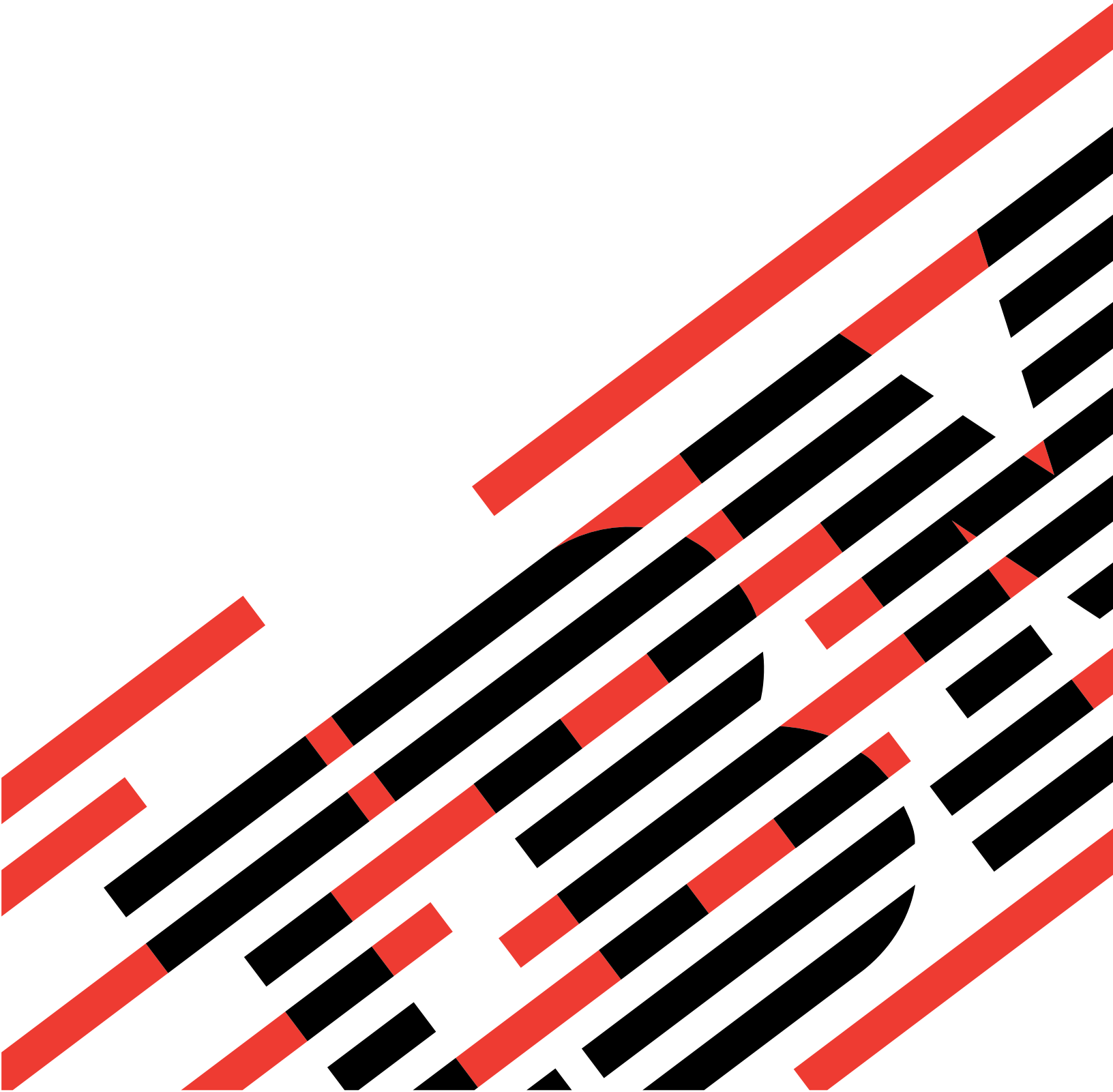
IBM

eserver

iSeries

グローバリゼーション
(グローバル・アプリケーションの開発)



IBM

@server

iSeries

グローバリゼーション
(グローバル・アプリケーションの開発)

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

http://www.ibm.com/jp/manuals/main/mail.html

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは

http://www.ibm.com/jp/manuals/ の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

原　典： RBAG–S000–01
iSeries
Globalization (Develop global applications)

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 2002.8

目次

OS/400 のグローバリゼーション

企業は世界的な規模で e-commerce と基本ビジネス・プロセスとの統合を推進しており、将来見込まれるカスタマー、既存のカスタマー、現行のパートナーは、ソフトウェア・グローバリゼーションを通して、収入の増加や支出の削減を図ることができます。また、カスタマーとのコミュニケーションの改善や、収益の増加も期待できます。グローバル・ソフトウェアにより、次のような利点が得られます。

- お客様の満足度が高まり、売上が伸びる
- カスタマー・サポートのコミュニケーションが良くなる。
- 全世界に広く情報を普及できる。
- 情報技術 (IT) への投資収益が高まる。

このトピックの目的は、次のとおりです。

- アプリケーションを効率的に最小の費用で作成する。
- 既存のアプリケーションをグローバリゼーションに対応したものに改造する、またはグローバリゼーションに対応した新たなアプリケーションを作成する。ただし、グローバリゼーション・アプリケーションを作成する場合は、既存アプリケーションを改造するよりも、新たなアプリケーションを設計する方が通常は費用が安くなります。
- 設計するアプリケーションが現行または計画中のほかの国際化対応アプリケーションに干渉しないようにする。

ここには、国内向けおよび海外向けのアプリケーションを作成するのに必要な情報をまとめてあります。このリリースの 新機能、および トピックの印刷 の方法も説明しています。

グローバリゼーションの概要
OS/400(R) にグローバリゼーションがどのように実装されているかについて説明します。システム上のグローバリゼーションに固有の値を説明するトピック、および OS/400 のサービスと機能がグローバリゼーションをどのようにサポートするかを説明するトピックが含まれています。

各国語バージョンを使用する OS/400 のセットアップ
OS/400 上に各国語バージョンを正しくインストールし、構成するために必要なステップについて説明します。ハードウェアの選択とインストール、ソフトウェアのインストール、およびグローバル設定で実行する環境を構成する方法を説明するトピックが含まれています。この情報は、ユーザー自身のサーバーをインストールする際にご使用いただけるだけでなく、それぞれ独自の言語バージョンを OS/400 にインストールするカスタマー向けのアプリケーションを開発する際にもこの原則を適用できます。

グローバル・アプリケーションの開発
グローバル・アプリケーションを設計、開発、納入するためのガイドラインを示します。

- 各機能を各国の言語に対応させる。
- さまざまなハードウェアをサポートする。
- アプリケーションに使用するテキスト・データを翻訳する。
- アプリケーションを世界中で使用できるようにする。

グローバル・アプリケーションのデータの処理
OS/400 で使用可能なグローバル環境でのデータの処理方法について説明します。ユニコードと

UCS-2 データの説明、中国語規格 GB18030、複数言語環境を一貫して統合するための CCSID の使用方法、両方向データ、DBCS データ、およびロケールの使用方法などのトピックが含まれています。

グローバリゼーションの参照情報
グローバリゼーション・カテゴリーで説明した概念とタスクに関する詳細なサポート情報を提供します。

グローバリゼーション・チェックリスト
グローバリゼーションに関連したすべてのチェックリストをまとめてあります。このチェックリストは、ユーザーがグローバル・アプリケーションを作成および処理するときに考慮する必要がある問題を確認するのに役立ちます。

グローバル・アプリケーションの開発

グローバル・アプリケーションとは、各国語サポートのあるアプリケーションのことです。各国語サポートでは、ユーザーが選択する言語で、データの入力、保存、処理、読み取り、印刷、および表示ができます。さらに、各国語サポートは、データ、コマンド、プロンプト、メッセージ、および文書資料について、ユーザーが選択する言語で、またそれぞれの文化に応じた形態で、表示や入力ができます。

異なる理由の場合もありますが、ほとんどの国際化対応アプリケーションは、次のような理由で作成します。

- 市場がローカル性のあるグローバル・ソフトウェア製品を求めている
- アプリケーションが複数の国の社会で使用される
- 収益チャンスを拡大する

次のリンクには、開発作業を開始する前に知っておく必要がある重要な情報を提供しています。

- グローバル・アプリケーションの開発: 目的と作業
- グローバル・アプリケーションの設計
- グローバル・アプリケーション設計のプログラミングに関する考慮事項
- グローバル・アプリケーションの納入

関連情報

グローバル環境でさまざまなタイプのデータを処理する方法については、グローバル・アプリケーションのデータの処理 を参照してください。

目的と作業

グローバル・アプリケーションの開発に時間と資金を投資する前に、計画を立てる段階で、グローバル・ユーザーに効率的かつ効果的に対応する方法を検討すると有益です。　以下のトピックは、計画を作成するのに役立ちます。

- 開発の目的
- マーケット・リサーチ作業
- 開発作業
- 文書作成
- 翻訳作業
- テスト作業

- パッケージングとインストール作業
- アプリケーション保守作業

グローバリゼーション開発の目的

国際化対応アプリケーションを計画または作成する場合は、このトピックを使用してください。このトピックの推奨項目では、以下のような基本目的を前提とします。

- アプリケーションを効率よく作成する。
- アプリケーションの作成費用を最小限にとどめる。既存のアプリケーションをグローバリゼーションに対応したものに改造する、またはグローバリゼーションに対応した新たなアプリケーションを作成する。ただし、グローバリゼーション・アプリケーションを作成する場合は、既存アプリケーションを改造するよりも、新たなアプリケーションを設計する方が通常は費用が安くなります。
- 設計するアプリケーションが、現行または計画中のほかの国際化対応アプリケーションに干渉しないようにする。
- 各国語サポートのアプリケーションを作成するときは、次のタスクを計画または実行する必要があります。
  - 各機能を各国の言語に対応させる。
  - さまざまなハードウェアをサポートする。
  - アプリケーションに使用するテキスト・データを翻訳する。
  - アプリケーションを世界中で使用できるようにする。

グローバリゼーション開発の計画作業

グローバル・アプリケーションの開発には、時間、労力、および費用を節約するために、各段階で十分な計画が必要です。プログラムを再コンパイルしたり、データ・オブジェクトを再パッケージすることは避けてください。使用する言語バージョンによっては、プロダクトに異なるデータ・オブジェクトが必要になります。1 組のプログラム・コードと、必要に応じて、複数の文化依存コードやテキスト依存コードが必要になります。

グローバル・アプリケーションを計画するときには、次の作業を考慮してください。

- マーケット・リサーチ
- 開発
- 資料作成
- 翻訳
- テスト
- パッケージングとインストール
- アプリケーションの保守

マーケット・リサーチ作業

どのような決定を行うときでも、常に重要なことは、アプリケーションの設計や開発を行う対象が誰であるかを理解することです。この質問に答えるために、自分自身と潜在顧客に対して次の質問をしてください。

現在および未来の目標市場は?

目標市場を複数の国とするか、1 言語の地域だけとするか、あるいは他の言語を使用する国も対象とするかによって、この質問の答が大きな違いを生みます。たとえば、ローマ字系の言語でアプリケーションをコーディングすると、ヘブライ語、中国語、日本語などのローマ字以外の言語を使用する国を対象としたとき

に、アプリケーションが複雑になります。 アプリケーションが複雑になる理由は、互換性のない文字セットやより複雑な入力方式の処理が必要になるからです。

言語の問題以外にも、考慮すべきことがあります。 目標市場の文化、風習、業務習慣、および法規制を理解する必要があります。ビジネス・パートナーとして受け入れてもらい、顧客の国で市場に参入して顧客をサポートするためには、顧客の生活様式を理解する必要があります。

次のようなことで影響を受けます。

- 必要技能 (技術、文化、言語、法律)
- 考慮すべき環境
- 自社の構造とサポート体制
- 他企業との関係
- 必要リソース (要員、時間、資金)

アプリケーションを使用するユーザーの要件は?

