは、FFI\_PATH で指定されなければならない。*filename* にリストされ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ない限り検索されません。
- FFI\_PATH ステートメントに絶対パスを使用する。

## 機密保護の勧告

Net.Data 初期設定ファイルの FFI\_PATH ステートメントを使ってアクセスできる、ファイル FFI 関数を指定することができます。FFI だけが、ステートメントにリストされたパスを検索します。したがって、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、下記のステートメントに似た FFI\_PATH を指定して、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することができます。

```
FFI_PATH      C:¥public;E:¥WWW;E:¥guest;A:
```

以下に、ユーザーのフラット・ファイル保護を作成するための勧告をリストします。

- フラット・ファイル操作のために使用するのに適したディレクトリーを選択する。これらのディレクトリーを FFI\_PATH に追加して、そのディレクトリーに対する検索を制限する必要があります。
- 人々にマクロ内で DTWF\_REMOVE または他のエクスポート操作を実行させる管理を使用して、拡張子 .dll および .cmd をもったファイルを除去したり変更することを防ぐ。このようなファイルは、ユーザーが現行ディレクトリー内にもっている可能性があります。
- マクロがシステムに追加される妥当な制御を使用して、システムでファイルを保護する適切なステップを取る。
- 匿名の FTP ユーザーがパスに書き込むようにさせる FFI\_PATH でパスを指定してはならない。これを行うと、以前には許可されなかったアクションを許可するシステムに、だれかが Net.Data マクロを書き込むことができます。
- Net.Data 初期設定ファイルのパスを FFI\_PATH に追加してはならない。

## 許可要件

FFI 組み込み関数が使用するファイルへのアクセス権限を、`Net.Data` が実行するユーザー ID に必ず持たせてください。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。

## 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 は、FFI が、要求された変換に従って、ファイルをパーツ (行の列など) に分割するとき使用するフラグまたは区切り記号です。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、ファイルの内容を表の行と列に分離します。書き込み操作の場合、区切り文字は、表の行と列にある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ない限り、文字の終わりにヌル文字を組み込みません。

区切り文字にヌル文字を使用するには、2 つの二重引用符 (『』) を使用した空ストリングではなく、二重引用符内の逆スラッシュとゼロ (『\0』) を `DELIMITER` パラメーターに指定します。`ASCITEXT` 変換を指定すると、`Net.Data` は、区切り文字に改行文字を使用して、すべての要求された区切り文字を無視します。

読みとり操作に対してより、書き込み操作に対して別の区切り文字を使用すると、有害な変更がファイルに発生する可能性があります。 `Net.Data` は新規の区切り文字を使っ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` 関数と `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。これらの関数を使って、`Net.Data` はフラット・ファイルを予約して、他のアプリケーションがそのファイルを読みとったり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` 関数を使用します。この関数は、ファイルが他のアプリケーションで使用できないようにして、ファイルを読みとり更新する間に変更できないようにします。

ファイルを解放するには、`DTWF_CLOSE` 関数を使用します。この関数はファイルを解放して、他のアプリケーションがそのファイルを読みとったり更新できるようにします。

## DTWF\_APPEND

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

### 目的

表変数の内容をファイルの終わりに書き込みます。

ファイルの現行内容は、DTWF\_APPEND を使用した結果に、特に、最後の行の最後の列の内容に影響を及ぼします。改行文字がファイルの最後の行の最後の列値に続くと、付加されたデータが新規の行に入れます。送でない場合は、付加されたデータは、ファイルの最後の行のパーツになります。

### 形式

@DTWF\_APPEND(filename, transform, delimiter, table, retry, rows)

@DTWF\_APPEND(filename, transform, delimiter, table, retry)

@DTWF\_APPEND(filename, transform, delimiter, table)

### 値

表 93. DTWF\_APPEND 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> ファイル内の改行文字は、ASCIITEXT と DELIMITED の変換に、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 表内の行の最大長は 16383 文字です。この制限には、Net.Data マクロ表内の各列ごとのヌル文字が含まれます。
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。

表 93. DTWF\_APPEND 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。

## 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, "ASCITEXT", " ;", myTable)
```

### 例 3:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
}%
@DTWF_APPEND(myFile, "ASCITEXT", " ;", myTable, "0", "10")
```

## DTWF\_CLOSE

AIX	HP-UX	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。

### 形式

@DTWF\_CLOSE(filename, retry)

@DTWF\_CLOSE(filename)

### 値

表 94. DTWF\_CLOSE 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	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	<i>retry</i>	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。

### 例

例 1:

```
@DTWF_CLOSE(myFile, "5")
```



## DTWF\_DELETE

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

### 目的

レコードをファイルから削除します。(空ファイルは削除しません。)

### 形式

@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 95. DTW\_DELETE 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> ファイル内の改行文字は、ASCIITEXT と DELIMITED の変換に、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。

## 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X		X	X	X		X	X

### 目的

レコードをファイルに挿入します。

### 形式

@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 96. DTWF\_INSERT 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> ファイル内の改行文字は、ASCIITEXT と DELIMITED の変換に、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 表内の行の最大長は 16383 文字です。この制限には、Net.Data マクロ表内の各列ごとのヌル文字が含まれます。
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 96. DTWF\_INSERT 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。この値がファイルの行数より大きい場合は、値は最終レコードに変更され、エラーが戻されます。0 を指定することは、ファイルの先頭に挿入することです。デフォルトでは、1 から開始されます。

## 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X		X	X	X		X	X

### 目的

明示的にファイルをオープンします。ファイル存在しない場合には、ファイル名の絶対パスを指定する必要があります。また、ファイルが作成されるディレクトリーは、FFI\_PATH で指定される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、ファイルは現行作業ディレクトリーでオープンされます。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。

**パフォーマンス上のヒント:** 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。

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。

### 形式

@DTWF\_OPEN(filename, mode, retry)

@DTWF\_OPEN(filename, mode)

### 値

表 97. DTWF\_OPEN 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。

表 97. DTWF\_OPEN パラメーター (続き)

データ型	パラメーター	用途	説明
文字列	<i>mode</i>	IN	<p>要求されるアクセスの型。</p> <ul style="list-style-type: none"> <li>• r - 読み取りのための既存のファイルを開きます。</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開きます。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開きます。</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開きます。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	<p>ファイルを即時に開け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。</p>

