



Net.Data 管理およびプログラミング の手引き OS/390



Net.Data 管理およびプログラミング の手引き OS/390

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、 95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-2APG-00
Net.Data
Administration and Programming Guide
for OS/390

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 1998.8

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 1998

目次

まえがき	v
Net.Data について	v
本書について	vi
本書の対象読者	vii
本書の例について	vii
 第1章 Net.Data for OS/390 とは何か ?	1
 第2章 Net.Data のインストールおよび構成	5
Net.Data の初期設定ファイルについて	5
Net.Data の初期設定ファイルのインストール	6
Net.Data の初期設定ファイルのカスタマイズ	6
構成変数ステートメント	7
パス構成ステートメントのカスタマイズ	11
環境構成ステートメント	15
DB2 との接続の管理	16
作業負荷管理の考慮事項	17
CGF と一緒に使用する Net.Data の構成	17
ICAPI あるいは GWAPI と一緒に使用するための Net.Data の構成	19
言語環境のセットアップ	20
REXX 言語環境のセットアップ	21
SQL および ODBC の言語環境のセットアップ	21
メッセージ・カタログを使用可能にする	22
Net.Data のファイルおよびデータ・セットへのアクセス権の指定	22
 第3章 ユーザー資産を保護する	25
ファイアウォールを使用する	25
ネットワーク上のユーザーのデータを暗号化する	26
認証を使用する	26
許可を使用する	27
Net.Data のメカニズムを使用する	27
 第4章 Net.Data を起動する	29
マクロ・ファイルで Net.Data を呼び出す (マクロ要求)	30
HTML リンク	31
HTML フォーム	32
マクロ・ファイルを使用しない Net.Data の起動 (直接要求)	33
直接要求の構文	33
直接要求の例	36
 第5章 Net.Data のマクロ開発	39
Net.Data のマクロ・ファイルの分析	40
DEFINE ブロック	41
FUNCTION 定義ブロック	42
HTML ブロック	43
Net.Data のマクロ変数	44
変数の定義	46
変数の参照	47
変数の型	48
Net.Data の関数	55

関数の定義	55
関数の呼び出し	57
ストアード・プロシージャの呼び出し	58
メッセージ・ブロック	60
マクロに HTML を作成	62
HTML ブロック	62
レポート・ブロック	63
マクロ・ファイルにおける条件付き論理とループ	65
条件付き論理	65
ループ構成体	66
第6章 組み込み関数の使用	69
汎用操作関数	69
第7章 言語環境の使用	71
第8章 パフォーマンスを向上させる	73
ICAPI または GWAPI を使用してパフォーマンスを向上させる	73
DB2 for OS/390 のメッセージを抑制する	73
付録A. Net.Data for OS/390 バージョン 2 SMP/E のインストール	75
インストール要件および考慮事項	75
駆動システム要件	76
ターゲットのシステム要件	76
削除される FMID	78
特別な考慮事項	78
SMP/E を使った Net.Data のインストール	78
インストールの概説	79
インストール・ステップの概説	80
ステップ 1: サンプルの JCL を、製品テープからアンロードする	81
ステップ 2: 希望する割り当てジョブの更新と実行	81
ステップ 3: 階層ファイル・システム構造の作成	84
ステップ 4: SMP/E の受信ジョブの更新と実行	84
ステップ 5: SMP/E の APPLY CHECK および APPLY Job の更新と実行	85
ステップ 6: SMP/E の ACCEPT CHECK および ACCEPT Job の更新と実行	85
インストールの検証	86
付録B. Net.Data for OS/390 を構成し、DataJoiner をアクセスする	89
付録C. Net.Data サンプル・マクロ	91
付録D. 特記事項	95
商標	96
用語集	97
索引	99

まえがき

Net.Data バージョン 2 をお買い上げいただきありがとうございます。本製品は、動的 Web ページを作成するための IBM 製開発ツールです。Net.Data を使用すれば、さまざまなデータ・ソースからデータを取り込んだり、既知のプログラム言語が持つ長所を生かしたりして、動的コンテンツを含んだ Web ページを迅速に開発することができます。

Net.Data バージョン 2 は、ユーザーのインターネット・ビジネス・ソリューションの開発および活用に関与する新規機能を提供するとともに、大幅なパフォーマンスの向上を実現しています。

Net.Data について

IBM の Net.Data プロダクトを用いると、DB2、IMS、および ODBC 対応のデータベースを含む関係データベース管理システムおよび非関係データベース管理システム (DBMS) の両方、ならびに Java、JavaScript、Perl、C、C++、および REXX などプログラミング言語で作成されたアプリケーションを使用して、動的 Web ページを作成することができます。

OS/390 用 Net.Data はマクロ処理プログラムで、Web サーバーを含む OpenEdition サーバー・マシン上のミドルウェアです。ユーザーは、Net.Data アプリケーション・プログラムである呼び出し先マクロを作成することができます。Net.Data は、ユーザーからの入力、データベースまたは Hierarchical File System (HFS) ファイルの現在の状態、既存のビジネス論理、およびマクロの設計にとり込むその他の要素に基づくカスタマイズされた内容を解釈しながら、動的 Web ページを作成します。

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

- ビジネス・ロジックを Perl スクリプト、C および C++ アプリケーション、もしくは REXX プログラム内でカプセル化する
- DB2 などのデータベ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。このページはネットワークを通してブラウザに転送され、表示されます。Net.Data ファミリーの他の製品は、ウィンドウズ NT、AIX、OS/2、AS/400、HP-UX、Sun Solaris、および Santa Cruz Operating System (SCO) オペレーティング・システムを実行するマシン上でも、同様の機能を提供します。

Net.Data は、HTTP (HyperText Transfer Protocol) およびコモン・ゲートウェイ・インターフェース (CGI) などの業界標準のインターフェースをサポートしています。HTTP はブラウザと Web サーバー間で使用され、CGI は Web サーバーと Net.Data 間で使用されます。このようなサポートによって、Net.Data で使用するブラウザお

よび Web サーバーは、ユーザーが好きなものを選択することができます。Net.Data for OS/390 バージョン 2 は、パフォーマンスを向上させるために ICAPI および GWAPI Web サーバー API もサポートします。

さらに Net.Data for OS/390 バージョン 2 では、パフォーマンスとスケーラビリティが拡張され、以下のようなユーザーのアプリケーションの要件を満たしています。

- SQL 言語環境によって使用される OS/390 用 DB2 接続の再利用
- ICAPI 環境での OS/390 用作業負荷管理プログラムとの統合
- マクロ・ファイルを使用しない Net.Data の呼び出し (直接要求)
- 生成された Web ページ内の、無駄になっている空白スペースの最小化
- OS/390 用 DB2 メッセージ・テキスト・ルックアップをう回する機能

Net.Data バージョン 2 では、以下のようなたくさんの機能も拡張されています。

- 言語環境は、ODBC 言語環境を使用して、ストアード・プロシージャーを実行する機能が拡張されています。
- マクロ言語環境は、次のものを含んでいます。
 - 注釈をどこでも追加する機能
 - ネストされた IF ブロック
 - WHILE ブロック
 - ストアード・プロシージャーから単一の結果セットを受信する機能
 - DBCS 対応のストリングおよびワード関数
 - ストアード・プロシージャー用パラメーター・リスト内の SQL 10 進数データ・タイプのサポート
- この他の拡張された機能は、データベースのデータを統合する Web ページを構成する機能です。このデータは、他のコード・ページで符号化されたマクロから、HTML タグで 1 つのコード・ページに符号化されたものです。

本書について

本書は Net.Data の管理とプログラミングの概念について解説しています。Net.Data およびその構成要素の構成方法、機密保護の設計、およびパフォーマンスの向上についても解説しています。

いままでのプログラム言語やデータベースの知識を基に、ユーザーは Net.Data マクロ言語を使用したマクロの開発方法を学習します。ユーザーは、Net.Data が提供する言語環境の使用方法を学習します。この言語環境を利用して、DB2 データベースや IMS トランザクションを IMS Web を使用してアクセスしたり、Java、REXX、Perl、および他のプログラム言語を使用してユーザー・データをアクセスします。

本書では、告知後、まだ発売されていない製品または機能について言及する場合があります。

サンプル Net.Data マクロ、デモ、および本書の最新バージョンの詳細な情報については、以下の World Wide Web サイトをご覧ください。

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

本書の対象読者

本書は、Net.Data アプリケーションを設計し、開発する方々を対象としています。オペレーティング・システムの差、Net.Data メッセージ、およびその他の情報は、*Net.Data 解説書* で解説しています。

本書で説明する概念を理解するには、C プログラミング言語、ならびに Web サーバーの仕組みに関する知識があり、単純な SQL ステートメント、HTML タグ、および HTML フォームを理解する必要があります。さらに、以下の解説書の情報に精通しておく必要があります。"*Net.Data 解説書*" および "*Net.Data 言語環境プログラム解説書*"

本書の例について

本書に記載されている例は、特定の概念を説明するために単純化されたものになっています。Net.Data の構成要素が使用されるすべてのケースが示されているわけではありません。例の中には、コードを追加しないと機能しない断片的なものもあります。

第1章 Net.Data for OS/390 とは何か？

Web ページは、HTML だけで作成することができます。したがって、ユーザーがこれらのページを編集しない限り、ページは変更されません。Web 上に“live”データとアプリケーション（現在の営業統計など）を組み込むには、Web サイトの開発者は通常、プログラムを作成します。このプログラムは、Web サーバーのミドルウェアとして Web ページを動的に構築します。この種のプログラムの作成は簡単ではありません。

Net.Data を使用すると、マクロを使用して、対話式 Web アプリケーションを簡単に作成することができます。Net.Data マクロによって、論理、変数、関数呼び出し、およびレポート生成ツールを使用することができます。マクロとは Net.Data マクロ言語の構成要素、HTML タグ、および SQL や Perl などの言語環境ステートメントが組み込まれているテキスト・ファイルです。Net.Data はマクロ・ファイル処理して HTML 出力を生成します。マクロは、単純な HTML と、Web サーバー・プログラムの動的な機能性を結合します。この結果、live データを静的 Web ページに簡単に追加することができるようになります。live データは、ローカルデータベースやリモートのデータベースから、さらにフラット・ファイルから抽出することができます。アプリケーションおよびシステム・サービスから生成することもできます。

1ページの図1は、Net.Data for OS/390および Web サーバーの関係、サポートされるデータおよびプログラム言語環境への関係を示しています。

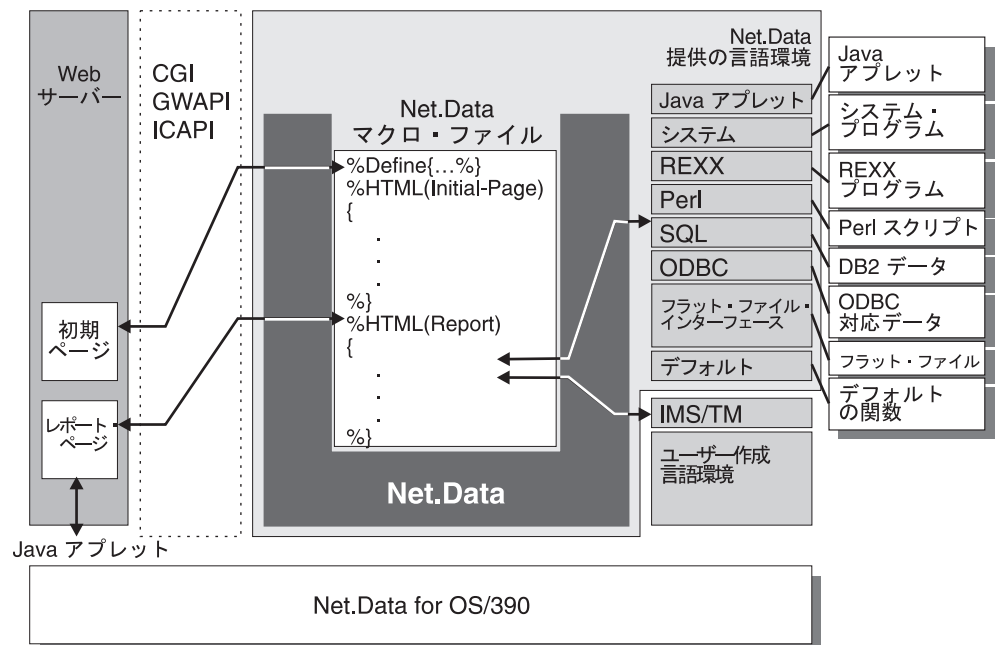


図1. Net.Data for OS/390、Web サーバー、およびサポートされるデータとプログラム・ソース間の関係

Web サーバーは、Net.Data サービスを要求する URL を受信した時に、CGI または 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) を使用して、Net.Data を起動します。URL には Net.Data 特有の情報が組み込まれてい

ます。これらの情報は、マクロ・ファイルとして処理されるか、SQL ステートメントまたはプログラムとして直接的に起動されます。Net.Data は URL の処理を終了し、結果の Web ページを Web サーバーに送信します。サーバーはそのページを Web クライアントに渡し、そのクライアントでブラウザを使って表示します。

Net.Data を利用すれば、動的な Web ページを非常に簡単に作成できます。マクロ言語を使用すると、ユーザーが独自に Web サーバー・アプリケーションをプログラムするよりも簡単です。Net.Data では、ユーザーが現在知っている HTML、SQL、Perl、REXX、および JavaScript などの言語を使用することができます。

この他 Net.Data の重要な利点としては、多数の異なるデータベース・ソースへのアクセスをサポートしていることです。これによって Web 開発者は、DB2、IMS、Oracle、および Sybase などのさまざまなデータベースのデータを利用することができます。Net.Data が提供する言語環境に関するさらに詳細な情報は、"*Net.Data* 言語環境解説書"を参照してください。

Net.Data Web アプリケーション環境には、以下の機能があります。

インタープリター・マクロ言語

Net.Data マクロ言語はインタープリター言語です。Net.Data はマクロを処理するために起動されると、各言語ステートメントを直接的にファイルの先頭から順次に解釈を開始します。このアプローチで、ユーザーがマクロに加えた変更は、そのマクロを実行する URL をユーザーが指定すると即時に反映されます。再度コンパイルをする必要はありません。

直接要求

単一の SQL ステートメント、DB2 ストアド・プロシージャ、REXX プログラム、C や C++ プログラム、または Perl スクリプトを実行するための単純な要求では、マクロを作成する必要はありません。このような要求は直接 URL で指定して、ブラウザから Web サーバーまでの流れを制御します。

フリー・フォーマット

Net.Data マクロ言語には、2、3 のプログラミング・フォーマットがあるだけです。この単純さは、プログラマーに自由度と柔軟性をもたらします。単一の命令が複数の行に分解されたり、複数の命令が単一の行にまとめられたりします。命令はどの列からも開始することができます。スペースやすべての行をスキップすることができます。コメントはどこでも使用できます。

型を持たない変数

Net.Data はすべてのデータを文字ストリングとして取り扱います。Net.Data は組み込み関数を使用して、有効な数字を表しているストリングを算術演算します。指数のフォーマットもサポートされます。マクロ言語の変数の詳細については、44ページの『Net.Data のマクロ変数』で解説されています。

Net.Data 関数

Net.Data は組み込み関数を使用して、テキストや数字に関するいろいろな処理、検索、および比較演算を行います。他の組み込み関数には、フォーマット機能や算術計算があります。

エラー処理

Net.Data がエラーを検出すると、説明の付いたメッセージがクライアントに戻されます。ブラウザーを使用するユーザーにエラー・メッセージが戻される前に、メッセージをカスタマイズすることができます。詳しくは、"*Net.Data* 解説書"を参照してください。

第2章 Net.Data のインストールおよび構成

SMP/E を使用することにより、Net.Data for OS/390 をインストールすることができます。Net.Data for OS/390 のプログラム資料説明書では、インストール・プロセスが説明されています。また、製品のインストール・テープが付属しています。

ユーザーのオペレーティング・システムに対応した Net.Data を、SMP/E を使用してインストール後、Net.Data を構成し、Web サーバーのための構成を変更しなければなりません。そのためには、以下を行います。

- Net.Data の初期設定 (INI) ファイルのインストールおよびカスタマイズ
- CGI、ICAPI、あるいは GWAPI との併用のための Net.Data の構成
- Web サーバーの構成および環境変数ファイルのカスタマイズ
- Net.Data 言語環境の設定
- アクセス権限の指定
- メッセージ・カタログの使用可能化

本章では、Net.Data の構成方法、Net.Data と一緒に使用するための Web サーバーの構成方法について説明します。

- 6ページの『Net.Data の初期設定ファイルのインストール』
- 6ページの『Net.Data の初期設定ファイルのカスタマイズ』
- 16ページの『DB2 との接続の管理』
- 17ページの『作業負荷管理の考慮事項』
- 17ページの『CGF と一緒に使用する Net.Data の構成』
- 19ページの『ICAPI あるいは GWAPI と一緒に使用するための Net.Data の構成』
- 20ページの『言語環境のセットアップ』
- 22ページの『メッセージ・カタログを使用可能にする』
- 22ページの『Net.Data のファイルおよびデータ・セットへのアクセス権の指定』

Net.Data の初期設定ファイルについて

Net.Data は、その初期設定ファイルを使用して、さまざまな構成変数の設定を確立し、言語環境と検索パスを構成します。構成変数の設定は、Net.Data 操作のさまざまな側面を制御します。たとえば、Web ページ内の文字データの符号化、ストリングおよびワード関数は、DBCS が使用可能かどうか、および DB2 と DRDA により使用可能となるデータへのアクセスのためのデフォルトのサブシステム ID の選択、などです。

言語環境のステートメントは、使用可能な Net.Data の言語環境を定義し、言語環境に入出力する特殊な入力および出力パラメーター値を識別します。パス・ステートメントは、Net.Data が使用する、マクロ、REXX プログラム、および Perl スクリプトなどの HFS ファイルのディレクトリー・パスを指定します。

Net.Data の初期設定ファイルのインストール

SMP/E のインストール処理を行うと、db2www.ini という名前の Net.Data の初期設定ファイルのサンプルが、ディレクトリー /usr/lpp/netdata/pub に作成されます。

Net.Data の初期設定ファイルをインストールするには、以下を行います。

1. Net.Data のサンプルの初期設定ファイルを、Web サーバーの文書のルート・ディレクトリーにコピーする。(Web サーバーの文書のルート・ディレクトリーは、Web サーバーの構成ファイル、/etc/httpd.conf において、要求テンプレート『/*』として、Pass ディレクティブにより指定されています。Web サーバーのデフォルトの文書のルート・ディレクトリーは、/usr/lpp/internet/server_root/pubです。しかし、これは、Web サーバーがインストールされた時点で変更されているかもしれません。Web サーバーの文書のルート・ディレクトリーが、internet/server_root/pub と異なっている場合、以下の指示の適切なものを選択して置き換えてください。)

Net.Data を、ディレクトリー /usr/lpp/netdata にインストールした場合は、OMVS 下で、以下のシェル・コマンドを実行して、初期設定ファイルをコピーすることができます。

```
cp /usr/lpp/netdata/pub/db2www.ini /usr/lpp/internet/server_root/pub
```

2. Net.Data の初期設定ファイルの許可番号は、必ず 644 になるようにください。

Net.Data の初期設定ファイルのカスタマイズ

初期設定ファイルに含まれている情報は、以下の節で説明する、3 つのタイプの構成ステートメントを使用して指定されます。

- 7ページの『構成変数ステートメント』
- 11ページの『パス構成ステートメントのカスタマイズ』
- 15ページの『環境構成ステートメント』

6ページの図2 に示されているサンプルの初期設定ファイルは、次のステートメントの例を含んでいます。

```
1 DB2SSID DBNC
2 DB2PLAN DTWGAV21
3 MACRO_PATH /usr/lpp/netdata/macros;
4 EXEC_PATH /usr/lpp/netdata/testcmd;
5 FFI_PATH /usr/lpp/netdata/file-data;

6 ENVIRONMENT (DTW_SQL) dtwsq1 (IN DB2SSID, IN DB2PLAN, IN LOCATION)
7 ENVIRONMENT (DTW_DEFAULT) defcd11 (IN DTW_MBMODE,OUT RETURN_CODE)
8 ENVIRONMENT (DTW_SYSTEM) sysd11 (OUT RETURN_CODE)
9 ENVIRONMENT (DTW_PERL) perld11 (OUT RETURN_CODE)
10 ENVIRONMENT (DTW_REXX) rexxd11 (OUT RETURN_CODE)
11 ENVIRONMENT (DTW_FILE) filed11 (OUT RETURN_CODE)
12 ENVIRONMENT (DTW_APPLET) appld11 (OUT RETURN_CODE)
13 ENVIRONMENT (DTW_ODBC) odbcd11 (OUT RETURN_CODE)
```

- 1 ～ 2 行は、構成変数を定義しています。
- 3 ～ 5 行は、HFS ファイルのパスを定義しています。
- 6 ～ 13 行は、使用可能な環境ステートメントを定義しています。

図2. Net.Data の初期設定ファイル

以下の節では、INI ファイルの構成ステートメントのカスタマイズ方法について説明します。

構成変数ステートメント

Net.Data の構成変数ステートメントは、構成変数の値を設定します。構成変数は、さまざまな目的に使用されます。変数の中には、適切な動作、あるいは代替モードでの動作のために、言語環境が必要とするものがあります。また変数によっては、文字の符号化や、構成中の Web ページの内容を制御するものもあります。さらに、構成変数ステートメントを使用して、アプリケーションに固有の変数を定義することもできます。

使用する構成変数は、環境変数、および使用しているDB2 システムに依存します。また、その他、アプリケーションに固有の要因にも依存します。

構成変数ステートメントを更新するには、アプリケーションが要求する構成変数を使って初期設定ファイルをカスタマイズします。構成変数は、以下の構文を持ちます。

(Web サーバーの文書のルート・ディレクトリーは、Web サーバーの構成ファイル/etc/httpd.conf において、要求テンプレートを 『/*』 として、Pass ディレクティブで指定されます。Web サーバーのデフォルトの文書のルート・ディレクトリーは、/usr/lpp/internet/server_root/pub ですが、これは、Web サーバーがインストールされた時点で変更されているかもしれません。Web サーバーの文書のルート・ディレクトリーが、internet/server_root/pub と異なっている場合は、以下の指示においては、適宜ユーザーの選択ディレクトリーに置き換えてください。)NAME[=]value-string

等号は、オプションです。大括弧で示されます。

以下のサブセクションは、初期設定ファイルで 사용할 ことができる構成変数ステートメントについて説明しています。

- 9ページの『DefaultDBCp: デフォルトのデータベース・コード・ページ変数』
- 8ページの『DB2MSGs: DB2 のメッセージ・テキスト変数』
- 8ページの『DB2PLAN: DB2 のプラン変数』
- 9ページの『DB2SSID: DB2 のサブシステム ID 変数』
- 10ページの『DSNAOINI: DB2 CLI の初期設定ファイル変数』
- 10ページの『DTW_REMOVE_WS: 余分な空白文字を削除するための変数』
- 10ページの『DTW_MBMODE: ネイティブ言語サポート変数』

Net.Data のサンプルの初期設定ファイルは、Net.Data の構成変数の設定のカスタマイズについて、幾つかの前提事項を設けます。これらの前提事項は、使用している環境では正しくないこともあります。

- DB2 サブシステムの ID 指定は、DBNC を使用する。ユーザーのアプリケーションの DB2SSID 構成変数を使用して、この値を置き換えます。
- DB2 プラン指定は、DTWGAV21 を使用する。ユーザー・アプリケーションの DB2PLAN 構成変数を使用して、この値を置き換えます。

DB2MSGSG: DB2 のメッセージ・テキスト変数

SQL の言語環境を使用して、DB2 for OS/390 にアクセスする場合に、DB2 提供のメッセージ・テキストを表示するかどうかを指定します。

サポートされるオペレーティング・システム：DB2MSGSG 構成変数は、OS/390 オペレーティング・システムのみでサポートされます。

構文：

DB2MSGSG [=] *message_level*

ここで、*message_level* は、Net.Data が表示する DB2 提供のメッセージのレベルを示しています。 *message_level* は、以下の値に設定することができます。

NONE	Net.Data は、メッセージ・テキストを表示しないことを指定します。
ERRORONLY	SQLCODE の値が負の場合にのみ、Net.Data はメッセージ・テキストを表示するよう指定します。
ALL	SQLCODE のすべての値の場合に、Net.Data がメッセージ・テキストを表示するように指定します。これがデフォルトです。上でリストした有効な値のいずれとも異なる値が DB2MSGSG に与えられる場合は、Net.Data は、デフォルト値 ALL を使用します。

例：DB2 のメッセージ・テキストのレベルを設定します。

DB2MSGSG = NONE

パフォーマンスのためのヒント：DB2 のメッセージ・テキストを、ブラウザーで表示する必要がない場合は、NONE を指定すると、パフォーマンスを改善することができます。DB2 の警告メッセージ・テキストを、ブラウザーで表示する必要がない場合は、ERRORONLY を指定すると、パフォーマンスを向上することができます。

DB2PLAN: DB2 のプラン変数

DB2 for OS/390 にアクセスする場合に、SQL 言語環境により使用される DB2 のデフォルトのプランを指定します。

構文：

DB2PLAN [=] *plan_name*

例：デフォルトの DB2 プラン名を設定します。

DB2PLAN = DTWGA21

マクロ・ファイルの初期設定ファイルの設定を上書きするには、以下のようになります。

1. DB2PLAN 変数を、初期設定ファイルの DTW_SQL ENVIRONMENT ステートメントのパラメーターとして、以下の例に示すように追加する。
ENVIRONMENT (DTW_SQL) dtwsq1 (IN DB2PLAN)
2. マクロ・ファイルにおいて、変数 DB2PLAN を、アプリケーションに要求される値に設定する。

DB2SSID: DB2 のサブシステム ID 変数

DB2 for OS/390 にアクセスする場合に、SQL 言語環境により使用されるデフォルトの DB2 のサブシステム ID を指定します。

構文：

```
DB2SSID [=] subsystem_id
```

例：デフォルトの DB2 サブシステム ID を設定します。

```
DB2SSID = DBNC
```

マクロ・ファイルの初期設定ファイルの設定を上書きするには、以下のようになります。

1. DB2SSID 変数を、初期設定ファイルの DTW_SQL ENVIRONMENT ステートメントのパラメーターとして、以下の例に示すように追加する。

```
ENVIRONMENT (DTW_SQL) dtwsq1 (IN DB2SSID)
```