作成するアプリケーションを使用するユーザーの要件を理解する必要があります。たとえば、次のようなことが考えられます。

- 複数の言語を使用して、個別のデータベースを処理する。
- すべての言語を対象に、共用データベースを処理する。
- データの交換、あるいは統合。
- エンド・ユーザー、会社、会社の顧客に応じて、複数の言語を使用する。
- エンド・ユーザーのデータベース・ツールを使用して、アプリケーション・データベースの照会をする。

こうしたことは、言語の切り替え方式、データの表示方式、あるいはデータの変換方式など、アプリケーションの設計に影響します。

どの程度のグローバリゼーション・サポートが必要か?

顧客やそのエンド・ユーザーの要件が分かれば、保存や管理が必要な国別情報の内容をはじめ、データの表示方法、翻訳すべき部分、異なる環境とのアプリケーションの統合性を決定できます。

費用はどのくらい必要か。

予想収益を見積もるには、目標市場として選択した場所を分析します。要件が分かれば、作業量や費用も分かるはずです。この金額を使用して、予想収益に対するコスト比較ができます。

対応アプリケーションの作成とアプリケーションの改造では、どちらが高価になるか?

各国語サポート対応アプリケーションを作成するほうが、初期費用は高くなると考えられます。ただし、対応化のためには、標準的なモジュール設計やデータ駆動型設計のテクニックを使用するので、NLS の対応とは無関係にアプリケーションの品質が改善されます。設計が優れていると、アプリケーション・システムの理解や説明が容易になるので、ある程度の投資回収を期待できます。 優れた設計は、開発や保守の生産性も改善します。多数の言語バージョンに対応させる場合でも、アプリケーションの設計や実装に関する作業が 1 回だけで済みます。既存アプリケーションの改造と比較すると、最初から対応を計画、設計する方がはるかに安価になります。

開発作業

開発を成功させるには、NLS 対応アプリケーションの開発を始める前に、次のことを考慮してください。

国際化対応アプリケーションを開発するための教育

NLS 対応アプリケーションを開発するには、これまで以上の初期教育が必要になります。重要な教育項目を以下に示します。

- 一般的なグローバリゼーションの概念
- OS/400 で使用できるグローバリゼーション・サポート
- 作成するアプリケーションの対象となるほかのシステムやアプリケーションで使用できるグローバリゼーション・サポート
- アプリケーション内の分離可能なパーツ
- 国別情報に応じたデータの表示方法
- テキスト・データ・パーツの設計とコーディング
- 翻訳作業
- プロダクトとシステムの統合
- パッケージング、インストール、およびセットアップ
- プロダクトのサポートと保守

グローバリゼーション対応ガイドラインに従って、まず試作アプリケーションを準備して、選択した特定のアプリケーション実装環境でアプリケーションをテストします。次に、アプリケーション作成の開発作業、ガイドライン、標準など全体に関して、グローバリゼーション対応ガイドラインの内容を統合してください。

国際化対応アプリケーションの実装

国際化対応アプリケーションを実装するときに最も重要なことは、実行コードを 1 セットだけにするということです。 実行コードとテキスト・データの間には、一貫した区別を付けてください。 アプリケーション全体にわたり、選択したアプローチを標準化することが重要です。命名規則には、固有の明確な定義を与えてください。こうしたデータをアプリケーションで理解、維持するためには、プログラムが呼び出すパラメーターを一貫した方法で処理してください。

文書作成

文書は、アプリケーション・システムを使用するエンド・ユーザーのために、エンド・ユーザーの言語で作成します。また、文書には、エンド・ユーザー、システム・オペレーター、およびアプリケーション・システム・マネージャーのために、インストール、セットアップ、およびカスタマイズの情報も含める必要があります。

ユーザー向け文書は、簡単に翻訳できるようなテキスト・データにします。できる限り、オンライン・ヘルプ情報とユーザー文書を組み合わせて、翻訳するテキストの量を削減してください。表示画面や印刷レイアウトの例は、アプリケーションで作成して文書に組み込みます。

翻訳作業

テキスト・データの翻訳には時間がかかります。テキスト・データは、コードが未確立の場合でも、開発の初期段階で翻訳者に提供してください。翻訳の計画では、次のことを考慮してください。

物理的な機器類

翻訳者は、翻訳する言語との互換性のある機器を使用する必要があります。翻訳に必要な文字がすべてそろっているディスプレイ装置とキーボード、および翻訳したテキストを印刷できるプリンターが必要です。

翻訳ツール

生産性が上がるようなツール、そしてテキスト以外のアプリケーション・データの翻訳を防止するようなツールを翻訳者に提供してください。翻訳ツールを購入するときは、次の機能を確保してください。

- エンド・ユーザーが見る画面を表示する機能、およびシステム上でテキスト・データを翻訳し、テキスト・データ以外のアプリケーション部分の翻訳を防止する機能を備えたエディター。エディターには、スキャン、置換、検索、コピー、移動、削除などの機能も必要です。
- プロダクト全体で単語や句の一貫性を得るための辞書機能。
- アプリケーションの誤動作につながる翻訳間違いをチェックするための機能。
- 翻訳文書を改訂版に組み込むためのマージ機能。このマージ機能を使用すると、新規テキストだけを翻訳すればよいので、時間と労力を節約できます。
- チェックのための印刷機能。

翻訳者の教育

翻訳者は、翻訳するプロダクトやツールの内容を知る必要があります。翻訳は、単なる単語の置換作業ではありません。翻訳は、概念をほかの言語で形成する作業です。翻訳するプロダクトの知識があると、エンド・ユーザーにさらに分かりやすいプロダクトを提供できます。翻訳者を教育するための時間とリソースを前もって計画してください。

翻訳に関するガイドラインと指示

正しい翻訳を得るためには、翻訳に関するガイドラインと指示を提供してください。たとえば、エラー・メッセージを正しく翻訳するためには、そのメッセージが表示される状況を知ることが必要です。メッセージが表示される原因のエラーについて注意書きがあると、翻訳者の助けになります。

翻訳のための用語集

翻訳を正確にするためには、一般に普及している標準的な辞書の定義に従った用語を使用します。標準的な辞書にはない用語、あるいは標準的な定義とは異なる用法の用語をアプリケーションに使用する場合は、翻訳者向けに非標準用語の用語集を提供してください。アプリケーションには、省略語や頭字語を使用しないようにします。アプリケーションに省略語や頭字語がどうしても必要な場合は、用語集に定義してください。省略語や頭字語は、ある言語では明白であっても、他の言語ではそうとは限らないことを覚えておいてください。

テスト作業

グローバリゼーション対応プロダクトは、次の 3 段階のテストを実行してください。

1. 実行コードのテスト
   実行コードは、グローバリゼーション対応環境でテストして、言語依存のすべての組み合わせを確認します。翻訳者は、プロダクトの機能をテストしないでください。
2. テキスト・データの確認
   テキスト・データについては、翻訳が正しいか、またプロダクト全体の整合性がとれているどうかをテストします。

3. 実行コードとテキスト・データの統合
   テキスト・データとコードを個別にテストした後、総合テストを実行して、グローバリゼーション関連の処理がすべてアプリケーションに組み込まれているかどうか、またテキスト・データの翻訳でプロダクトが誤動作しないかどうかをテストします。

   アプリケーションを複数の国や複数言語のシステムで実行する場合は、複数のテキスト・データを使用した個別テストを計画してください。

パッケージングとインストール作業

アプリケーションのパッケージングの際には、実行コード、翻訳テキスト・データ、およびインストール文書について考慮してください。アプリケーションのパッケージングとインストールを簡単にするための提案を以下に示します。

- 実行コードとテキスト・データは分離して保存します。
- テキスト・データをパッケージするときは、お客様が受け取るテキスト・データをお客様が注文した言語だけにします。 (すべてのお客様にすべての言語のテキスト・データを送ることは、システム・リソースが無駄となり、さらに保守上の問題が発生することが考えられます。)
- インストール資料は、オペレーター関連の不要なトラブルを回避するため、また、アプリケーションの信頼性について最初から間違った印象を与えないように、完全なものを提供してください (プロダクトをインストールする人の言語に翻訳します)。

  インストール資料には、次のトピックを含めます。
  - ハードウェアおよびソフトウェアなど、アプリケーションのインストールと実行に必要な条件。
  - アプリケーションのインストール方法と失敗したときの対処方法。
  - 次の項目について必要な変更内容
    - サブシステムの定義
    - 装置記述
    - ユーザー・プロファイル
    - システム値
    - ライブラリー・リスト
  - アプリケーションの制約事項

アプリケーションの保守

多国語アプリケーションの保守を計画する場合には、次の点を考慮してください。

- 実行コードは、テキスト・データとは分離して保守してください。分離したコンポーネントは、完全な同期化が必要です。 1 つのコンポーネントを再設計すると、他のコンポーネントの再設計が必要になることがあります。
- テキスト・データを変更したときは、テキスト・データを翻訳したすべての言語に対して、その変更内容を加えてください。これで、1 つの保守レベルですべてのプロダクトに対応できます。
- 変更したテキスト・データを配布する前に、必ず実行コードをテストしてください。

グローバル・アプリケーションの設計

国際化対応アプリケーション・コンポーネントを設計することは、各国語を個別にサポートするコンポーネントを作成することを意味します。ある言語をサポートするために、ほかの言語へのサポートに干渉することのないようにしてください。ある言語をサポートするために、ほかの言語向けのプロダクトで機能を縮小しないでください。

作成したアプリケーションは、複数の言語で同時にサポートする必要があります。たとえば、2 バイト・コード化文字セット (DBCS) 言語をサポートするために、1 バイト・コード化文字セット (SBCS) 言語へのサポートを省略しないでください。 ライブラリーをセットアップするときには、テスト、パッケージング、納品などで動的に割り当てできるように、複数のテキスト・データ・ライブラリーを使用することを考えてください。

iSeries サーバー用のグローバル・アプリケーションを開発する際には、上述のことに加えて、グローバル・ユーザー向けにアプリケーションを作成およびコーディングする方法に影響を与える固有の設計上の問題も考慮する必要があります。以下のトピックは、こうした問題を識別し、作業を進める上で役立つガイダンスを提供します。

- チェックリスト: アプリケーションの設計
- グローバリゼーションとローカライズ
- アプリケーションの配置とアーキテクチャー
- ユーザー・インターフェース

チェックリスト: アプリケーションの設計

次の表は、各国語サポート対応アプリケーションを設計するときのガイドラインです。

適合	該当せず	規則
		システムまたはシステム・コンポーネント内に特定の文字セットがあることを前提としない。
		大文字小文字の変換は、言語およびコード・ページごとに定義可能とする。
		フォールディングは、言語およびコード・ページごとに定義可能とする。 フォールディングは、特定の装置で印刷や表示ができない文字を印刷や表示ができる文字に置換する処理です。
		ソフトウェアを制御するためのグラフィック文字は、メッセージ、メニュー、プロンプト、入力フィールド、または出力フィールドでも使用できるようにする。
		データ入力に使用できる文字セットは、システム・オペレーター、ユーザー、またはアプリケーションが定義できるようにする。
		グラフィック記号とアイコンを翻訳可能にする。
		アクティブ・コード・ページのすべての文字にアクセスできるようにする。
		プロダクトの言語依存部分は、簡単に変更できるようにするため、言語非依存部分とは分離する。
		プロダクトの各コンポーネントにある各国語サポートがそれぞれ独立するようにプロダクトを設計する。
		戦略ポイントには、各国語出口を準備する。
		診断を使用可能にする。
		物理的なキーボード・レイアウトとは異なる論理レイアウトをユーザーが使用できるようにする。
		ユーザー・インターフェース・テキストおよび表示制御情報は、すべて実行コードと分離する。