## 例

### 例 1:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myMode = "r+"
}%
@DTWF_OPEN(myFile, myMode, "1000")
```

## DTWF\_READ

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

### 目的

レコードをファイルから表変数へ読み取ります。

### 形式

@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 98. DTWF\_READ 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> ファイル内の改行文字は、ASCIITEXT と DELIMITED の変換に、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。この制限には、Net.Data マクロ表内の各列ごとのヌル文字が含まれます。
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 98. DTWF\_READ 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。

## 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	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。

### 形式

@DTWF\_REMOVE(filename, retry)

@DTWF\_REMOVE(filename)

### 値

表 99. DTW\_REMOVE 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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X		X	X	X		X	X

### 目的

ストリング検索の結果を表変数に戻します。

### 形式

@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 100. DTWF\_SEARCH 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> ファイル内の改行文字は、ASCIITEXT と DELIMITED の変換に、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 表内の行の最大長は 16383 文字です。この制限には、Net.Data マクロ表内の各列ごとのヌル文字が含まれます。
ストリング	<i>searchFor</i>	IN	検索対象の文字ストリング。

表 100. DTWF\_SEARCH 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。

## 使用法

- DTWF\_SEARCH に戻される表の列は 3 つです。最初の 2 つの列には、一致が検出された行と列の番号が入っています。最後の列には、*SearchFor* パラメーターで指定される文字を含む列値が入っています。たとえば、ファイルの 4 番目の行に列 3 と一致する文字が入っている場合には、戻された表の行の最初の列には数字 4 があり、一致が検出された行を示しています。戻された表の行の 2 番目の列にある数字 3 は、同じものが含まれているファイルの列を示しています。3 番目の列には、完全な列値が入っています。
- SearchFor* パラメーターには、*delimiter* パラメーターの内容を組み込むことができません。

## 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X		X	X	X		X	X

### 目的

表変数のファイル・レコードを更新します。ファイル存在しない場合には、ファイル名の絶対パスを指定する必要があります。また、ファイルが作成されるディレクトリは、FFI\_PATH で指定されるディレクトリと一致しなければなりません。絶対パスを使用しないと、ファイルは現行作業ディレクトリでオープンされます。

### 形式

@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 101. DTWF\_UPDATE 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> ファイル内の改行文字は、ASCIITEXT と DELIMITED の変換に、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・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。この制限には、Net.Data マクロ表内の各列ごとのヌル文字が含まれます。

表 101. DTWF\_UPDATE 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。
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。

## 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X		X	X	X		X	X

### 目的

表変数の内容をファイルに書き込みます。ファイル存在しない場合には、ファイル名の絶対パスを指定する必要があります。また、ファイルが作成されるディレクトリーは、`FFI_PATH` で指定される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、ファイルは現行作業ディレクトリーでオープンされます。

### 形式

@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 102. DTWF\_WRITE 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> ファイル内の改行文字は、ASCIITEXT と DELIMITED の変換に、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 表内の行の最大長は 16383 文字です。この制限には、Net.Data マクロ表内の各列ごとのヌル文字が含まれます。

表 102. DTWF\_WRITE 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。
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。

## 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 登録関数

Web 登録は、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 登録を作成することができます。各登録は名前をもっています。各登録には複数の項目を含めることができます。Net.Data は、登録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。

- 251ページの『DTWR\_ADDENTRY』
- 252ページの『DTWR\_CLEARREG』
- 253ページの『DTWR\_CLOSEREG』
- 254ページの『DTWR\_CREATEREG』
- 255ページの『DTWR\_DELENTY』
- 256ページの『DTWR\_DELREG』
- 257ページの『DTWR\_LISTREG』
- 258ページの『DTWR\_LISTSUB』
- 259ページの『DTWR\_OPENREG』
- 260ページの『DTWR\_RTVENTRY』
- 261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。



## DTWR\_ADDENTRY

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

### 目的

項目を Web 登録に追加します。

### 形式

@DTWR\_ADDENTRY(registry, registryVariable, registryData, index)

@DTWR\_ADDENTRY(registry, registryVariable, registryData)

### 値

表 103. DTWR\_ADDENTRY 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	項目追加先の登録の名称。
ストリング	<i>registryVariable</i>	IN	追加する登録項目の <i>registryVariable</i> ストリング部分の値。
ストリング	<i>registryData</i>	IN	追加する登録項目の <i>registryData</i> ストリング部分の値。
ストリング	<i>index</i>	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登録に追加されます。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X		X		X			X

### 目的

Web 登録から項目を消去します。

### 形式

@DTWR\_CLEARREG(registry)

### 値

表 104. DTWR\_CLEARREG 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	消去する登録の名前。

### 例

例 1:

```
@DTWR_CLEARREG("Myregistry")
```

## DTWR\_CLOSEREG

AIX	HP-UX	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。

### 形式

@DTWR\_CLOSEREG(registry)

### 値

表 105. DTWR\_CLOSEREG 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 (*) や円記号 (¥) などの特殊文字を使用してはいけません。

### 例

例 1: レジストリーをクローズする。

```
@DTWR_CLOSEREG("/qsys.lib/mylib.lib/myreg.file")
```

## DTWR\_CREATEREG

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

### 目的

新規 Web 登録を作成します。

Re

### 形式

@DTWR\_CREATEREG(registry, security)

@DTWR\_CREATEREG(registry)

### 値

表 106. DTWR\_CREATEREG 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	作成する登録の名前。  制約事項：Web レジストリー名にアスタリスク (*) や円記号 (¥) などの特殊文字を使用してはいけません。
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、登録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。

### 例

例 1:

```
@DTWR_CREATEREG("myRegistry")
```

例 2:

```
@DTWR_CREATEREG("URLLIST", "*RWX, *RWX, *R")
```

## DTWR\_DELENTY

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

### 目的

項目を Web 登録から削除します。

### 形式

@DTWR\_DELENTY(registry, registryVariable, index)

@DTWR\_DELENTY(registry, registryVariable)

### 値

表 107. DTWR\_DELENTY 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	項目が削除される登録の名前。
ストリング	<i>registryVariable</i>	IN	除去する項目の <i>registryVariable</i> ストリング部分の値。
ストリング	<i>index</i>	IN	索引付き項目の <i>registryVariable</i> 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、索引付き項目が登録から除去されます。

### 例

#### 例 1:

```
@DTWR_DELENTY("Myregistry", "Jones")
```

#### 例 2:

```
@DTWR_DELENTY("URLLIST", "SMITH", "WORK_URL")
```

## DTWR\_DELREG

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

### 目的

Web 登録を削除します。

### 形式

@DTWR\_DELREG(registry)

### 値

表 108. DTWR\_DELREG 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	削除する登録の名前。

### 例

例 1:

```
@DTWR_DELREG("Myregistry")
```

## DTWR\_LISTREG

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

### 目的

Web 登録全体をリストします。DTWR\_LISTREG は、ユーザーが渡した OUT 表変数の登録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 登録操作のために 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登録項目が、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、指定された登録に 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 の値が指定登録の使用可能項目数よりも大きい場合は、すべての登録項目がリストされます。

表には、常に、2 つの列があります。表の列ヘッダーは、Web 登録言語環境によって "REGISTRY\_VARIABLE" および "REGISTRY\_DATA" に設定されます。

### 形式

@DTWR\_LISTREG(registry, registryTable)

### 値

表 109. DTWR\_LISTREG 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	リストする登録の名前。
ストリング	registryTable	OUT	登録項目を入れる表変数の名前。

### 例

#### 例 1:

```
%DEFINE RegistryTable = %TABLE(ALL)
@DTWR_LISTREG("URLLIST", RegistryTable)
```

## DTWR\_LISTSUB

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

### 目的

Web 登録の即時サブキー項目をリストします。DTWR\_LISTSUB は、ユーザーが渡した OUT 表パラメーターの登録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 登録操作に渡されます。

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登録項目が、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、指定された登録に 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 の値が指定登録の使用可能項目数よりも大きい場合は、すべての登録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。

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。

### 形式

@DTWR\_LISTSUB(registry, registryTable)

### 値

表 110. DTWR\_LISTSUB 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	リストする登録の名前。
ストリング	registryTable	OUT	登録項目を入れる表変数の名前。

### 例

#### 例 1:

```
%DEFINE RegistryTable = %TABLE(ALL)
@DTWR_LISTSUB("URLLIST", RegistryTable)
```



## DTWR\_OPENREG

AIX	HP-UX	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。

### 形式

@DTWR\_OPENREG(registry, commit)

@DTWR\_OPENREG(registry)

### 値

表 111. DTWR\_OPENREG 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	オープンするレジストリーの名前。
ストリング	<i>commit</i>	IN	レジストリーをコミットメント制御下でオープンするかしないかを指定する、単一シンボルまたはリテラル・ストリング。可能な値は以下のとおりです。  <b>Y</b> コミットメント制御下で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下でレジストリーをオープンしません。 デフォルトは N です。

### 例

**例 1:** コミットメント制御下でレジストリーをオープンする。

```
@DTWR_OPENREG("/qsys.lib/mylib.lib/myreg.file", "Y")
```

## DTWR\_RTVENTRY

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

### 目的

Web 登録項目から registryData ストリングを取りだします。

### 形式

@DTWR\_RTVENTRY(registry, registryVariable, registryData, index)

@DTWR\_RTVENTRY(registry, registryVariable, registryData)

@DTWR\_rRTVENTRY(registry, registryVariable, index)

@DTWR\_rRTVENTRY(registry, registryVariable)

### 値

表 112. DTWR\_RTVENTRY 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	検索対象の項目が含まれている登録の名前。
ストリング	registryVariable	IN	取り出す registryData ストリングが含まれている登録項目の registryVariable ストリング部分の値。
ストリング	registryData	OUT	registryVariable と一致する登録項目の registryData ストリング部分の値を返します。
ストリング	index	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。

### 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
X		X		X			X

### 目的

指定された登録項目の既存の *registryData* スtring値を、呼び出し元が指定した新規の値と置き換えます。*registryVariable* Stringは変更できません。

### 形式

@DTWR\_UPDATEENTRY(registry, registryVariable, newData, index)

@DTWR\_UPDATEENTRY(registry, registryVariable, newData)

### 値

表 113. DTWR\_UPDATEENTRY パラメーター

データ型	パラメーター	用途	説明
String	<i>registry</i>	IN	更新する項目が含まれている登録の名前。
String	<i>registryVariable</i>	IN	更新する登録項目の <i>registryVariable</i> String部分の値。
String	<i>newData</i>	IN	更新する登録項目の <i>registryData</i> String部分の新規値。
String	<i>index</i>	IN	更新する索引付き項目の <i>registryVariable</i> String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。

### 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永続的なマクロ・ブロックを定義する際に役立って、Net.Data の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わり、トランザクション間で永続的な HTML ブロック、およびトランザクション内で変更をコミットまたはロールバックするかどうかを定義します。

- 263ページの『DTW\_ACCEPT』
- 265ページの『DTW\_COMMIT』
- 266ページの『DTW\_ROLLBACK』
- 267ページの『DTW\_RTVHANDLE』
- 268ページの『DTW\_STATIC』
- 269ページの『DTW\_TERMINATE』

## DTW\_ACCEPT

AIX	HP-UX	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するトランザクション・ハンドルを定義します。Net.Data では、Web ブラウザーからの応答としてマクロを呼び出す URL にトランザクション・ハンドルが組み込まれている必要があります。要求が Web サーバーに入ってくると、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の経路を定めます。

トランザクション・ハンドルが呼び出されるのは、DTW\_TERMINATE() への呼び出しが含まれている最後の論理ブロックまで、マクロのそれぞれの HTML ブロックの開始時でなければなりません。DTW\_ACCEPT() または DTW\_TERMINATE() への呼び出しのいずれかが、テキストがブラウザーに出力される前に検出されない場合には、Net.Data のエラーが発生します。

@DTW\_STATIC() 関数で指定されるタイムアウト値をオーバーライドするタイムアウト値を、このページに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に指定する時間 (秒) の間待機します。

マクロが永続的状态でないときにこの関数を呼び出すと、Net.Data のエラーが発生します。

**ヒント：**トランザクション・ハンドルが含まれている URL を、フォーム・プッシュボタン上のアクションとして、またはブラウザーに表示されるページのハイパーテキスト・リンクとして、コード化することができます。

### 形式

@DTW\_ACCEPT(handle, timeout)

@DTW\_ACCEPT(handle)

### 値

表 114. DTW\_ACCEPT 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>handle</i>	IN	この永続的なトランザクションで後続のマクロを呼び出すために URL で使用す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	<i>timeout</i>	IN	応答を待つこのポートをジョブが提供する時間 (秒) を指定する変数またはリテラル・ストリング。この値は、DTW_STATIC() 関数で指定されたすべてのタイムアウト値をオーバーライドします。

## 例

### 例 1:

```
%DEFINE handle = ""
@DTW_RTVHANLDE(handle)
%HTML(REPORT) {
@DTW_ACCEPT(handle)
...
%}
```

## DTW\_COMMIT

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

### 目的

直前のコミットメント境界以降コミットメント制御下にある資源に対して加えられた変更を永久に保留し、新規のコミットメント境界を再設定します。

### 形式

@DTW\_COMMIT()

### 値

なし

### 例

**例 1:** コミットを指定する。

```
@DTW_COMMIT()  
%HTML(report){  
%}
```

## DTW\_ROLLBACK

AIX	HP-UX	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確立します。 Net.Data が実行されているプロセスに対して、最後のコミットメント境界以降に行われた、コミットメント制御下の資源に加えられたすべての変更がバックアウトされます。

### 形式

@DTW\_ROLLBACK()

### 値

なし

### 例

**例 1:** ロールバックを指定する。

```
@DTW_ROLLBACK()  
%HTML(report){  
%}
```



## DTW\_RTVHANDLE

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

### 目的

別の呼び出しを介したこのマクロに固有で、スレッド情報、タイムスタンプ、および現行ユーザー (ある場合には) の組み合わせをベースにして計算されるトランザクション・ハンドルを生成し戻します。トランザクション・ハンドルを使用して、永続的トランザクションのパーツとして指定される URL が HTTP サーバーに固有であり、有効な要求として安全に確実に識別されるようにすることができます。

### 形式

@DTW\_RTVHANDLE(handle)

### 値

表 115. DTW\_RTVHANDLE 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>handle</i>	OUT	現行の永続的マクロのための固有トランザクション・ハンドルを含む変数。

### 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する `handle` 変数を定義する。

```
%DEFINE handle = ""
@DTW_RTVHANDLE(handle)
```

## DTW\_STATIC

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

### 目的

全マクロが永続的であることを示します。この関数は、マクロで最初のステートメントでなければなりません。この関数呼び出しの後にマクロで定義されるすべての変数は、他の方法で指定されない限り、`DTW_TERMINATE()` が呼び出されるかまたはプロセスが終了するまで、複数のマクロ呼び出しにわたって永続的になります。

タイムアウト値 (秒) を関数呼び出しで指定し、`Net.Data` が実行されているプロセスがブラウザからの応答を待つ時間を示すことができます。タイムアウト値の有効期限が切れると、プロセスは終了し、最後のコミットメント境界以降、コミット制御下の資源に対して加えられたすべての変更がロールバックされます。

以降の `@DTW_ACCEPT()` 呼び出しでタイムアウト値を指定すると、`Net.Data` はこの値を以降の呼び出しの値でオーバーライドします。この呼び出しまたは以降の `@DTW_ACCEPT()` 呼び出しでタイムアウト値を指定しないと、Web サーバーのデフォルトのタイムアウト値が使用されます。

### 形式

`@DTW_STATIC(timeout)`

`@DTW_STATIC()`

### 値

表 116. `DTW_STATIC` 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たなければならない時間 (秒) を指定する変数またはリテラル・ストリング。

### 例

例 1: 60 秒のタイムアウト値を指定する `DTW_STATIC()` への呼び出し

```
@DTW_STATIC("60")
```

## DTW\_TERMINATE

AIX	HP-UX	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。最後のコミットメント境界以降コミットメント制御下にある資源に対して加えられた、すべての変更が永続的になります。

すべてのテキストがブラウザーに出力される前に、永続的なトランザクションの論理的なすぐ前の HTML ブロックを開始するときに、DTW\_TERMINATE 関数が呼び出されます。すべてのテキスト出力が、ブロック内で関数の前に表示されると、Net.Data のエラーが発生します。アプリケーションを作成する方法によって、直前の論理 HTML ブロックが複数存在する可能性があることに注意してください。マクロが永続的状态でないときにこの関数を呼び出すと、Net.Data のエラーが発生します。

### 形式

@DTW\_TERMINATE()

### 値

なし

### 例

**例 1:** 永続的なトランザクションを終了する。

```
%HTML(QUIT){  
@DTW_TERMINATE()  
...  
%}
```



---

## 付録A. Net.Data テクニカル・ライブラリー

Net.Data テクニカル・ライブラリーは、以下の Net.Data Web サイトから使用できます。

<http://www.software.ibm.com/data/net.data/library.html>

文書	説明
OS/390 用 Net.Data 管理およびプログラミングの手引き	Net.Data のインストール、構成、および起動に関する概念とタスクの情報が入っています。また、Net.Data マクロの作成、Net.Data パフォーマンス技法の使用、Net.Data 言語環境の使用、接続の管理、およびトラブルシューティングおよびパフォーマンス調整のための Net.Data ログおよび追跡の使用方法について説明します。
OS/2、Windows NT、および UNIX 用 Net.Data 管理およびプログラミングの手引き	
OS/400 用 Net.Data 管理およびプログラミングの手引き	
Net.Data 解説書	Net.Data マクロ言語、変数および組み込み関数について説明します。
Net.Data 言語環境解説書	Net.Data 言語環境インターフェースについて説明します。
Net.Data メッセージおよびコード	Net.Data エラー・メッセージおよび戻りコードをリストします。



---

## 付録B. DB2 WWW 接続

DB2 WWW 接続がある場合は、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』
- 274ページの『SQL\_MESSAGE』
- 274ページの『SQL\_REPORT』
- 275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 接続からも呼び出されます。DB2 WWW 接続は名前付き %SQL ブロックをサポートしません。

例:

### 例 1: DB2 WWW 接続マクロ

```
%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 接続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、126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 117. Net.Data の言語環境