2. マクロ・ファイルでは、変数 DB2SSID を、アプリケーションに要求される値に設定する。

DefaultDBCp: デフォルトのデータベース・コード・ページ変数

データベースのデータにアクセスするときに、Net.Data が使用するデフォルトのコード・ページ Net.Data は、この変数の設定を、以下の目的のために使用します。

- SQL ステートメント・テキストおよびストアド・プロシージャの呼び出しのための入力変数の値を、デフォルトのファイル・システムのコード・ページから、デフォルトのデータベースのコード・ページに変換する。
- ストアド・プロシージャの呼び出しおよび結果表からの出力変数の値を、デフォルトのデータベースのコード・ページから、デフォルトのファイル・システムのコード・ページに変換する。

Web サーバーの構成ファイル (/etc/httpd.conf) は、DefaultFsCp および DefaultNetCp ディレクティブを介して、デフォルトのコード・ページ環境を指定します。DefaultFsCp ディレクティブは、サーバー上のデフォルトのファイル・システムを指定します。このコード・ページは、EBCDIC のコード・ページです。Web サーバーは、この EBCDIC コード・ページで、Net.Data からテキスト・ストリームを受け取るものと予想しています。デフォルトの DefaultNetCp ディレクティブは、デフォルトのネットワークのコード・ページを指定します。このコード・ページは、ASCII のコード・ページです。この ASCII のコード・ページは、Web サーバーにより提供されるテキスト・ストリームを符号化するのに使用されます。

パフォーマンスのためのヒント：コード・ページ変数 DefaultDBCp は、アプリケーションに必要なければ、構成しないでください。この変数を定義すると、Net.Data は、特殊な変換が必要であることを前提にしています。

DefaultDBCp が、INI ファイル内で指定されていない場合、Net.Dataは、データベースのデータのコード・ページは、デフォルトのファイル・システムのコード・ページと等価であるとみなし、変換が行われません。

構文：

DefaultDBCp [=] *code_page*

DSNAOINI: DB2 CLI の初期設定ファイル変数

DB2 CLI の初期設定ファイルの名前を指定します。この構成変数の名前は、順次データ・セットあるいは区分データ・セットのメンバーのいずれにもなることができます。

Net.Data の ODBC 言語環境を使用したい場合は、この変数を使用して、DB2 CLI の初期設定ファイルの名前を指定します。ODBC の言語環境を ICAPI と一緒に使用するつमりの場合は、DB2 CLI の初期設定ファイルの MVSATTACHTYPE 変数を RRSAP に設定します。また、PLANNAME 変数を、DB2PLAN で指定された名前と同じプラン名に設定します。

構文 :

DSNAOINI [=] *CLI_initialization_file_name*

例 1: 順次データ・セット CLI の初期設定ファイル名

DSNAOINI DBNC.DSNAOINI

例 2: 区分データ・セット (PDS) のメンバー

DSNAOINI DBNC.CLI(DSNAOINI)

DTW_REMOVE_WS: 余分な空白文字を削除するための変数

タブ、ブランク、および改行文字で構成される不必要なスペースを削除して、動的に生成された Web ページのサイズを削減します。この変数が YES に設定されていると、Net.Data は、2 つ以上の連続した空白文字を、1 つの改行文字に圧縮し、より短くした HTML のページを生成します。デフォルトは、NO です。

構文 :

DTW_REMOVE_WS [=] YES|NO

例: 空白文字の圧縮

DTW_REMOVE_WS = YES

DTW_MBMODE: ネイティブ言語サポート変数

ワードおよびストリング関数の各国語サポートをアクティブにします。この変数の値が YES の場合、すべてのストリングおよびワード関数は、ストリングを混合データとして (すなわち、1 バイト文字集合および 2 バイト文字集合の両者の文字を含む可能性のあるストリングとして) 扱うことにより、ストリング内の DBCS 文字を正しく処理します。デフォルト値は、NO です。Net.Sata のマクロ・ファイルで DTW_MBMODE 変数を設定することにより、初期設定ファイルで設定されている値を上書きすることができます。

構文 :

DTW_MBMODE [=] NO|YES

例：各国語サポートをアクティブにします。

```
DTW_MBMODE = YES
```

マクロ・ファイルの初期設定ファイルの設定を上書きするには、以下のようになります。

- DTW_MBMODE 変数を、初期設定ファイルの DEFAULT ENVIRONMENT ステートメントの IN パラメーターとして、以下の例に示すように追加します。

```
ENVIRONMENT (DTW_DEFAULT) defcd11  
  (IN DTW_MBMODE, OUT RETURN_CODE )
```

- マクロ・ファイルで、変数 DTW_MBMODE を、アプリケーションに必要な値に設定する。

パス構成ステートメントのカスタマイズ

Net.Data は、Net.Data のマクロ・ファイルが使用するファイルと実行可能プログラムの位置を、パス構成ステートメントの設定から決定します。パス・ステートメントは以下の通りです。

- MACRO_PATH
- EXEC_PATH
- INCLUDE_PATH
- FFI_PATH

これらのパス・ステートメントは、マクロ・ファイル、実行可能ファイル、HFS ファイル、および組み込みファイルを配置しようとしたときに、Net.Data が検索する 1 つ以上のディレクトリーを識別します。必要とするパス・ステートメントは、マクロが使用する Net.Data の能力に依存します。

Net.Data のサンプルの初期設定ファイルは、Net.Data の検索パスの設定のカスタマイズについて、幾つかの前提事項を設けます。これらの前提事項は、使用環境によっては正しくないこともあり、次のようにパス構成ステートメントの変更を要求することもあります。

- Net.Data のマクロ・ディレクトリーのパスが、/usr/lpp/netdata/macros と異なる場合は、そのパスを、MACRO_PATH ステートメントのマクロ・ディレクトリーのパスに置き換える。

Net.Data の /usr/lpp/netdata/macros ディレクトリーに含まれるファイルは、SMP/E の制御下にあり、変更することはできません。これらのファイルのどれか 1 つでも変更した場合は、作成するディレクトリーに格納されるファイルのコピーに変更を加えてください。SMP/E が作成したディレクトリーの検索前に、Net.Data に、ユーザーの専用ディレクトリー内のこれらのファイルを検索するように指示しなければなりません。これを行うには、db2www.ini ファイルにおいて、SMP/E が作成したディレクトリーの前に、ユーザー専用のディレクトリーを追加します。たとえば、SMP/E のインストール中に提供されたマクロをカスタマイズし、そのマクロを、ディレクトリー /u/SYSADM/macros に配置し、デフォルトの MACRO_PATH ステートメントを、以下のステートメントに置き換えます。

```
MACRO_PATH /u/SYSADM/macros;/usr/lpp/netdata/macros;
```

- Net.Data の外部プログラムのディレクトリーのパスが、`/usr/lpp/netdata/testcmd` と異なる場合は、そのパスを、EXEC_PATH ステートメントのユーザーの外部プログラムのディレクトリーのパスに置き換える。
- Net.Data のフラット・ファイルのディレクトリーのパスが、`/usr/lpp/netdata/file-data` と異なる場合は、そのパスを、FFI_PATH ステートメントのユーザーのフラット・ファイルのディレクトリーのパスに置き換えます。

更新のガイドライン：

幾つかの一般的なガイドラインは、すべてのパス・ステートメントに適用されます。

- 指定された各ディレクトリーは、セミコロン (;) で区切られる。
- 各パス・ステートメントは、複数のパスを指定することができる。パスの検索は、指定された順で左から右に行われます。この複数パス能力により、ユーザーのファイルを複数のディレクトリーに編成することができます。たとえば、複数の Web アプリケーションをそれぞれ、それ自身のディレクトリーに配置することができます。

ヒント：Net.Data は、指定されたディレクトリーをすべて検索します。ただし、サブディレクトリーは検索しません。たとえば、Net.Data のマクロが以下のディレクトリーにあるとした場合、パス・ステートメントには、各サブディレクトリーを指定しなければなりません。

```
/usr/test/client
/usr/test/assoc
/usr/test/partner
```

MACRO_PATH ステートメントは、以下のようになります。

```
MACRO_PATH [=] /usr/test/client;usr/test/assoc;usr/test/partner
```

以下の節では、各パス・ステートメントの目的と構文を説明し、有効なパス・ステートメントの例を提供します。

MACRO_PATH

MACRO_PATH 構成ステートメントは、Net.Data が Net.Data のマクロ・ファイルを検索するディレクトリーを識別します。たとえば、以下の URL 指定は、パスおよびファイル名 `/WWW/macro/sqlm.d2w` を持つ Net.Data のマクロを要求します。

```
http://server/netdata-cgi/db2www/WWW/macro/sqlm.d2w/report
```

構文：

```
MACRO_PATH [=]
path1;path2;...;pathn
```

等号 (=) は、オプションで、大括弧で示されます。

Net.Data は、パス `/server/macro` を、MACRO_PATH 構成ステートメントのパスに、Net.Data がマクロ・ファイルを検出するまで、あるいはすべてのパスを検索するまで、左から右へ、追加していきます。Net.Data のマクロの起動に関する情報については、"29ページの『第4章 Net.Data を起動する』" を参照してください。

例：以下の例は、初期設定ファイルの MACRO PATH ステートメントと、Net.Data を起動する関連リンクを示しています。

Net.Data の初期設定ファイル：

```
MACRO_PATH = /u/SYSADM/macros;/usr/lpp/netdata/macros;
```

HTML リンク

```
<A HREF="http://server/netdata-cgi/db2www/query.d2w/input">Submit another query.</A>
```

Net.Data が、ディレクトリー */server/netdata-cgi* から実行され、ファイル *query.d2w* がディレクトリー */u/SYSADM/macros* で検出された場合は、完全修飾パスは、*/u/SYSADM/macros/query.d2w* になります。

EXEC_PATH

EXEC_PATH 構成ステートメントは、EXEC ステートメントにより起動される外部プログラムを、Net.Data が検索する 1 つ以上のディレクトリーを識別します。プログラムが検出されれば、外部プログラム名がパス指定に追加され、実行のために言語環境に渡される完全修飾ファイル名になります。

構文：

```
EXEC_PATH [=]  
path1;path2;...;pathn
```

例：以下の例は、初期設定ファイルの EXEC PATH ステートメントと、外部プログラムを起動するマクロ・ファイルの EXEC ステートメントを示しています。

Net.Data の初期設定ファイル：

```
EXEC_PATH = /u/SYSADM/prgms;/usr/lpp/netdata/prgms;
```

Net.Data のマクロ

```
%FUNCTION(DTW_REXX) myFunction() {  
    %EXEC{ myFunction.cmd %}  
%}
```

Net.Data が、ディレクトリー */server/netdata-cgi* から実行され、ファイル *myFunction.cmd* が、*/usr/lpp/netdata/prgms* ディレクトリーで検出される場合、プログラムの修飾名は、*/usr/lpp/netdata/prgms/myFunction.cmd* になります。

INCLUDE_PATH

INCLUDE_PATH 構成ステートメントは、Net.Data が検索する 1 つ以上のディレクトリーを識別し、Net.Data のマクロの INCLUDE ステートメントで指定されたファイルを検出します。ファイルを検出すると、Net.Data は、組み込みファイル名を、パス指定に追加し、修飾された組み込みファイル名を作成します。

構文：

```
INCLUDE_PATH [=]  
path1;path2;...;pathn
```

ヒント：ローカルな Web サーバーから HTML ファイルを組み込む場合、*Net.Data* 解説書の INCLUDE_URL のローカル Web サーバーの例に示されているように、

INCLCLUDE_URL 構成要素を使用します。例示された構文を使用することにより、INCLUDE_PATH を更新して、Web サーバーにとっては既知のディレクトリーを指定する必要がなくなります。

例 1: 以下の例は、初期設定ファイルの INCLUDE PATH ステートメントと、組み込みファイルを指定する INCLUDE ステートメントの両方を示しています。

Net.Data の初期設定ファイル：

```
INCLUDE_PATH = /u/SYSADM/includes;/usr/lpp/netdata/includes;
```

Net.Data のマクロ

```
%INCLUDE "myInclude.txt"
```

Net.Data が、ディレクトリー */server/netdata-cgi* から実行され、*myInclude.txt* が、*/u/SYSADM/includes* ディレクトリーで検出された場合、組み込みファイルの完全修飾名は、*/u/SYSADM/includes/myInclude.txt* になります。

例 2: 以下の例は、INCLUDE PATH ステートメントおよび サブディレクトリー名で完全修飾された INCLUDE ファイルを示しています。

Net.Data の初期設定ファイル：

```
INCLUDE_PATH = /u/SYSADM/includes;/usr/lpp/netdata/includes;
```

Net.Data のマクロ

```
%INCLUDE "/OE/oeheader.inc"
```

Net.Data が、ディレクトリー */server/netdata-cgi* から実行された場合、組み込みファイルが、ディレクトリー */u/SYSADM/includes/OE* および */usr/lpp/netdata/includes/OE* で検索されます。ファイルが、*/usr/lpp/netdata/includes/OE* で検出される場合、組み込みファイルの完全修飾名は、*/usr/lpp/netdata/includes/OE/oeheader.inc* になります。

FFI_PATH

FFI_PATH 構成ステートメントは、Net.Data がフラット・ファイル・インターフェース (FFI) 関数で参照される HFS ファイルを検索する 1 つ以上のディレクトリーを識別します。

構文：

```
FFI_PATH [=]  
path1;path2;...;pathn
```

例：以下の例は、初期設定ファイルの FFI PATH ステートメントを示しています。

Net.Data の初期設定ファイル：

```
FFI_PATH = /u/SYSADM/ffi;/usr/lpp/netdata/ffi;
```

FFI 言語環境が呼び出されると、Net.Data は、FFI_PATH ステートメントで指定されたパス内を調べます。

環境構成ステートメント

ENVIRONMENT ステートメントは、言語環境を構成します。言語環境とは、Net.Data の構成要素です。Net.Data は、この構成要素を使用して、DB2 データベースのようなデータ・ソースにアクセスしたり、あるいは、REXX のような言語で書かれたプログラムを実行します。Net.Data は、言語環境のセットと、ユーザー自身の言語環境の作成を可能にしてくれるインターフェースを提供してくれます。これらの言語環境および言語環境インターフェースは、*Net.Data* の言語環境解説書で説明されています。

Net.Data は、ENVIRONMENT ステートメントが、言語環境のための Net.Data の初期設定ファイルに存在することを要求します。それによって初めて、言語環境を起動することができるのです。

Net.Data は、Net.Data 言語環境が行う、FUNCTION ブロックで定義された関数の呼び出しの解釈方法に影響を与える幾つかの変数を指定します。これらの変数の設定が有効になるためには、言語環境に渡されなければなりません。

たとえば、マクロは、LOCATION 変数を定義して、DTW_SQL 関数内の SQL ステートメントを実行する、リモートの DBMS の場所名を指定することができます。LOCATION の値は、SQL の言語環境 (DTW_SQL) に渡されなければなりません。これにより、SQL の言語環境は、指定されたリモートの DBMS に接続することができます。変数を言語環境に渡すには、LOCATION 変数を、DTW_SQL の環境ステートメントのパラメーター・リストに追加しなければなりません。

また、構成変数として INI ファイル内に設定し、マクロ・ファイル内で上書きすることができる変数もあります。たとえば、SQL 言語環境が呼び出されたときに、マクロ・ファイルの DB2PLAN および DB2SSID 変数のデフォルトの設定をマクロに上書きさせたい場合、それらの変数を、DTW_SQL の ENVIRONMENT ステートメントに組み込みます。

Net.Data のサンプルの初期設定ファイルは、Net.Data の環境構成ステートメントの設定のカスタマイズについて、幾つかの前提事項を設けます。これらの前提事項は、使用する環境では正しくないこともあります。ステートメントを使用環境に合わせて適切に変更します。

ENVIRONMENT の追加と更新を行うには、以下のようにします。

ENVIRONMENT ステートメントは以下の構文を持っています。

```
ENVIRONMENT(type) library_name (parameter_list, ...)
```

パラメーター :

- *type*

Net.Data が、Net.Data のマクロで定義された FUNCTION ブロックと、この言語環境とを関連付けるための名前です。FUNCTION ブロックの定義で、言語環境のタイプを指定し、Net.Data が関数を実行するために使用しなければならない言語環境を識別しなければなりません。

- *library_name*

Net.Data が呼び出す言語環境インターフェースを含む DLL の名前。

- *parameter_list*

FUNCTION ブロック定義で指定されたパラメーターに加えて、関数呼び出しごとに言語環境に渡されるパラメーターのリスト。

これらのパラメーターは、FUNCTION ブロック定義で指定されたパラメーターに続けて、*dtw_lei* 構造の *parm_data_array* フィールドで渡されます。(これらの構造に関する情報については、“*Net.Data* の言語環境解説書” を参照してください。)

言語環境で処理される関数を実行する前に、これらのパラメーターを、ユーザーの *Net.Data* のマクロで、構成変数、あるいは変数として定義しなければなりません。関数が、その出力パラメーターのどれかを変更すると、パラメーターは、関数が完了後も、それらパラメーターの値を保持します。

Net.Data が INI ファイルを処理する場合、*Net.Data* は、言語環境の DLL をロードすることはありません。*Net.Data* が言語環境 DLL をロードするのは、それが最初に、言語環境を識別する関数を実行するときです。DLL は、*Net.Data* がロードされている限り、ロードされた状態を続けます。

例: *Net.Data* 提供の言語環境の ENVIRONMENT ステートメント

アプリケーションの ENVIRONMENT ステートメントをカスタマイズする場合、ユーザーの初期設定ファイルから言語環境に渡す必要のある変数、あるいは *Net.Data* のマクロ記述者が、自分のマクロを設定したり上書きしたりする必要のある変数を、ENVIRONMENT ステートメントに追加します。

```
ENVIRONMENT (DTW_SQL)      dtwsql      ( IN LOCATION, DB2SSID, DB2PLAN,
TRANSACTION_SCOPE)
ENVIRONMENT (DTW_DEFAULT)  defcd11    (IN DTW_MBMODE,OUT RETURN_CODE)
ENVIRONMENT (DTW_ODBC)     odbcd11    (IN LOCATION, TRANSACTION_SCOPE)
ENVIRONMENT (DTW_APPLET)   ap1ld11    (OUT RETURN_CODE)
ENVIRONMENT (DTW_PERL)     perl1d11   (OUT RETURN_CODE )
ENVIRONMENT (DTW_FILE)     filed11    (OUT RETURN_CODE)
ENVIRONMENT (DTW_REXX)     rexxd11    (OUT RETURN_CODE )
ENVIRONMENT (DTW_SYSTEM)   sysd11     (OUT RETURN_CODE )
```

各 ENVIRONMENT ステートメントは、途中で改行を入れず単一の行にしなければなりません。

DB2 との接続の管理

Net.Data のようなアプリケーション・プログラムは、DB2 により管理されるデータにアクセスしたり、DB2 のストアード・プロシージャを実行するには、DB2 for OS/390 に接続しなければなりません。、*Net.Data* は、DB2 製品の一部として提供される、リソース回復サービス接続機能 (RRSAF) を使用し、この目標達成します。DB2 のサブシステムへの接続を確立すると、大きなオーバーヘッドが発生するため、既存の接続の再利用が、ユーザーの要求ごとに新規に接続を生成し直す方法に代わる魅力的な代替案になります。

Net.Data は、それが ICAPI あるいは GWAPI との使用のために構成されるときに SQL 言語環境が使用する接続の再利用をサポートします。Web サーバーのスレッドが、DB2 へのアクセスを必要とする *Net.Data* のユーザー要求を処理する場合、DB2 接続が必要に応じて取得され、スレッドのコンテキストに取り込まれます。サーバーが、DB2 へのアクセスを必要とする連続要求をこのスレッドに割り当てる場合、スレッドは、既存のローカルの DB2 への接続がすでに存在していれば、それを再使用します。スレッドは、DB2 のプラン名およびユーザー ID を、必要に応じて変更し、要

求の要件を満足します。各スレッドと関連付けられている接続のプールは、サーバーが定常状態に達するまで、拡張し続けます。定常状態では、各スレッドは、DB2 のサブシステムごとに 1 つの接続を持ちます。

接続管理機能の利用には、*Net.Data* の構成は必要ありません。しかし、作業負荷管理プログラム (WLM) を使用して、*Net.Data* の要求を処理する Web サーバーのアドレス空間を管理したいのであれば、何らかの WLM 構成を追加する必要があります。

作業負荷管理の考慮事項

作業負荷管理プログラム (WLM) は、OS/390 オペレーティング・システムの構成要素で、ビジネスの目標に向かってのシステム性能の定義、実装、およびモニターのための機能を提供してくれます。WLM は、処理作業のリソースを割り当てます。そのために、ユーザーのアプリケーションのパフォーマンスおよびスケーラビリティが、確実にユーザーの要求を満たすように、ユーザーが定義する方針を使用します。

Net.Data を ICAPI あるいは GWAPI とともに使用するために構成する場合、IBM Internet Connection Server あるいは Lotus Domino Go Webserver いずれかにより、WLM を使用して、ユーザーの *Net.Data* 作業負荷を管理するための方針を確立します。ユーザーは、与えられたテンプレートに一致する URL 要求の処理のためのアプリケーション環境 および WLM トランザクション・クラスにより、これらの方針を確立することができます。

WLM を使用することにより得られる利点は、*Net.Data* の初期設定ファイル (db2www.ini) に変更を加え、WLM の REFRESH あるいは WLM の QUIESCE コマンドを使用することにより、Web サーバーを停止したり、再始動しなくても、その変更を有効にすることができる、ということです。

WLM の詳細については、"*OS/390 MVS 計画: ワークロード管理サービス V2, GC880-6582-04*" を参照してください。

WLM を用いた IBM Internet Connection Server および Lotus Domino Go Webserver の使用の詳細については、以下を参照してください。

- *ICSS OS/390:Webmaster の手引き V2R, GC88-7534-00*
- *Go Webserver for OS/390 Webmaster の手引き リリース 4.6.1, SD88-7826-00*

CGF と一緒に使用する *Net.Data* の構成

コモン・ゲートウェイ・インターフェース (CGI) は、*Net.Data* のようなアプリケーション・プログラムを、Web サーバーから起動することができるようにする業界標準のインターフェースです。*Net.Data* の CGI サポートにより、*Net.Data* をお気に入りの Web サーバーと一緒に使用することができます。

Net.Data のための階層ファイル・システム (HFS) のディレクトリーを作成したときに、ディレクトリーの構造や名前を変更していない限り、SMP/E のインストール・プロセスは、*Net.Data* の実行可能ファイルおよび DLL を、ディレクトリー /usr/lpp/netdata/cgi-bin にインストールしています。/usr/lpp/netdata は、ユ

ユーザーの Web サーバーのルート・ディレクトリーではないので、Web サーバーは、ユーザーが Web サーバーの構成に変更を加えていなければ、クライアントの Net.Data 要求を処理することはできません。

Web サーバーを変更するには、以下を行います。

1. Web サーバーを停止する。
2. 以下のアプローチのいずれかを使用して、実行可能ファイルと DLL のインストールを完了する。

- **Net.Data のディレクトリーの使用**

- a. Net.Data の要求を /usr/lpp/netdata/cgi-bin ディレクトリーにリダイレクトする Web サーバーの構成ファイル /etc/httpd.conf に、Exec ディレクティブを追加する。たとえば、以下のようになります。

```
Exec /netdata/cgi/* /usr/lpp/netdata/cgi-bin/*
```

- b. ユーザーの Net.Data の cgi-bin ディレクトリーを、Web サーバーの環境変数ファイル /etc/httpd.envvars の LIBPATH ステートメントに追加します。ユーザーの Net.Data の cgi-bin ディレクトリーが、/usr/lpp/netdata/cgi-bin の場合、LIBPATH ステートメントは、以下のステートメントと類似するはずです。

```
LIBPATH=/usr/lpp/internet/bin:/usr/lpp/netdata/cgi-bin
```

- **Web サーバーのディレクトリーの使用**

- 実行可能なファイルと DLL (ap1d11、db2www、defcd11、dtwle、dtwlei、dtwsq1、filed11、odbcd11、perl11、rex1d11、sysd11、dtwiod11) を、Web サーバーの cgi-bin ディレクトリーに移動する。Web サーバーのデフォルトの cgi-bin ディレクトリーは、/usr/lpp/internet/server_root/cgi-bin です。

Web サーバーのデフォルトのルート・ディレクトリーは、Web サーバーの構成ファイル /etc/httpd.conf の ServerRoot ディレクティブによって指定されますが、Web サーバーのインストール時に、変更された可能性もあります。Web サーバーのデフォルトの cgi-bin ディレクトリーは、Web サーバーの構成ファイルの Exec ディレクティブで指定されますが、Web サーバーのインストール時に、変更された可能性もあります。Web サーバーのルート・ディレクトリーが /usr/lpp/internet/server_root と異なっていたり、Web サーバーの cgi-bin ディレクトリーが、/usr/lpp/internet/server_root/cgi-bin と異なっている場合は、以下の指示では、適宜ユーザーの選択ディレクトリーに置き換えてください。

- Web サーバーの cgi-bin ディレクトリーを、Web サーバーの環境変数ファイル /etc/httpd.envvars の LIBPATH ステートメントに追加する。ユーザーの Web サーバーの cgi-bin ディレクトリーが、/usr/lpp/internet/server_root/cgi-bin の場合、ユーザーの LIBPATH ステートメントは、以下のステートメントと類似するはずです。

```
LIBPATH=/usr/lpp/internet/bin:/usr/lpp/internet/server_root/cgi-bin
```

3. Net.Data の実行可能ファイルと DLL、および実行可能なファイルと DLL へのパスの各ディレクトリーに対する許可番号が、確実に 755 になるようにする。
4. Web サーバーを再始動する。

制約事項： Web サーバーの環境変数ファイルの LIBPATH ステートメントでは、cgi-bin と icapi-lib ディレクトリーを、同時に指定しないでください。

Web サーバーのインストールと Web サーバーの構成ファイルのディレクティブの詳細については、以下の資料を参照してください。

- *ICSS for OS/390 概説およびインストール V2R2*, GC88-7949-00
- *ICSS OS/390:Webmaster の手引き V2R2*, GC88-7534-00
- *Go Webserver 概説およびインストール V4R6*, SD88-7825-00
- *Go Webserver for OS/390 Webmaster の手引き リリース 4.6.1*, SD88-7826-00