適合	該当せず	規則
		表示フィールドの長さや表示フィールドの位置に依存する機能、または表示フィールドの位置のみに依存する機能は、ユーザー・インターフェース・テキストが拡張しても影響を受けないように設計する。
		翻訳処理には、パネルやメッセージの識別およびトラッキングができるような手段を提供する。
		変数は、表示フィールド内で任意の位置と順序で配置できるようにする。
		メッセージおよびその他の表示する単語やフレーズは、個別の単語やフレーズから構成するのではなく、完全な形のものを使用する。
		エンド・ユーザー・コマンド、キーワード、または応答などの入力は、大文字小文字とは無関係に使用できるようにする。
		各国語依存型の機能があるプロダクトは、国や言語を追加できるように設計する。
		小文字のアルファベットを不変とは想定しない。
		文字セットは、オペレーター、ユーザー、またはアプリケーションが定義できるようにする。
		句読記号を含む特殊文字は、定義可能にして、プログラムには依存しない。
		ユーザー・インターフェース・テキスト・モジュールは、実行コードとは個別にパッケージ化する。

グローバリゼーションとローカライズ

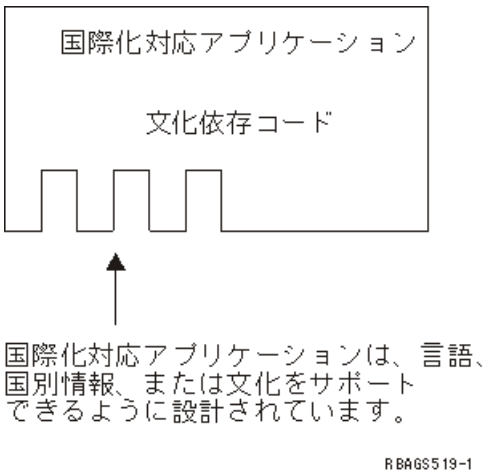
OS/400 は、プログラムの動作を制御し、リソースの制御、ジョブの計画、入出力の制御、データ管理などのサービスを提供します。これは、iSeries サーバーの機能を補足、拡大して、対話式アプリケーションおよびバッチ・アプリケーションについて完全な統合サポートを提供します。

OS/400 の機能の多くは、対話式データ処理に直接使用できます。たとえば、次の機能があります。

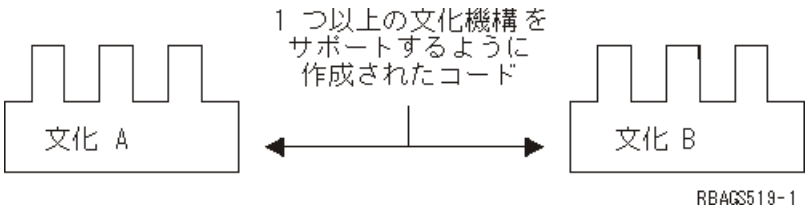
- データベース・サポートにより、任意のワークステーションから高速読み取りを行って、最新のビジネス・データを使用できます。
- 実行管理機能により、すべてのワークステーション・ユーザーを対象に要求処理を計画します。
- アプリケーション開発サポートにより、通常の生産活動を実行しながら、新規アプリケーション・プログラムのオンラインによる開発やテストを実行できます。
- システム・オペレーション・サポートにより、システム・オペレーション担当者は、すべてのコマンドについてプロンプトやヘルプを完備した単一の制御言語を使用してディスプレイ装置を操作できます。
- ヘルプおよび索引の検索サポートにより、ユーザーは、広範囲に渡るトピックについて、オンライン情報を要求できます。
- メッセージ処理サポートにより、システム、システム・オペレーション担当者、ワークステーション・ユーザー、およびシステム内で実行中のプログラムの間で通信ができます。
- セキュリティー・サポートにより、データやその他のシステム・リソースを無許可アクセスから守ります。

これらの機能に加えて、OS/400 プログラムは各国語サポートを提供します。各国語サポートを使用すると、選択した言語でシステムを操作でき、各国の文化に応じた結果を得られます。各国語サポートは、グローバリゼーションとローカライズで構成されています。

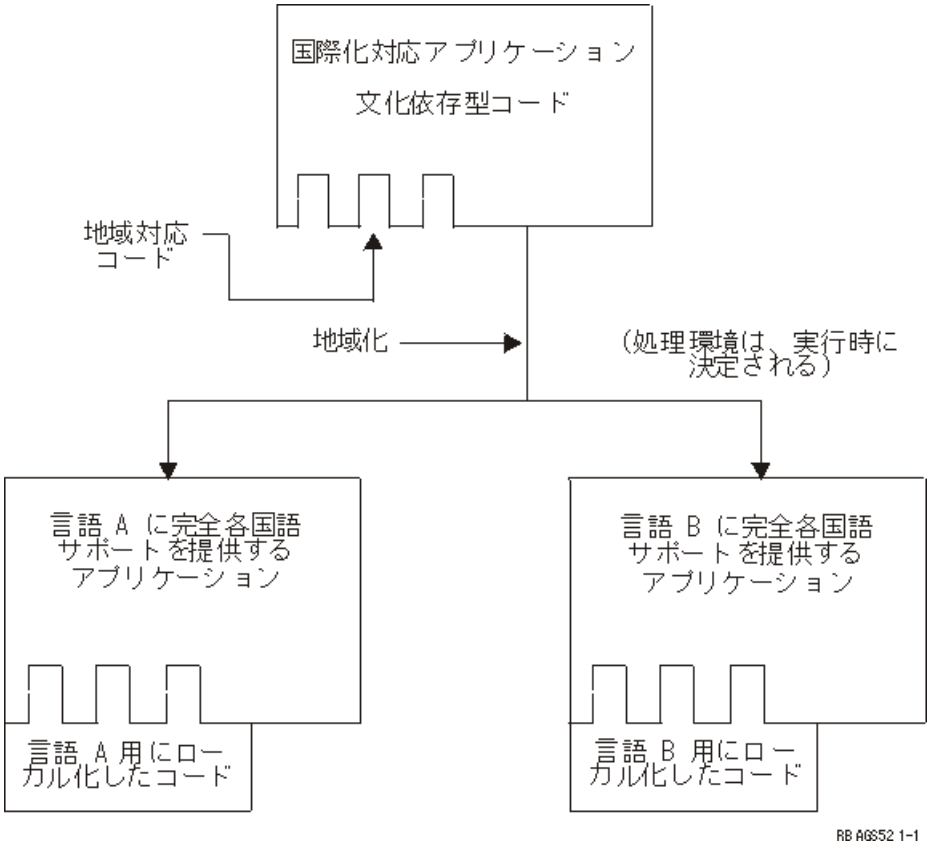
グローバリゼーション は、アプリケーションを変更することなく、アプリケーションをあらゆる言語環境で使用できるサポートです。このタイプの設計では、各国語サポート対応アプリケーションも使用できます。グローバル・アプリケーションは、次の図に示すように、文化的に中立です。



一方、**ローカライズ** では、特定の言語、国、文化などを対象にアプリケーションを操作します。アプリケーションをローカライズするには、アプリケーションのグローバリゼーション以上の操作が必要です。



ローカル・コードとグローバル・コードを実行時に統合すると、ユーザーは各国語サポート完全対応のアプリケーションを得られます。次の図に示すように、実行時にグローバル・コードと組み合わせるローカル・コードを決定するのは処理環境です。

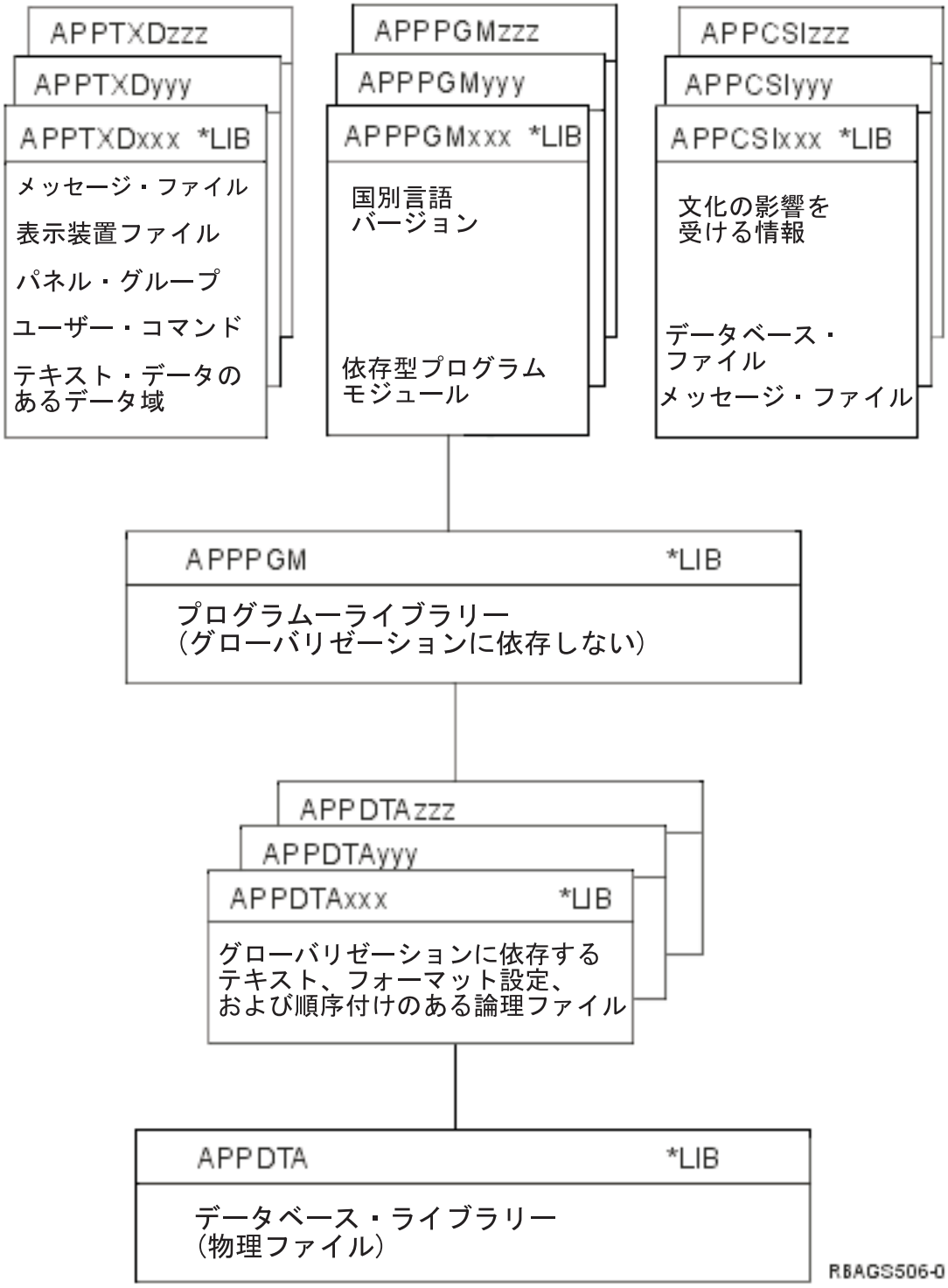


アプリケーションの配置とアーキテクチャー

国際化対応アプリケーションを設計するときは、アプリケーションを国際環境で使用できるように編成および構成することを考慮してください。特に、以下のことを考慮してください。

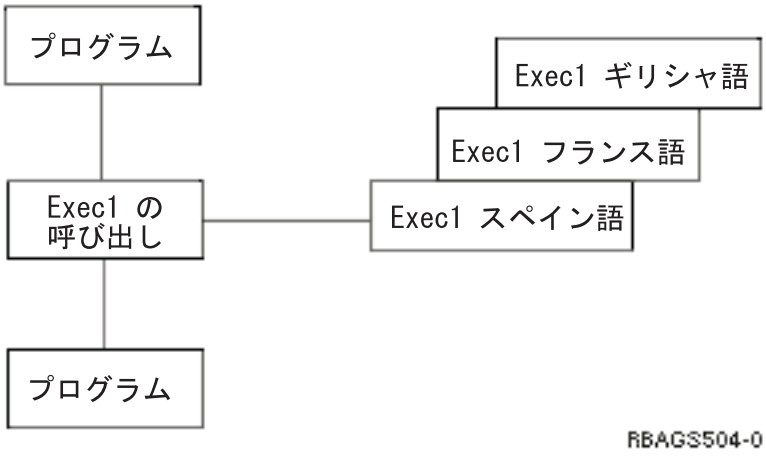
- プログラム・モジュール を適切な場所で分離する。
- アプリケーション・パーツ に、多国語環境に応じた名前を付ける。
- 常に 仕様書 を参照する。
- データベースの定義 については、個別のライブラリーに論理ファイルの複数の組み合わせを準備する。