言語環境	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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	X
Java アプレット	X	X	X	X		X	X	X
Java アプリケーション	X		X				X	X
ODBC	X	X	X	X		X	X	X
Oracle	X							X
Perl	X	X	X	X			X	X
REXX	X		X	X	X	X	X	X
SQL	X	X	X	X	X	X	X	X
Sybase	X							X
システム	X	X	X	X	X	X	X	X
Web 登録	X		X		X	X	X	X

表 118. Net.Data の構成変数

構成変数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
CACHE_MACHINE	X							X
CACHE_PORT	X							X
DB2INSTANCE	X	X	X			X	X	X
DB2MSG				X				
DB2PLAN				X				
DB2SSID				X				
DefaultDBCp				X				
DSNAOINI				X				
DTW_CACHE_MACRO				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
DTW_LOG_DIR	X	X	X			X	X	X
DTW_LOG_LEVEL	X	X	X			X	X	X
DTW_MBMODE	X	X	X	X		X	X	X
DTW_REMOVE_WS				X				

表 118. Net.Data の構成変数 (続き)

構成変数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_SHOWSQL	X	X	X	X	X	X	X	X
DTW_SMTP_CHARSET					X			
DTW_SMTP_SERVER	X	X	X	X		X	X	X
DTW_SQL_ISOLATION					X			
DTW_SQL_NAMING_MODE					X			
DTW_VARIABLE_SCOPE	X	X	X			X	X	X
DTWR_CLOSE_REGISTRIES					X			
DTWR_CLOSE_REGISTRIES					X			
DTW_UNICODE	X	X	X			X	X	X
DTWR_CLOSE_REGISTRIES					X			

表 119. Net.Data の変数

変数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
ALIGN	X	X	X	X	X	X	X	X
DATABASE	X	X	X		X	X	X	X
DB_CASE	X	X	X	X	X	X	X	X
DB2PLAN				X				
DB2SSID				X				
DTW_APPLET_ALTTEXT	X	X	X	X		X	X	X
DTW_CURRENT_FILENAME	X	X	X	X	X	X	X	X
DTW_CURRENT_LAST_MODIFIED	X	X	X	X	X	X	X	X
DTW_DEFAULT_MESSAGE	X	X	X		X	X	X	X
DTW_DEFAULT_REPORT	X	X	X	X	X	X	X	X
DTW_EDIT_CODES					X			
DTW_HTML_TABLE	X	X	X	X	X	X	X	X
DTW_LOG_LEVEL	X	X	X			X	X	X
DTW_MACRO_FILENAME	X	X	X	X	X	X	X	X
DTW_MACRO_LAST_MODIFIED	X	X	X	X	X	X	X	X
DTW_MBMODE	X	X	X	X		X	X	X
DTW_MP_PATH	X	X	X	X	X	X	X	X
DTW_MP_VERSION	X	X	X	X	X	X	X	X
DTW_PRINT_HEADER	X	X	X	X	X	X	X	X
DTW_REMOVE_WS	X	X	X	X	X	X	X	X
DTW_SAVE_TABLE_IN	X	X	X	X	X	X	X	X
DTW_SET_TOTAL_ROWS	X	X	X	X	X	X	X	X
LOCATION				X				
LOGIN	X	X	X		X	X	X	X
Nn	X	X	X	X	X	X	X	X

表 119. Net.Data の変数 (続き)

変数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
NLIST	X	X	X	X	X	X	X	X
NULL_RPT_FIELD					X			
NUM_COLUMNS	X	X	X	X	X	X	X	X
NUM_ROWS					X			
PASSWORD	X	X	X		X	X	X	X
RETURN_CODE	X	X	X	X	X	X	X	X
ROW_NUM	X	X	X	X	X	X	X	X
RPT_MAX_ROWS	X	X	X	X	X	X	X	X
SHOWSQL	X	X	X	X	X	X	X	X
SQL_CODE	X	X	X	X	X	X	X	X
SQL_STATE	X	X	X	X	X	X	X	X
START_ROW_NUM	X	X	X	X	X	X	X	X
TOTAL_ROWS	X	X	X	X	X	X	X	X
TRANSACTION_SCOPE	X	X	X	X	X	X	X	X
V_columnName	X	X	X	X	X	X	X	X
VLIST	X	X	X	X	X	X	X	X
Vn	X	X	X	X	X	X	X	X

表 120. Net.Data の関数

関数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_ACCEPT					X			
DTW_ADD	X	X	X	X	X	X	X	X
DTW_ADDQUOTE	X	X	X	X	X	X	X	X
DTW_ASSIGN	X	X	X	X	X	X	X	X
DTW_CACHE_PAGE	X							X
DTW_COMMIT					X			
DTW_CONCAT	X	X	X	X	X	X	X	X
DTW_DATE	X	X	X	X	X	X	X	X
DTW_DELSTR	X	X	X	X	X	X	X	X
DTW_DELWORD	X	X	X	X	X	X	X	X
DTW_DIVIDE	X	X	X	X	X	X	X	X
DTW_DIVREM	X	X	X	X	X	X	X	X
DTW_EXIT	X	X	X	X	X	X	X	X
DTW_FORMAT	X	X	X	X	X	X	X	X
DTW_GETCOOKIE	X	X	X	X	X	X	X	X
DTW_GETENV	X	X	X	X	X	X	X	X
DTW_GETINIDATA	X	X	X	X	X	X	X	X
DTW_HTML_ENCODE	X	X	X	X	X	X	X	X
DTW_INSERT	X	X	X	X	X	X	X	X
DTW_INTDIV	X	X	X	X	X	X	X	X
DTW_LASTPOS	X	X	X	X	X	X	X	X

表 120. Net.Data の関数 (続き)