ICAPI あるいは GWAPI と一緒に使用するための Net.Data の構成

CGI ではなく、Web サーバーのアプリケーション・プログラミング・インターフェース (API) を使用することにより、Net.Data のパフォーマンスを大きく改善することができます。Net.Data は、IBM Internet Connection Server API (ICAPI) および Lotus Domino Go Webserver API (GWAPI) をサポートしています。

CGI を使用して実行に成功するマクロならどれでも、ICAPI あるいは GWAPI の実行に成功します。これらのマクロに対して変更を加える必要はありません。

Net.Data のための HFS ディレクトリーの作成時に、ディレクトリー構造や名前を変更していなければ、SMP/E のインストール・プロセスは、Net.Data の実行可能ファイルおよび DLL を、ディレクトリー /usr/lpp/netdata/icapi-lib にインストールしています。/usr/lpp/netdata は、ユーザーの Web サーバーのルート・ディレクトリーではないので、Web サーバーの構成に何らかの追加変更をしていない限り、Web サーバーは、Net.Data に対するクライアントの要求には応えることができません。

Web サーバーを変更するには、以下を行います。

1. Web サーバーを停止する。
2. 以下のアプローチのいずれかを使用して、実行可能ファイルと DLL のインストールを完了する。

- **Net.Data のディレクトリーの使用**

- a. Web サーバーの構成ファイル /etc/httpd.conf に、ServerInit ディレクティブを追加し、Web サーバーがその初期化ルーチンを実行するときに、Net.Data に固有の初期化を実行するよう、Web サーバーに指示する。想定される ServerInit ディレクティブを 1 つ挙げると、以下のようになります。

```
ServerInit    /usr/lpp/netdata/icapi-lib/db2www:dtw_init
```

- b. Web サーバーの構成ファイル/etc/httpd.conf に Service ディレクティブを追加し、/usr/lpp/netdata/icapi-lib ディレクトリーに Net.Data の要求をリダイレクトする。想定される Service ディレクティブを 1 つ挙げると、以下のようになります。

```
Service /netdata-cgi/db2www* /usr/lpp/netdata/icapi-lib/db2www:dtw_icapi*
```

- c. ユーザーの Net.Data の icapi-lib ディレクトリーを、Web サーバーの環境変数ファイル /etc/httpd.envvars の LIBPATH ステートメントに追加する。ユーザーの Net.Data の icapi-lib ディレクトリーが、/usr/lpp/netdata/icapi-lib の場合、LIBPATH ステートメントは、以下と類似するはずです。

LIBPATH=/usr/lpp/internet/bin:/usr/lpp/netdata/icapi-lib

• **Web サーバーのディレクトリーの使用**

- a. 実行可能なファイルと DLL (appldll、db2www、defcdll、dtwice、dtwle、dtwlei、dtwsql、filedll、odbcdll、perl.dll、rex.dll、sysdll、dtwiodll) を、Web サーバーの cgi-bin ディレクトリーに移動する。Web サーバーのデフォルトの cgi-bin ディレクトリーは、/usr/lpp/internet/server_root/cgi-bin です。

- b. Web サーバーの構成ファイル /etc/httpd.conf に、ServerInit ディレクティブを追加し、Web サーバーがその初期化ルーチンを実行するときに、Web サーバーに、Net.Data に固有の初期化を実行するよう指示する。想定される ServerInit ディレクティブを 1 つ挙げると、以下のようになります。

ServerInit /usr/lpp/internet/server_root/cgi-bin/db2www:dtw_init

- c. Web サーバーの構成ファイル /etc/httpd.conf に Service ディレクティブを追加し、Net.Data の要求を /usr/lpp/internet/server_root/cgi-bin ディレクトリーにリダイレクトする。想定される Service ディレクティブを 1 つ挙げると、以下のようになります。

Service /cgi-bin/db2www* /usr/lpp/internet/server_root/cgi-bin/db2www:dtw_icapi*

- d. Web サーバーの cgi-bin ディレクトリーを、Web サーバーの環境変数ファイル /etc/httpd.envvars の LIBPATH ステートメントに追加する。ユーザーの Web サーバーの cgi-bin ディレクトリーが、/usr/lpp/internet/server_root/cgi-bin の場合、ユーザーの LIBPATH ステートメントは、以下と類似するはずです。

LIBPATH=/usr/lpp/internet/bin:/usr/lpp/internet/server_root/cgi-bin

制約事項： Web サーバーの環境変数ファイルの LIBPATH ステートメントでは、cgi-bin と icapi-lib ディレクトリーを、同時に指定しないでください。

3. Net.Data の実行可能ファイルと DLL、および実行可能なファイルと DLL へのパスの各ディレクトリーに対する許可番号が、必ず 755 になるようにしてください。
4. Web サーバーを再始動する。

Web サーバーのインストールと Web サーバーの構成ファイル・ディレクティブの詳細については、以下の資料を参照してください。

- *ICSS for OS/390 概説およびインストール V2R2*, GC88-7949-00
- *ICSS OS/390:Webmaster の手引き V2R2*, GC88-7534-00
- *Go Webserver 概説およびインストール V4R6*, SD88-7825-00
- *Go Webserver for OS/390 Webmaster の手引き リリース 4.6.1*, SD88-7826-00

言語環境のセットアップ

Net.Data 言語環境の構成変数および ENVIRONMENT 構成ステートメントの変更後、REXX、SQL、および ODBC 言語環境が適切に機能するようになるには、幾つかのセットアップが必要になります。

REXX 言語環境のセットアップ

The REXX の言語環境 (DTW_REXX) は、DTWGENOU と呼ばれる MVS のロード・モジュールを使用します。このモジュールは、DTW.V2R1M0.SDTWLOAD 区分データ・セット内のメンバー DTWGENOU として、SMP/E のインストール・プロセス中に作成されています。Net.Data の REXX 言語環境は、DTWGENOU が、SYS1.LINKLIB などのシステム・ライブラリー、あるいは Web サーバーの開始プロシージャの STEPLIB DD ステートメントで指定される私用ライブラリーに常駐していることを要求します。Web サーバーの開始プロシージャの名前と位置は、使用しているシステムの構成に依存します。

SQL および ODBC の言語環境のセットアップ

SQL の言語環境 (DTW_SQL) および ODBC の言語環境 (DTW_ODBC) は、DB2 のロード・モジュール・ライブラリー SDSNLOAD を使用します。Net.Data の SQL および ODBC の言語環境は、このライブラリーが、LINKLIST に常駐すること、あるいは、このライブラリーが、Web サーバーの始動プロシージャの STEPLIB DD ステートメントで指定されていることを要求します。Web サーバーの開始プロシージャの名前と位置は、使用しているシステムの構成に依存します。

必要事項：Net.Data の SQL および ODBC の言語環境を使用する前に、Net.Data のプランを作成し、ストアード・プロシージャを呼び出すか、他のタイプの SQL ステートメントを実行します。このプランを作成するのに必要なバインドは、使用を計画している言語環境およびストアード・プロシージャの位置に依存します。

以下のアプローチの 1 つを使用して、Net.Data の DBRM をパッケージにバインドします。

- Net.Data の DBRM のパッケージへのバインド、SQL 言語環境の使用をサポートする Net.Data のプランの作成、およびそのプラン上での EXECUTE 権限の PUBLIC への授与、を行うためのサンプルの JCL は、DTW.V2R1M0.SDTWBASE(DTWBIND) にあります。
- SQL および ODBC 言語環境の両方を使用する計画なら、DB2 CLI 対応の DBRM を Net.Data の DBRM と同じプランにバインドする。Net.Data の DBRM および DB2 CLI の DBRM のパッケージへのバインド、SQL および ODBC の言語環境の使用をサポートする Net.Data のプランの作成、およびプラン上での EXECUTE 権限の PUBLIC への授与のためのサンプルの JCL は、DTW.V2R1M0.SDTWBASE(DTWOBIND) にあります。

使用している環境内で JCL の実行を成功させるためには、サンプルの JCL に幾つかの小さな変更を加える必要があるかもしれません。

- JOBLIB DD ステートメントで指定される SDSNEXIT および SDSNLOAD のデータ・セット名の接頭部は、使用している DB2 のバージョンに依存し、インストールには誤っている可能性があります。
- DSN コマンドの SYSTEM オプションおよび RUN コマンドの PLAN オプションで指定される値も、インストールには誤っている可能性があります。DSN コマンドの SYSTEM オプションは、DB2 のサブシステムの名前を指定し、db2www.ini ファイルの DB2SSID ステートメントで指定されるサブシステムの ID と同じでなければなりません。

- RUN コマンドの PLAN オプションは、DSNTIAD プログラムのためのアプリケーション・プランの名前を指定します。ストアード・プロシージャを使用する計画であるのならば、ストアード・プロシージャのためのパッケージを、Net.Data のプランにバインドする必要があるかもしれません。

適切な変更をすべて行い、JCL を処理依頼します。

メッセージ・カタログを使用可能にする

Net.Data for OS/390 は、英語、日本語、および韓国語のメッセージ・カタログを提供します。

Web サーバーの環境変数ファイル `/etc/httpd.envvars` 内の環境変数を設定して、メッセージ・カタログの使用を可能にし、Net.Data のメッセージの取得先の特定のメッセージ・カタログを指定します。

Net.Dataのための階層ファイル・システム (HFS) のディレクトリを作成したときに、そのディレクトリ構造や名前を変更していなければ、Net.Data の英語、日本語、および韓国語のメッセージ・カタログは、すでに、それぞれ `/usr/lpp/netdata/C/d2w.cat`、`/usr/lpp/netdata/Ja_JP/d2w.cat`、および `/usr/lpp/netdata/Ko_KR/d2w.cat` にインストールされています。

ディレクトリの構造あるいは名前を変更した場合、以下のステップでは、`/usr/lpp/netdata` を、ユーザーの選択ディレクトリに置き換えてください。

- Net.Data のメッセージ・カタログを使用可能にするには、`/usr/lpp/netdata/%L/%N` を、Web サーバーの環境変数ファイルの `NLSPATH` ステートメントに追加します。`NLSPATH` ステートメントは、以下と類似のものになるはずです。

```
NLSPATH=/usr/lib/nls/msg/%L/%N:/usr/lpp/internet/%L/%N:/usr/lpp/netdata/%L/%N
```

- Net.Data が使用する特定のカタログを選択するには、Web サーバーの環境変数ファイル `/etc/httpd.envvars` で、`LANG` ステートメントの値を指定します。ステートメントの構文は、次のようになります。

```
LANG = catalog
```

22ページの表 1 を使用して、`catalog` の正しい値を指定します。

表 1. `LANG` ステートメント値

	英語	日本語	韓国語
<code>LANG =</code>	<code>C</code>	<code>Ja_JP</code>	<code>Ko_KR</code>

Net.Data のファイルおよびデータ・セットへのアクセス権の指定

Net.Data を使用する前に、Net.Dataが実行されるユーザー ID が、必ず、Net.Data が使用するファイルおよびデータベースへの適切なアクセス権を持つようにする必要があります。この意味は、これらのファイルは、ユーザー ID が明示的なアクセス権を持つ MVS データ・セットあるいは HFS ファイル、およびディレクトリにない限り、ということなのです。

さらに具体的にいえば、Net.Data が実行するときのユーザー ID は、以下の許可を持っていなければなりません。

- DSNAOINI 構成変数で指定される DB2 CLI の初期設定ファイルの読み取り
- MVS のロード・モジュール DTWGENOU の実行
- Net.Dataの初期設定ファイル、db2www.ini の読み取り
- Net.Data の実行可能ファイルおよび DLL の実行、および実行可能ファイルおよび DLLへのパスにあるディレクトリーの検索
- 適切な Net.Data のマクロ・ファイルの実行、および MACRO_PATH パス構成ステートメントで識別される適切なディレクトリーの検索
- 適切なファイルの実行、および EXEC_PATH パス構成ステートメントで識別される適切なディレクトリーの検索
- 適切なファイルの読み取り、および INCLUDE_PATH パス構成ステートメントで識別される適切なディレクトリーの検索
- 適切なファイルの読み取りと書き込み、および FFI_PATH パス構成ステートメントで識別される適切なディレクトリーの検索
- /tmp HFS ディレクトリーのファイルの読み込み、書き込み、および実行

第3章 ユーザー資産を保護する

OS/390 環境のインターネット機密保護は、ファイアウォール・テクノロジー、オペレーティング・システム機能、Web サーバー機能、Net.Data メカニズム、およびデータ・ソースの一部であるアクセス制御メカニズムの組み合わせで提供されます。

ユーザーの資産には、機密保護の適切なレベルを決定する必要があります。本章では、ユーザー資産を保護するために使用できるメソッドを説明し、Web サイトの機密保護のプランをたてるために使用できるその他のリソースのリファレンスも提供します。

以下のセクションには、ユーザーの資産保護のガイドラインが含まれています。ここで解説する機密保護のメカニズムは、次の通りです。

- 25ページの『ファイアウォールを使用する』
- 26ページの『ネットワーク上のユーザーのデータを暗号化する』
- 26ページの『認証を使用する』
- 27ページの『許可を使用する』
- 27ページの『Net.Data のメカニズムを使用する』

ファイアウォールを使用する

ファイアウォール は、ハードウェア、ソフトウェア、および、ネットワーク環境のリソースへのアクセスを制限するように設計されたポリシーのコレクションです。

ファイアウォール

- 侵入または割り込みから、内部のネットワークを保護します。
- 内部ユーザーが持ち込むデータとプログラムから、内部のネットワークを保護します。
- 外部データへの内部ユーザーのアクセスを制限します。
- ファイアウォールが侵害された場合に起こる損傷を、限定的な部分に制限します。

Net.Data は、 OS/390 ファイアウォール・テクノロジー 、または OS/390環境で実行する同等のファイアウォール製品とともに、使用されます。

OS/390 ファイアウォール・テクノロジーは、いろいろなアーキテクチャーおよびストラテジーを実装するために使用するツールキットです。このキットには、以下のツールがあります:

- IP フィルター
- プロキシ・サーバー
- Socks サーバー
- 定義域名サービス (DNS)
- 仮想私設ネットワーク

保護方法に関するファイアウォールのインストールおよび構成方法について詳しくは、"OS/390 ファイアウォール技術解説書 V2R5、SD88-7094" を参照してください。

ネットワーク上のユーザーのデータを暗号化する

Secured Sockets Layer (SSL)をサポートする Web サーバーを使用しているときに、クライアント・システムと Web サーバー間で送信されるすべてのデータを暗号化できます。この機密保護の基準は、ログイン ID、パスワード、およびクライアント・システムから Web サーバーに HTML フォームで転送されるすべてのデータと、Web サーバーからクライアント・システムに送信されるすべてのデータの、暗号化をサポートしています。Internet Connection Secure Server、バージョン 2 リリース 2 またはそれ以降、および Lotus Domino Go Webserver、4.6.1 またはそれ以降のバージョンは、ともに SSL をサポートします。

認証を使用する

Web サーバーは、サーバーが処理するそれぞれの Net.Data 要求と、ユーザー ID を関連付けます。次に、その要求を処理しているプロセスまたはスレッドは、ユーザー ID が許可されているすべてのリソースにアクセスすることができます。

OS/390 環境では、次の 3 つのうちの 1 つの方法で Net.Data 要求を処理しているスレッドまたはプロセスと、ユーザー ID は関連付けられるようになります。

クライアントを基にした認証

ユーザーは、ローカル OS/390 ユーザー ID とパスワードをクライアント側で入力するようにプロンプト指示されます。次に Web サーバーはローカルの機密保護サブシステム (RACF など) を起動して、ユーザーを認証します。正常に認証された場合には、提供されたユーザー ID は要求と関連付けられます。特別な Web サーバー %%CLIENT%% アクセス制御ユーザー ID を使用することによって、このタイプの認証を使用可能にすることができます。

サーバーを基にした認証

Web サーバーのユーザー ID はそれぞれの要求と関連付けられているため、ユーザー ID またはパスワードを入力するようなユーザーへのプロンプト指示はされません。この選択は、Web サーバーのユーザー ID と通常関連付けされる権限レベルであるため、お勧めできません。特別な Web サーバー %%SERVER%% アクセス制御ユーザー ID を使用することによって、このタイプの認証を使用可能にすることができます。

代理の認証

いくつかの事前に定義されたコレクションのリソースにアクセスする権限を持つ代理ユーザー ID は、クライアント要求に関連づけられています。この認証タイプには、ユーザーのグループまたは要求のクラスに適切なアクセス権限をもつ代理ユーザー ID の作成が必要です。

Web サーバーがクライアント要求とユーザー ID を関連付けるために使用するアプローチは、Web サーバーの構成時に指定します。アクセス制御ユーザー ID、Web サーバーのインストール、および Web サーバーを構成するための Protect、Protection、DefProt、UserId 指示の使用について詳しくは、以下を参照してください。

- "ICSS for OS/390 概説およびインストール V2R2, GC88-7949-00"
- "ICSS OS/390:Webmaster の手引き V2R2, GC88-7534-00"
- "Go Webserver 概説およびインストール V4R6, SD88-7825-00"
- "Go Webserver for OS/390 Webmaster の手引き リリース 4.6.1, SD88-7826-00"

許可を使用する

DB2 および HFSのようなデータ・ソースは、独自に許可のメカニズムを備えていて、データ・ソースが管理する情報を保護しています。このようなメカニズムは、Net.Data 要求を実行しているプロセスやスレッドに関連付けられているユーザー ID が、26ページの『認証を使用する』で説明されているように正しく認証されていることを前提にしています。これらのデータ・ソースに対する既存のアクセス制御メカニズムは、次に、認証されたユーザー ID によって保持されている認証に基づいて、アクセスを許可または拒否します。

Net.Data のメカニズムを使用する

上記で説明したメソッドの他に、パス・ステートメントおよび隠蔽変数などの Net.Data 提供のメカニズムを使用することができます。

Net.Data は、パス構成ステートメントの設定から、Net.Data マクロ・ファイルが使用するファイルと実行可能プログラムのロケーションを判別します。これらのパス・ステートメントは、マクロ・ファイル、実行可能ファイル、組み込みファイル、およびその他の HFS ファイルを見つけようとするとき、Net.Data が検索する 1 つまたは複数のディレクトリーを示します。このパス・ステートメントにディレクトリーを選択して組み込むことによって、ブラウザーで明示的にユーザーがアクセスできるファイルを制御することができます。パス・ステートメントの詳細については "5ページの『第2章 Net.Data のインストールおよび構成』" を参照してください。

Web ブラウザーで HTML ソースを見ることがあるユーザーから、Net.Data マクロのいろいろな特性を隠すために、隠蔽変数を使用することができます。たとえば、データベースの内部構造を隠すことができます。詳しくは "50ページの『隠し変数』" を参照してください。

第4章 Net.Data を起動する

ユーザーは Net.Data を、Common Gateway Interface (CGI)、または Lotus Domino Go Webserver (GWAPI) などの Web サーバー API や Internet Connection Server (ICAPI) とともに使用するよう、構成することができます。Net.Data の構成を理解するには、"5ページの『第2章 Net.Data のインストールおよび構成』" を参照してください。

本章では、CGI と構成された Net.Data の起動について説明します。ICAPI または GWAPI と構成された Net.Data を起動する方法について理解するには、"73ページの『ICAPI または GWAPI を使用してパフォーマンスを向上させる』" を参照してください。

さらに Net.Data が、マクロ・ファイルやただ 1 つの SQL ステートメント、ストアド・プロシージャ、または関数を実行するようにするかどうかも指定できます。このようなタイプの呼び出しは、それぞれマクロ要求および直接要求と呼ばれます。

マクロ要求

Net.Data マクロ言語で書かれているマクロ・ファイルを指定することによって Net.Data を起動します。

直接要求

次のように指定して Net.Data を起動します:

- 言語環境の名前
- 関数の呼び出しに必要なすべてのパラメーター値と、SQL ステートメントまたは関数の名前。
- SQL ステートメントまたは関数の呼び出しに必要なフォーム・データ

30ページの図 3 は、マクロ要求と直接要求の違いを示しています。マクロ要求は通常、その要求の URL 内のマクロ・ファイルを指定します。フォーム・データを使用することもできます。直接要求では URL 内のマクロ・ファイルは指定されませんが、フォーム・データを引き続き使用できます。

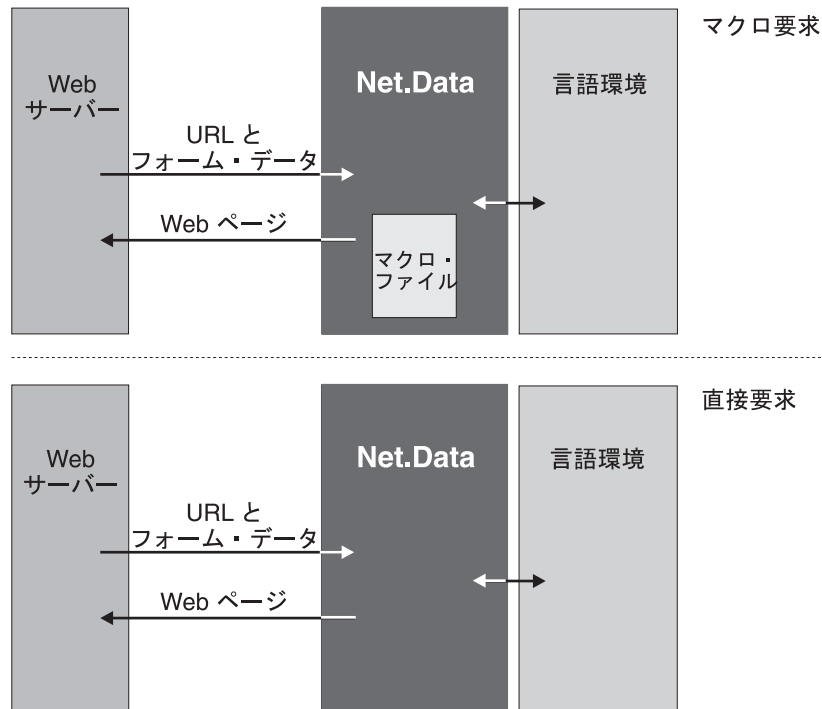


図3. マクロ要求対直接要求

Net.Data を起動する構文は、Net.Data の構成と、作成する要求のタイプによって異なります。マクロ要求と直接要求の場合は共に、Net.Data は HTML リンク、HTML フォーム、または URL によって Web ブラウザーから起動されます。Web サーバーは、CGI、ICAPI、または GWAPI を使用して Net.Data を起動します。

マクロ要求の場合、Net.Data マクロ・ファイルの名前、および Net.Data マクロ内で実行される HTML ブロックの名前は、リンク、フォーム、または URL 内で指定されます。

直接要求の場合、Net.Data 言語環境の名前、実行する SQL ステートメントまたは関数の名前、および追加の必須パラメーター値が、Net.Data によって定義された構文を使用して URL 内で指定されます。

本章では、両方の呼び出し方を説明します:

- 30ページの『マクロ・ファイルで Net.Data を呼び出す (マクロ要求)』
- 33ページの『マクロ・ファイルを使用しない Net.Data の起動 (直接要求)』

マクロ・ファイルで Net.Data を呼び出す (マクロ要求)

このセクションでは、マクロ・ファイルを指定した Net.Data の起動の方法を示します。マクロ要求スタイルによる起動に対しては、URL、HTML フォーム、または HTML リンクを使用して、Net.Data を呼び出すことができます。

次の例は、Net.Data の異なる起動の方法を示しています。この例では、すでに 17ページの『CGF と一緒に使用する Net.Data の構成』 および 19ページの『ICAPI あるい

は GWAPI と一緒に使用するための Net.Data の構成』で記述してあるように、Net.Data は Net.Data ディレクトリーを使用することを前提としています。

- HTML リンク:

```
<A HREF="http://server/netdata-cgi/db2www/filename.ext/block/
[?name=val&...]">any text</A>
```

- HTML フォーム:

```
<FORM METHOD=method
ACTION="http://server/netdata-cgi/db2www/
filename.ext/block/[?name=val&...]">any text</FORM>
```

- URL:

```
http://server/netdata-cgi/db2www/filename.ext/block/[?name=val&...]
```

パラメーター:

server Web サーバーの名前を指定します。サーバーがローカル・サーバーであれば、このサーバー名を省略し、相対 URL を使用することができます。

filename.ext Net.Data マクロ・ファイルの名前および拡張子を指定します。

block 参照される Net.Data マクロ・ファイルの中の HTML ブロックの名前を指定します。

method フォームで使用される HTML メソッドを指定します。METHOD=POST を推奨します。

?name=val&...