次の図は、アプリケーション・パーツに関する推奨構成方法です。



プログラム・モジュールの分離: 文化依存パーツを実行コードから分離して、文化依存環境を設定できます。これには、システム値、ユーザー・プロファイル属性、ジョブ属性、およびオブジェクト属性を使用します。

各国語や文化依存パーツを実行コードと分離できない場合は、各国語に依存する機能を必要とするすべての場所に各国語の出口か呼び出し機能を準備してください。次の図は、各国語出口を示しています。



アプリケーション・パーツの名称: 作成したアプリケーションをさまざまな言語や国で使用するには、目標システムの環境について、命名規則を考慮する必要があります。使用する文字は、目標の環境で使用可能であることが必要で、また表示や印刷もできなければなりません。次の名称を指定するときは、使用する文字は不変文字セットだけにしてください。

- ライブラリー
- データベース・ファイル
- 装置ファイル (表示装置または印刷装置)
- ヘルプ・パネル
- メッセージ・ファイル
- ユーザー・コマンド
- プログラム
- レコード様式
- フィールド

その他の文字は、意味が異なるか、またはキーボードに存在しない文字です。

国際化対応アプリケーションを作成するには、アプリケーション・オブジェクトをそれぞれの関連パーツに分割する必要がありますが、パーツは、テキスト・データの場合もあれば、非テキスト・データの場合もあります。 この 2 種類のパーツに対して、異なる命名規則を使用する必要があります。 さらに、テキスト・データは、言語別の区別ができるようにしてください。オブジェクトを個別のライブラリーに分割するとこの操作ができます。

シナリオ: ライブラリーの命名規則

ライブラリーの命名規則は次のようになっています。

```
AAATTTLLL
```

ここで、**AAA** はアプリケーションの識別コード、**TTT** はオブジェクトのタイプ、**LLL** は言語コードです。

この命名規則では、先頭に固有の識別コード (AAA) が付いているので、同じアプリケーションに属するすべてのライブラリーを 1 つにまとめることができます。

2 番目の部分 (TTT) は、オブジェクト・タイプを識別できます。

テキスト・データ

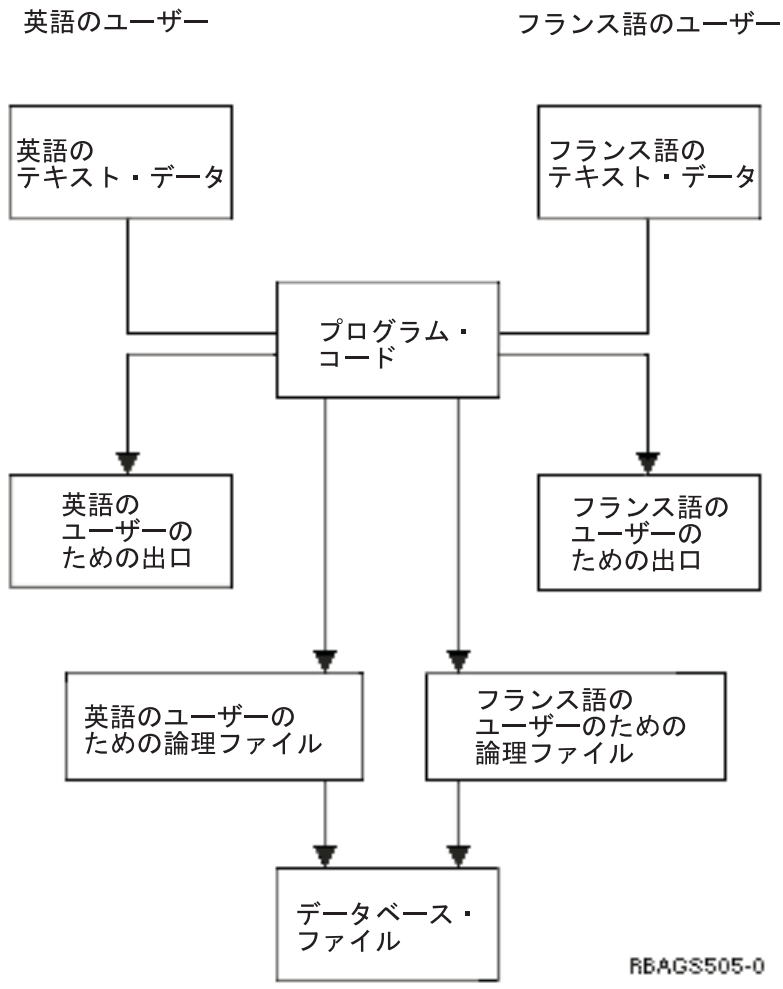
- 表示装置ファイル
- 印刷装置ファイル
- メッセージ・ファイル
- ヘルプ・パネル
- ユーザー・コマンド
- 文化値
- NLS 対応情報と仕様を含むデータベース・ファイル
- NLS 依存型のプログラム・モジュール

非テキスト・データ
プログラム

データ データベース・ファイル

3 番目の部分 (LLL) は、すべてのテキスト・データ部分について、各国語バージョンを指定できます。これにより、異なるライブラリーの間で、異なる各国語バージョンに同一のオブジェクト名を使用できます。ジョブを実行するときに、必要に応じてライブラリー・リストを再編成するだけで、プログラムがさまざまなオブジェクトを使用できるようになります。

最初のライブラリー・リストは、ジョブ記述から取り込むことができます。新しいジョブ記述を作成するときは「ジョブ記述の作成」 (CRTJOBD) コマンド、既存のジョブ記述を変更するときは「ジョブ記述の変更」 (CHGJOBD) コマンドで、INLLIBL パラメーターにライブラリー・リストを指定すると、新しいライブラリー・リストを作成できます。この例を次の図に示します。



仕様の参照: アプリケーションのフィールド参照ファイルにすべてのフィールドを定義してから、データベース仕様、装置ファイル仕様、および高水準言語プログラムなどを必要なときに参照してください。このテクニックを使用すると、フィールド仕様を 1 回定義しておけば、再び使用できます。ソースの異なる同じフィールドを識別する必要がある場合は、フィールドを名前変更するか、または修飾してください。特定のフィールドについて定義を変更する必要がある場合は、フィールド参照ファイルでそのフィールドの属性を変更し、オブジェクトをもう一度作成するだけで済みます。これで、フィールドが使用されるすべての場所で自動的に変更が行われます。