関数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_LENGTH	X	X	X	X	X	X	X	X
DTW_LOWERCASE	X	X	X	X	X	X	X	X
DTW_MULTIPLY	X	X	X	X	X	X	X	X
DTW_POS	X	X	X	X	X	X	X	X
DTW_POWER	X	X	X	X	X	X	X	X
DTW_QHTMLENCODE	X	X	X	X	X	X	X	X
DTW_REVERSE	X	X	X	X	X	X	X	X
DTW_ROLLBACK					X			
DTW_RVTHANDLE					X			
DTW_SENMAIL	X	X	X	X	X	X	X	X
DTW_SETCOOKIE	X	X	X	X	X	X	X	X
DTW_SETENV	X	X	X	X	X	X	X	X
DTW_STATIC					X			
DTW_STRIP	X	X	X	X	X	X	X	X
DTW_SUBSTR	X	X	X	X	X	X	X	X
DTW_SUBTRACT	X	X	X	X	X	X	X	X
DTW_SUBWORD	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
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
DTW_TB_DUMPV	X	X	X	X	X	X	X	X
DTW_TB_GETN	X	X	X	X	X	X	X	X
DTW_TB_GETV	X	X	X	X	X	X	X	X
DTW_TB_HTMLLENCODE	X	X	X	X	X	X	X	X
DTW_TB_INPUT_CHECKBOX	X	X	X	X	X	X	X	X
DTW_TB_INPUT_RADIO	X	X	X	X	X	X	X	X
DTW_TB_INPUT_TEXT	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
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
DTW_TB_SELECT	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

表 120. Net.Data の関数 (続き)

関数	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
DTW_TB_TABLE	X	X	X	X	X	X	X	X
DTW_TB_TEXTAREA	X	X	X	X	X	X	X	X
DTW_TERMINATE					X			
DTW_TIME	X	X	X	X	X	X	X	X
DTW_TRANSLATE	X	X	X	X	X	X	X	X
DTW_UPPERCASE	X	X	X	X	X	X	X	X
DTW_URLESCSEQ	X	X	X	X	X	X	X	X
DTW_WORD	X	X	X	X	X	X	X	X
DTW_WORDINDEX	X	X	X	X	X	X	X	X
DTW_WORDLENGTH	X	X	X	X	X	X	X	X
DTW_WORDPOS	X	X	X	X	X	X	X	X
DTW_WORDS	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 121. Net.Data のインターフェース

インターフェース・タイプ	AIX	HP	OS/2	OS/390	OS/400	SCO	SUN	Win NT
FastCGI	X						X	
CGI	X	X	X	X	X	X	X	X
Java Beans								X
Internet Connection API (ICAPI)	X		X	X				X

表 121. 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	OS/2	OS/390	OS/400	SCO	SUN	Win NT
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
Netscape API (NSAPI)	X						X	X
Servlets	X			X				X

表 122. Net.Data のツール

ツール	AIX	HP	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

以下の情報は、米国で提供される製品とサービスに関するものです。本書で説明されている製品、サービス、または機能でも、米国以外の国では提供されていないことがあります。お客様の地域で現在使用可能な製品およびサービスに関する情報については、その地域の IBM 担当員にお尋ねください。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

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	Lotus
DataJoiner	MVS
DB2	Net.Data
Domino	OS/2
IBM	OS/390
IMS	OS/400

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 HotJava は Sun Microsystems, Inc. の商標です。

Microsoft、Windows、Windows NT®、および Windows 95 のロゴは、Microsoft Corporation の商標です。

UNIX は、X/Open Company Ltd. のみに許諾された米国およびその他の国の登録商標です。

その他の会社名、製品名、およびサービス名は、その他の会社の商標またはサービス・マークです。

## 用語集

**絶対パス.** オブジェクトの全パス名。絶対パス名は、上位、すなわち "ルート" ディレクトリー (下方スラッシュ (/) または円記号 (¥) 文字で識別される) で始まる。

**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 では、CGI プロセスを介してパフォーマンスを改良するために、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・スペースのパーツ。同一データに続けてアクセスする場合に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)。

**CLLETTE.** Web サーバーからの要求を満たす Net.Data Live コネクションで長期にわたって対応する処理。この接続管理プログラムは、これらの要求に対応する CLLETTE 処理のスケジュールを行う。

**コミットメント制御 (commitment control).** Net.Data が、資源での操作が作業単位のパーツである場所で実行しているプロセス内で、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 (CGI) (Common Gateway Interface).** 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。

**クッキー (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つど、送り返す情報のバケット。クッキーには、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が解決され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.** Lotus Corp. および IBM が提供する Web サーバー。正規のセキュアの両方の接続を提供する。ICAPI および GWAPI は、このサーバーが提供するインターフェースである。

**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.** 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)。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).** 会社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).** 接続管理プログラムと複数のクライアントで構成される 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 仮想マシン接続の再利用を管理する。

**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間の対話を管理する。

**NSAPI.** Netscape API。

**ヌル (null).** 情報がな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).** オブジェクトを突き止める方法をシステムに示す。パス名はディレクトリー名のシーケンスとして表され、オブジェクト名がその後に続く。個々のディレクトリーおよびオブジェクト名は、下方スラッシュ (/) または円記号 (¥) 文字で区切られる。

**Perl.** 解釈済みプログラミング言語。

**永続 (persistence).** トランザクション全体に割り当てられた値を保持している状態。この場合、トランザクションは複数の Net.Data 起動にまたがる。永続的なのは変数だけである。さらに、コミットメント制御の影響を受けた資源での操作が、コミットまたはロールバックが明示的に行われるまで、すなわち、トランザクションが終了するまで、活動状態のままである。

**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を保管し検索できるリポジトリ。

**相対パス名 (relative path name).** 上位の、すなわち "ルート"・ディレクトリーで始まらないパス名。システムでは、パス名はプロセスの現行作業ディレクトリーで始まると想定される。

**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 を使用すると、トランザクションは複数の Net.Data 起動にまたがることができる。

**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、その操作が単一の操作であるかのように、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ている資源での操作だけである。

**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]

値のグループの受け渡し 75

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 263

    DTW\_COMMIT 265

    DTW\_ROLLBACK 266

    DTW\_RTVHANDLE 267

    DTW\_STATIC 268

    DTW\_TERMINATE 269

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 275

## [カ行]

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 122

    DTW\_MP\_VERSION 123

    DTW\_PRINT\_HEADER 124

    DTW\_REMOVE\_WS 125

    RETURN\_CODE 126

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

    一般 129

    永続的な 262

    数学 156

    ストリング 169

    説明 127

    表 196

    フラット・ファイル・インターフェース (FFI) 227

関数 (続き)