Net.Data に渡される 1 つまたは複数のオプション・パラメーターを指定します。

HTML リンク

マクロ・ファイルで HTML リンク・タグ `<a>` を使用することによって、HTML ブロックを実行する Web ページの中のリンクを作成します。マクロおよび HTML ブロックを指定する HREF 属性を使用することによって、さらにいくつかのテキストまたはリンク・タグ内のイメージまでも含むことによって、ユーザーはこのリンクを作成することができます。このメソッドは、Web ページがブラウザに表示される時に、『hot spot』としてテキストまたはイメージを識別します。ブラウザでユーザーがテキストやイメージをクリックする場合には、Net.Data はそのマクロ内の HTML ブロックを実行します。

次の例は、ユーザーが Web ページ上でテキスト「モニターをすべてリストする (List all monitors)」を選択したときに実行される SQL 照会とのリンクを示しています。

```
<a href="http://server/netdata-cgi/db2www/listA.d2w/report">
「モニターをすべてリストする (List all monitors)」</a>
```

このリンクは、以下のマクロを呼び出します。

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

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

この照会は、EQPTABLE 表の中に記述された各モニターに関する型式番号、コスト、および記述情報を持つ表を戻します。この例は、照会の結果をデフォルトのレポートで表示します。REPORT ブロックを使用してレポートをカスタマイズする方法に関しては、"63ページの『レポート・ブロック』" を参照してください。

一般的に、Net.Data の各ブロックは、%block_name{ で始まり、%} で終わります。Net.Data マクロ言語のその他詳細な構文に関しては、"Net.Data 解説書" を参照してください。

HTML フォーム

HTML フォームを使用して、Net.Data マクロの実行を動的にカスタマイズすることができます。フォームによって値を入力することが可能になり、マクロの実行と Net.Data が生成する Web ページの内容を変更することができます。

次の例は、31ページの『HTML リンク』でのモニター・リストの例と同様のものですが、ユーザーはブラウザで、簡単な HTML フォームを使用して製品タイプを選択し、その情報を表示することができます。

```
<H1>Hardware Query Form</H1>
<HR>
<FORM METHOD=POST ACTION="/netdata-cgi/db2www/equip1st.d2w/report">
<P>What type of hardware do you want to see?
<MENU>
<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="MON" checked> Monitors
<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="PNT"> Pointing devices
<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="PRT"> Printers
<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="SCN"> Scanners
</MENU>

<INPUT TYPE="SUBMIT" VALUE="Submit">
</FORM>
```

ブラウザで選択し、「実行 (Submit)」ボタンをクリックすると、Web ブラウザーは FORM タグの ACTION パラメーターを処理し、Net.Data が起動します。次に Net.Data は、equip1st.d2w マクロの中の HTML レポート・ブロックを実行します。

```
%FUNCTION(DTW_SQL) myQuery(){
SELECT MODNO, COST, DESCRIP FROM EQPTABLE
WHERE TYPE='${hardware}'
%REPORT{
<H3>Here is the list you requested</H3>
%ROW{
<HR>
$(N1): $(V1), $(N2): $(V2)
<P>$(N3): $(V3)
%}
%}
%}

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

1

—

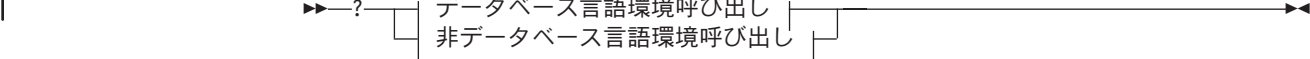
1
2
3
4
5
6

1
2
3
4
5
6
7
8

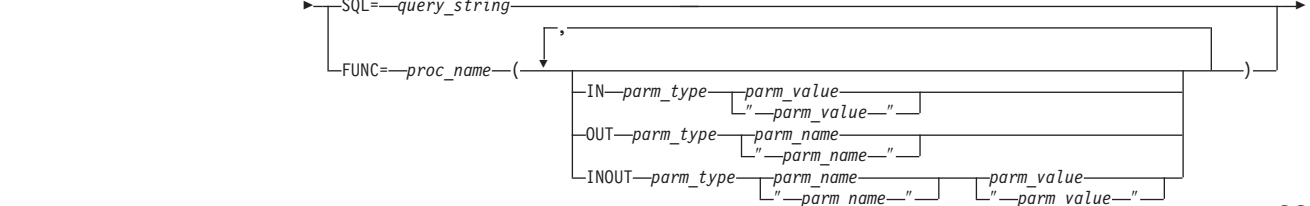
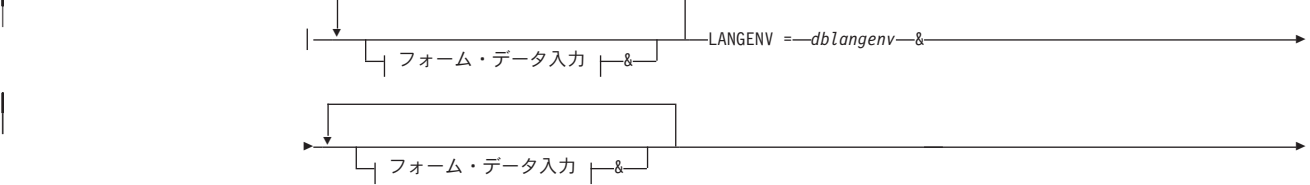
1

11

1

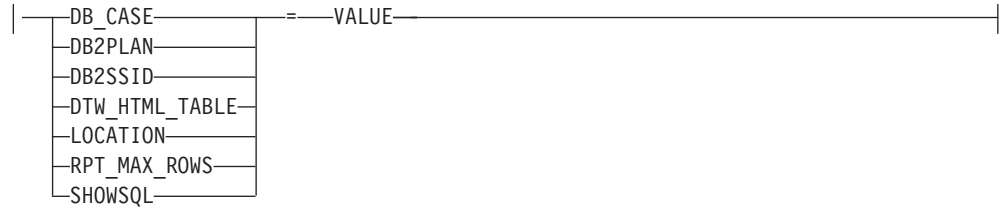


1

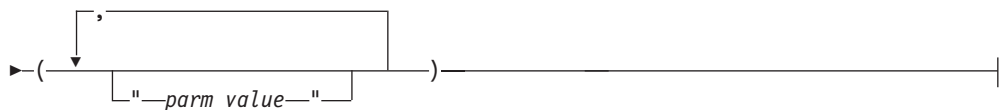
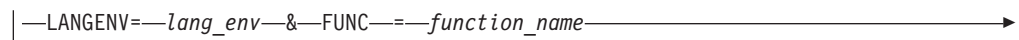




フォーム・データ入力



非データベース言語環境呼び出し



パラメーター:

データベース言語環境呼び出し

データベース言語環境を起動する Net.Data への直接要求を指定します。

フォーム・データ入力

オプション・パラメーター。SQL 変数の設定を指定する、または単純な HTML フォーマットを要求することができます。

DB2CASE

SQL ステートメントの文字ケース (大文字または小文字) を指定します。

DB2PLAN

ローカルの DB2 サブシステムをアクセスするときに使用する DB2 プランを指定します。

DB2SSID

ローカルの DB2 サブシステムをアクセスするときに使用する DB2 サブシステム ID を指定します。

DTW_HTML_TABLE

Net.Data が HTML テーブルを戻すべきかどうかを指定します。

LOCATION

ローカルの DB2 サブシステムが SQL 要求を渡すリモート・サーバーの名前を指定します。

RPT_MAX_ROWS

関数がレポートに戻すテーブルの最大行数を指定します。

SHOWSQL

Net.Data が実行される SQL ステートメントを隠すか表示するかを指定します。

VALUE

Net.Data 変数の値を指定します。

LANGENV

SQL ステートメントまたはストアド・プロシージャ呼び出しの、ターゲット言語環境を指定します。

dblangenv

データベース言語環境の名前:

- DTW_SQL
- DTW_ODBC

SQL

直接要求がインライン SQL ステートメントの実行を指定していることを示しています。

照会ストリング

動的 SQL を使用して実行されるすべての有効な SQL ステートメントを含むストリングを指定します。

FUNC

直接要求が、ストアド・プロシージャの実行を指定していることを示しています。

proc_name

有効な DB2 ストアド・プロシージャを指定します。

parm_type

DB2 ストアド・プロシージャに対する有効なパラメーター・タイプを指定します。

parm_name

有効なパラメーター名を指定します。

parm_value

DB2 ストアド・プロシージャに対する有効なパラメーターの値を指定します。

IN Net.Data がパラメーターを使用して入力データを言語環境に渡すことを指定します。

INOUT

Net.Data がパラメーターを使用して、入力データを言語環境に渡し、さらに言語環境からの出力データを戻すことを指定します。

OUT

言語環境がパラメーターを使用して、言語環境からの出力データを戻すことを指定します。

非データベース言語環境呼び出し

非データベース言語環境を起動する Net.Data への直接要求を指定します。

LANGENV

その関数を実行するターゲット言語環境を指定します。

lang_env

以下の非データベース言語環境の名前を指定します。

- DTW_PERL
- DTW_REXX
- DTW_SYSTEM

FUNC

直接要求が関数の実行を指定していることを示しています。

function_name

実行される関数を含んでいるファイルを指定します。

parm_value

その関数に有効なパラメーターの値を指定します。

直接要求の例

次の例は、直接要求メソッドを使用する場合に、Net.Data を起動するいろいろな方法を示しています。この例では、すでに 17ページの『CGF と一緒に使用する Net.Data の構成』 および 19ページの『ICAPI あるいは GWAPI と一緒に使用するための Net.Data の構成』 で記述してあるように、Net.Data は Net.Data ディレクトリーを使用して構成されていることを前提にしています。

HTML リンク

例 1: Perl 言語環境を起動して、Net.Data 初期設定ファイルの EXEC パス・ステートメントの Perl スクリプトを呼び出すリンク

```
<A HREF="http://server/netdata-cgi/db2www/?LANGENV=DTW_PERL&FUNC=my_perl(hi)">
すべてのテキスト</A>
```

例 2: 直前の例のように、Perl 言語環境を起動しますが、二重引用符およびスペース文字に対して、URL 形式で符号化された値でストリングを渡します。

```
<A HREF="http://server/netdata-cgi/db2www/?LANGENV=DTW_PERL&FUNC=my_perl(%22Hello+World%22)">
すべてのテキスト</A>
```

ヒント: URL では、スペースや二重引用符などの特定の文字は符号化しなければなりません。この例では、パラメーター値の中の二重引用符やスペースは、%22 および + 文字に符号化します。URL で符号化しなければならないすべての文字のリストに関しては、"Net.Data 解説書" の DTW_URLESCSEQ 関数の記述を参照してください。

HTML フォーム

例 1: SQL 言語環境を使用する SQL 照会を実行する HTML フォーム。

```
<FORM METHOD="POST"
  ACTION="http://server/netdata-cgi/db2www/>
<INPUT TYPE=hidden NAME=LANGENV VALUE=DTW_SQL>
<INPUT TYPE=hidden NAME=SQL VALUE="select * from Table1 where col1=$(InputName)">
Enter Customer name:
<INPUT TYPE=text NAME=InputName VALUE="John">
<INPUT TYPE=SUBMIT>
</FORM>
```

ヒント: Net.Data マクロで作成される HTML フォームを使用する直接要求呼び出しに、いろいろな変数置換を使用することができます。

URL

例 1: SQL 言語環境を使用する SQL 照会を実行する URL

`http://server/netdata-cgi/db2www/?LANGENV=DTW_SQL&SQL="select+++from+customer"`

制約事項: URL を使用する直接要求呼び出しでは、変数置換は使用できません。

例 2: Perl 言語環境を起動して、Net.Data 初期設定ファイルの EXEC パス・ステートメントにない実行可能ファイルを呼び出す URL

`http://server/netdata-cgi/db2www/?LANGENV=DTW_PERL&FUNC=/u/MYDIR/macros/myexec.pl`

ヒント: EXEC パス構成ステートメントで指定されていないパスでファイル名を参照するには、*function_name* の値としてファイル名と完全な修飾パスを指定してください。

例 3: システム言語環境を起動し、外部 Perl スクリプトを呼び出す URL

`http://server/netdata-cgi/db2www/?LANGENV=DTW_SYSTEM&FUNC=perl+/u/MYDIR/macros/myexec.pl`

例 4: REXX 言語環境を起動し、プログラムにパラメーターを渡す URL

`http://server/netdata-cgi/db2www/?LANGENV=DTW_REXX&FUNC=myexec.cmd(parm1,parm2)`

例 5: ストアード・プロシージャを呼び出し、SQL 言語環境にパラメーターを渡す URL

`http://server/netdata-cgi/db2www/?LANGENV=DTW_SQL&FUNC=MY_STORED_PROC(IN+CHAR(30)+Salaries)`

第5章 Net.Data のマクロ開発

Net.Data のマクロは、Net.Data のマクロ言語の連続した構成要素で構成されるテキスト・ファイルで、以下を行います。

- 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・ファイルで定義された関数、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
- 処理出力の HTML 形式へのフォーマットし、それを Web ブラウザーに戻す

Net.Data のマクロには、39ページの図 4 に示されているように、宣言部分および HTML 部分 という 2 つの部分 が含まれます。

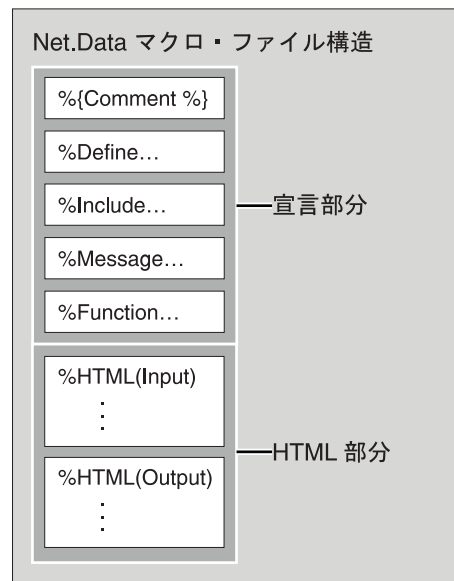


図4. マクロ・ファイル構造

- 宣言部分 には、マクロ・ファイルにおける変数および関数の定義が含まれる。
- HTML 部分 には、Web ページのレイアウトを指定するHTML のステートメントで構成される HTML のブロックが含まれる。

これらの部分は、複数回、任意の順序で使用することができます。マクロ・ファイルの部分および構成要素の構文については、" Net.Data 解説書" を参照してください。

本章では、Net.Data のマクロ・ファイルを構成しているさまざまなブロックと、マクロ・ファイルを記述するために使用できる方法について考察します。

- 40ページの『Net.Data のマクロ・ファイルの分析』
- 44ページの『Net.Data のマクロ変数』
- 55ページの『Net.Data の関数』
- 62ページの『マクロに HTML を作成』
- 65ページの『マクロ・ファイルにおける条件付き論理とループ』

Net.Data のマクロ・ファイルの分析

マクロ・ファイルは、次の 2 つのセクションで構成されています。

- 宣言部分。これは、HTML 部分で使用される定義を含みます。宣言部分は、以下の 2 つの主要なブロックを使用します。

- 41ページの『DEFINE ブロック』

- 42ページの『FUNCTION 定義ブロック』

これらのブロックには、言語構成要素のブロックとそれに含まれるステートメント、たとえば、EXEC ステートメント、IF ブロック、INCLUDE ステートメント、および MESSAGE ブロック、などが含まれます。言語構成要素についての詳細は、*Net.Data 解説書* の言語構成要素に関する章を参照してください。

- Web ページのレイアウトを定義し、参照変数と関数を参照する HTML 部分。HTML 部分は、入出力に使用される HTML ブロックを含みます。HTML ブロックは、43ページの『HTML ブロック』で説明されています。

本節では、簡単な Net.Data のマクロを使って、マクロ言語の要素を例示します。このマクロの例は、REXX プログラムに渡す情報をプロンプト指示するフォームを提供します。このマクロは、この情報を、OMPSAMP.CMD と呼ばれる REXX のプログラムに渡します。このプログラムは、ユーザーが入力したデータを、そのまま返します。次に、この結果が、HTML の 2 ページに表示されます。

まず最初に、マクロ全体を見渡し、次に各ブロックを詳細に見てみましょう。

```
%{ ***** 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">

    </form>

    < hr>
    <p>[<a href="/">Home page</a>]
    </body></html>
```

```
%}

%{ *****          HTML Block: Output          *****%}
%HTML (OUTPUT) {
  <html>
  <head>
  <title>$(page_title)</title>
  </head><body>
  <h1>Output Page</h1>
  <p>@rex1(input_data)
  <p><hr>
  <p>[<a href="/">Home page</a> |
  <a href="input">Previous page</a>]
  </body></html>
}%
```

サンプルのマクロは、DEFINE、FUNCTION、および 2 つの HTML ブロック、という 4 つの主要なブロックで構成されます。1 つの Net.Data のマクロに、複数の DEFINE、FUNCTION、および HTML ブロックを持つことができます。

2 つの HTML ブロックは、おなじみの HTML のタグを含みます。このため、Web のマクロを書くのが簡単になります。HTML について詳しくれば、マクロの作成は、単に、サーバーで動的に処理されるマクロ・ステートメントと、データベースに送信する SQL ステートメントの追加を行うだけになります。

マクロは HTML 文書に類似しているように見えますが、Web サーバーは、CGI あるいは Web サーバーの API を介してマクロにアクセスします。Net.Data には、処理すべきマクロの名前、およびそのマクロの表示すべき HTML ブロック、という 2 つのパラメーターが必要です。

マクロ・ファイルが呼び出されると、Net.Data はそのマクロ・ファイルを最初から処理していきます。以下の節では、Net.Data がマクロ・ファイルを処理していくとどうなるかを見てみます。

DEFINE ブロック

DEFINE ブロックには、後で HTML のブロックで使用する DEFINE 言語構成要素および変数の定義が含まれます。以下の例は、1 つの変数定義を持つ DEFINE ブロックを示しています。

```
%{ *****          DEFINE Block          *****%}
%DEFINE {
  page_title="Net.Data macro Template"
}%
```

1 行目はコメントです。コメントは、%{ と %} で囲まれた任意のテキストです。コメントは、マクロ・ファイルの任意の場所に置くことができます。その次のステートメントが、DEFINE ブロックの始まりです。1 つの定義ブロックに、複数の変数を定義することができます。この例では、page_title という 1 つの変数だけが定義されています。変数の定義を行うと、\$(page_title) という構文を使用して、この変数をマクロの任意の場所で参照することができます。変数を使用すると、後でマクロを全体にわたって変更することが容易にできるようになります。このブロックの最後の行 %} は、DEFINE ブロックの終了を識別します。

FUNCTION 定義ブロック

次のブロックは、FUNCTION 定義ブロックです。これには、HTML ブロックで呼び出される関数の関数定義が含まれます。関数は、言語環境によって処理され、プログラム、SQL 照会、あるいは ストアド・プロシージャを実行することができます。

以下の例は、外部の REXX プログラムの関数呼び出し、およびマクロ・ファイルに含まれる関数の関数呼び出しを定義している FUNCTION 定義ブロックを示しています。

```
%{ ***** FUNCTION Definition Block *****%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result) { <-- This function accepts
                                                         one parameter and returns a
                                                         result which is substituted
                                                         for the associated function
                                                         call
                                                         %EXEC{ompsamp.cmd %} <-- The function executes an external REXX program
                                                         called "ompsamp.cmd"
                                                         %}

%FUNCTION(DTW_REXX) today () RETURNS(result) {
    result = date() <-- The single source statement for this function is
                      contained inline.
%}
```

この FUNCTION 定義ブロックには、2 つの関数宣言が含まれています。最初の rexx1 は、REXX 関数の宣言です。この宣言は、次に ompsamp.cmd と呼ばれる、外部の REXX プログラムを実行します。1 つの入力変数 input は、この関数によって受け取られ、自動的に外部の REXX コマンドに渡されます。REXX コマンドはまた、result と呼ばれる 1 つの変数を戻します。REXX コマンドにおける result 変数の内容は、Output ブロックに含まれる @rexx1() 関数呼び出しの起動に置き換わります。ompsamp.cmd のソース・コードに示されるように、REXX のプログラムにより、変数 input および result に直接アクセスすることができます。

```
/* REXX */
result = 'The REXX program received "'input'" from the macro.'
```

この関数のコードは、その関数に渡されたデータをそのまま返します。この結果のテキストは自分の好きなようにフォーマットすることができます。それには、@rexx1() 関数呼び出し要求を、通常の HTML スタイルのタグでくくります (たとえば、 あるいは のように)。result 変数を使用しなくても、REXX のプログラムは、REXX の SAY ステートメントを使用すれば、HTML のタグを標準出力に書き込むことができたはずです。

2 番目の関数宣言 today も、REXX のプログラムを参照しています。しかし、この場合の REXX のプログラム全体 (1 行全部) は、関数宣言それ自身に含まれています。外部プログラムは必要ありません。インライン・プログラムは、REXX および Perl の関数では許可されています。その理由は、これらのプログラムは、動的な解析および実行ができるインタープリター言語だからです。インライン・プログラムは、管理すべき独立したプログラムの必要がないので、単純であるという優位点を持っています。最初の REXX の関数は、インラインとしてハンドルすることも可能です。

HTML ブロック

マクロの最後の 2 つのブロックは、入出力ブロックとして使用される HTML ブロックです。このブロックには、好きな名前を付けることができます。たとえば、『Report ブロック』は、Net.Data 文書の多くの例で使用されています。

最初の例の HTML ブロックは、HTML 入力ブロックです。このブロックには、1 つの入力フィールドを持つ簡単なフォームの HTML が含まれます。

```
%{ ***** HTML Block: Input *****%}
%HTML (INPUT) { <--- Identifies the name of this HTML block.
  <html>
  <head>
  <title>$(page_title)</title> <--- Note the variable substitution.
  </head><body>
  <h1>Input Form</h1>
  Today is @today() <--- This line contains a call to a function.

  <FORM METHOD="post" ACTION="output"> <--- When this form is submitted,
                                          the "output" HTML block is called.
  Type some data to pass to a REXX program:
  <INPUT NAME="input_data" <--- "input_data" is defined when the form
  TYPE="text" SIZE="30"> is submitted and can be referenced elsewhere in
                                          this macro. It is initialized to whatever the
                                          user types into the input field.

  <p>
  <INPUT TYPE="submit" VALUE="Enter">

  < hr>
  <p>
  [
  <a href="/">Home page</a>]
  </body><html>
  %} <--- Closes the HTML block.
```

ブロック全体は、HTMLのブロック識別子、%HTML (INPUT) {...%} で囲まれます。INPUT は、このブロックの名前を識別します。これに名前を付けることができます。HTML の <title>タグは、変数置換の例を含んでいます。変数 page_title の値が、フォームの表題に取って代わります。

このブロックはまた、関数呼び出しを持っています。式 @today() は、関数 today への呼び出しです。この関数は、後で説明する FUNCTION ブロックで定義されます。Net.Data は、today 関数の結果、すなわち現在日付を、@today() 式が配置されているのと同じ場所に挿入します。

FORM ステートメントの ACTION パラメーターは、HTML ブロック間あるいはマクロ間のナビゲーションの例を提供してくれています。ACTION パラメーターの別のブロックの名前を指定すると、フォームが処理依頼されたときに、そのブロックにアクセスします。HTML フォームからの入力データはどれも、暗黙の変数としてブロックに渡されます。これは、このフォーム上で定義される単一の入力フィールドにもあてはまります。フォームが処理依頼されると、このフィールドに入力されたデータは、変数 input_data に入れられ、HTML 出力ブロックに渡されます。

マクロ・ファイルが同じ Web サーバー上にある場合、相対参照を持つ他のマクロ・ファイルの HTML ブロックにアクセスすることができます。たとえば、ACTION パラメーター ACTION="../../othermacro.d2w/main" は、マクロ・ファイル othermacro.d2w

の main と呼ばれる HTML ブロックにアクセスすることができます。フォームに入力されたデータはどれも、もう一度変数 *input_data* に入れられて、このマクロに渡されます。

Net.Data を起動する場合、変数を URL の一部として渡します。たとえば、以下のようになります。

```
<a href="/netdata-cgi/db2www/othermacro.d2w/main?input_data=value">Next macro</a>
```

ほとんどの CGI プログラムの場合と同じように、入力データを受け取るために、環境変数を定義する必要はありません。Net.Data は、ユーザーに代わって環境変数をハンドルしてくれます。ユーザーに要求されるのは、単に変数の名前を参照することだけです。

この例における次の HTML ブロックは、Output ブロックです。これには、HTML のタグ付け、および Input ブロックの要求により処理された出力を定義する Net.Data のマクロ・ステートメントが含まれます。

```
%{ ***** HTML Block: Output *****%}
%HTML (OUTPUT) {
  <html>
  <head>
  <title>$(page_title)</title> <--- More substitution.

  </head><body>
  <h1>Output Page</h1>
  <p>@rex1(input_data) <--- This line contains a call to function rex1
                                passing the argument "input_data".