たとえば、次のようにします。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                     REF(field-ref-file-name)
     A          R record
     A field    R line  pos
or
     A field    R line  pos                REFFLD(ref-field-name)
```

データベースの定義: 特定の項目を指定するには、ファイルを定義してから、その指定内容をデータベース・ファイルに対して使用します。 次のような指定ができます。

- ファイルに関するオブジェクト記述テキスト
- レコード形式とフィールド記述に関する説明テキスト (TEXT キーワード)
- フィールド記述上の列のヘッディング (COLHDG キーワード)

- 日時の形式と区切り記号
- 分類順序
- 言語識別コード

オブジェクト記述のテキストは、DB2(R) UDB for iSeries SQL、iSeries Access などのあらゆるデータベース・ツール、さらにファイル選択表示画面上のデータ・ファイル・ユーティリティーで表示できます。

列のヘッディングは、出力フィールド定義表示画面上で、データベース・ツールにより表示されます。列のヘッディングは、画面設計機能 (SDA) および報告者設計ユーティリティー (RLU) 上でフィールド・プロンプトのテキストまたはヘッディングとしても使用されます。

日付タイプおよび時刻タイプのフィールドについては、ユーザーの要求やジョブ要求により、アプリケーションやデータベース・ツールで表示形式の変換をしない限り、フィールド作成時に指定された形式でデータ管理機能が処理します。

これらのすべての情報をユーザーの言語と文化に従って表示するには、複数の論理ファイルの組み合わせを異なるライブラリーに準備しておく必要があります。翻訳テキストとともに、異なる日時形式や異なる分類順序を指定すれば、データ管理機能がその変換を行います。数値タイプの日付フィールドについては (圧縮されている場合は異なります)、サブストリング (SST) 機能を使用して、同様のテクニックを使用できます。ユーザーは、指定された論理ビューを使用しないと、データにアクセスできません。複数の分類順序を使用して論理ファイルを定義する場合は、共用重みテーブルに対して固有の索引を使用しないでください。この方法は可能ですが、固有の索引を使用すると、重みが同じで異なる文字のキーが使用できなくなります。

アプリケーション・パーツの名称のシナリオには、複数ユーザー向けの論理ファイルの組み合わせの使用例があります。

ユーザー・インターフェース

ユーザー・インターフェースは、カスタマーが実際に見ることができるソフトウェア・プロダクトの一部です。ユーザー・インターフェースには、表示画面や印刷出力のレイアウト、表示テキストや印刷テキスト、コマンド、オンライン・ヘルプ、メッセージなどがあります。 また、ソフトウェア・プロダクトの一部であるユーザー・インターフェースは、国や文化の異なるユーザーのために、翻訳するか文化に応じた変更を加える必要があります。

OS/400 には、ユーザー・インターフェースに使用するテキストを組織し、翻訳しやすいようにテキストをライブラリーに保存するソフトウェア機能があります。 また、オペレーティング・システムのユーザー・インターフェース・マネージャーは、一貫性のあるユーザー・インターフェースを提供します。ユーザー・インターフェース・マネージャーは、表示画面やオンライン・ヘルプなどのパネルについて、定義および実行のための総合的なサポートを提供します。

このセクションでは、国際化対応アプリケーションのユーザー・インターフェースを設計するときのガイドラインを提供します。このガイドラインは、設計作業の初期に適用してください。ガイドラインでは、次の内容を扱います。

- チェックリスト: ユーザー・インターフェースの設計
- テキストの翻訳の設計
- テキスト・データ・コードの設計
- ユーザー・インターフェース・マネージャー
- プログラム・メッセージの設計

- メニューの設計
- コマンドの設計
- 文化依存型の設計
- 表示装置ファイルの設計
- 印刷装置ファイルの設計と翻訳
- ソース・ファイルの設計
- CDRA の設計
- NLV サポート対象外の言語の処理

チェックリスト: ユーザー・インターフェースの設計: グローバル・サポート対応のユーザー・インターフェースを作成する場合は、次の表のガイドラインに従ってください。

適合	該当せず	規則
		ソフトウェアを制御するためのグラフィック文字は、メッセージ、メニュー、プロンプト、入力フィールド、または出力フィールドでも使用できるようにする。
		グラフィック記号とアイコンを翻訳可能にする。
		プロダクトの言語依存部分は、簡単に変更できるようにするため、言語非依存部分とは分離する。
		ユーザー・インターフェース・テキストおよび表示制御情報は、すべて実行コードと分離する。
		ユーザー・インターフェース・テキストが翻訳により拡大した場合に備えて十分なスペースを確保する。
		表示フィールドの長さや表示フィールドの位置に依存する機能、または表示フィールドの位置のみに依存する機能は、ユーザー・インターフェース・テキストが拡張しても影響を受けないように設計する。
		翻訳処理中にパネルやメッセージの識別およびトラッキングができるような手段を提供する。
		変数は、表示フィールド内で任意の位置と順序で配置できるようにする。
		メッセージおよびその他の表示する単語やフレーズは、個別の単語やフレーズから構成するのではなく、完全な形のものを使用する。
		エンド・ユーザー・コマンド、キーワード、または応答などの入力は、大文字小文字とは無関係に使用できるようにする。
		日時形式は、さまざまな形式を選択できるようにする。
		数値に使用する句読点の形式は、さまざまな形式を選択できるようにする。
		数値の丸め方式と数学形式は、さまざまな形式を選択できるようにする。
		通貨形式は、定義可能にする。
		通貨の記号と省略形のデフォルト値は選択可能にする。
		通貨記号の位置を選択可能にする。
		通貨金額のフィールド・サイズを選択可能にする。
		度量衡方式を選択可能にする。

適合	該当せず	規則
		小文字のアルファベットを不変とは想定しない。
		句読記号を含む特殊文字は、定義可能にして、プログラムには依存しない。
		ユーザー・インターフェース・テキスト・モジュールは、実行コードとは個別にパッケージ化する。
		1 バイト・コード化文字セット・システムのユーザー・インターフェース・テキスト・モジュールは、実行コードとは別にロードする。
		変数および入力フィールドの指示について、プロダクト全体に一貫性のある規則を適用する。
		数値を文字で表現しない。
		ユーザー・インターフェース・テキストに使用する用語について、プロダクト全体に一貫性を与える。
		省略形を使用しない。
		スラング、業界用語、ユーモアを使用しない。
		商標を示して、説明を記述する。
		あいまいな単語を使用しない。
		ユーザー・インターフェース・テキストには、正しい文章構成を使用する。
		否定形の疑問文は使用しない。

テキストの翻訳の設計: 以下の情報は、テキスト・データの翻訳を簡単にするための一般的なヒントです。

テキスト・データを実行コードと分離する

翻訳を容易にし、また間違って実行コードを翻訳しないように、テキスト・データと実行コードを分離してください。必要な実行コードは 1 つだけですが、テキスト・データの翻訳は何度も行われることがあります。

拡大スペースを提供する

テキストをほかの言語に翻訳するときに必要なスペースの量は、言語により異なります。翻訳後のバージョンで、もともとの概念や使いやすさを維持するには、テキスト・データの拡大を考えて十分な表示スペースを確保してください。 次の表は、米国英語を使用したときのスペースの推奨拡大率を示しています。

テキスト内の文字数	必要となる追加スペース
10 以下	100 ～ 200%
11 ～ 20	80 ～ 100%
21 ～ 30	60 ～ 80%
31 ～ 50	40 ～ 60%
51 ～ 70	31 ～ 40%
70 以上	30%

画面上のオブジェクトの位置を変更する

表示項目の位置は、他の表示項目の位置やサイズの影響を受けることが多いので、表示項目は、翻訳した後に位置の変更が必要になる場合があります。位置を変更しても、プログラムは正しく応答する必要があります。

変数の順序を柔軟にする

情報を動的にするために、通常、メッセージには置換変数を使用します。しかし、言語にはそれぞれ固有の構文 (品詞の順序) があります。メッセージをほかの言語に翻訳するときには、翻訳先言語の構文に従って、置換変数の位置や順序を変更する必要があります。

完全なテキスト・データとする

定数テキストの最終的な形態が複数の部分で構成されている場合、そのテキストを翻訳できなくなることがあります。その理由は、翻訳者にとって単語の使用形態が分からなかったり、言語が異なる場合に各部分を組み合わせることができないことがあるからです。

たとえば、表示画面に使用する列見出しは、完全なエンティティーとして定義する必要があります。列見出しを定義するときに、単語と単語の部分を組み合わせないでください。 たとえば、月曜日から金曜日までの仕事をスケジューリングするアプリケーションを作成するとします。このアプリケーションをフランス語で作成します。レポートと画面表示の列見出しを曜日の最初の部分と「DI」を組み合わせて作成します。アプリケーションの列とレポートの見出しは次のようになります。

曜日の最初の部分	組み合わせる文字	結果
LUN	DI	LUNDI
MAR	DI	MARDI
MERCRE	DI	MERCREDI
JEU	DI	JEUDI
VENDRE	DI	VENDREDI

このアプリケーションをフランス語からドイツ語に翻訳するときに、2 つの単語の部分を組み合わせて MONTAG、DIENSTAG、MITTWOCH、DONNERSTAG、および FREITAG という単語は作成することはできません。

コマンド、応答、およびキーワードをテキスト・データのように扱う

コマンド、応答、およびキーワードは、ユーザーが通常使用する言語に翻訳します。たとえば、英語のアプリケーションをドイツ語に翻訳するとします。ドイツ語のユーザーは、応答に「Ja」と「Nein」を使用するので、応答を英語のまま「Yes」および「No」とすると、よくわからない、使いにくい、と感じます。

テキストをできるだけ単純に明確に表現する

- 簡単な句や文を使用して、句の組み合わせを避けてください。簡単な単語を使用すると、翻訳が容易になります。
- プロダクト全体を通して、一貫した用語を使用してください。

  プロダクトを通して用語に一貫性がないと、翻訳者は、正しい訳語を決定するために時間を浪費します。
- 誤解を防ぐためには、単語の正しい使用方法について、翻訳者向けの注意書きを加えてください。
- 省略語は避けてください。

  省略語の規則は、言語により異なります。省略語を使用すると、翻訳者およびエンド・ユーザーの誤解の原因となります。
- スラング、業界用語、ユーモアを使用しないでください。

スラング、業界用語、ユーモアなどは、特定の言語に限られた表現です。ほかの言語に翻訳するのは困難です。

- 否定形の質問はしないでください。

  否定形の質問は、ユーザーが誤解することがよくあります。質問するときは、肯定形で問い合わせてください。

テキスト・データ・コードの設計: アプリケーション・ディスプレイ、印刷装置ファイル仕様、およびユーザー作成のコマンドには、通常は大量の定数テキストが含まれています。さらに、アプリケーション・ディスプレイ、印刷装置ファイル仕様、およびユーザー作成のコマンドには、ヘッディング、フィールド・プロンプト、指示行、およびファンクション・キーの記述などの入出力フィールドも含まれています。

定数テキストの指定、保存、および使用の方法については、複数のテクニックがあります。 テキスト・データのコンポーネントに応じて、異なるテクニックを使用できます。テクニックには、それぞれ長所と短所があります。次のトピックでは、それぞれのテクニックの使用方法と各コンポーネントに使用できるテクニックが示されています。

- メッセージの早期バインディング
- メッセージの実行時バインディング
- 名前のない出力フィールドとして直接コーディングする
- データベース・ファイルに保存するテキスト

メッセージの早期バインディング: テキストは、ソース・コード外部の個別のメッセージ・ファイルに保存できます。この場合、テキストは作成時にオブジェクトにバインドされます。このテクニックは、次の場合に使用できます。

表示装置ファイル
: 表題、命令行、オプション定義、ヘッディング、フィールド・プロンプト、コマンド・キー記述などの定数

印刷装置ファイル
: 表題、ヘッディング、合計行の記述などの定数

ユーザー・コマンド
: コマンド定義ステートメント上のプロンプト記述

装置ファイルの場合は (ディスプレイとプリンター)、メッセージは、DDS ソース仕様の「メッセージ定数」キーワード (MSGCON) によって参照されます。

たとえば、次のようにします。

```
A          line pos   MSGCON(length message-ID [*libl/]message-file-name)
                  ^
           includes expansion space
```

ユーザー・コマンドの場合は、リテラルの代わりに、メッセージ識別コード *xxxnnnn* が PROMPT キーワードに指定されます。 メッセージ・ファイルは、「コマンドの作成」 (CRTCMD) コマンド上で参照されます。

たとえば、次のようにします。

```
CMD       PROMPT(xxxnnnn)
```

メッセージ・ファイル名の *message-file-name* は、ソース・ファイルにあり、次のコマンドによって参照されます。

```
CRTCMD CMD(command-name)  PGM(library-name/program-name)  +
     PMTFILE([*libl/]message-file-name)
```

オブジェクトを作成する前に、指定されたメッセージ・ファイルにメッセージ記述を入力する必要があります。「メッセージ記述の追加」(ADDMSGD) コマンドを使用して、メッセージ記述を入力します。

たとえば、次のようにします。

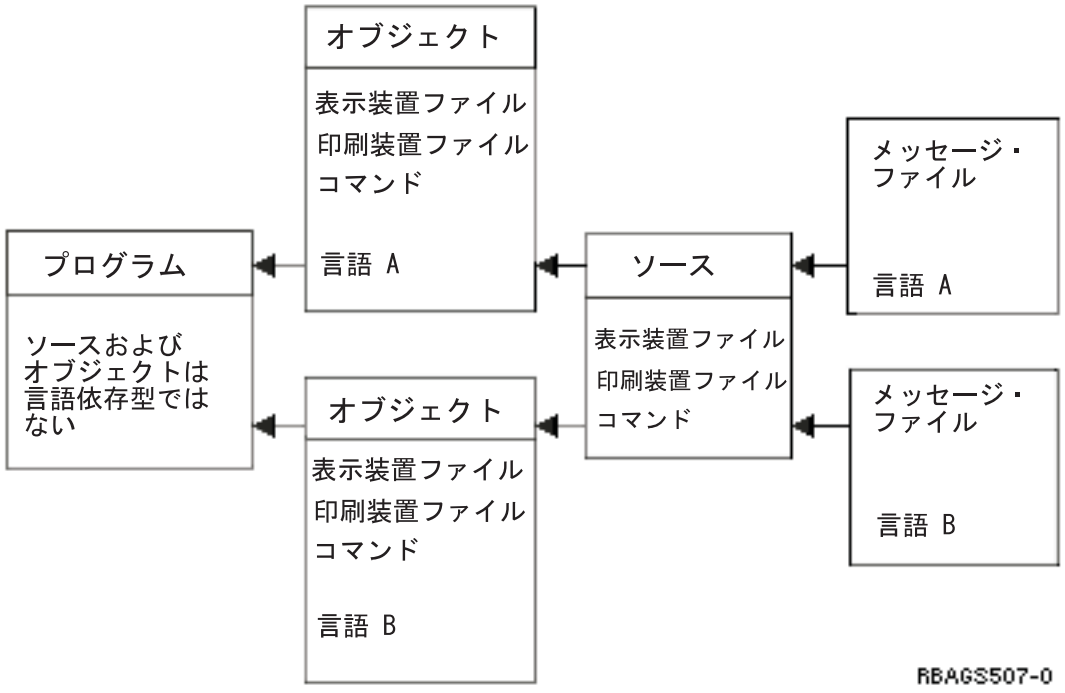
```
ADDMSGD   MSGID(xxxnnnn) MSGF(library-name/message-file-name)  +
MSG('Text                                          ')
```

ここで、*xxxnnnn* はメッセージ識別コードです。

このテクニックを使用すると、複数の言語で任意の数のオブジェクトを作成できます。また、オブジェクト作成時にほかのメッセージ・ファイルを割り当てるだけで、同じソース・コードを使用してこれらのオブジェクトを異なるライブラリーに入れることができます。

メッセージ・ファイルが必要になるのは、オブジェクトを作成するときだけです。MSGCON キーワードでは、それぞれの言語に応じて、適切な長さを指定するように考慮してください。 次に、長さ情報を翻訳者に知らせてください。

次の図は、メッセージの早期バインディングの方法を示しています。



ファイル作成時には、テキスト・データとプログラム・ライブラリーの入った特定のライブラリーをライブラリー・リストに設定して、使用する言語バージョン用のテキスト・データを選択できます。

メッセージの実行時バインディング: テキストは、DDS ソース・コード外部のメッセージ記述に保存できます。この場合、テキストは、実行時に表示形式だけにバインドされます。

このテクニックは、次の場合に使用できます。

表示装置ファイルのみ

表題、命令行、オプション定義、ヘッディング、フィールド・プロンプト、コマンド・キー記述などの定数 (MSGID キーワード)

入力フィールドのデフォルト値 (MSGID キーワード)

フィールド妥当性検査の指定項目 (CHKMSGID キーワード)

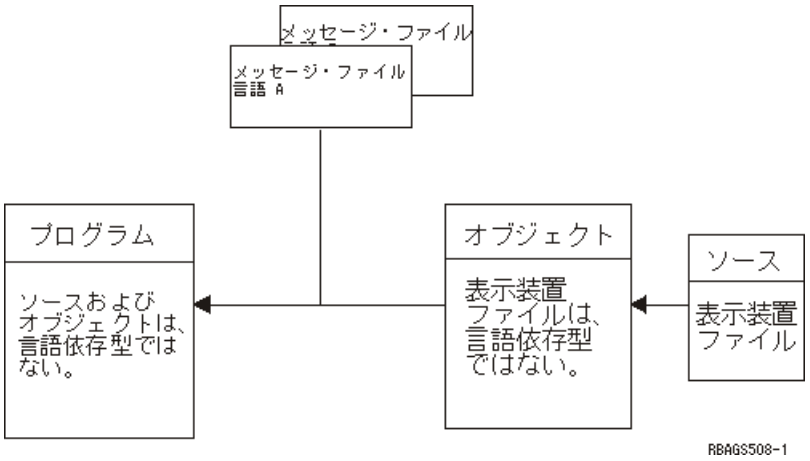
エラー・メッセージ (ERRMSGID および SFLMSGID のキーワード)

表示装置ファイルの DDS にメッセージを指定するには、MSGID キーワード (メッセージ識別コード) を使用します。ADDMSGD (メッセージ記述の追加) コマンドを使用して、メッセージを指定メッセージ・ファイルに入力する必要があります。

たとえば、次のようにします。