    命名規則 75

    ワード 186

    Web 登録 250

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

機密保護の勧告、FFI\_PATH 229

行の長さの制限、マクロ 4

許可要件、FFI\_PATH 230

区切り文字、FFI 言語環境

    ASCIITEXT 230

    DELIMITED 230

区切られた値のストリング 74

区切られたリスト・ストリング 73

クッキー

    送信 124

    DTW\_GETCOOKIE 137

    DTW\_PRINT\_HEADER 124

    DTW\_SETCOOKIE 148

組み込み関数 127

組み込みファイル 40

現行ディレクトリー、フラット・ファイルのために決定する 228

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 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
  - ストリング 6
  - 変数参照 5
  - マクロ
    - 構文 1
    - 説明 7
  - COMMENT ブロック 9
  - DB2 WWW 接続 273
  - 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
  - variable name 5
  - WHILE ブロック 65
- 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 275
- 時刻形式、UTF-8 152
- 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 157
- DTW\_DIVIDE 158
- DTW\_DIVREM 159
- DTW\_FORMAT 161
- DTW\_INTDIV 164
- DTW\_MULTIPLY 165
- DTW\_POWER 166

## 数学関数 (続き)

- DTW\_SUBTRACT 157
- 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 170
  - DTW\_CONCAT 171
  - DTW\_DELSTR 172
  - DTW\_INSERT 173
  - DTW\_LASTPOS 175
  - DTW\_LENGTH 176
  - DTW\_LOWERCASE 177
  - DTW\_POS 178
  - DTW\_REVERSE 179
  - DTW\_STRIP 180
  - DTW\_SUBSTR 181
  - DTW\_TRANSLATE 183
  - DTW\_UPPERCASE 185
  - MBCS サポート 169
- 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・ファイルのための 228
- 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
- データベース・アクセスの制限 108, 110
- 特記事項 283

## [ハ行]

- パフォーマンス、DTW\_EXIT 136
- パラメーター、渡す 25
- 汎用関数 129
  - DTW\_ADDQUOTE 130
  - DTW\_CACHE\_PAGE 132
  - DTW\_DATE 135
  - DTW\_EXIT 136
  - DTW\_GETCOOKIE 137
  - DTW\_GETENV 139
  - DTW\_GETINIDATA 140
  - DTW\_HTMLENCOD 141
  - DTW\_QHTMLENCOD 143



汎用関数 130 (続き)

DTW\_SENDMAIL 130  
DTW\_SETCOOKIE 148  
DTW\_SETENV 151  
DTW\_TIME 152  
DTW\_URLESCSEQ 154

日付形式、UTF-8 135

日付変数 115

表

HTML での結果 89  
Net.Data が行数を指定 90

表関数

DTW\_TB\_APPENDROW 197  
DTW\_TB\_COLS 198  
DTW\_TB\_DELETECOL 200  
DTW\_TB\_DELETEROW 199  
DTW\_TB\_DLIST 201  
DTW\_TB\_DUMPH 203  
DTW\_TB\_DUMPV 204  
DTW\_TB\_GETN 205  
DTW\_TB\_GETV 206  
DTW\_TB\_HTMLENCODER 207  
DTW\_TB\_INPUT\_CHECKBOX 208  
DTW\_TB\_INPUT\_RADIO 209  
DTW\_TB\_INPUT\_TEXT 210  
DTW\_TB\_INSERTCOL 212  
DTW\_TB\_INSERTROW 213  
DTW\_TB\_LIST 214  
DTW\_TB\_MAXROWS 216  
DTW\_TB\_QUERYCOLNONJ 217  
DTW\_TB\_ROWS 218  
DTW\_TB\_SELECT 219  
DTW\_TB\_SETCOLS 221  
DTW\_TB\_SETN 222  
DTW\_TB\_SETV 223  
DTW\_TB\_TABLE 224  
DTW\_TB\_TEXTAREA 226

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

表変数 (続き)

例 75

ファイルの解放、FFI 関数 231

ファイルのロック、FFI 関数 231

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 275

フラット・ファイル

アクセスのための勧告 228

機密保護の勧告 229

許可要件 230

区切り文字 230

現行ディレクトリーでの作成 228

構成規則 229

絶対パス 228

データ・ソース 227

定義 227

ファイルのロック 231

ロケーション

現行ディレクトリー 228

FFI\_PATH 228

FFI\_PATH のマッチング 228

フラット・ファイルへのアクセス 227

フラット・ファイルflat files

アクセス 227

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 73

レポート 86

Net.Data、概要 69

変数参照 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 136

宣言パーツ 2

表示パーツ 2

マクロからの電子メールの送信 144  
メッセージ、デフォルト・テキスト 118

## [ヤ行]

用語集 284  
呼び出し  
外部プログラム 16  
関数 27

## [ラ行]

リスト変数  
値区切り記号 74  
説明 73  
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、フラット・ファイル 228  
ロケーション、DB2 サブシステムへの接続 107

## [ワ行]

ワード関数  
DTW\_DELWORD 187  
DTW\_SUBWORD 188  
DTW\_WORD 190  
DTW\_WORDINDEX 191  
DTW\_WORDLENGTH 192  
DTW\_WORDPOS 193  
DTW\_WORDS 195  
MBCS サポート 186  
渡すパラメーター、System 言語環境 25

## A

ALIGN 87  
APPLET タグ、代替テキスト 101

## C

COMMENT ブロック  
構文 9

COMMENT ブロック (続き)  
説明 9

## D

DATABASE 96  
DB2 WWW 接続、言語構成要素 273  
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 231  
DTWF\_CLOSE 231, 234  
DTWF\_DELETE 235  
DTWF\_INSERT 237  
DTWF\_OPEN 231, 239  
DTWF\_READ 241  
DTWF\_REMOVE 243  
DTWF\_SEARCH 244  
DTWF\_UPDATE 246  
DTWF\_WRITE 248  
DTWR\_ADDENTRY 250  
DTWR\_CLEARREG 252  
DTWR\_CLOSEREG 253  
DTWR\_CREATEREG 254  
DTWR\_DELENTY 255  
DTWR\_DELREG 256  
DTWR\_LISTREG 257  
DTWR\_LISTSUB 258  
DTWR\_OPENREG 259  
DTWR\_RTVENTRY 260  
DTWR\_UPDATEENTRY 261  
DTW\_ACCEPT 263  
DTW\_ADD 156  
DTW\_ADDQUOTE 130  
DTW\_APPLET\_ALTTEXT 101  
DTW\_ASSIGN 77, 169, 170  
DTW\_CACHE\_PAGE 132  
DTW\_COMMIT 265  
DTW\_CONCAT 171  
DTW\_CURRENT\_FILENAME 116  
DTW\_CURRENT\_LAST\_MODIFIED 117  
DTW\_DATE 135  
DTW\_DEFAULT\_MESSAGE 118  
DTW\_DEFAULT\_REPORT 88  
DTW\_DELSTR 172  
DTW\_DELWORD 186