  <p>
  < hr>
  <p>
  [
  <a href="/">Home page</a> |
  <a href="input">Previous page</a>]
  %}
```

Input ブロックと同じように、このブロックも、変数および関数呼び出しを置換するための Net.Data のマクロ・ステートメントを持つ標準の HTML です。再度、*page_title* 変数はこの title ステートメントに置換されます。そして、前と同じように、このブロックは関数呼び出しを含みます。この場合、このブロックは、関数 *rex1* を呼び出し、変数 *input_data* の内容をこの関数に渡します。この変数は、このブロックが Input ブロックで定義されたフォームから受け取ったものです。任意の数の変数を関数に渡したり、関数から受け取ったりすることができます。関数定義は、渡される変数の数と型を決定します。

Net.Data のマクロ変数

Net.Data により、Net.Data でマクロの変数を定義したり、参照することができます。これらの変数を、マクロから言語環境に渡したり、また言語環境から受け取ったりすることができます。変数名と値、および文字列リテラルなどの Net.Data のトークンは、256 KB までのデータを含むことができます。

マクロ変数は、以下のタイプに分類されます。

- **DEFINE** ブロックで **DEFINE** ステートメントを使用して、明示的に定義された変数。

- 定義済みの変数。これは、Net.Data により使用可能になり、ある値が設定されます。
- 暗黙的に定義された変数。これには、以下の 4 つのタイプがあります。
 - 明示的には定義されていないが、最初の参照時にインスタンス化される変数。
 - FUNCTION ブロック定義の一部で、FUNCTION ブロック内でしか参照できないパラメーター変数。
 - Net.Data によりインスタンス化され、フォーム・データあるいは URL のデータの名前と値のペアに対応する変数。
 - Net.Data の表と関連付けられ、ROW ブロックあるいは REPORT ブロック内でしか参照することができない変数。

識別子は、変数あるいは関数呼び出しで、可視 になります。すなわち、識別子が宣言されたり初期化されたときに参照することができる、という意味です。識別子が可視になっている領域は、その効力範囲 と呼ばれます。効力範囲には、次の 5 つのタイプがあります。

- グローバル

識別子は、マクロ・ファイル内の任意の場所でそれを参照できる場合、グローバルな効力範囲を持ちます。グローバルな効力範囲を持つ識別子には、以下のようなものがあります。

- Net.Data の組み込み関数
- フォームのデータ
- URL データ
- HTML ブロック内からインスタンス化された変数

- マクロ・ファイル

識別子は、その宣言がブロックの外に現れる場合、グローバルな効力範囲を持ちます。ブロックは、左大括弧 ({) で始まり、パーセント記号と大括弧 (% }) で終わります。(DEFINE ブロックは、この定義から除外され、独立した DEFINE ステートメントとして扱わなければなりません。) マクロ・ファイルの効力範囲を持つ識別子は、それが宣言された点から、マクロ・ファイルの終わりまでが可視になります。

- FUNCTION ブロック

識別子は、関数定義のパラメーター・リスト内で宣言された場合、関数ブロックの効力範囲を持ちます。同じ名前を持つ識別子が、関数定義の外に存在する場合は、Net.Data は、その関数ブロック内の関数パラメーター・リストの識別子を使用します。Net.Data は、関数ブロックの外部からの識別子あるいはその値を使用したり、変更したりすることはありません。

識別子は、それが関数の外で宣言されたり、初期化されている場合、および関数のパラメーター・リストで宣言されていもない場合は、ローカルな効力範囲を持ちません。識別子が、関数ブロックの内部で使用される場合、関数呼び出しの前に割り当てられていた値を保持します。関数ブロック内で更新された場合、識別子は、関数呼び出しの後では新規の値を保持します。

識別子は、それがパラメーター・リストで宣言されておらず、関数呼び出しの前に宣言されたり初期化されたりしていない場合は、関数のブロック・レベルの効力範囲を持ちます。そのような識別子は、関数ブロックの外では参照することができません。

- REPORT ブロック

識別子は、それが REPORT ブロック内からだけしか参照できない場合は、レポート・ブロックの効力範囲を持ちます (たとえば、表の列の名前 N1、N2、...、Nn)。Net.Data が、その表処理の一部として暗黙的に定義する変数だけが、レポート・ブロックの効力範囲を持つことができます。インスタンス化されるそれ以外の変数はどれも、関数ブロックの効力範囲を持ちます。

- ROW ブロック

識別子は、それが ROW ブロック内からしか参照できない場合、行ブロックの効力範囲を持ちます (たとえば、表値の名前 V1、V2、...、Vn)。Net.Data が、その表処理の一部として暗黙的に定義する変数だけが、行ブロックの効力範囲を持つことができます。インスタンス化されるそれ以外の変数はどれも、関数ブロックの効力範囲を持ちます。

識別子が参照されると、識別子は、その識別子の値で置き換えられます変数への参照が、それに関連付けられている値を持たない場合、あるいは関数呼び出しが戻り値を持たない場合、その参照は空ストリングで置き換えられます。

以下の節では、変数の定義およびその参照方法について説明します。また、さまざまな変数の型とその使用方法についても説明します。

変数の定義

Net.Data のマクロにおける変数の定義方法には、以下の 3 つの方法があります。

- **DEFINE** ステートメントまたはブロック

Net.Data のマクロで使用するための最も簡単な変数定義の方法は、以下のように **DEFINE** ステートメントを使うことです。この構文は、Net.Data に固有のもので

```
%DEFINE variable_name="variable value"
```

```
%DEFINE variable_name={ variable value on multiple  
lines of text %}
```

variable_name は、ユーザーが変数に与える名前です。変数名は、文字あるいはアンダースコアで始まらなければなりません。そして、任意の英数字およびアンダースコアを含むことができます。すべての変数名は、大文字小文字の区別をします。ただし、表変数の、*N_columnName* および *V_columnName* は除きます。

ストリングに引用符を組み込むためには、2 つの引用符を続けて使用します。連続する引用符が 2 つだけの場合は、ヌル・ストリングに等しくなります。たとえば、以下のようにします。

```
%DEFINE HI="say ""hello"""
```

変数 HI は、say "hello" を表示します。

```
%DEFINE reply="hello"
```

変数 reply は、hello を表示します。

```
%DEFINE empty=""
```

変数 empty は、ヌルです。

幾つかの変数を、DEFINE ステートメントを使用して定義するには、DEFINE ブロックを以下のように使用します。

```
%DEFINE{
    variable1="value1"
    variable2="value2"
    variable3="value3"
    variable4="value4"
%}
```

- **HTML のフォームの SELECT および INPUT タグ**

HTML のフォームに使用される SELECT および INPUT タグを使用することができます。以下の例では、標準の HTML フォームのタグを使用して、変数を定義しています。

```
<INPUT NAME="variable_name" TYPE=...>
```

あるいは

```
<SELECT NAME="variable_name">
```

variable_name は、その変数に与えた名前です。そして、その変数の値は、フォームで受け取った入力から決定されます。Net.Data のマクロにおける、このような変数定義のタイプの使用方法の例については、「32ページの『HTML フォーム』」を参照してください。

INPUT あるいは SELECT タグから受け取った変数の値は、Net.Data のマクロにおいて DEFINE ステートメント により設定された変数の値を上書きします。

- **URL のデータ**

Net.Data のマクロを URL 要求として呼び出し、ユーザー ID のような変数を URL に組み込み、Net.Data に送信することができます。たとえば、以下のようにします。

```
http://www.ibm.com/netdata-cgi/db2www/stdqry1.d2w/input?field=custno
```

上の例では、変数名 *field* とその変数値 *custno* は、Net.Data が入力ステートメントから受け取った追加データを指定します。Net.Data は、データを受け取り、そのデータを Net.Data がデータを形成するように処理します。

変数の参照

定義済みの変数を参照して、その値を戻すことができます。

Net.Data のマクロにおいて変数を参照するには、その変数名を \$(および) 内に指定します。たとえば、以下のようにします。

```
$(variableName)
$(homeURL)
```

Net.Data が変数参照を検出すると、Net.Data は、その変数参照をその値で置き換えます。

変数を HTMLの一部として使用するには、HTML ブロックでそれらの変数を参照します。たとえば、すでに変数 *homeURL* を定義している場合、以下のようにします。

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

ホーム・ページは \$(homeURL) として指定することができます、以下のようにリンクを作成します。

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

変数は、Net.Data のマクロの任意の部分で参照することができます。変数が、それが参照されたときに未定義である場合、Net.Data は、変数を定義し、その変数に初期値としてヌルを与えます。Net.Data は、マクロの構文解析をしながら、変数参照を評価し、その変数参照を、その変数の現在の値とインラインで置き換えます。

制約事項： 循環参照（あるいは、循環）は許可されていません。たとえば、以下の DEFINE ステートメントは、値が参照され、最終値が評価されたときに、エラーとなります。

```
%DEFINE a="$(b)"  
%DEFINE b="$(a)"
```

変数の型

マクロ・ファイルでは、以下の変数参照のタイプを使用することができます。

- 48ページの『条件変数』
- 49ページの『環境変数』
- 49ページの『実行可能な変数』
- 50ページの『隠し変数』
- 51ページの『リスト変数』
- 52ページの『表変数』
- 53ページの『各種変数』
- 53ページの『表処理変数』
- 54ページの『レポート変数』
- 54ページの『言語環境変数』

条件変数

条件変数を使用すると、IF、THEN 構成要素と類似の方法を使用して、変数に対して条件値を定義することができます。条件変数を定義する場合は、変数を取ることができる 2 つの値を指定することができます。参照する最初の変数が存在する場合は、条件変数は最初の値を取得します。そうでない場合は、条件変数は 2 番目の値を取得します。条件変数の構文は、次のようになります。

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

varB が定義される場合は、varA="value_1"、そうでない場合は、varA="value_2" となります。これは、次の例のように、IF ブロックを使用するのと同じこととなります。

```
%IF ( varB )
    varA = "value_1"
%ELSE
    varA = "value_2"
%ENDIF
```

条件変数をリスト変数と一緒に使用する例については、"51ページの『リスト変数』" を参照してください。

環境変数

Net.Data が実行されている処理に存在する Net.Data の環境変数を参照することができます。それには、以下の構文を使用します。

```
%define VAR=%ENVVAR
```

たとえば、SERVER_NAMEは、以下のようにして環境変数として定義することができます。

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

そして、参照は以下のように行います。

```
The client is $(SERVER_NAME)
```

出力は以下のように見えます。

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

ENVVAR ステートメントについての詳細は、"*Net.Data* 解説書" を参照してください。

実行可能な変数

実行可能な変数を使用すると、変数参照から他のプログラムを呼び出すことができます。

DEFINE ブロックのEXEC 言語構成要素を使用して、Net.Data のマクロに実行可能な変数を定義します。EXEC 言語要素のさらに詳しい情報については、"*Net.Data* 解説書"の、言語構成要素の章を参照してください。以下の例では、変数 runit が定義され、実行可能なプログラム testProg が実行されます。

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

runit は、実行可能な変数になります。

Net.Data は、Net.Data のマクロ内で、有効な変数参照に出会うと、実行可能なプログラムを実行します。たとえば、有効な変数参照が、Net.Data のマクロの変数 runit に対して行われると、プログラム testProg が実行されます。

簡単な方法は、別の変数定義から、実行可能な変数を参照することです。以下の例は、この方法の一例を示しています。変数 date は、実行可能な変数として定義され、dateRpt は、変数参照として定義されています。この変数参照には、実行可能な変数が含まれます。

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


`$(dateRpt)` が `Net.Data` のマクロに現れるごとに、`Net.Data` は、実行可能なプログラム `date` を検索し、それを見つけると、戻ります。

Today is Tue 11-07-1999

`Net.Data` がマクロ・ファイルで実行可能な変数に出会うと、`Net.Data` は、以下の方法によって、参照されている実行可能なプログラムを探します。

1. `Net.Data` は、`Net.Data` の初期設定ファイルの `EXEC_PATH` を検索する。詳細については、"13ページの『`EXEC_PATH`』" を参照してください。
2. `Net.Data` がプログラムを見つけなかった場合は、システムの `PATH` 環境変数、あるいはライブラリー・リストで定義されているディレクトリーを検索する。実行可能なプログラムが見つければ、`Net.Data` は、そのプログラムを実行します。

制約事項： 実行可能変数を、それが呼び出す実行可能なプログラムの出力値に設定しないでください。前の例では、変数 `date` の値はヌルです。DTW_ASSIGN 関数呼び出しでこの変数を使用し、その値を別の変数に割り当てると、割り当て後の新規の変数の値も、ヌルになります。実行可能変数の目的はただ 1 つ、その変数が定義するプログラムを呼び出すことです。

実行するプログラムにパラメーターを渡すことができます。それには、変数定義の際に、そのプログラム名と一緒にそのパラメーターを指定します。次の例では、距離と時間の値が、プログラム `calcMPH` に渡されています。

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

この次の例では、システム日付を HTML リポートの一部として戻します。

```
%DEFINE tstamp=%exec "date"
```

```
%FUNCTION(DTW_SQL) myQuery() {  
  SELECT CUSTNO, CUSTNAME from dist1.customer  
  %REPORT{  
    %ROW{  
      <A HREF="/netdata-cgi/db2www/exmp.d2w/report?value1=$(V1)&value2=$(V2)">  
        $(V1) $(V2) </A> <BR>  
    %}  
    %}  
    %}  
  
  %HTML (report){  
    <H1>Report made: $(tstamp) </H1>  
    @myQuery()  
  %}
```

各レポートは、追跡がしやすくなるように日付を表示します。この例ではまた、顧客番号と名前を、別の `Net.Data` のマクロのリンクに書き込んでいます。レポートの任意の顧客をクリックすると、`exmp.d2w` という `Net.Data` のマクロが呼び出され、顧客番号と名前を `Net.Data` のマクロに渡します。

隠し変数

隠し変数を使用すると、変数の実際の名前を、Web ブラウザーを使用して HTML のソースを見ているアプリケーション・ユーザーから隠すことができます。隠し変数を定義するには、以下のようにします。

1. HTML における最後の変数参照の後に、隠したいストリングごとに変数を指定する。変数は、HTML ブロックで使用された後、以下の例のように、常にDEFINE 言語要素を使用して定義されます。 \$\$ (variable) 変数が参照され、次に定義されます。
2. 変数が参照されている HTML ブロックで、1 つのドル記号ではなく、2 つのドル記号を使用して、変数を参照します。たとえば、\$(X)ではなく、\$\$ (X) とします。

```
%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
%}

.
.
.
```

Web ブラウザーが HTML のフォームを表示すると、\$(name) および \$(addr) は、\$(name) および \$(addr) にそれぞれ置き換えられます。その結果、実際の表と列名は、HTML のフォームに表示されることはありません。アプリケーションのユーザーは、真の変数名が隠されているのかどうかを知ることができません。ユーザーがフォームを処理依頼すると、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、\$(Field) は、SQL ステートメントにおいて、SQL 照会の customer.name あるいは customer.addr で置き換えられます。

リスト変数

リスト変数を使用して、値が区切られたストリングを作成します。リスト変数は、WHERE あるいは HAVING 節に見られるような複数項目を持つ SQL 照会を構成するのに特に役に立ちます。リスト変数の構文は、次のようになります。

```
%LIST " value_separator " variable_name
```

推奨：ブランクは重要です。ほとんどの場合、値の区切り文字の前後にスペースを挿入します。ほとんどの照会は、値の区切り文字に、ブールあるいは数学演算子を使用します (たとえば、AND、OR、あるいは >)。次の例は、条件、隠し、およびリスト変数の使用例を示したものです。

```
%HTML(INPUT) {
<FORM METHOD="POST" ACTION="/netdata-cgi/db2www/example2.d2w/report">
<H2>Select one or more cities:</H2>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(cond1)">Sao Paola<BR>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(cond2)">Seattle<BR>
```

```

<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(cond3)">Shanghai<BR>
<INPUT TYPE="submit" VALUE="Submit Query">
</FORM>
%}

%DEFINE{
%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 のフォームでは、ボックスがチェックされていない場合、conditions はヌルとなり、照会では、whereClause もまたヌルになります。そうでない場合は、whereClause は、OR で区切られた選択値を持ちます。たとえば、3 都市がすべて選択される場合は、SQL は以下のようにになります。

```

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

```

次の例は、Seattle が選択されると、その結果、次のような SQL 照会になることを示しています、

```

SELECT name, city FROM citylist
WHERE cond1='Seattle'

```

表変数

表変数は、関係しあうデータの集合を定義します。表変数は、同一のレコード、すなわち行の配列と、各行のフィールドを説明する列名の配列を含みます。表は、Net.Data のマクロでは、以下のステートメントのようにして定義されます。

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

TABLE の後に続く数字は、この表が含むことができる行数の制限です。行数に制限のない表を指定するには、次の例のように、デフォルトを使用するか、ALL を指定します。

```

%DEFINE myTable2=%TABLE
%DEFINE myTable3=%TABLE(ALL)

```

表を定義するときに、表がゼロの行とゼロの列を持つと、記憶域が割り当てられません。表に値を代入するには、表を OUT あるいは INPUT パラメーターとして、関数に渡す以外に方法はありません。DTW_SQL 言語環境は、SELECT ステートメントの結果を表に自動的に書き込みます。

DTW_REXX あるいは DTW_PERL など、他のすべての環境の場合は、言語環境が自動的に表の値を設定することはありません。そのかわり、表の個々の要素は、ネイティブの言語のインタープリターから出力パラメーターとして利用できるようになります。そのためには、表の個々の要素は、スクリプトあるいはプログ

ラムを実行して設定されなければなりません。詳細については、“言語環境解説書”の言語環境の説明を参照してください。

表変数の名前を参照することにより、関数間で表を渡すことができます。表の個々の要素は、関数の REPORT ブロックで参照することができます。詳細については、“53ページの『表処理変数』”を参照してください。表変数は、通常、SQL 関数の値が代入され、次に、SQL 関数あるいは別の関数のいずれかにパラメーターとして渡された後、その関数におけるレポートへの入力として使用されます。表変数を、IN、OUT、あるいは INOUT パラメーターとして、任意の非 SQL 関数に渡すことができます。表は、SQL関数には、OUT パラメーターとしてのみ渡すことができます。

表の列名とフィールドの値は、1 を原点に持つ配列要素としてアドレス指定されます。配列は 0 から開始する、という標準 C および C++ の言語規則とは異なります。

各種変数

これらの変数は、Net.Data で定義された変数で、以下の目的のために使用することができます。

- Net.Data の処理に影響を与える
- 関数呼び出しの状態を検出する
- データベース照会の結果セットに関する情報を取得する
- ファイル場所と日付に関する情報を決定する

各種変数は、Net.Data が決定する定義済みの値、あるいはユーザーが設定する値を持つことができます。たとえば、Net.Data は、Net.Data が処理中の現行ファイルに基づいた DTW_CURRENT_FILENAME 変数の値を決定します。これに対して、Net.Data は、タブや改行文字で作られた余分なスペースを削除するかどうかを指定することができます。

定義済みの変数は、マクロ・ファイル内では変数参照として使用され、ファイルの現在の状態、日付、あるいは関数呼び出しの状態に関する情報を提供します。たとえば、現行ファイルの名前を検索するには、以下を使用することができます。

```
<p>This file is <i>$(DTW_CURRENT_FILENAME)</i>.</P>
```

変更可能な変数値は、一般には、DEFINE ステートメント、あるいは @DTW_ASSIGN() 関数を使用して設定され、Net.Data のマクロ・ファイルの処理方法に影響を与えます。たとえば、空白文字を削除するかどうかを指定するには、以下の DEFINE ステートメントを使用することができます。

```
%DEFINE DTW_REMOVE_WS="YES"
```

有効な各種変数のリストおよびその構文と例については、“Net.Data 解説書”を参照してください。

表処理変数

Net.Data は、REPORT および ROW ブロックで使用するための表変数を定義します。これらの変数を使用して、SQL 照会および関数呼び出しからの値を参照します。

表処理変数は、Net.Data が決定する事前定義値を持ち、これにより、SQL 照会あるいは関数呼び出しの結果セットからの値を、処理中の列、行、あるいはフィールドで

参照することができます。また、処理されている行の数に関する情報、あるいはすべての列名のリストにアクセスすることができます。

たとえば、Net.Data は、SQL 照会からの結果セットを処理しながら、現行の列名ごとに、変数 Nn の値を、N1 を最初の列に、N2 を 2 番目の列に、というようにして割り当てます。HTML 出力の現行列名を参照することができます。

表処理変数を、変数参照としてマクロ・ファイル内で使用します。たとえば、処理中の現行列の名前を検索するには、以下を使用することができます。

```
<p>Column 1 is <i>$(N1)</i>.</P>
```

有効な表処理変数のリストおよびその構文と例については、“*Net.Data 解説書*”を参照してください。

レポート変数

レポート変数は、HTML の出力の表示方法をカスタマイズするのに役に立ちます。レポート変数は、使用する前に、DEFINE ステートメント、あるいは @DTW_ASSIGN() で定義されなければなりません。

これらの変数は、スペーシングを指定し、デフォルトのレポート・フォーマットを上書きし、HTML の表の出力対デフォルトの表出力、および他の表示機能を指定します。たとえば、ALIGN 変数を使用すると、表処理変数に対して前後のスペースを制御することができます。以下の例では、ALIGN 変数を使用して、照会により戻されるリストの各列名をスペースで区切ります。

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

また、Net.Data が、表出力に HTML のタグを使用するかどうかを決定することもできます。DTW_HTML_TABLE を YES に設定すると、テキスト・フォーマットの表ではなく、HTML の表が作成されます。

```
%DEFINE DTW_HTML_TABLE="YES"

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

有効なレポート変数のリストおよびその構文と例については、“*Net.Data 解説書*”を参照してください。

言語環境変数

言語環境変数は、言語環境と共に使用され、言語環境による要求処理の方法に影響を与えます。言語環境変数は、それらを参照する前に、DEFINE ステートメント、あるいは @DTW_ASSIGN() 関数で定義されなければなりません。適切な Net.Data のマクロ・ブロックにおいて、言語環境変数を設定あるいは参照します。

言語環境変数を使用すると、データベースとの接続の確立、OS/390 のサブシステムの DB2 のプランの割り当て、それに NLS サポートを使用可能にする、などのタスクを実行することができます。

たとえば、SQL_STATE 変数を使用すれば、データベースから戻される SQL の状態値にアクセスしたり、表示することができます。

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

有効な言語環境変数のリストおよびその構文と例については、*Net.Data* 解説書 を参照してください。

Net.Data の関数

Net.Data は、Web アプリケーションで有益だと思われる多くの関数を提供します。ユーザー自身が関数を書くことも簡単です。55ページの図 5は、Net.Data 関数およびマクロ・ファイルがどのようにして対話しているかを示しています。

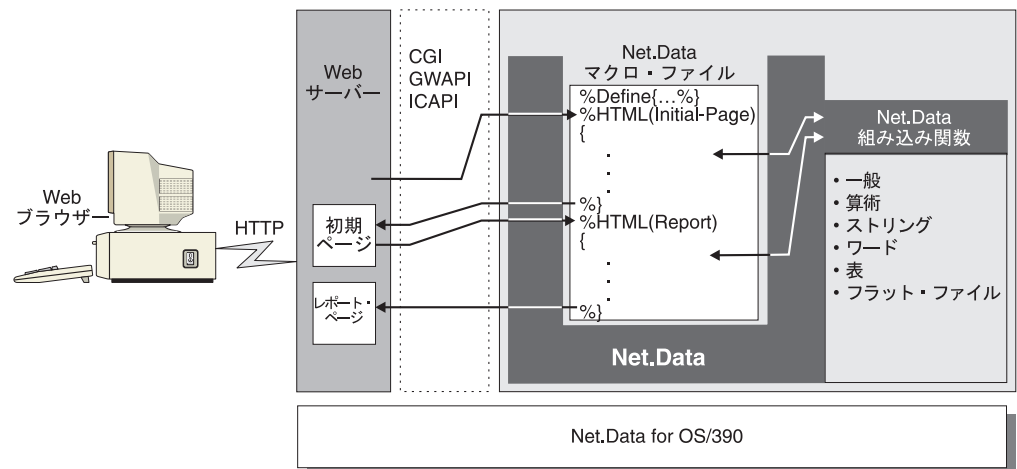


図 5. Net.Data の組み込み関数

関数の定義

ユーザー自身の関数を定義したり、NetData の関数ライブラリーを使用することができます。(Net.Data ライブラリー内にはない) ユーザー定義関数の場合、FUNCTION ブロックを使用して、Net.Data のマクロ:

詳細は、*Net.Data* 解説書 を参照してください。

構文を以下に示します。

FUNCTION ブロック:

```
%FUNCTION(type) function-name(usage parameter, ...) [RETURNS(return-var)] {
    executable-statements
    report-block
    message-block
%}
```

type

初期設定ファイルにおいて構成されている言語環境を識別します。言語環境は、特定の言語プロセッサ（これは、実行可能なステートメントを処理します）および Net.Data と言語プロセッサとの標準インターフェースを提供します。

幾つかのデフォルトの言語環境が Net.Data で提供されています。

function-name

FUNCTION ブロックの名前を指定します。FUNCTION ブロック を、関数呼び出しと共に、マクロ・ファイルのどこか別の場所で行います。関数呼び出しは、@ 記号を前に付けた *function-name* を参照します。詳細については、“57ページの『関数の呼び出し』” を参照してください。

複数の FUNCTION ブロックを、同じ名前で定義し、それらを同時に実行することができます。各ブロックは、すべて、同じパラメーター・リストを持たなければなりません。Net.Data が関数を呼び出すと、同じ名前を持つすべての FUNCTION ブロックは、Net.Data のマクロにおける定義順に実行されます。

usage

パラメーターが、入力 (IN) パラメーターか、出力 (OUT) パラメーターか、それとも両方のタイプ (INOUT) かを指定します。この指定は、パラメーターが、FUNCTION ブロック に渡されるのか、あるいはそこから受け取るのかを示しています。usage のタイプは、別のusage のタイプにより変更されるまで、パラメーター・リストのその後に続くパラメーターすべてに適用されます。デフォルトのタイプは IN です。

parameter

関数呼び出し時に指定される対応する引き数の値で置き換えられる、ローカルな効力範囲を持つ変数の名前。実行可能ステートメント、あるいは REPORT ブロック内の、たとえば \$(*parm1*) というパラメーター参照は、そのパラメーターの実際の値で置き換えられます。さらに、パラメーターは、言語環境に渡されて、その言語の自然構文を使用する実行可能ステートメントから、あるいは環境変数として、アクセス可能となります。パラメーター変数の参照は、FUNCTION ブロックの外では無効です。

また、ユーザーが指定したタイプの関数呼び出し時に、暗黙のパラメーターを渡します。初期設定ファイルの ENVIRONMENT ステートメントに、パラメーターを定義しなければなりません。

return-var

このパラメーターを、RETURNS キーワードの後に指定して、特殊な OUT パラメーターを識別します。戻り変数の値は、関数呼び出しに割り当てられ、Net.Data のマクロ処理時に関数呼び出しが、この値に置き換えられます。RETURNS 節を指定しなければ、呼び出しから言語環境への戻りコードがゼロの場合、あるいは戻りコードの値がそれ以外の場合でも、関数呼び出しの値は、ヌル・ストリングになります。

executable-statements

変数の置換および関数処理の後で、処理のための指定された言語環境に渡される言語ステートメントのセット。

FUNCTION ブロックの場合、Net.Data は、すべての変数参照を変数の値に置き換え、すべての関数呼び出しを実行し、関数呼び出しをその結果値に置き換えます。その後で、実行可能なステートメントが言語環境に渡されます。各言語環境は、ステートメントを異なる方法で処理します。実行可能なステートメントあるいは実行可能なプログラムの呼び出しについての詳細は、"49ページの『実行可能な変数』" を参照してください。

report-block

FUNCTION ブロックの出力をハンドルするための REPORT ブロックを定義します。"63ページの『レポート・ブロック』" を参照してください。

message-block

MESSAGE ブロックを定義します。このブロックは、FUNCTION ブロックによって戻された任意のメッセージをハンドルします。"60ページの『メッセージ・ブロック』" を参照してください。

最外部の Net.Data のマクロ・レイヤーと、Net.Data のマクロで呼び出される前の関数を定義します。

関数の呼び出し

FUNCTION ブロック名の前に付いている @ 文字を使用して、Net.Data のマクロから関数を呼び出します。

@function_name([argument,...])

function_name

これは、呼び出される FUNCTION ブロック、 の名前です。関数は、それが組み込み関数でなければ、Net.Data のマクロであらかじめ定義されていなければなりません。

argument

これは、定義済みの変数、リテラル文字ストリング、変数参照、あるいは関数呼び出しの名前です。関数呼び出し時の引き数は、FUNCTION ブロックのパラメーターと一致しており、各パラメーターは、FUNCTION ブロックの処理中に、対応する引き数の値が割り当てられます。引き数は、対応するパラメーターと同じ数および型でなければなりません。

Net.Data は、FUNCTION ブロック、あるいは関数呼び出しに関連付けられている組み込み関数を、次の順で処理します。

1. Net.Data は、FUNCTION ブロックの実行可能ステートメントのセクションにおける変数参照と関数呼び出しを処理する。Net.Data は、すべての変数参照を、変数の現行値で置き換え、すべての関数呼び出しを実行し、すべての関数呼び出しをその値と置き換えます。変数参照と関数呼び出しは、それらが指定された順に処理されます。このステップ中は、Net.Data は、組み込み関数を処理しません。
2. ネイティブの言語プロセッサは、実行可能ステートメントのセクションを処理する。FUNCTION ブロックの場合、プロセッサは、SQL、REXX、あるいは

Perl などの FUNCTION ブロックで指定された言語環境に対応します。組み込み関数は、実行可能ステートメントを持ちません。Net.Data は、関数名で組み込み関数を処理します。

Net.Data は、関数のパラメーターを、ネイティブの言語プロセッサに渡します。Net.Data は、IN あるいは INOUT の場合にのみ、ネイティブ言語プロセッサに値を渡し、OUT あるいは INOUT の場合にのみ、ネイティブの言語プロセッサから戻り値を受け取ります。

3. Net.Data は、言語プロセッサからの戻りコードに基づき、暗黙の RETURN_CODE および DTW_DEFAULT_MESSAGE 変数を設定する。
4. FUNCTION ブロックは、REPORT ブロックを含む場合、あるいはデフォルトのレポートを生成するように指定する場合、Net.Data は、任意の参照出力パラメーターを使用して、レポート・セクションを処理する。Net.Data は、組み込み関数に対してはレポートを生成しません。
5. FUNCTION ブロックが、ローカルな MESSAGE ブロックを含む場合、Net.Data は、その MESSAGE ブロックを処理する。Net.Data は、以下の条件のどれか 1 つが発生した場合、グローバルな MESSAGE ブロックを処理します。
 - グローバルな MESSAGE ブロックは指定されるが、戻りコードは、ローカルな MESSAGE ブロックではハンドルされない。
 - 組み込み関数が呼び出される。
6. Net.Data は、関数呼び出しを、関数呼び出しの戻り値で置き換える。FUNCTION ブロックの場合、この値は以下のどれか 1 つになります。

RETURNS パラメーター値

RETURNS キーワードを持つ FUNCTION ブロックに置き換えられます。

空ストリング ("")

RETURN_CODE がゼロの場合、RETURNS キーワードを持たない FUNCTION ブロックに置き換えられます。

RETURN_CODE

RETURN_CODE がゼロでない 場合、RETURNS キーワードを持たない FUNCTION ブロックで置き換えられます。

組み込み関数の場合、その値は、組み込み関数のフォーマットに依存します。

ストアード・プロシージャの呼び出し

ストアード・プロシージャは、コンパイル済みのプログラムで、DB2 のローカルあるいはリモート・サーバーに格納され、SQL ステートメントを実行することができます。Net.Data では、ストアード・プロシージャは、CALL ステートメントを使用して、Net.Data の関数から呼び出されます。ストアード・プロシージャのパラメーターは、Net.Data の関数パラメーター・リストから渡されます。

ストアード・プロシージャは、複数の値を出力パラメーターとして戻すことができ、Net.Data の表に結果セットを戻して、レポートや行セクションで使うことができます。

ストアド・プロシージャは、以下のデータ型を使用することができます。

表 2. ストアド・プロシージャのデータ型

CHAR	FLOAT	SMALLINT
DECIMAL	INTEGER	VARCHAR
DOUBLE	GRAPHIC	VARGRAPHIC
DOUBLEPRECISION		

これらのデータ型に関するさらに詳しい情報については、データベースの文書を参照してください。

例 1: ストアド・プロシージャを呼び出し、出力パラメーターに、1 つの値を戻します。