```
A    FLD-name  length  line pos    MSGID(message-ID [*libl/]message-filename)
                   ^
           includes expansion space
ADDMSGD   MSGID(xxxnnnn) MSGF(library-name/message-file-name)  +
MSG('Text                                          ')
```

このテクニックを使用すると、DDS ソース・コードと表示装置ファイルのオブジェクト 1 つで、複数の言語で複数のライブラリーにメッセージ・ファイルをいくつでも作成できます。 実行時には、必要に応じてライブラリー・リストを設定して、ほかのメッセージ・ファイルを割り当てます。次の図で例を示します。



注: このテクニックでは、アプリケーションが国別情報に従ってすべての編集作業を実行する必要があります。

名前のない出力フィールドとして直接コーディングする: 定数テキストを定義する最も一般的な方法は、テキストをリテラルとしてソース・コードに直接指定することです。定数テキストの定義は、この方法が最も一般的ですが、翻訳するのには最も困難な方法です。 アプリケーションをコーディングするときには、翻訳する計画がない場合でも、この方法は避けてください。

翻訳対象外のアプリケーションをコーディングする場合は、この方法は次の場合に使用してください。

表示装置ファイル

表題、命令行、オプション定義、ヘッディング、フィールド・プロンプト、コマンド・キー記述などの定数

入力フィールドのデフォルト値 (DFT キーワード)

エラー・メッセージ (ERRMSG/SFLMSG キーワード)

印刷装置ファイル
表題、ヘッディング、合計行の記述などの定数

ユーザー・コマンド
コマンド定義ステートメント上のプロンプト記述

装置ファイルの場合は、テキストを名前のないフィールドとして指定し、開始行、列、定数テキストを指定します。

たとえば、次のようにします。

```
A              line pos       'Text . . . . . . . . . . . . . :  '
```

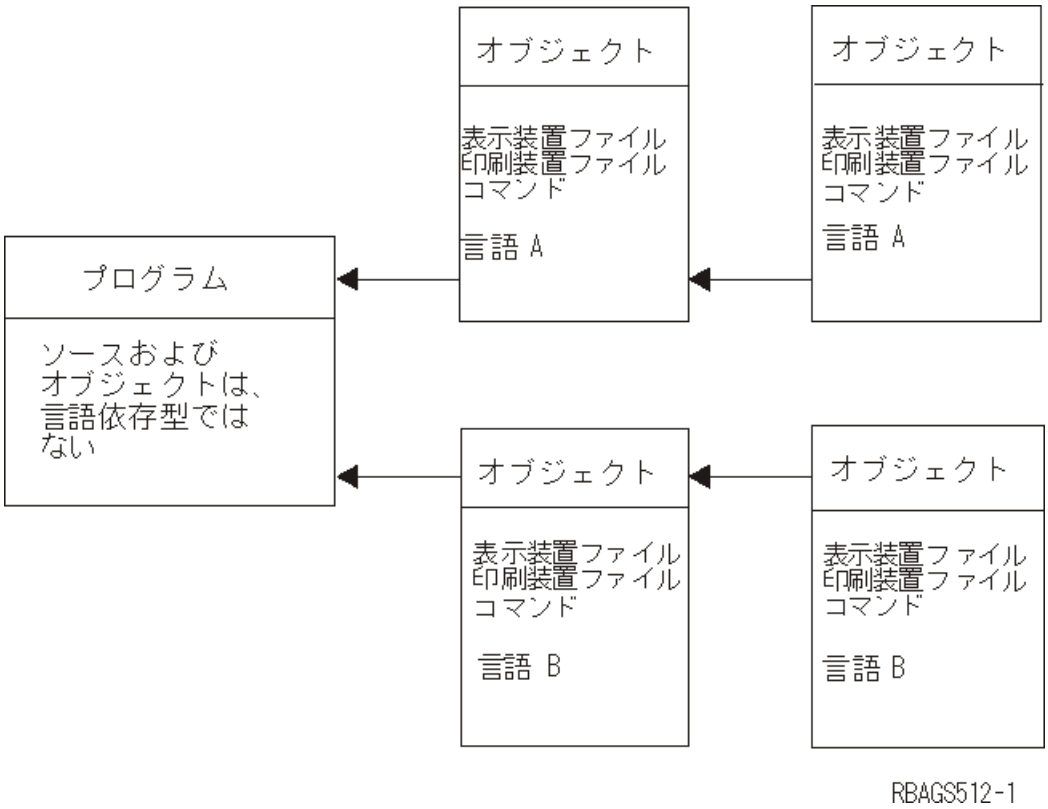
ユーザー作成のコマンドの場合も同じような規則を適用します。コマンド・ソース・ステートメントのキーワード上にテキストを直接定義します。

たとえば、次のようにします。

```
CMD          PROMPT('    Command description       ')
```

キーワード上にテキストを直接定義するときは、個々の単語のように多数の小さなリテラルを個別に指定するのでなく、大きなリテラルにさまざまな要素のセットを標準化してください。 これで、ソース・コードを変換するときに、読みやすくなり、柔軟性が得られます。

説明テキストに必要なスペースは、言語により異なります。変換後に十分な場所を残すためには、スペースをはじめから準備してください。次の図に示されているように、多様な言語に対応するためには、ソース・メンバーを変換し、オブジェクトを作成する必要があります。



それぞれの各国語バージョンには 1 組のプログラムがありますが、ソース・メンバーとデータ・オブジェクトについては複数のセットを持つことができます。アプリケーションを実行したときに、使用する言語バージョンのテキスト・データを選択できます。このためには、テキスト・データとプログラム・ライブラリーの両方を含む特定のライブラリーを使用して、ライブラリー・リストのシステム部分を設定する必要があります。

データベース・ファイルに保存するテキスト: テキストは、ソース・コードの外部のデータベース・ファイルに保存して、アプリケーション・プログラムで読み取り、実行時に表示形式または印刷形式にすることができます。DDS 上に定数をコーディングする代わりに、プログラムでファイル可能な出力フィールドを指定できます。出力フィールドは、それぞれの言語に応じて、翻訳のことも考慮して、適切な長さに指定してください。

このテクニックは、次の場合に使用できます。

表示装置ファイル
: すべての定数テキスト
入力フィールドのデフォルト値
エラー・メッセージ

印刷装置ファイル
: すべての定数テキスト

プログラム
: 値の比較、文字のスキャン、テーブルなど、すべての固定情報

このテクニックを使用すると、DDS ソース・コードと表示装置ファイルのオブジェクト 1 つで、複数の言語で複数のライブラリーにデータベース・ファイルをいくつでも作成できます。 実行時には、必要に応じてライブラリー・リストを設定して、ほかのメッセージ・ファイルを割り当てられます。

注: このテクニックでは、アプリケーションが国別情報に従ってすべての編集作業を実行する必要があります。

ユーザー・インターフェース・マネージャー: OS/400 のユーザー・インターフェース・マネージャー (UIM) は、システムの一部として、アプリケーションのパネルやダイアログの定義に使用します。UIM には、次の機能があります。

- データおよびパネルを記述するためのタグ・ベースの言語。
- タグ・ベースの言語を使用して、パネル・グループ・オブジェクトやメニュー・オブジェクトを作成するコンパイラー。
- パネルを表示および印刷するためのパネル・グループ・オブジェクトとして使用するアプリケーション・プログラミング・インターフェース (API) のセット。

UIM には、次の機能もあります。

- 画面管理のためのダイアログ・コマンド
- コンテキストによるオンライン・ヘルプ
- ポップアップ・ウィンドウ
- メニュー・バー
- CL コマンドを入力するためのコマンド行
- ユーザーや環境に応じたパネルの内容の調整
- メニュー・ネットワークを使用した高速パス

- 2 バイト文字セット (DBCS) の言語
- 両方向 (BIDI) 言語のサポート

UIM は、メニュー、情報の表示、リストの表示、入力項目の表示など、共通のパネル・タイプをサポートします。表示タイプやインターフェースに一貫性があると、ユーザーは新しいアプリケーションに早く慣れることができます。

UIM は、UIM 制御以外のオープン表示装置ファイルを使用した要求ディスプレイ装置と共存または共用が可能です。ただし、UIM パネルと DDS 定義のレコード形式を同時に画面に表示することはできません。UIM パネルと DDS パネルが置き換わると、システムは、ファイルまたはパネル・グループの操作を中断し、必要に応じて画面を復元します。

ユーザー・インターフェース・マネージャーについては、次のトピックにも情報があります。

- オンライン・ヘルプの設計
- 索引検索タグ
- 索引検索と DBCS

オンライン・ヘルプの設計: オンライン・ヘルプの定義には、次のいずれかを使用できます。

パネル・グループ
: ユーザー・インターフェース・マネージャー (UIM) ソースを入力するオブジェクトです。

レコード
: ソース・ファイル・メンバーに含まれている DDS キーワードの組み合わせです。

オンライン・ヘルプを定義するときに、ユーザー・インターフェース管理機能を使用する場合は、DDS または表示装置ファイルの代わりにパネル・グループが定義されます。どちらの場合も、表示するデータのエンコード方式を表示装置ファイルかパネル・グループに CHRID 値で示す必要があります。

パネル・グループは、ヘルプ情報を含めるときに使用するオブジェクトです。OS/400 では、ヘルプ情報の集合体を含むオブジェクト・タイプの識別コードとして *PNLGRP を使用します。

ガイドライン: オンライン・ヘルプ

各国語バージョン向けに翻訳するオンライン・ヘルプ情報を定義する場合は、パネル・グループとレコードについて、次のことに注意してください。

- レコードには文書処理機能を使用できません (スペル・チェックやワード・ラップなどの機能。ただし、システムの API にスペル・チェック機能はあります)。
- OS/400 のメッセージやパネル・グループの内容は、各国語の規則や翻訳に影響を与えます。 OS/400 プログラムの各国語バージョンが存在しない国もあります。完全に翻訳されずに、ほとんどが英語のままの各国語バージョンもあります。メッセージやパネル・グループが翻訳されていないと、各国語の国別情報が反映されません。コマンドの設計 には、未翻訳の NLV パーツがあるために、パネルの一部が英語のままになっている翻訳の例があります。
- 翻訳によるスペースの拡大分の余裕をとっておいてください。

ガイドライン: DDS オンライン・ヘルプの設計

1 つのシステムに複数の言語をインストールする場合、ヘルプ文書は個別のフォルダーに保存します。DDS ソース・ファイルは、システム上の各言語について、コピー、変更、およびコンパイルが必要になります。

索引検索タグ: ヘルプ・パネル・グループには、索引検索モジュールを含めることができます。索引検索は、各表示画面のヘルプ情報を補足します。ヘルプ・パネル・グループ内の情報を索引検索機能に使用するには、ヘルプ・モジュールに UIM タグを正しく追加する必要があります。

ユーザーは、索引検索機能が使用可能と指定されたヘルプ画面から索引検索機能を使用できます。

ISCH タグ

ISCH タグは、索引トピックの表題を定義し、ユーザーが入力する検索語 (同義語) とトピックをリンクするルート・ワードを指定します。このタグは、対応する HELP タグのすぐ後に指定します。 1 つのヘルプ・モジュールには、1 つの ISCH タグを付けられます。

それぞれの ISCH タグには、数行のルート・ワードを付けることができます。ただし、ルート・ワードは 50 文字以内です。複数行のルート・ワードを使用する場合は、2 行目以降の行頭に ROOTS= が必要です。