DTW\_DIVIDE 158  
DTW\_DIVREM 159  
DTW\_EDIT\_CODES 102  
DTW\_FORMAT 161  
DTW\_GETCOOKIE 137  
DTW\_GETENV 139  
DTW\_GETINIDATA 140  
DTW\_HTMLENCODER 141  
DTW\_HTML\_TABLE 89  
DTW\_INSERT 173  
DTW\_INTDIV 164  
DTW\_LASTPOS 175  
DTW\_LENGTH 176  
DTW\_LOG\_LEVEL 119  
DTW\_LOWERCASE 177  
DTW\_MACRO\_FILENAME 120  
DTW\_MACRO\_LAST\_MODIFIED 121  
DTW\_MBMODE 103  
DTW\_MP\_PATH 122  
DTW\_MP\_VERSION 123  
DTW\_MULTIPLY 165  
DTW\_POS 178  
DTW\_POWER 166  
DTW\_PRINT\_HEADER 124  
DTW\_QHTMLENCODER 143  
DTW\_REMOVE\_WS 125  
DTW\_REVERSE 179  
DTW\_ROLLBACK 266  
DTW\_RTVHANDLE 267  
DTW\_SAVE\_TABLE\_IN 104  
DTW\_SENMAIL 144  
DTW\_SETCOOKIE 148  
DTW\_SETENV 151  
DTW\_SET\_TOTAL\_ROWS 105  
DTW\_STATIC 268  
DTW\_STRIP 180  
DTW\_SUBSTR 181  
DTW\_SUBTRACT 167  
DTW\_SUBWORD 188  
DTW\_TB\_APPENDROW 197  
DTW\_TB\_COLS 198  
DTW\_TB\_deleteCOL 200  
DTW\_TB\_DELETEROW 199  
DTW\_TB\_DLIST 201  
DTW\_TB\_DUMP 203  
DTW\_TB\_DUMPV 204  
DTW\_TB\_GETN 205  
DTW\_TB\_GETV 206  
DTW\_TB\_HTMLENCODER 207  
DTW\_TB\_INPUT\_CHECKBOX 208  
DTW\_TB\_INPUT\_RADIO 209  
DTW\_TB\_INPUT\_TEXT 210

DTW\_TB\_INSERTCOL 212  
DTW\_TB\_INSERTROW 213  
DTW\_TB\_LIST 212  
DTW\_TB\_MAXROWS 216  
DTW\_TB\_QUERYCOLNONJ 217  
DTW\_TB\_ROWS 218  
DTW\_TB\_SELECT 219  
DTW\_TB\_SETCOLS 221  
DTW\_TB\_SETN 222  
DTW\_TB\_SETV 223  
DTW\_TB\_TABLE 224  
DTW\_TB\_TEXTAREA 226  
DTW\_TERMINATE 269  
DTW\_TIME 152  
DTW\_TRANSLATE 183  
DTW\_UPPERCASE 185  
DTW\_URLESCSEQ 154  
DTW\_WORD 190  
DTW\_WORDINDEX 191  
DTW\_WORDLENGTH 192  
DTW\_WORDPOS 193  
DTW\_WORDS 195

## 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 273

## F

FFI 関数  
    ファイルの解放 231  
    ファイルのロック 231  
DTWF\_APPEND 232  
DTWF\_CLOSE 234  
DTWF\_DELETE 235  
DTWF\_INSERT 237  
DTWF\_OPEN 239  
DTWF\_READ 241  
DTWF\_REMOVE 243  
DTWF\_SEARCH 244  
DTWF\_UPDATE 246  
DTWF\_WRITE 248  
FFI 言語環境  
    機密保護の勧告 229

## FFI 言語環境 (続き)

許可要件 229

区切り文字 230

現行ディレクトリー 228

構成規則 229

ファイルへのアクセス 227

ファイル・ロケーション 228

FFI 言語環境の構成 229

FFI 言語環境の呼び出し 227

## FFI\_PATH

機密保護の勧告 229

構成規則 229

構文 227

フラット・ファイルへのアクセス 227

フラット・ファイル・ロケーション 228

*filename* パラメーターを使用したパスのマッチング  
228

## 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

表変数の表示 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 273

HTML\_REPORT ブロック 273

# I

## IF ブロック

構文 33

説明 33

IN キーワード 21, 49, 127

## 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, 127

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 169

ワード関数 186

## 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, 127

# 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

DTW\_HTML\_TABLE 89

NLIST 78

NUM\_COLUMNS 79

NUM\_ROWS 80

REPORT ブロック (続き)  
    Nn 57  
    RPT\_MAX\_ROWS 90  
    START\_ROW\_NUM 92  
    TOTAL\_ROWS 82  
RETURNS キーワード 22  
RETURN\_CODE 126  
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 273  
SQL\_CODE 275  
SQL\_MESSAGE ブロック 274  
SQL\_REPORT ブロック 274  
SQL\_STATE 112  
START\_ROW\_NUM 92  
System 言語環境、渡すパラメーター 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  
    時刻 152  
    日付 135

## V

variable name 5  
VLIST 84

Vn 85  
V\_columnName 83

## W

Web 登録関数  
    DTWR\_ADDENTRY 251  
    DTWR\_CLEARREG 252  
    DTWR\_CLOSEREG 253  
    DTWR\_CREATEREG 254  
    DTWR\_DELENTY 255  
    DTWR\_DELREG 256  
    DTWR\_LISTREG 257  
    DTWR\_LISTSUB 258  
    DTWR\_OPENREG 259  
    DTWR\_RTVENTRY 260  
    DTWR\_UPDATEENTRY 261  
WHILE ブロック  
    構文 65  
    説明 65







Printed in Japan

SB88-7370-01



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