```
%FUNCTION(DTW_SQL) STORED_PROC1 ( IN SMALLINT arg1,
                                   OUT VARCHAR(9) retval)
    RETURNS (RESULT) {

CALL STATSRPT1
%}

%HTML(REPORT) {
@STORED_PROC1(arg1, retval)
.
.
.
%}
```

ストアド・プロシージャの名前は、*STATSRPT1*で、*CALL* ステートメントで呼び出されます。 *STORED_PROC1*は、ストアド・プロシージャを呼び出す関数の名前です。

例 2: ストアド・プロシージャを呼び出し、*REPORT* ブロックに表をプリントします。

```
%FUNCTION(DTW_SQL) STORED_PROC2 (IN CHAR(30) arg1,
                                   IN VARCHAR(100) arg2,
                                   OUT CHAR(30) retval)
    RETURNS (RESULT) {

CALL STATSRPT2

%REPORT {
    $(N1)      $(N2)
    %ROW {
        $(V1)      $(V2)
    %}

%}

%}
```

ストアド・プロシージャ *STATSRPT2* は、*CALL* ステートメントで呼び出されます。 *STORED_PROC2* は、ストアド・プロシージャを呼び出します。

例 3: ストアド・プロシージャを呼び出し、続けて処理を行うために、表を *HTML* ブロックに戻します。

```
%FUNCTION(DTW_SQL) stored_proc3 (IN CHAR(30) arg1,
                                   IN VARCHAR(100) arg2,
                                   OUT CHAR(30) retval,
                                   OUT myTable)
    RETURNS (RESULT) {

CALL STATSRPT3
```

```

%REPORT {
    $(N1)      $(N2)
    %ROW {
        $(V1)      $(V2)
    %}
%}

%}
%
```

ストアド・プロシージャ *STATSRPT3* は、CALL ステートメントで呼び出されます。 *STORED_PROC3* は、ストアド・プロシージャを呼び出します。ストアド・プロシージャは、HTML セクションに表をプリントし、さらに処理を行うために、OUT パラメーター *myTable* に表を戻します。このパラメーターは、表変数でなければなりません。

メッセージ・ブロック

MESSAGE ブロックにより、関数呼び出しの成功あるいは失敗を基にして、関数呼び出し後の進め方を決定することができ、関数の呼び出し側に情報を表示することができます。 *Net.Data* は、以下のメッセージ・ブロック処理を使用します。

1. *Net.Data* は、FUNCTION ブロックへの関数呼び出しごとに、言語環境変数 *RETURN_CODE* を設定する。
2. 言語環境が、戻りコード値を *Net.Data* に渡すと、*Net.Data* は、*RETURN_CODE* の値を、戻りコード値に設定する。
3. 関数呼び出しが完了すると、MESSAGE ブロックは、*RETURN_CODE* の値を使用して、進め方を決定する。

MESSAGE ブロックは、連続したメッセージ・ステートメントで構成され、各メッセージ・ステートメントは、戻りコード値、メッセージ・テキスト、および取るべきアクションを指定します。 MESSAGE ブロックの構文は、*Net.Data* 解説書の言語構成要素の章に示されています。

MESSAGE ブロックは、グローバルあるいはローカルな効力範囲を持つことができます。 MESSAGE ブロックが、FUNCTION ブロックで定義されている場合は、その効力範囲は、その FUNCTION ブロックに対してはローカルです。 MESSAGE ブロックが、最外部のマクロ・レイヤーで指定されている場合は、MESSAGE ブロックは、グローバルな効力範囲を持ち、*Net.Data* のマクロで実行されるすべての関数呼び出しに対してアクティブになります。 2 つ以上のグローバルな MESSAGE ブロックを定義している場合は、最後に定義されたブロックがアクティブになります。

Net.Data は、以下のルールを使用し、関数呼び出しからの *RETURN_CODE* 変数の値を処理します。

1. ローカルな MESSAGE ブロックを 完全一致で検査する。指定に応じて、抜け出るか、続行します。
2. *RETURN_CODE* が 0 でない場合は、ローカルな MESSAGE ブロックを、+default あるいは -defaultで検査する。これは、*RETURN_CODE* の符号に依存し、指定に応じて、抜け出るか、続行します。
3. *RETURN_CODE* が 0 でない場合、ローカルな MESSAGE ブロックを、default で検査する。指定に応じて、抜け出るか、続行します。
4. グローバルな MESSAGE ブロックを 完全一致で検査する。指定に応じて、抜け出るか、続行します。

5. RETURN_CODE が 0 でない場合は、グローバルな MESSAGE ブロックを、+defaultあるいは -defaultで検査する。これは、RETURN_CODE の符号に依存し、指定に応じて、抜け出るか、続行します。
6. RETURN_CODE が 0 でない場合、グローバルな MESSAGE ブロックを、defaultで検査する。指定に応じて、抜け出るか、続行します。
7. RETURN_CODE が 0 でない場合、Net.Data の内部デフォルト・メッセージを発行し、抜け出ます。

以下の例は、グローバルな MESSAGE ブロックと、関数の MESSAGE ブロックを持つ Net.Data のマクロの部分を表示しています。

```
%{ 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
%}
```

my_function() が RETURN_CODE 値 50 で戻れば、Net.Data は次の順にエラーを処理します。

1. ローカルな MESSAGE ブロックを、完全一致で検査する。
2. ローカルな MESSAGE ブロックを +defaultで検査する。
3. ローカルな MESSAGE ブロックを defaultで検査する。
4. グローバルな MESSAGE ブロックを、完全一致で検査する。
5. グローバルな MESSAGE ブロックを +defaultで検査する。

Net.Data が一致を検出した場合、Net.Data はメッセージ・テキストを Web ブラウザーに送信し、要求されたアクションを検査します。

continue を指定した場合は、Net.Data は、メッセージ・テキストをプリントしてから、Net.Data マクロの処理を継続します。たとえば、マクロが my_functions() を 5 回呼び出し、エラー 100 が、上の例の MESSAGE ブロックの処理中に検出された場合、プログラムからの出力は、次のようになります。

```
.
.
.
11 May 1997                $245.45
13 May 1997                $623.23
19 May 1997                $ 83.02
return code 100 message
22 May 1997                $ 42.67

Total:                     $994.37
```

マクロに HTML を作成

Net.Data により、標準 HTML をアプリケーション・ユーザーのブラウザーに簡単に提供することができます。以下の節では、マクロの HTML および REPORT ブロックについて説明し、Net.Data のマクロにおける HTML のフォーマット方法を示します。これらのブロックの構文情報については、“*Net.Data 解説書*”の言語構成要素の章を参照してください。

HTML ブロック

Net.Data のマクロ・ファイルには、HTML ブロックおよび Web ブラウザーへの HTML 出力を生成する、HTML ブロックの関数が含まれます。マクロあるいは直接要求のいずれかの方法を使用して Net.Data を起動するときは常に、HTML ブロックを指定しなければなりません。HTML ブロックの内容は、それ以降の Net.Data の起動を制御します。

有効な HTML ならどれでも、HTML ブロックに現れることができます。さらに、INCLUDE ステートメント、関数呼び出し、および HTML ブロックの変数参照を使用することができます。以下の例は、Net.Data のマクロにおける HTML ブロックの一般的な使われ方を示しています。

```
%DEFINE DATABASE="MNS96"

%HTML(INPUT) {
<H1>Hardware Query Form</H1>
<HR>
<FORM METHOD="POST" ACTION="/netdata-cgi/db2www/equip1st.d2w/report">
<dl>
<dt>What hardware do you want to list?
<dd><input type="radio" name="hardware" value="MON" checked>Monitors
<dd><input type="radio" name="hardware" value="PNT">Pointing devices
<dd><input type="radio" name="hardware" value="PRT">Printers
<dd><input type="radio" name="hardware" value="SCN">Scanners
</dl>
<HR>
<input type="submit" value="Submit">
</FORM>
%}

%FUNCTION(DTW_SQL) myQuery() {
SELECT MODNO, COST, DESCRIP FROM EQPTABLE WHERE TYPE=$(hardware)
%REPORT{
<B>Here is the list you requested:</B><BR>
%ROW{
<HR>
$(N1): $(V1)    $(N2): $(V2)
<P>
$(V3)
%}
%}
%}

%HTML(REPORT) {
@myQuery()
%}
```

Net.Data のマクロを、以下の例のように、HTML のリンクから起動することができます。

```
<a href="http://www.ibm.com/netdata-cgi/db2www/equip1st.d2w/input">List of hardware</a>
```

アプリケーション・ユーザーがこのリンクをクリックすると、Web ブラウザーは、Net.Data を起動し、Net.Data は、マクロ・ファイルを解析します。Net.Data が起動に指定された HTML ブロック、この場合は HTML (INPUT) ブロック、の処理を開始すると、Net.Data は、そのブロック内のテキストの処理を開始します。Net.Data は、Net.Data のマクロ言語構成要素として認識できないものはどれも、HTML とみなし、表示するためにブラウザーに送信します。

ユーザーが、選択を行い、「処理依頼 (Submit)」ボタンを押すと、Net.Data は、HTML の FORM 要素の ACTION 部分を実行します。この部分は、Net.Data の マクロの HTML(REPORT) ブロックへの呼び出しを指定します。次に Net.Data は、HTML(INPUT) ブロックの場合と同様に、HTML(REPORT) ブロックを処理します。

Net.Data は次に、myQuery() 関数呼び出しを処理します。この関数呼び出しは、次に SQL FUNCTION ブロックを起動します。SQL ステートメントの \$(hardware) 変数参照を、入力フォームで戻された値と置き換えた後、Net.Data は照会を実行します。この時点で、Net.Data は再度 HTML のブラウザーへの送信を開始し、REPORT ブロックで指定された HTML に従って、照会結果を表示します。

Net.Data が REPORT ブロックの処理を完了した後で、Net.Data は、HTML(REPORT) ブロックに戻り、処理を終了します。

Net.Data は、起動されるごとに、1 つのHTMLだけを処理します。しかし、HTML のリンクとフォームを使用することにより、他の人が別の HTML ブロックで、簡単にもう 1 つの Net.Data の起動を開始することができます。すべては、ユーザーにより制御されます。

レポート・ブロック

REPORT ブロックの言語構成要素を使用して、FUNCTION ブロックからのデータ出力をフォーマットし、表示することができます。この出力は、基本的には、表データです。ただし、HTML タグ、マクロ変数参照、および関数呼び出しの有効な組み合わせを指定することはできます。表の名前は、オプションで REPORT ブロックで指定することができます。表の名前を指定しない場合は、Net.Data は、FUNCTION ブロックのパラメーター・リストの最初の出力表の表データを使用します。FUNCTION ブロックで表を指定しない場合は、Net.Data は、デフォルトの表データを使用します。

REPORT ブロックは、次の 3 つの部分を持ち、各部分はオプションです。

- 表の行データの前に一度だけ表示される HTML データを含むヘッダー情報。
- 結果表の行ごとに一度だけ表示される HTML および表変数を含む ROW ブロック。
- 表の行データの後に一度だけ表示されるデータを含むフッター情報。

ROW ブロックからの表出力はどれも表示をしないようにするには、ROW ブロックを空にしておくか、ROW ブロックを完全に省略します。

Net.Data により提供される REPORT ブロック内の幾つかの変数を使用して、Net.Data のマクロの結果表のデータにアクセスすることができます。これらの変数は、53 ページの『表処理変数』で説明されています。追加詳細については、“Net.Data 解説” のレポート変数のセクションを参照してください。

ヘッダーおよびフッター情報を提供するには、ROW ブロックの前後にテキストを提供します。Net.Data は、ROW ブロックの前で検出したものはすべて処理し、それをヘッダー情報として扱います。そして、ROW ブロックの後で検出したものはすべて、フッター情報として扱います。HTML ブロックの場合と同様、Net.Data は、マクロ言語構成要素として認識できない、ヘッダー、ROW およびフッター・ブロックにおけるすべてのものを、HTML として扱い、HTML データをブラウザに送信します。

REPORT ブロックの関数および変数を使用することもできます。

Net.Data に、フォーマット済みのテキストを使用してデフォルトのレポートをプリントさせるには、マクロ・ファイルに REPORT ブロックを組み込まないようにします。以下の例は、デフォルトのレポート・フォーマットを示しています。

SHIPDATE	RECDATE	SHIPNO
25/05/1997	30/05/1997	1495194B
25/05/1997	28/05/1997	2942821G

HTML のタグをフォーマット済みテキストの代わりに使用するには、DTW_HTML_TABLE を YES に設定します。

デフォルトのレポートのプリントを使用禁止にするには、DTW_DEFAULT_REPORT を NO に設定するか、空の REPORT ブロックを指定します。たとえば、以下のようになります。

```
%REPORT{%
```

以下の例は、特殊変数と HTML のタグを使用した、レポート・フォーマットのカスタマイズ方法を示しています。この例では、CustomerTbl の表から、名前、電話番号、および FAX 番号を表示しています。

```
%FUNCTION(DTW_SQL) custlist() {
    SELECT Name, Phone, Fax FROM CustomerTbl
}%REPORT{
<I>Phone Query Results:</I>
<BR>
=====
<BR>
%ROW{
    Name: <B>$(V1)</B>
<BR>
    Phone: $(V2)
<BR>
    Fax: $(V3)
<BR>
-----
<BR>
    %}
    Total records retrieved: $(NUM_ROWS)
    %}
    %}
```

この結果作成されるレポートは、Web ブラウザーでは、次のように表示されます。

```
Phone Query Results:
=====
Name: Doen, David
Phone: 422-245-1293
Fax: 422-245-7383
-----
```


Name: Ramirez, Paolo
Phone: 955-768-3489
Fax: 955-768-3974

Name: Wu, Jianli
Phone: 525-472-1234
Fax: 525-472-1234

Total records retrieved: 3

Net.Data は、以下を行い、レポートを生成しました。

1. *Phone Query Results:* を、レポートの最初に 1 回プリントする。
2. 変数 V1、V2、そして V3 に、Name、Phone、および Fax の値をそれぞれ、検索時に各行ごとに与える。
3. 各取得行の後ろに罫線を引き、読みやすくする。
4. スtring *Total records retrieved:* および NUM_ROWS の値を、レポートの最後に 1 回だけプリントする。

マクロ・ファイルにおける条件付き論理とループ

Net.Data により、IF および WHILE ブロックを使用して、条件論理およびループを Net.Data のマクロに取り込むことができます。以下の議論では、HTML および MACRO IF ブロック共に、IF ブロックと呼ぶことにします。そして、特にそうでないと指定しない限り、その振る舞いも同じです。

条件付き論理

IF ブロックを使用して、Net.Data のマクロで、条件付き処理を行います。IF ブロックは、ほとんどの高級言語の IF ステートメントに類似しています。その理由は、この IF ブロックは、1 つ以上の条件をテストし、次に条件テストの結果に基づき、ステートメントのブロックを実行することができるからです。

IF ブロックは、マクロ内のほとんどどこにでも指定することができ、それらをネストすることができます。IF ブロックの構文は、*Net.Data* 解説書の言語構成要素の章に示されています。

IF ブロックの実行可能なステートメント・ブロックに許される要素は、IF ブロック自身の位置に依存します。IF ブロックを含むブロック内で有効な要素ならどれでも、その IF ブロックで有効です。たとえば、IF ブロックを、HTML ブロック内部に指定する場合、HTML ブロックに許される要素はどれでも、IF ブロックに許されます。同様に、IF ブロックを、Net.Data のマクロの宣言文の他のブロックの外で指定する場合は、その他のブロック（たとえば、DEFINE ブロック、あるいは FUNCTION ブロック）の外で許される要素のみが、IF ブロック内で許されます。

例外：IF ブロックが REPORT ブロック内にある場合は、IF ブロック内に、ROW ブロックを指定しないでください。

Net.Data は、IF ブロック条件リストを、条件を構成している項の内容に基づき、2 つの方法のうちのいずれか 1 つで処理します。デフォルトのアクションは、すべての項を、String として処理し、条件で指定された String 比較を実行します。しかし、以下の 2 つの条件が満足される場合は、Net.Data は、数値比較を実行します。

- 条件がバイナリー操作 (<, >, <=, >=, !=, ==) の場合。
- 条件に含まれる項が共に整数を表している場合。この意味は、項は数字からなるストリングで、オプションとして、その前に、'+' あるいは '-' の文字がくるということです。ストリングは、'+' あるいは '-' 以外の非数字文字を含むことができません。

有効なストリングの例：

```
+1234567890
-47
000812
92000
```

無効なストリングの例：

```
- 20      (空白文字を含んでいる)
234,000   (コンマを含んでいる)
57.987    (小数点を含んでいる)
```

Net.Data は、IF ブロックを、そのブロックを実行したときに評価します。これは、Net.Data によって最初に読み取られるときとは異なる場合があります。たとえば、IF ブロックを REPORT ブロックに指定すると、Net.Data は、REPORT ブロックを含む FUNCTION ブロック定義を読み取るときに、IF ブロックに関連付けられた条件リストを評価しません。これを行うのは、関数を呼び出して、それを実行するときなのです。これは、IF ブロックの条件リストの部分および実行されるステートメントのブロックの両方に対してもあてはまります。

例： 次の例は、他のブロックに、IF ブロックを含むマクロ・ファイルを示しています。

```
%{ This macro is called from another macro, passing the operating system
    and version variables in the form data.
}%

%IF (platform == "AS400")
  %IF (version == "V3R2")
    %INCLUDE "as400v3r2_def.hti"
  %ELIF (version == "V3R7")
    %INCLUDE "as400v3r7_def.hti"
  %ELIF (version == "V4R1")
    %INCLUDE "as400v4r1_def.hti"
%ENDIF
%ELSE
  %INCLUDE "default_def.hti"
%ENDIF

%HTML (report){
  %WHILE (a < "10") {
    outer while loop #$(a)<BR>
    %IF (@dtw_rdivrem(a,"2") == "0")
      this is an even number loop<BR>
    %ENDIF
    @DTW_ADD(a, "1", a)
  }
}%
```

ループ構成体

WHILE ブロックを使用して、Net.Data のマクロでループを実行します。IF ブロックと同様、WHILE ブロックにより、1 つ以上の条件をテストし、次に、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実行することができます。IF ブロックと異なり、ステートメントのブロックは、条件テスト結果に基づき、何回でも実行することができます。

WHILE ブロックを HTML ブロック、REPORT ブロック、ROW ブロック、および HTML の IF ブロック内で指定し、それらをネストすることができます。WHILE ブロックの構文は、*Net.Data* 解説書 の言語構成要素の章に示されています。

Net.Data は、WHILE ブロックを、IF ブロックを処理するのと全く同じ方法で処理します。しかし、ループを 1 回完了するごとに、条件リストを再評価します。そして、どの条件付きループ構成要素の場合も同じですが、条件のコード化に誤りがある場合は、処理は無限ループに陥ることがあります。

例：次の例は、WHILE ブロックを持つマクロ・ファイルを示しています。

```
%DEFINE loopCounter = "1"

%HTML(build_table) {
%WHILE (loopCounter <= "100") {
    %{ generate table tag and column headings %}
    %IF (loopCounter == "1")
        <TABLE BORDER>
        <TR>
        <TH>Item #
        <TH>説明
    %ENDIF

    %{ generate individual rows %}
    <TR>
    <TD>$(loopCounter)
    <TD>@getDescription(loopCounter)

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

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

第6章 組み込み関数の使用

Net.Data は、Web ページ開発を容易にするための大きな組み込み関数のセットを提供します。これらの関数は、すでに Net.Data により定義されているので、FUNCTION ブロックではそれらの関数を定義する必要はありません。ユーザー定義の関数を呼び出すことができるのなら、マクロのどこからでもこれらの関数を簡単に呼び出すことができます。

組み込み関数は、3 つの方法で、その結果を戻すことができます。各関数とその結果をどのように戻すかを接頭部により示すことができます。

- **DTW_ and DTWF_:** 呼び出し結果は、出力パラメーターで戻されます。あるいは、結果は戻されません。(DTWF_ は、フラット・ファイル関数に対応する接頭部です。)
- **DTW_r および DTWF_r:** マクロにおける関数呼び出しは、関数呼び出しの結果に置き換えられます。その方法は、RETURNS キーワードを指定したユーザー関数の関数呼び出しが、RETURNS キーワードの値で置き換えられるのと同じです。
- **DTW_m:** 複数の結果が、関数に渡されたパラメーターで戻されます。

組み込み関数によっては、タイプを持たないものがあります。詳細は、"Net.Data 解説書" を参照してください。

Net.Data は、以下の組み込み関数を提供します。

- 汎用関数
- 数学関数
- スtring関数
- ワード関数
- 表関数
- フラット・ファイル関数

以下の節では、Net.Data の組み込み関数について高度な概説を行います。各関数の構文および例の説明については、"Net.Data 解説書" を参照してください。

汎用操作関数

以下のタイプの関数を使用して、汎用、数学、String、ワード、あるいは表操作の各関数を実行します。

汎用関数

この関数セットは、データを変更したり、システム・サービスを利用することにより、Web ページの開発に役に立ちます。これらの関数を使用して、照会、環境変数の設定、HTML のエスケープ・コードの使用、およびシステムからの有益な情報の取得、を行うことができます。

数学関数

これらの関数は、数学操作を実行し、数値データの計算あるいは変更を行うことができます。標準的な

数学操作の他にも、法による除算の実行、演算結果の精度の指定、科学表記の使用、などを行うことができます。

ストリング関数

これらの関数により、ストリング内の文字を操作することができます。ストリングの大文字小文字の変更、文字の挿入あるいは削除、別の変数へのストリング値の割り当てを行うことができます。さらに、その他の役にたつ関数を実行することができます。

ワード関数

これらの関数により、ストリング内のワードを操作することができます。これらの関数のほとんどは、ストリング関数と同じ働きをします。ただし、ワード全体に対して働きます。たとえば、これらの関数を使用して、ストリング内のワード数のカウント、ワードの削除、ストリングからのワードの取得、などを実行することができます。

表関数

これらの関数を使用して、Net.Data の表変数のデータを使い、レポートやフォームを生成することができます。表変数には、値の配列、およびそれらに関連付けられている列の名前が含まれます。これらの関数は、値のグループを関数に渡すのに都合の良い方法を提供してくれます。

フラット・ファイル関数

フラット・ファイル・インターフェース (FFI) 関数を使用して、フラット・ファイル内の保管データだけでなく、フラット・ファイルのソース (テキスト・ファイル) のデータのオープン、読み取り、および操作をすることができます。

第7章 言語環境の使用

Net.Data はデータとプログラム言語環境を提供します。これによって、ユーザーはデータ・ソースをアクセスし、ビジネス論理を持つアプリケーション・プログラムを実行します。たとえば、SQL 言語環境によって、ユーザーは SQL ステートメントを DB2 サブシステムに渡すことが可能になり、REXX 言語環境によって、ユーザーは REXX プログラムを起動できるようになります。ユーザーは、さらに SYSTEM 言語環境を使用して、C や C++ プログラムを実行することができます。たとえば、External CICS Interface (EXCI) インターフェースを使用して、CICS プログラムを実行します。

Net.Data によって、ユーザー作成データとプログラム言語環境をプラグイン方式で簡単に追加することもできます。それぞれのユーザー作成言語環境は、Net.Data によって定義された標準的なインターフェースのセットをサポートしていなければなりません。さらに、ダイナミック・リンク・ライブラリー (DLL) として実装されていなければなりません。Net.Data 初期設定ファイル内の環境ステートメントは、DLL に言語環境名を関連付けます。Net.Data は、FUNCTION ブロックへの関数呼び出しが最初に検出されると、DLL をロードして実行します。FUNCTION ブロックは言語環境名を指定します。同じ言語環境名を指定する FUNCTION ブロックへのこれ以降の関数呼び出しでは、DLL のロードは 1 度しか行われないため、Net.Data は DLL を実行するだけになります。