```
 :PNLGRP.
 :HELP name=entry1.
 :ISCH ROOTS='root1 root2 root3 root4 root5'
ROOTS='root6 root7 root8 root9 root10'
ROOTS='root11 root12 root13 ... root50'.
Title of First Topic

This is the first index search module in this panel group.
 :EHELP.
 :EPNLGRP.
```

ルート・ワードは、どの行の場合もアポストロフィで囲みます。また、ルート・ワードの最後の行の末尾にはピリオドを入力します。トピックの表題は ISCH タグのピリオドの後に続けます。また、ピリオドのすぐ次の行に置くこともできます。

ISCHSYN タグ

ISCHSYN タグは、ユーザーが入力した特定のルート・ワードと一致させる単語 (同義語) を定義します。ユーザーが入力した単語がルート・ワードの同義語の場合は、そのルート・ワードを含む ISCH タグに一致するトピックを検索します。

ルート・ワードとして使用する単語を同義語としても使用する場合は、その単語を ISCHSYN タグの同義語として含める必要があります。たとえば、次のようにします。

```
 :ISCHSYN ROOT='ocean'.ocean water sea
```

ISCHSYN タグの同義語は、同じ行に入力する必要があり、各ルート・ワードに少なくとも 1 つの ISCHSYN タグが必要です。行数が複数行になる場合は、同じルート・ワードに ISCHSYN タグを複数入力できます。

UIM では、同義語を大文字だけ、小文字だけ、あるいは大文字小文字混合で入力しても違いはありません。このため、大文字小文字を区別するために、複数の同義語を入力する必要はありません。

同義語には英字と数字を使用できますが、次の文字は (16 進数によるそれぞれの等価文字を含む) 同義語として使用したり、同義語の一部に使用することはできません。

- . (ピリオド)
- ((左括弧)
-) (右括弧)
- ; (セミコロン)

- , (コンマ)
- ? (疑問符)
- : (コロン)

ISCHSYN タグは、パネル・グループのどこにでも置けますが、保守や翻訳などの作業を簡単にするために、同じ場所にまとめてください (パネル・グループの先頭、あるいは ISCHSYN タグだけのパネル・グループ・オブジェクトなど)。

例: ISCH と ISCHSYN の使用方法

次の例は、ISCHSYN および ISCH のタグの例です。

```
  :PNLGRP.
 :ISCHSYN ROOT='ocean'.ocean water sea
 :ISCHSYN ROOT='lake'.lake water pond
 :ISCHSYN ROOT='definition'.definition define description what
 :ISCHSYN ROOT='definition'.summary concept information explanation
 :HELP name='defocean'.
 :ISCH ROOTS='definition ocean'.
Definition of ocean

An ocean is one of the five large bodies of salt water, which
together cover nearly three-fourths of the world.
 :EHELP.
 :HELP name='deflake'.
 :ISCH ROOTS='definition lake'.
Definition of lake

A lake is a body of standing water that is enclosed by land.
 :EHELP.
 :EPNLGRP.
```

索引検索と DBCS: 索引検索機能は、2 バイト文字 (DBCS) と 1 バイト文字 (SBCS) のデータに使用できます。 DBCS データを使用するときは、要求を出す装置が DBCS データを入力、処理できなければなりません。索引検索データを含むオブジェクトには、DBCS データが含まれていることを示す記号が付きます。装置が DBCS データを処理する能力があるかどうかは、システムが判別します。

DBCS 形式で準備したデータで索引検索機能を使用するときには、次のことに注意してください。

- 索引検索データが DBCS システム用に準備されている場合は、ISCHSYN タグに入力する同義語は、2 バイト文字で入力してください。つまり、タグの次の最初のバイトはシフトアウト文字、そしてデータの最後のバイトはシフトイン文字にします。システムは、ISCHSYN タグ上のデータを 2 バイト文字データには変換しません。
- 単語の間に 1 バイトのブランクを入れて、単語を分離します。 1 ～ 19 個の 2 バイト文字を組み合わせて単語を形成できます。シフトアウト文字とシフトイン文字を間にはさむことはできますが、索引検索では無視されます。
- ISCH と ISCHSYN のタグをリンクするのに使用する単語 (ISCH タグの ROOTS 属性と ISCHSYN タグの ROOT 属性) には同一の単語を使用して、入力には DBCS を使用しないでください。
- 検索語の入力には、1 バイトまたは 2 バイトのいずれかを使用できます。単語と単語の区切りには、1 バイトのブランクを入力できます。

検索語は、画面では 2 バイト文字 (検索に実際に使用される文字) で表示されます。索引検索には、大文字小文字を区別しないように、特別な処理が行われます。ISCHSYN タグの検索語は、PNLGRP タグの TXTCHRID 属性で指定されたコード・ページの変換テーブルを使用して、大文字に変換されます。検索語が DBCS の場合は、大文字には変換されません。シフトアウト文字とシフトイン文字は、構文解析のとき

にはブランクとして処理されて、先頭および末尾のブランクは削除されます。すべての SBCS 単語は、装置記述コード・ページの変換テーブルを使用して、大文字に変換されます。

プログラム・メッセージの設計: OS/400 上のメッセージには、事前定義メッセージと即時メッセージがあります。メッセージの設計やコーディングには、次のことを考慮してください。

- 即時メッセージは使用しないでください。即時メッセージは、送信元またはプログラムが送信時に作成し、メッセージ・ファイルには保存されません。 このため、翻訳者が即時メッセージを訳すことはできません。
- 次の 2 つの条件を満たす事前定義メッセージ記述を使用してください。
  - メッセージを使用するプログラムの外に存在する。
  - メッセージ・ファイルに保存できる。
- メッセージ・ファイルの最大サイズを指定しないでください。メッセージ・ファイルが満杯になったときに、メッセージ・ファイルのサイズを変更できなくなります。 ほかのメッセージ・ファイルを作成して、このファイルにメッセージ記述をもう一度追加しなければならなくなります。

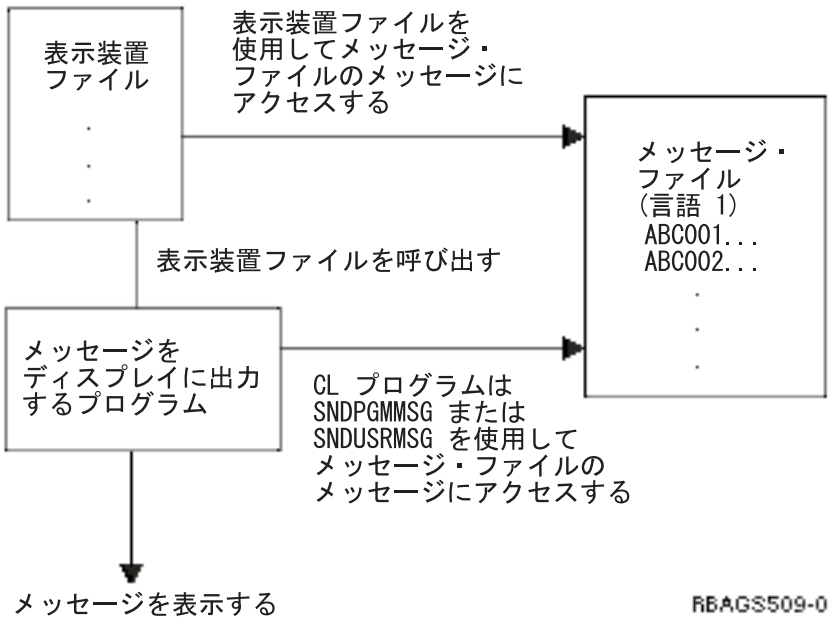
  メッセージ・ファイルを作成して、事前定義メッセージ記述を維持するには、「メッセージ・ファイルの作成」 (CRTMSGF) コマンドを使用してください。 「メッセージ記述の追加」 (ADDMSGD) コマンドを使用すると、事前定義メッセージ記述の内容をメッセージ・ファイルに入れることができます。 詳しくは、制御言語 に関する情報を参照してください。
- 置換変数の扱いには注意が必要です。置換変数の順序は、言語により異なります。 たとえば、英語のメッセージでは、次のようになります。

  ```
  File &1 in Library &2 not found.
  ```

  &1; と &2; が置換変数です。この置換変数は、言語により表示される場所が異なります。
- 異なる言語の応答コードを理解できるような設計とコーディングを使用してください。 たとえば、次のようになります。