71ページの図6では、Web サーバー、Net.Data、および Net.Data 言語環境間の関連を表示しています。

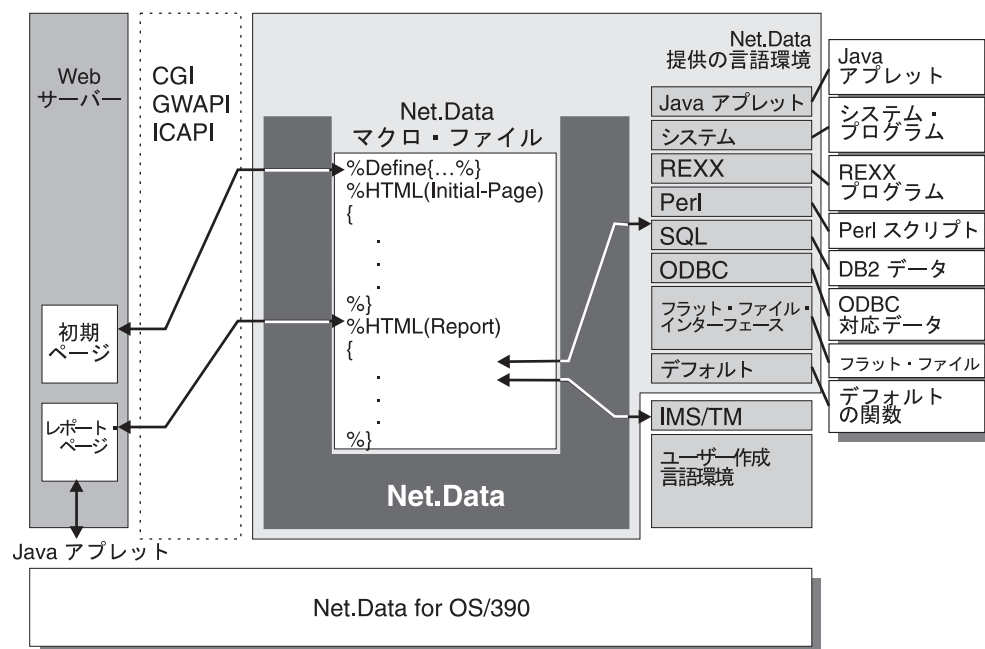


図 6. *Net.Data* 言語環境

Net.Data 提供の言語環境および言語環境のユーザー作成方法の詳細については、*"Net.Data 言語環境解説書"*を参照してください。

第8章 パフォーマンスを向上させる

パフォーマンスを向上させることは、システム・チューニングの重要な部分です。本章では、パフォーマンスを向上させるための戦略と、パフォーマンスを低下させる可能性のある構成に関する説明をします。本章で説明するトピックは、以下の通りです。

- 73ページの『ICAPI または GWAPI を使用してパフォーマンスを向上させる』
- 73ページの『DB2 for OS/390 のメッセージを抑制する』

ICAPI または GWAPI を使用してパフォーマンスを向上させる

CGIY の代わりに Web server API で Net.Data を起動すると、パフォーマンスを向上させることができます。Net.Data が Web サーバーの API モードで実行されている場合には、Net.Data は Web サーバーのアドレス空間でスレッドとして実行され、CGI プロセスとして Net.Data を起動するオーバーヘッドがなくなります。Net.Data のパフォーマンス・オプションの中で、Web サーバー API は最高のパフォーマンスを提供します。

Net.Data for OS/390 では、以下のものを使用して構成する時に、Web サーバーが Net.Data を起動する方法を決定します。

- CGI を使用する
- サポートされている Web サーバー API (ICAPI または GWAPI) を使用する

API を使用するための Net.Data と Web サーバーの構成方法を理解するには、“19ページの『ICAPI あるいは GWAPI と一緒に使用するための Net.Data の構成』”を参照してください。

考慮事項: Web サーバー API を使用すると、アプリケーションの分離をせずにパフォーマンスを向上させることができます。Net.Data はマルチ・スレッド・モードで実行します。このため、ユーザー開発言語環境、不適切な呼び出し、またはデータベースの故障によるエラーは、Web サーバーに問題を発生させ、サーバーをダウンさせることがあります。CGI に対して Web サーバー API の内の 1 つを使用するかどうかは、ユーザーのアプリケーションが、パフォーマンスを重視するかアプリケーションの分離を重視するかによって決定します。

Net.Data を ICAPI または GWAPI とともに使用するような構成にするために、Net.Data を起動するための構文は、CGI とともに使用するような構成をするために Net.Data を起動する構文と同じです。詳細と例については“29ページの『第4章 Net.Data を起動する』”を参照してください。

DB2 for OS/390 のメッセージを抑制する

SQL 言語環境を使用して非ゼロ SQLCODES から DB2 メッセージを抑制することによって、Net.Data for OS/390 のパフォーマンスを向上させることができます。DB2MSGs 変数を使用して、ユーザー・アプリケーションに必要なメッセージ・レベルを指定します。通常、実動モードで DB2MSGs を NONE に設定して、DB2

SQLCODE 処理をう回することができます。DB2MSGSGS を NONE または ERRORONLY に設定する場合には、ユーザー・マクロから MESSAGE ブロックを使用して非ゼロ SQLCODES をキャッチすることができます。ユーザー・マクロで MESSAGE ブロックを使用する方法を理解するには、"Net.Data 解説書" を参照してください。

SQLCODES メッセージ・レベルを指定するには、Net.Data 初期設定ファイルの DB2MSGSGS 構成変数を使用します。構成情報については "8ページの『DB2MSGSGS: DB2 のメッセージ・テキスト変数』" を参照してください。

指定可能な値:

DB2MSGSGS [=] *message_level*

message_level は Net.Data が提供する DB2 メッセージのレベルを指示します。次のように指定することができます。

NONE Net.Data はメッセージを出力しないことを指定します

ERRORONLY Net.Data が負の SQLCODE 値に対するメッセージだけを出力するように指定します。

ALL Net.Data がすべての SQLCODE 値に対するメッセージを出力するように指定します。これはデフォルトです。上記にリストされた有効な値以外を DB2MSGSGS に指定した場合には、Net.Data はデフォルトで ALL を使用します。

付録A. Net.Data for OS/390 バージョン 2 SMP/E のインストール

本付録は、プログラムのインストールと保守の責任者となっているシステム・プログラマーに向けたものです。本付録は、Net.Data for OS/390 バージョン 2 のユーザー・ライセンスと一緒に納入された、Net.Data for OS/390 バージョン 2 のプログラム資料説明書にあるインストール情報のサブセットです。本付録には、以下の情報が含まれています。

- 75ページの『インストール要件および考慮事項』
- 78ページの『SMP/E を使った Net.Data のインストール』

以下の Net.Data のためのインストールおよび構成ステップは、5ページの『第2章 Net.Data のインストールおよび構成』にあります。

- 6ページの『Net.Data の初期設定ファイルのインストール』
- 6ページの『Net.Data の初期設定ファイルのカスタマイズ』
- 17ページの『作業負荷管理の考慮事項』
- 17ページの『CGF と一緒に使用する Net.Data の構成』
- 19ページの『ICAPI あるいは GWAPI と一緒に使用するための Net.Data の構成』
- 20ページの『言語環境のセットアップ』
- 22ページの『メッセージ・カタログを使用可能にする』
- 22ページの『Net.Data のファイルおよびデータ・セットへのアクセス権の指定』

Net.Data を、MVS カスタムビルト・インストレーション・プロセス・オフアリング (CBIPO*)、SystemPac*、あるいは ServicePac と一緒にインストールしている場合は、本付録は使用しないでください。これらのオフアリングを使用する場合は、オフアリングとともに提供されているジョブと文書を使用してください。この文書は、必要に応じて、プログラム資料説明書の特定のセクションを指示することがあります。

MVS カスタムビルト・デリバリー・オフアリング (CBPDO*) (5751-CS3) を使用して、Net.Data をインストールする場合は、CBPDO 磁気テープで提供されるソフトコピーのプログラム資料説明書を使用してください。ご購入の CBPDO には、本製品のソフトコピー予防サービス・プラン (PSP) のアップグレード版が含まれています。

インストール要件および考慮事項

以下の節は、Net.Data のインストールとアクティブ化のためのシステム要件を確認します。以下の用語が使用されます。

駆動システム

プログラムをインストールするために使用されるシステム。

ターゲット・システム

プログラムがインストールされるシステム。

多くの場合、同じシステムを、駆動システムおよびターゲット・システムとして使用することができます。しかし、自分のシステムのクローンをセットアップし、実行システムの IPL 可能なコピーを別に作成して、ターゲット・システムとして使用したい場合もあります。このクローンは、SMP/E が更新するシステム・ライブラリーすべてのコピー、そのシステム・ライブラリーを記述している SMP/E SCI データ・セットのコピー、およびユーザーの PARMLIB と PROCLIB を含んでいなければなりません。

2 つのシステムは、以下の場合に使用しなければなりません。

- すでにインストールされている製品の新規レベルをインストールする場合、新規製品は旧製品を削除する。別のターゲット・システムにインストールすることにより、旧製品を生産に使用し続けながら、新規製品を試験することができます。
- ライブラリーあるいはロード・モジュールを他の製品と共有する製品をインストールする場合は、インストールにより他の製品を破壊する場合がある。試験システムあるいはクローンにインストールすると、生産システムを破壊することなく、これらの強力な製品を評価することができます。

駆動システム要件

本節では、Net.Data のインストールに要求される駆動システムの環境について説明します。

マシン要件

駆動システムは、必要なソフトウェアをサポートするハードウェア環境ならどのような環境でも実行することができます。

プログラミング要件

76ページの表 3 は、駆動システムのプログラミング要件について説明しています。

表 3. 駆動システムのソフトウェア要件

プログラム番号	製品名および最低の VRM/サービス・レベル
5645-001	OS/390 システム修正変更プログラム/拡張 (SMP/E) リリース 1

ターゲットのシステム要件

本節では、Net.Data のインストールおよびその使用に要求される駆動システムの環境について説明します。

マシン要件

ターゲット・システムは、必要なソフトウェアをサポートするハードウェア環境ならどのような環境でも実行することができます。

プログラミング要件

ターゲット・システムには、以下の節で説明しているプログラミング要件があります。

• 最低要件

最低要件は、以下のいずれかとして定義されます。

- **インストール時の要件**：インストール時に必要な製品。言い換えれば、本製品は、この要件が満足されなければ、正常にインストールされません。この条件には、REQ、PRE、あるいは CALLLIB として指定されている製品が含まれます。
- **実行時の要件**：本製品の正常なインストールには必要ではないけれども、本製品が動作するには、実行時に必要となる製品。

77ページの表 4 は、ターゲット・システムの最低要件について説明しています。

表 4. ターゲット・システムの最低要件。

プログラム番号	製品名および最低の VRM/サービス・レベル	インストールの必要性
5688-198	言語環境/370 バージョン 1.5 あるいはそれ以降。LE 1.5 サービスを適用しなければなりません。APARの PN80033、PN79307、PN80015、PN76475。	はい
5695-039	RACF 2.1 あるいは必要なサポートが受けられる、それと等価な機密保護製品	いいえ
5645-001	APAR の PQ12091、OW32688 が付属する OS/390 リリース 3 あるいはそれ以上	はい
以下のどれか 1つ		
5697-B14	APAR の PQ15294、PQ0598130 が付属する Internet Connection Secure Server for OS/390 バージョン 2 リリース 2	はい
	Lotus Domino Go Webserver for OS/390 バージョン 4.6.1 http://www.ics.raleigh.ibm.com/dominogowebserver	はい

• 機能要件

機能要件とは、本製品の正常なインストールあるいは本製品の基本機能には必要はないけれども、本製品の特定の機能が動作するためには実行時に必要となる製品、として定義されます。これには、IF REQ として指定される製品が含まれます。77ページの表 5 は、機能要件を説明しています。

表 5. ターゲット・システムの機能要件。

プログラ ム番号	製品名および最低の VRM/サービス・レベル	機能	インストールの 必要性
5655-DB2	APAR の PQ06950、PQ09901 が付属する DB2 for OS/390 バージョン 5 or あるいは 6	<ul style="list-style-type: none"> – 呼び出し接続機能および RRS 接続機能 – SQL アプリケーション・プログラミング・インターフェース – DB2 CLI 機能 	いいえ

- オプションのソフトウェア

- IMS アプリケーションおよびデータ・アクセスの場合：各企業向け IMS/ESA バージョン 5.1 あるいはそれ以上
- DataJoiner を介した異機種のデータ・ソースへのアクセスの場合：PTF U447593 が付属する DataJoiner for AIX バージョン 1.2、あるいは DataJoiner for HP-UX バージョン 1.1

- 非互換性 (マイナスの) 要件

マイナスの要件は、本製品と同じシステムにインストールする必要のない製品を識別します。Net.Data には、マイナスの要件はありません。

- DASD 記憶要件

Net.Data ライブラリーは、3380 あるいは 3390 DASD に常駐することができます。78ページの表 6 は、ライブラリーのタイプごとに必要なスペースの合計をリストしています。

表 6. DASD 記憶要件

ライブラリーのタイプ	必要なスペースの合計
ターゲット	110
配布	1040
HFS	45827

データ・セット、ライブラリー、ディレクトリー、およびディレクトリー記憶要件の詳細な評価については、“Net.Data for OS/390 バージョン 2 プログラム資料説明書”を参照してください。

削除される FMID

Net.Data をインストールすると、78ページの表 7 にリストされている FMID が削除されます。

表 7. 削除される FMID

削除される FMID	FMID の削除	説明
H242210	H24C110	Net.Data バージョン 5 基本
H242210	H24C111	Net.Data バージョン 5 日本語

特別な考慮事項

Net.Data には、ターゲット・システムに対する特別な考慮事項はありません。

SMP/E を使った Net.Data のインストール

本節では、NetData のインストール方法と、その機能をインストールしてアクティブにするためのステップを踏んだ手順について説明します。

以下の考慮事項に注意してください。

- Net.Data をそれ自身の SMP/E 環境にインストールする場合は、SMPCSI および SMP/E 制御データ・セットの作成と初期化に関する SMP/E のマニュアルの指示を参照してください。
- サンプルのジョブは、インストール作業の幾つか、あるいはすべてを実行するのを支援するために提供されています。SMP/E のジョブは、SMP/E の実行に必要なすべての DDDEF エントリーが、適切なゾーンで定義されていることを前提としています。
- サンプルのジョブの代わりに、SMP/E のダイアログを使用しても、SMP/E のインストールのステップを完了することができます。

インストールの概説

Net.Data をインストールする前に、以下の概説情報を再検討します。

Net.Data のインストールのための SMP/E の考慮事項

SMP/E の RECEIVE、APPLY、および ACCEPT コマンドを使用して、Net.Data をインストールします。別の方法としては、SMP/E のダイアログを使用して、SMP/E のインストールのステップを完了することができます。

SMP/E 環境

すべての Net.Data SMP/E のインストール・ジョブは、SMP/E の実行に必要なすべての DD ステートメントが、DDDEF を使用して定義されることを前提としています。

サンプルのジョブは、Net.Data のインストールを支援するために提供されています。RECEIVE ステップの完了後、サンプルのジョブを、SMPTLIB:IBM.H242210.F1 に見い出すことができます。これらのジョブをユーザー自身のライブラリーにコピーし、それらを変更して、Net.Data のインストール中に使用します。サンプルのジョブには、以下のものがあります。

DTWALA	新規の SMP/E CSI を割り当てるためのサンプルのジョブ (オプション)
DTWALB	SMP/E のデータ・セットを割り当て、CSI ゾーンを初期化するためのサンプルのジョブ (オプション)
DTWALC	基本ターゲットおよび配布ライブラリーを割り当てるためのサンプルのジョブ
DTWALJ	日本語ターゲットおよび配布ライブラリーを割り当てるためのサンプルのジョブ
DTWALK	韓国語の配布ライブラリーを割り当てるためのサンプルのジョブ
DTWDDEF	SMP/E の DDDEF を定義するためのサンプルのジョブ
DTWMKDIR	階層ファイル・システム構造を作成するためのサンプル・ジョブ
DTWREC	サンプルの RECEIVE ジョブ
DTWAPPLY	サンプルの APPLY ジョブ
DTWACPT	サンプルの ACCEPT ジョブ

サンプルの SMP/E ジョブでは、

- SMP/E CSI の名前は、hlqual.SMPCSi.CSI となる。
- SMP/E CSI におけるグローバル・ゾーンの名前は、GLOBAL となる。
- 配布ゾーンの名前は、distlib となる。
- ターゲット・ゾーンの名前は、targlib となる。

サンプル・ジョブを更新して、インストール時に使用する CSI およびゾーンの名前を反映させなければなりません。

SMP/E のオプションのサブエントリーの値

SMP/E CSI の幾つかのサブエントリーの推奨値は、80ページの表 8 に示されています。これらよりも低い値を使用すると、インストール処理に失敗する場合があります。DSSPACE は、GLOBAL オプションのエントリーのサブエントリーです。PEMAX は、GLOBAL オプションのエントリーの GENERAL エントリーのサブエントリーです。グローバル・ゾーンの更新に関する指示については、SMP/E のマニュアルを参照してください。

表 8. SMP/E のサブエントリーの値

サブエントリー	値	コメント
DSSPACE	(200,200,500)	3390 DASD トラック
PEMAX	5500	SMP/E のデフォルトは、ここで指定できるものよりも大きくなります。

SMP/E の CALLIBS 処理

Net.Data は、インストール中の外部参照を解決するために、SMP/E のリリース 8 で提供される CALLLIBS 関数を使用します。

Net.Data が初期化される場合には、

- SMP/E SMPLTS データ・セットが割り当てられていることを検証する。SMPLTS データ・セットの割り当てに関する情報については、“SMP/E 解説書”を参照してください。

Net.Data の場合、SMPLTS には、3380/3390 の DASD のスペースのうち、120 シリンダーで十分です。

- SCEELKED ライブラリーに DDDEF を提供します。(この DDDEF が使用されるのは、CALLLIBS を使用して、Net.Data のリンク・エディットを解決する場合だけです。) このデータ・セットは、Net.Data のインストール中は更新されません。

インストール・ステップの概説

Net.Data のインストールは、以下のステップから構成されます。

81ページの『ステップ 1: サンプルの JCL を、製品テープからアンロードする』

81ページの『ステップ 2: 希望する割り当てジョブの更新と実行』

84ページの『ステップ 3: 階層ファイル・システム構造の作成』

84ページの『ステップ 4: SMP/E の受信ジョブの更新と実行』

85ページの『ステップ 5: SMP/E の APPLY CHECK および APPLY Job の更新と実行』

85ページの『ステップ 6: SMP/E の ACCEPT CHECK および ACCEPT Job の更新と実行』

ステップ 1: サンプルの JCL を、製品テープからアンロードする

サンプルのインストール・ジョブは、配布テープで提供され、Net.Data のインストールを支援してくれます。81ページの図7のサンプルの JCL は、テープから Net.Data のジョブをコピーします。

ジョブ・カードを追加し、太文字のパラメーターを英大文字の値に変更して、JCL の処理依頼の前に、ユーザー・サイトの要件を満足します。

```
//STEP1 EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=A
//IN DD DSN=IBM.H242210.F1,UNIT=tunit,VOL=SER=242210,
// LABEL=(2,SL),DISP=(OLD,KEEP)
//OUT DD DSN=DTW.V2R1M0.INSTALL,
// DISP=(NEW,CATLG,DELETE),
// VOL=SER=dasdvol,UNIT=dunit,
// DCB=*.STEP1.IN,SPACE=(blksiz, (primary,secondary,dir))
//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN DD *
COPY INDD=IN,OUTDD=OUT
SELECT MEMBER=(DTWALC,DTWDEF,DTWALA,DTWALB)
SELECT MEMBER=(DTWREC,DTWAPPLY,DTWACPT)
SELECT MEMBER=(DTWMKDIR,DTWALJ,DTWALK)
/*
```

図7. Net.Data のジョブをテープからコピーするためのサンプルの JCL

上で示された JCL において、

- *tunit* は、製品のテープあるいはカートリッジと一致する装置の値。
- **DTW.V2R1M0.INSTALL** は、サンプルのジョブが常駐するデータ・セットの名前。
- *dasdvol* は、データ・セットを常駐させる DASD デバイスのボリューム通し番号。
- *dunit* は、DASD のボリュームのユニット・タイプです。

以下の方法によって、サンプルのインストール・ジョブにアクセスすることもできます。

1. FMID の *fmid* に対して SMP/E の RECEIVE を実行する
2. データ・セット **hlq.IBM.H242210.F1** から、編集と処理依頼のための作業データ・セットにジョブをコピーする

ステップ 2: 希望する割り当てジョブの更新と実行

以下の節では、更新および実行に必要な割り当てについて説明します。割り当てジョブの幾つかは、オプションです。

- 新規の **SMP/E CSI** を割り当てます(オプション)

このインストールのために、SMP/E の新規のデータ・セットを割り当てる場合は、**DTWALA** を変更して実行し、SMP/E CSI のデータ・セットの割り当てと準備を行います。

インストール・ジョブ・ストリームのコメント・セクションの注は、必ず読んでおいてください。

ジョブ・カードを追加し、ジョブの処理依頼をします。

予想される戻りコードとメッセージ： このジョブを完了させるには、戻りコードは 0 でなければなりません。

- **SMP/E のデータ・セットの割り当てとCSI ゾーンの初期化 (オプション)**

ジョブ **DTWALB** を変更後、これを実行し、SMP/E のデータ・セットの割り当てとカタログ化を行い、CSI ゾーンを初期化します。

既存の SMP/E CSI を使用している場合は、**DTWALB** のメンバーに、以下の変更を加えます。

- NEW,CATLG となっている部分をすべて、OLD に変更する。
- SMPCNTL の入力ステートメントをすべて削除し、以下と置き換える。

```
SET BDY(GLOBAL).  
UCLIN .  
  ADD GLOBALZONE  
    FMID(H242210).  
ENDUCL.
```

インストール・ジョブ・ストリームのコメント・セクションの注は、必ず読んでおいてください。

予想される戻りコードとメッセージ： このジョブを完了させるには、戻りコードは 0 でなければなりません。

- **配布およびターゲット・ライブラリーの割り当て - 基本**

テープ上でのインストール・ジョブはすべて、私用ライブラリーを参照します。システム・ライブラリーをターゲット・ライブラリーとして選択する場合は、インストール・ジョブを変更して、システム・ライブラリーを参照するようにしなければなりません。

ジョブ **DTWALC** を変更して実行し、Net.Data のための SMP/E のターゲットおよび配布ライブラリーを割り当てます。

インストール・ジョブ・ストリームのコメント・セクションの注は、必ず読んでおいてください。

ジョブ・カードを追加し、ジョブの処理依頼をします。

予想される戻りコードとメッセージ： このジョブを完了させるには、戻りコードは 0 でなければなりません。

- **Net.Data の NLV の配布およびターゲット・ライブラリーの割り当て**

テープ上でのインストール・ジョブはすべて、私用ライブラリーを参照します。日本語製品を使用している場合は、**DTWALJ** を使用します。韓国語製品を使用している場合は、**DTWALK** を使用します。システム・ライブラリーをターゲット・ライブラリーとして選択する場合は、インストール・ジョブを変更して、システム・ライブラリーを参照するようにしなければなりません。

インストール・ジョブ・ストリームのコメント・セクションの注は、必ず読んでおいてください。

ジョブ・カードを追加し、ジョブの処理依頼をします。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

• HFS データ・セットの割り当て - オプション

ルートOfFile・システムにインストールするのではなく、別の HFS データ・セットに製品をインストールするつもりならば、83ページの図 8 の JCL を使用して、HFS のデータ・セットを割り当てます。

```
/******  
/* JOB NAME = DTWHFS */  
/* */  
/* DESCRIPTIVE NAME = ALLOCATE HFS DATA SET */  
/* */  
/* STATUS = VERSION 02 RELEASE 01 MODIFICATION LEVEL 00 */  
/* */  
/* FUNCTION = TARGET AND DISTRIBUTION DATA SET ALLOCATIONS */  
/* */  
/* LICENSED MATERIALS - PROPERTY OF IBM */  
/* 5655-DB2 (C) COPYRIGHT IBM CORP 1997, 1998 */  
/* ALL RIGHTS RESERVED. */  
/* US GOVERNMENT USERS RESTRICTED RIGHTS - */  
/* USE, DUPLICATION OR DISCLOSURE RESTRICTED */  
/* BY GSA ADP SCHEDULE CONTRACT WITH IBM CORP. */  
/* */  
/* NOTES = */  
/* */  
/******  
//DTWHFS PROC HFSPRE='DTW.Version 2R1M0',HFSVOL=OP1HFS,  
// HFSUNIT=3390,HFSSTOR=ALL  
//IEFBR14 EXEC PGM=IEFBR14  
/*  
/* NOTE: THIS HFS DATA SET MUST BE DFSMS MANAGED .  
/*  
//SDTWHFS DD DSN=&HFSPRE..SDTWHFS,DISP=(NEW,CATLG),  
// VOL=SER=&HFSVOL,UNIT=&HFSUNIT,  
// SPACE=(512,(39850,4000,1)),  
// DSNTYPE=HFS,STORCLAS=&HFSSTOR  
/******  
//DTWHFS PEND  
//ALLOCATE EXEC DTWHFS  
/*
```

図 8. HFS のデータ・セットを割り当てするための JCL

ジョブ・カードを追加し、ジョブの処理依頼をします。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

• Net.Data のためのターゲットおよび配布ターゲット・ライブラリーの DDDEF エントリーの定義

メンバー名が **DTWDDDEF** のサンプルのジョブを編集後、実行します。このジョブは、**DTW.V2R1M0.INSTALL** ジョブで定義されるデータ・セットにあります。このジョブは、Net.Data のデータ・セットをユーザーの SMP/E システムに定義するのに役に立ちます。

インストール・ジョブ・ストリームのコメント・セクションの注は、必ず読んでおいてください。

重要：通常、JCL はすべて、英大文字でなければなりません、HFS のパス名は、大文字と小文字を混合して使用することができます。

ジョブ・カードを追加し、ジョブの処理依頼をします。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

ステップ 3: 階層ファイル・システム構造の作成

SMP/E が Net.Data をインストールできるようになるには、その前に HFS にディレクトリーのスケルトンを持っていなければなりません。

TSO の MKDIR コマンドを使用して、以下のディレクトリーを作成します。

```
TSO MKDIR '/usr/lpp/netdata' MODE (7 5 5)
```

/usr/lpp/netdata にインストールする場合は、ルート権限を使用して実行する必要があります。

重要：MODE(7 5 5) では、スペースを保持してください。

オプション：ユーザーの製品を、ルートのファイル・システムではなく、別の HFS のデータ・セットにインストールする場合には、以下のコマンドを使用して、作成された新規の HFS をマウント・ポイントにマウントします。(コマンドは、1 行になければなりません。)

```
TSO MOUNT FILESYSTEM('dtw.v2r1m0.sdtwhfs')  
MOUNTPOINT('/usr/lpp/netdata') TYPE(HFS)
```

サンプル・ジョブ **DTWMKDIR** は、**DTW.V2R1M0.INSTALL** にあり、スケルトンを作成します。**DTWMKDIR** は、デフォルトのディレクトリー /usr/lpp/netdata にインストールする場合は、ルート権限で実行されなければなりません。

別のディレクトリー構造あるいは名前をインストールに使用する場合は、それに対応して、提供されるデータ・セットを編集します。

インストール・ジョブ・ストリームのコメント・セクションの注は、必ず読んでおいてください。

ジョブ・カードを追加し、ジョブの処理依頼をします。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

ステップ 4: SMP/E の受信ジョブの更新と実行

- データ・セット **DTW.V2R1M0.INSTALL** に進み、メンバー **DTWREC** を編集します。必要な変更を加え、**DTWREC** メンバーを実行します。

インストール・ジョブ・ストリームのコメント・セクションの注は、必ず読んでおいてください。

ジョブ・カードを追加し、ジョブの処理依頼をします。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

- SMP/E を使用して、累積サービス・テープを受け取ります。CBPDO から Net.Data を受け取る場合は、このステップを実行する必要はありません。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

ステップ 5: SMP/E の APPLY CHECK および APPLY Job の更新と実行

以下の 2 つのステップでは、Apply Check および Apply Job の更新および実行方法について説明しています。

- **SMP/E の APPLY CHECK の実行**

Net.Data が、ディレクトリー /usr/lpp/netdata にインストールされる場合は、**SMP/E DTWAPPLY** ジョブは、UID の値を 0 (ゼロ) にしたユーザー ID で実行しなければなりません。

サンプル・ジョブ **DTWAPPLY** を編集後、処理依頼をして、Net.Data に対して SMP/E の APPLY CHECK を実行します。詳しくは、サンプル・ジョブの指示を参考にしてください。

APPLY での CHECK 処理の場合、CHECK オペランドを APPLY ステートメントに追加します。

```
APPLY SELECT(H242210)
CHECK GROUPEXTEND.
```

SMP/E の Causer SYSMOD 要約報告を全面的に享受するには、APPLY CHECK の PRE、ID、REQ、および IFREQ をバイパスしないようにします。この理由は、SMP/E のルート原因解析は、エラーの原因のみを識別し、警告の原因は識別しないからです。(バイパスされる SYSMOD は、SMP/E では、警告として処理され、エラーとしては処理されません。)

GROUPEXTEND オペランドは、SMP/E は、必要な SYSMOD をすべて適用することを示しています。必要な SYSMODS は、他の関数に適用可能な場合があります。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

- **SMP/E の APPLY の実行**

サンプル・ジョブ **DTWAPPLY** から CHECK オペランドを削除し、サンプル・ジョブの処理依頼をして、Net.Data に対して、SMP/E の APPLY を実行します。詳しくは、サンプル・ジョブの指示を参考にしてください。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

ステップ 6: SMP/E の ACCEPT CHECK および ACCEPT Job の更新と実行

以下の 2 つのステップでは、ACCEPT CHECK および ACCEPT Job の更新および実行方法について説明しています。

• SMP/E ACCEPT CHECK の実行K

サンプル・ジョブ **DTWACPT** を編集し、処理依頼をして、Net.Data に対して SMP/E の ACCEPT CHECK を実行します。詳しくは、サンプル・ジョブの指示を参考にしてください。

ACCEPT における CHECK 処理の場合、CHECK オペランドを ACCEPT ステートメントに追加します。

```
ACCEPT SELECT(H242210)
CHECK GROUPEXTEND.
```

SMP/E の Causer SYSMOD 要約報告を全面的に享受するには、ACCEPT CHECK の PRE、ID、REQ、および IFREQ をバイパスしないようにします。この理由は、SMP/E のルート・原因解析は、エラーの原因のみを識別し、警告の原因は識別しないからです。(バイパスされる SYSMOD は、SMP/E では、警告として処理され、エラーとしては処理されません。)

GROUPEXTEND オペランドは、SMP/E は、必要な SYSMOD をすべて受け取ることを示しています。必要な SYSMODS は、他の関数に適用可能な場合があります。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

• SMP/E の ACCEPT の実行

サンプル・ジョブ **DTWACPT** から CHECK オペランドを削除し Net.Data に対して、SMP/E の ACCEPT を実行します。詳しくは、サンプル・ジョブの指示を参考にしてください。

推奨：SMP/E を使用して新規の配布ライブラリーをロードする前に、配布ゾーンに ACCJCLIN インディケーターを設定します。これにより、インラインの JCLIN を含む SYSMOD で ACCEPT が実行されるたびに、JCLIN から作成されたエントリが配布ゾーンに保管されます。ACCJCLIN インディケーターについての詳細は、SMP/E のマニュアルの インライン JCLIN の説明を参照してください。

置き換えモジュールを含む PTF 上で ACCEPT が実行される場合は、SMP/E の ACCEPT 処理は、モジュールをリンク編集し、配布ライブラリーにバインドします。この処理中、リンケージ・エディターあるいはバインダーは、未解決の外部参照を文書化したメッセージを発行し、ACCEPT のステップから 戻りコード 4 を戻すことがあります。これらのメッセージは無視してもかまいません。その理由は、配布ライブラリーは、実行可能ではなく、未解決の外部参照は、実行可能なシステム・ライブラリーには影響を与えないからです。

予想される戻りコードとメッセージ：このジョブを完了させるには、戻りコードは 0 でなければなりません。

インストールの検証

インストールの検証に先立って、5ページの『第2章 Net.Data のインストールおよび構成』における以下のプロシーチャーを完了させ、そして89ページの『付録B. Net.Data for OS/390 を構成し、DataJoiner をアクセスする』におけるプロシーチャーをオプションとして完了させます。

- 6ページの『Net.Data の初期設定ファイルのインストール』

- 6ページの『Net.Data の初期設定ファイルのカスタマイズ』
- 17ページの『作業負荷管理の考慮事項』
- 17ページの『CGF と一緒に使用する Net.Data の構成』
- 19ページの『ICAPI あるいは GWAPI と一緒に使用するための Net.Data の構成』
- 20ページの『言語環境のセットアップ』
- 22ページの『メッセージ・カタログを使用可能にする』
- 22ページの『Net.Data のファイルおよびデータ・セットへのアクセス権の指定』

Net.Data for OS/390 には、サンプル・マクロおよびデータが付属しています。そのため、インストールがスムーズに行われたときに、インストールを検証することができます。検証プロシージャについては、“*Net.Data for OS/390 バージョン 2 プログラム資料説明書*” を参照してください。

付録B. Net.Data for OS/390 を構成し、DataJoiner をアクセスする

Net.Data for OS/390 と DataJoiner を使用して、DB2/6000、Oracle、および Sybase などのリモート・データベースをアクセスすることができます。このセクションでは、DataJoiner for AIX バージョン 1.2 PTF U447593 または DataJoiner for HP-UX バージョン 1.1 を使用して、システムの構成方法を説明しています。

構成の各ステップ:

1. DataJoiner との遠隔通信に、通信データベース (CDB) で必要な情報を入力します。CDB 内の情報は、次のガイドに記述されています。 *DB2 Installation Guide*.
2. DataJoiner がインストールされているリモート・ロケーションに、BIND PACKAGE コマンドを使用して、Net.Data DBRM をバインドします。
3. BIND PLAN コマンドを使用して、Net.Data DBRM を DB2 にバインドします。PKLIST オプションを使用して、リモート・ロケーションで作成されたパッケージを組み込みます。
4. Net.Data 初期設定ファイルを変更し、SQL 関数への入力変数として LOCATION 変数を指定します。このファイルは Web サーバーのドキュメント・ルート・ディレクトリにあります。新規の DTW_SQL 環境ステートメントは以下のようになります。

```
ENVIRONMENT (DTW_SQL) dtwsq1 (IN LOCATION)
```

DataJoiner を使用してリモート・データをアクセスする Net.Data マクロは、LOCATION に値を指定する必要があります。次の Net.Data マクロの例では、DataJoiner を使用してリモート・データベースを照会します。

```
%{ ***** Define Block ***** %}  
%DEFINE {  
    DB2SSID="NDA1"  
    LOCATION="QMFDJ00"  
    DTW_DEFAULT_REPORT="YES"  
%}  
  
%{ ***** Function Definition Block ***** %}  
%FUNCTION(DTW_SQL) selectall() {  
    SELECT * FROM $(tabnam)  
%}  
  
%{ ***** HTML Block: Table_Input ***** %}  
%HTML(Table_Input) {  
<Title>DJ Test #1</Title>  
<Body>  
<h1 align=center>Table Selection</h1>  
<br>  
<form method="post" action="Column_Output">  
<p>Enter Table Name: <input type="text" name="tabnam"></p>  
<p><input type="submit"></p>  
</form>  
</Body>  
%}  
  
%{ ***** HTML Block: Column_Output ***** %}  
%HTML(Column_Output) {  
<Title>DJ Test #1</Title>  
<Body>
```



```
@selectAll()  
</Body>  
%}
```

付録C. Net.Data サンプル・マクロ

このサンプル・マクロ・アプリケーションでは、リストで社員の名前を選択して個々の社員の追加情報を得られるアプリケーションから、社員名のリストを表示します。このマクロは、SQL 言語環境を使用して、社員名および特定の社員に関する情報の EMPLOYEE 表を照会します。

```

%{***** Sample Macro *****)
*   FileName = sqlsamp1.d2w
*   Description:
*       This Net.Data macro file queries...
*       - The EMPLOYEE table to create a selection list of
*         employees for display at a browser
*       - The EMPLOYEE table to obtain additional information
*         about an individual employee
*
*****%}
%{*****
*   Include for global DEFINES -
*****%}
%INCLUDE "sqlsamp1.hti"
%{*****
*   Function: queryDB           Language Environment: SQL
*   Description: Queries the table designated by the variable myTable and
*   creates a selection list from the result. The value of the variable
*   myTable is specified in the include file sqlsamp1.hti.
*****%}
%FUNCTION(DTW_SQL) queryDB() {
  SELECT * FROM $(myTable)
%MESSAGE {
  -204: {<p><b>ERROR -204: Table $(myTable) not found. </b>
        <p>Be sure the correct include file is being used.</p>
        %} : exit
  +default: "WARNING $(RETURN_CODE)" : continue
  -default: "Unexpected ERROR $(RETURN_CODE)" : exit
%}

%REPORT {
<select name=emp_name>
%ROW{
<option>$(V2)
%}
</select>
%}
%}

%{*****
*   Function: fname           Language Environment: SQL
*   Description: Queries the table designated by the variable myTable for
*   additional information about the employee identified by the
*   variable emp_name.
*****%}
%FUNCTION(DTW_SQL) fname(){
SELECT EMPNME, PHONENO, JOB FROM $(myTable) WHERE EMPNME='$(emp_name)'
%MESSAGE {
  -204: "Error -204: Table not found "
  -104: "Error -104: Syntax error"
  100: "Warning 100: No records" : continue
  +default: "Warning $(RETURN_CODE)" : continue
  -default: "Unexpected SQL error" : exit
%}
%}

```

```

%{*****
*   HTML block: INPUT                               Title: Dynamic Query Selection
*
*   Description: Queries the EMPLOYEE table to create a selection list of
*               the employees for display at the browser
*   *****/}
%HTML(INPUT) {
<html><head><title>Generate Employee Selection List</title></head><body><h3>$(exampleTitle)</h3>
<p>This example queries a table and uses the result to create
a selection list using a <em>%REPORT</em> block.
< hr>
<form method="post" action="report">
@queryDB()<input type="submit" value="Select Employee">
</form>
< hr>
</body>
</html>
%}

%{*****
*   HTML block:   REPORT
*   Description:  Queries the EMPLOYEE table to obtain additional information
*               about an individual employee
*   *****/}
%HTML(REPORT) {
<html>
<head>
<title>Obtain Employee Information</title>
</head>
<body>
<h3>You selected employee name = $(emp_name)</h3>
<p>Here is the information for that employee:
<PRE>
@fname()
</PRE>
<hr><a href="input">Return to previous page</a>
</body>
</html>
%}

%{      End of Net.Data macro 1 %}
=====
%{***** Include File *****/}
*   FileName = sqlsamp1.hti
*   Description:
*   This include file provides global DEFINES for the sqlsamp1.d2w
*   Net.Data macro.
*   *****/}
%define {
    emp_name      = ""
    exampleTitle = "Sample Macro"
    myTable = "MRZ.EMPLOYEE"
%}

%{      End of include file %}

```


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

本書で参照される IBM 以外の Web サイトは、参考として掲載するものです。これらの Web サイトは IBM 社が認定しているものではありません。これら Web サイトの資料は、本 IBM 製品の資料の一部ではありません。これらの Web サイトの利用は、ユーザーの責任で行ってください。

(i) 独自に作成されたプログラムおよび (本プログラムを含む) その他のプログラムの間の情報交換、および (ii) 交換された情報の相互利用、を目的に、本プログラムの情報の入手を希望するライセンス所有者は、次の窓口までご連絡ください。

IBM Corporation

W92/H3

555 Bailey Avenue

P.O. Box 49023

San Jose, CA 95161-9023

_U.S.A.

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」または「IBM 海外使用契約書」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。

IBM の今後の方向性および予定に関する記述は、目標や目的を表明したものであり、予告なく変更または取り消しされることがあります。

この情報は、今後の計画の参考としてのみお取り扱いください。ここで記述された製品に関する情報は、製品化の前に変更されることがあります。

ここでの情報は、日々の業務で使用されるデータやレポートの例を含んでいます。これらの例は、なるべく実情に合うように、個人名、会社名、ブランド名、および製品名が使用されています。これらの名前はすべてフィクションであり、実際のビジネスの世界で使用されている名前と、偶発的に類似していることがあります。

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	IBM
AS/400	IMS
CBIDO	Language Environment
CBPDO	MVS/ESA
CICS	Net.Data
CustomPac	OpenEdition
DB2	OS/2
DB2 Universal Database	OS/390
DataJoiner	OS/400
Distributed Relational Database Architecture	RACF
DRDA	SystemPac

以下の用語は、米国またはその他の国における IBM Corporation の商標です。

Java および すべての Java ベースの商標とロゴは、米国ならびにその他の国における Sun Microsystems, Inc. の商標です。

Lotus および Domino Go Webserver は、米国ならびにその他の国における Lotus Development Corporation の商標です。

Microsoft、Windows、Windows NT、および Windows のロゴは、米国ならびにその他の国における Microsoft Corporation の商標あるいは登録商標です。

2 個のアスタリスク (**) で示されている他社名、製品名およびサービス名は、他社の商標またはサービス・マークです。

用語集

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)。

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 などの Java 対応のブラウザを処理し、HTML ページのロード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 (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。

CGI. コモン・ゲートウェイ・インターフェース (Common Gateway Interface)。

コモン・ゲートウェイ・インターフェース (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。

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)。

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。

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)。

ICS. インターネット接続サーバー (Internet Connection Server)。

ICSS. Internet Connection Secure Server。

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。

インターネット接続サーバー (Internet Connection Server). IBM の無保護 Web サーバー。

Internet Connection Secure Server. IBM の保護付き Web サーバー。

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。

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。言語環境によっては、REXX、Perl、および Oracle などの Net.Data と一緒に提供されるものもある。また、独自の言語環境を作成することもできる。

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。異機種混合のオペレーティング・システム全体で、異なるアプリケーション間の相互作用を管理する。 *Free Online Dictionary of Computing* より。

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。

Perl. 解釈済みプログラミング言語。

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。

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)。

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>。

Web サーバー (Web server). インターネット接続 (Internet Connection) などの HTTP サーバー・ソフトウェアを実行しているコンピューター。

索引

日本語, 英字, 数字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

アクセス権、Net.Data のファイルへの指定 22
暗号化、ネットワーク 26
インストール
 OS/390 上での 75
隠蔽変数
 資産の保護 27

[カ行]

開始、Net.Data の 29
隠し変数 50
各種変数 53
型、変数 48
環境変数 49
関数
 組み込み 67
 説明 55
 定義 55
 ユーザー定義 55
 呼び出し 57
 呼び出し、ストアド・プロシーチャーの 58
 FUNCTION ブロックの構文 55
関数の呼び出し 57
起動、Net.Data の
 概説 29
 構文 30
 直接要求 29
 フォーム 31, 36
 マクロ要求 29
 マクロ・ファイルで 30
 マクロ・ファイルを使用しないで 33
 リンク 31, 36
 CGI の使用 29
 GWAPI 73
 HTML ブロック 62
 ICAPI 73
 URL 31, 37
機密保護
 概説 25
 許可 27
 指定、アクセス権の 22
 認証 26
 ネットワーク暗号化 26

機密保護 (続き)
 ファイアウォール 25
 Net.Data のメカニズム 27
許可
 指定、Net.Data のファイルへのアクセス権の 22
許可、機密保護 27
空白文字、変数、不要文字を削除するための 10
組み込み関数 67
グローバルな識別子の効力範囲 45
言語環境 71
 構成、初期設定ファイル内の 15
 変数 54
 例 15
 ENVIRONMENT ステートメントの構成 15
 REXX、SQL、および ODBC のセットアップ 20
向上、パフォーマンスの 71
構成、DataJoiner 用 89
構成、Net.Data の
 概説 5
 作業負荷管理プログラム (WLM) 17
 初期設定ファイル
 更新 6
 構成変数ステートメント 7
 説明 5
 パス・ステートメント 11
 ENVIRONMENT ステートメント 15
 使用、ICAPI および GWAPI と 19
 接続管理 16
 メッセージ・カタログ 22
 CGI のための 17
 Net.Dataのファイルおよびデータ・セットへのアクセス権 22
 SQL、ODBC、および REXX のセットアップ 20
構成変数ステートメント
 構成、初期設定ファイル内の 7
 説明 7
 DB2MSGs 8
 DB2PLAN 8
 DB2SSID 9
 DefaultDBCp 9
 DSNAOINI 10
 DTW_MBMODE 10
 DTW_REMOVE_WS 10
効力範囲
 変数 45
 REPORT ブロック 45
効力範囲、識別子の
 グローバルな 45
 マクロ・ファイル 45

効力範囲、識別子の (続き)

FUNCTION ブロック 45

ROW ブロック 46

コモン・ゲートウェイ・インターフェース (Common Gateway Interface)。CGI を参照 17

[サ行]

サンプル・マクロ 91

実行可能な変数 49

初期設定ファイル

更新 6

構成変数ステートメント 7

説明 5

パス・ステートメント 11

フォーマット 6

ENVIRONMENT ステートメント 15

条件付き

変数 48

論理、IF ブロック 65

ストアード・プロシーチャー

有効なデータ型 58

呼び出し、マクロ・ファイルからの 58

接続管理

構成 16

作業負荷管理プログラムの考慮事項 16

宣言部分、マクロ・ファイル構造の 39

[タ行]

直接要求

構文 33

説明 29

例 36

データ型、ストアード・プロシーチャーに対して有効な 58

データ・セット、アクセス権 22

定義、変数の

DEFINE ステートメントまたはブロック 46

HTML のフォームの SELECT および INPUT タグ 47

URL データ 47

デフォルトのレポート、プリント 64

トークンのサイズ 44

特記事項 95

[ナ行]

ナビゲーション、マクロ内およびマクロ間の 43

認証、機密保護 26

ネイティブ言語サポート、関数のための 10

[ハ行]

パス・ステートメント

更新のガイドライン 12

パス・ステートメント (続き)

構成、初期設定ファイル内の 11

資産の保護 27

EXEC_PATH 13

FFI_PATH 14

INCLUDE_PATH 13

MACRO_PATH 12

パフォーマンス 71

SQLCODE メッセージ 73

Web サーバー API 73

表処理変数 53

表変数 52

ファイアウォール 25

ファイル、Net.Dataへのアクセス権の指定 22

フォーマット、データ出力の 63

フォーム

Net.Data の起動 31, 36

Net.Data を起動するための Web ページ内の 32

フッター情報、REPORT ブロックの 63

部分、マクロ・ファイルの

宣言 39

HTML 39

ブランク、変数、不要文字を削除するための 10

プリント、デフォルトのレポートの禁止 64

プログラム資料説明書、OS/390 89

プログラム資料説明書、OS/390 の 75

ブロック、マクロ・ファイルの 41

ヘッダー情報、REPORT ブロックの 63

変数

隠し 50

各種 53

型 44, 48

環境 49

言語環境 54

構成、ステートメントの

削除、不要なブランクの (DTW_REMOVE_WS) 10

初期設定ファイル 7

説明 7

データベースのコード・ページ変数

(DefaultNetCp) 9

ネイティブ言語サポート (DTW_MBMODE) 10

DB2 CLI の初期設定ファイル (DSNAOINI) 10

DB2 Subsystem ID (DB2SSID) 9

DB2 のプラン変数 (DB2PLAN) 8

DB2 のメッセージのパフォーマンス変数 (DB2MSGs) 8

効力範囲 45

参照 47

実行可能 49

条件付き 48

説明 44

変数 (続き)
 定義 46
 トークンのサイズ 44
 表の処理 53
 リスト 51
 レポート 54
 table 52
変数参照、処理順 57
変数の参照 47
保護、資産の 25

[マ行]

マクロ要求
 構文 30
 説明 29
 例 30
マクロ・ファイル
 開発 39
 関数 55
 サンプル 40
 識別子の効力範囲 45
 条件付き論理 65
 生成、HTML の 62
 説明 1
 宣言部分 39
 内部およびその間のナビゲーション 43
 ブロック 41
 分析 40
 変数 44
 ループ 66
 DEFINE ブロック 41
 FUNCTION ブロック 42
 HTML
 部分 39
 ブロック 43
 IF ブロック 65
 WHILE ブロック 66
メッセージ・カタログ、使用可能にする 22

[ヤ行]

ユーザー定義関数 55
用語集 96
呼び出し、関数の
 構文 57
 処理順序 57
呼び出し、ストアード・プロシージャの 58

[ラ行]

リスト変数 51

リンク
 Net.Data の起動 31, 36
 Net.Data を起動するための Web ページ内の 31
ループ、WHILE ブロックの 66
レポートのフォーマット、カスタマイズ 64
レポート変数 54

C

CGI、Net.Data for OS/390 の構成 17

D

DB2MSG 8, 73
DB2PLAN 8
DB2SSID 9
DBCS サポート、関数のための 10
DEFINE ステートメントまたはブロック、変数の定義 46
DEFINE ブロック 41
DSNAOINI 10
DTW_MBMODE 10
DTW_REMOVE_WS 10

E

ENVIRONMENT ステートメント
 言語環境のタイプ 15
 構成、初期設定ファイル内の 15
 構文 15
 説明 15
 例 16
 DLL あるいはライブラリー名 15
 parameter list 15
EXEC_PATH、初期設定ファイル内の構成 13

F

FFI_PATH、初期設定ファイル内の構成 14
FUNCTION ブロック
 関数の呼び出し 57
 識別子の効力範囲 45
 処理、関数呼び出しの 57
 説明 42
 フォーマット、出力の 63
 変数参照の処理 57

G

GWAPI
 構成、Net.Data の 19
 Net.Data の起動 73

H

HTML

- 生成、マクロ・ファイル内に 62
- タグ、表の 64
- 認識されていないデータ 63
- フォーム
 - について 32
 - Net.Data の起動 31, 36
 - SELECT および INPUT タグ、変数定義 47
- 部分、マクロ・ファイル構造の 39
- ブロック
 - 起動、Net.Data の 62
 - 処理 63
 - 説明 43
 - 例 62
- リンク
 - について 31
 - Net.Data の起動 31, 36
- FORM の処理依頼ボタン 63
- URL、Net.Data の起動 37

I

ICAPI

- および Domino Go Webserver (GWAPI) 19
- 構成、Net.Data の 19
- Net.Data の起動 73
- IF ブロック 65
- IN パラメーター 58
- INCLUDE_PATH、初期設定ファイル内の構成 13
- INOUT パラメーター 58

M

- MACRO_PATH、初期設定ファイル内の構成 12
- MBCS サポート、関数のための 10
- MESSAGE ブロック
 - 構文 60
 - 効力範囲 60
 - 処理 60
 - 説明 60
 - 例 61

N

Net.Data

- インストール 5
 - OS/390 75
- 概説 1
- 起動 29
- 機密保護のメカニズム 27
- 構成 5

Net.Data (続き)

- ファイル、アクセス権 22
- マクロ・ファイル、開発 39
- OS/390 のインストール 89

Net.Data のインストール 5

Net.Data マクロ。マクロ・ファイルを参照してください。 1

O

- ODBC、言語環境のセットアップ 21
- OS/390、Net.Data 75
- OS/390、Net.Data for 89
- OUT パラメーター 58

R

REPORT ブロック

- 効力範囲 45
- 説明 63
- フォーマット、データ出力の 63
- ヘッダーおよびフッター情報 63

RETURNS パラメーター 58

RETURN_CODE 変数 58, 60

REXX、言語環境のセットアップ 21

ROW ブロック、識別子の効力範囲の 46

S

- SQLCODE メッセージ、取り消し 73
- SQL、言語環境のセットアップ 21

U

URL

- 定義、変数の 47
- Net.Data の起動 31, 37

W

Web サーバー

- 構成
 - CGI のための 17
 - 構成、ICAPI および GWAPI の 19
 - 設定、メッセージ・カタログのための環境変数の 22

Web サーバー API

- 考慮事項 73
- によるパフォーマンスの向上 73
- Net.Data の起動
 - GWAPI 73
 - ICAPI 73

Web サーバーの API

- 構成、Net.Data の
 - GWAPI 19

Web サーバーの API (続き)

構成、Net.Data の (続き)

ICAPI 19

WHILE ブロック 66



Printed in Japan