  ```
  English     Y = Yes
  Danish      J = Ja (means Yes)
  ```

次の図は、メッセージ・ファイルからさまざまな NLV メッセージを作成する方法を示しています。



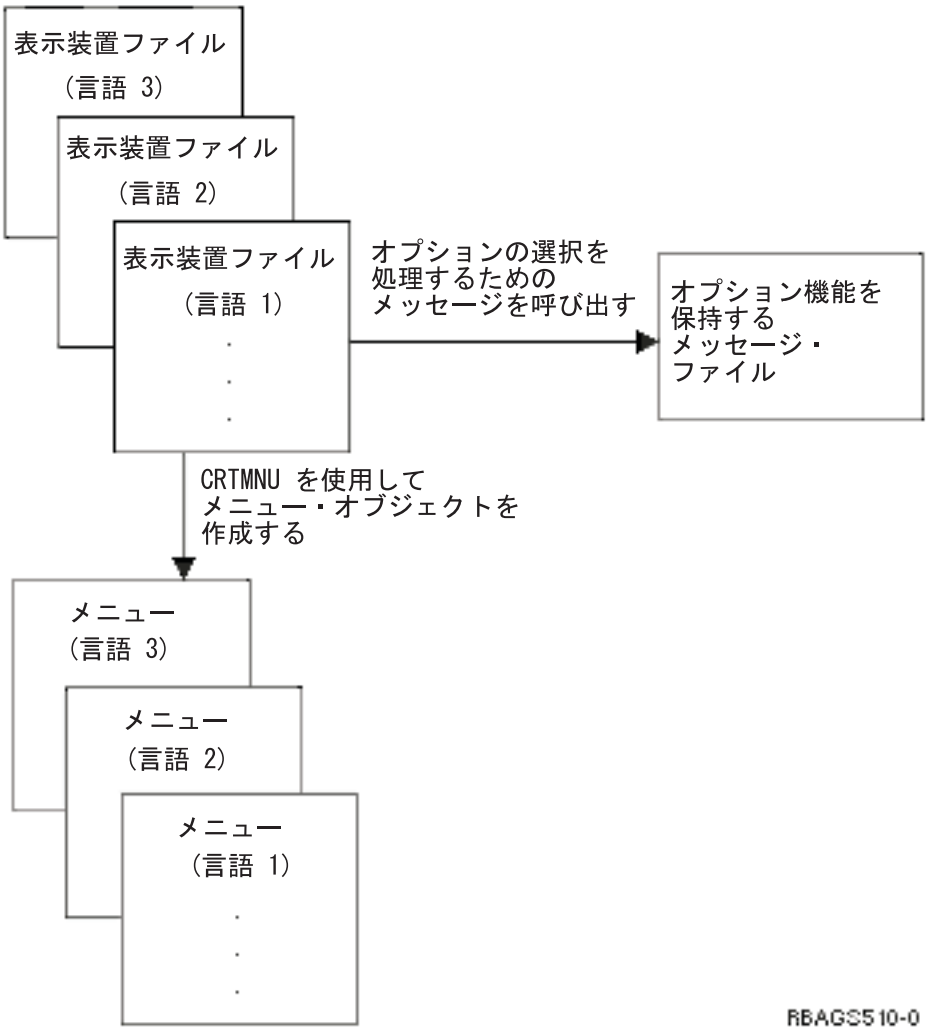
プログラムは、プログラム・メッセージのメッセージ・ファイルには直接アクセスできます。または、プログラム・メッセージの表示装置ファイルを使用して、メッセージ・ファイルに間接的にアクセスできます。メッセージ・ファイルについて詳しくは、メッセージに関する CCSID サポート を参照してください。

メニューの設計: OS/400 上では、独自のメニューを定義できます。ユーザー定義のメニューには、表示装置ファイル・メニュー、UIM (参照) メニュー、およびプログラム・メニューの 3 種類があります。

アプリケーション・システムを使用するには、ユーザーはたくさんのメニューや表示画面を操作する必要があります。アプリケーションを他の言語に翻訳するときに、翻訳するリテラル・テキストの多くはメニュー項目です。

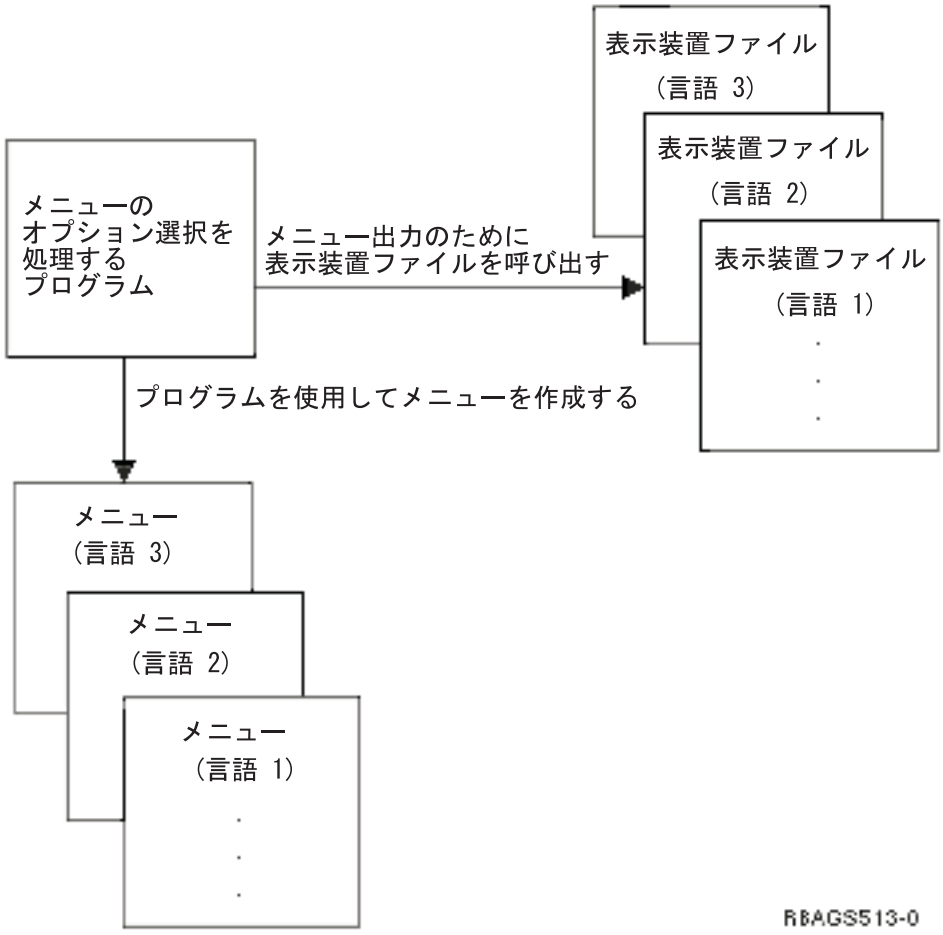
表示装置ファイル・メニュー

表示装置ファイル・メニューは、DDS によって定義されるディスプレイを使用して、メニュー形式を表示します。メニュー機能は、メニュー・オブジェクトによって制御されます。メッセージ・オブジェクトには、メニュー・オプションを実行するコマンドが含まれています。次の図は、さまざまな各国語バージョンでの表示装置ファイル・メニューの構成を示しています。



プログラム・メニュー

プログラム・メニューは、プログラムを使用して、メニュー形式 (DDS が定義します) を表示し、メニュー・オプションを実行するのに必要な機能を提供します。次の図は、さまざまな各国語バージョンでプログラム・メニューがどのように作成されるかを示しています。



メニューの翻訳

各国語バージョンのメニューの翻訳を容易にするため、次のようにしてください。

- メニューのリテラル・テキストは外部に置きます。このためには、定数テキストを外部定義のメッセージ記述としてメッセージ・ファイルに保管し、プログラムの実行時にメニュー・ファイルに組み込みます。
- メニューを他の言語に翻訳すると、スペースが余分に必要になることがあるので注意してください。メニューを設計するときには、翻訳で余分に必要になるスペースを確保してください。
- メニューに日付、時刻、または編集フィールドを表示するときには、国別情報に注意してください。
- オプション・フィールドの選択項目には、英語の大文字や小文字 (A ～ Z) を使用せずに、0 ～ 9 の数字を使用してください。さまざまな言語では、数値の方が標準的です。

コマンドの設計: OS/400 プログラムでは、ユーザー独自のコマンドを定義、作成できます。コマンドを作成するには、まずはじめに、コマンド定義ステートメントを使用して、コマンドを定義します。 次に、「コマンドの作成」 (CRTCMD) コマンドを使用して、コマンド定義ステートメントを処理し、コマンド定義オブジェクトを作成します。

コマンドを定義、作成するときには、次のことを考慮してください。

- ヘルプ・パネル・グループを使用して、コマンドのオンライン・ヘルプ情報を作成してください。各国語バージョンのヘルプ・パネルについては、 16 ページの『ユーザー・インターフェース』を参照してください。
- CL CMD、PARM、ELEM、および QUAL のコマンド定義ステートメントの PROMPT キーワードには、リテラル・テキストではなく、メッセージ識別コードを使用してください。
- プロンプトに表示される各パラメーターのプロンプト行の右側に表示されるテキストを翻訳してください。このテキストは、PARM コマンド定義の CHOICE パラメーターにより指定されるので、コマンド・プロンプトの表示方法について整合性をとれます。
- それぞれの各国語について、コマンド・プロンプト・テキストを個別のコマンド定義オブジェクト・バージョンにコンパイルします。コマンドを作成する前に、「システム・ライブラリー・リストの変更」(CHGSYSLIBL) コマンドを使用して、正しい各国語バージョン・ライブラリーから各国語バージョンのプロンプト・テキストを獲得してください。
- コマンド・プロンプト表示のファンクション・キーは、OS/400 プログラムが提供します。 OS/400 プログラムの NLV がコマンドの NLV と異なる場合は、コマンド・プロンプト表示に 2 つの言語が表示されます。たとえば、英語の表示をドイツ語に翻訳すると、コマンド・プロンプト表示に英語とドイツ語の両方が表示されます。

  コマンドの作成および定義については、制御言語 にも情報があります。

文化依存型の設計: NLS 対応アプリケーションを開発するときには、各国のさまざまな標準を考慮する必要があります。このような国別情報は、テキスト・データの処理と同様に、プログラムの外で対応してください。

A から Z のアルファベット以外の文字を使用する言語がたくさんあります (単語のスペルを正しくするために必要な共通母音など)。照合処理を実行するには、このことを考慮する必要があります。

システムは、言語、文化、およびデータ順序などをサポートするためにシステム値を使用します。それぞれの各国語バージョンのデフォルト・システム値については、デフォルト・システム値 を参照してください。

次のトピックには、文化依存型のアプリケーションを設計するときに考慮すべき属性について説明があります。

- データベース・ファイル属性
- ジョブ属性
- プログラム属性
- メッセージ CPX8416 内の情報
- 日付形式
- 日付区切り記号
- 日付表示の編集
- 時刻形式
- 時刻区切り記号
- 時刻表示の編集
- 小数点形式
- NLS 分類順序

データベース・ファイル属性: 文化依存型データベース属性には、次の属性があります。

- コード化文字セット識別コード (CCSID)
- 分類順序 (SRTSEQ)
- 言語識別コード (LANGID)

CCSID 属性は、物理ファイルだけに適用されます。SRTSEQ および LANGID の属性は、物理ファイルと論理ファイルの両ファイルに使用できます。論理ファイルに CCSID が存在するのは、物理ファイルから CCSID を引き継いだときだけです。データベース属性は、データとともに保存されます。これらの属性は、データにアクセスして動的に変更することができないので静的な属性といえます。

ジョブ属性: 文化依存型ジョブ属性には、次の属性があります。

- コード化文字セット識別コード (CCSID)
- 分類順序 (SRTSEQ)
- 言語識別コード (LANGID)
- 国別または地域別識別コード (CNTRYID)
- 日付形式 (DATFMT)
- 日付区切り記号 (DATSEP)
- 小数点形式 (DECFMT)
- 時刻区切り記号 (TIMSEP)

CCSID、SRTSEQ、LANGID、および CNTRYID の各属性のデフォルト値は、ジョブが開始されたときにユーザー・プロファイルから設定されます。CCSID、DATFMT、DATSEP、DECFMT、SRTSEQ、および TIMESEP の値は、ユーザー・プロファイルに対応する LOCALE および SETJOBATR の属性から設定できます。 ジョブ変更コマンド (CHGJOB) を使用すると、これらのジョブ属性に指定された値をオーバーライドできます。

プログラム属性: SRTSEQ および LANGID のパラメーターは、*PGM オブジェクト・タイプに属するプログラム属性として指定することもできます。LANGID パラメーターは、SRTSEQ 値が *LANIDUNQ または *LANGIDSHR に設定されている場合に限り、SRTSEQ パラメーターとともに使用できます。これ以外の場合では、LANGID パラメーターは使用されません。

プログラムが分類順序または言語識別コードに明示的な参照を行うと、プログラムに保存されているそれらの属性が有効になります。 プログラムを実行しているジョブの属性を参照するには、これらのパラメーターの *JOBRUN 値を使用します。 *JOBRUN を使用すると、複数のプログラムを組み合わせて、さまざまな分類順序によるデータ処理が可能になります。*JOBRUN 値は、データ処理に影響を与えますが、データの読み取りシーケンスには影響しません。読み取りシーケンスは、データベース属性により決定されます。データベースに定義されている分類順序とは異なる分類順序でデータを読み取るには、個別に構築した論理ファイルを使用します。

メッセージ CPX8416 内の情報: アプリケーションをほかの言語に変換する場合は、QCPFMSG メッセージ・ファイルに含まれている CPX8416 というメッセージを使用して、ほかの言語の文化値を正しく設定してください。1 次言語ライブラリー用、およびインストールされているすべての 2 次言語ライブラリー用のメッセージがあります。 システム・メッセージには、次の値が含まれます。

- コード・ページと文字セット
- 通貨記号
- 日付形式
- 日付区切り記号
- 小数点形式

- うるう年調整
- コード化文字セット識別コード
- 時刻区切り記号
- 言語識別コード
- 国別または地域別識別コード

パネルやディスプレイ上の文化依存型フィールドには、ハードコーディングされた値を含むことはできません。これらのフィールドは、画面上のフィールド最大表示長さに従って定義します。

使用するアプリケーションで 1 次言語以外の言語を使用してユーザーをサポートする場合は、呼び出し可能ルーチンに CPX8416 のメッセージ値を使用してください。呼び出し可能ルーチンは、1 次言語の文化値を使用してフィールドの内容 (日付形式など) を決定し、その値を画面に表示します。文化依存型フィールドに表示される文化値の形式は、メッセージ CPX8416 に維持されている NLS システム値が決定します。

アプリケーションは、システム・メッセージの詳細情報を使用できます。

次の表は、メッセージ CPX8416 のレイアウトです。この例では、テキスト欄の値に 英語の大文字と小文字 による NLV (機能 2924) を使用しています。

	フィールド	開始	長さ	位置調整
記述 値	QCHRID 697 37	0001 0012	10 21	L L
記述 値	QCURSYM $	0034 0045	10 01	L L
記述 値	QDATFMT MDY	0047 0058	10 03	L L
記述 値	QDATSEP /	0062 0073	10 01	L L
記述 値	QDECFMT	0075 0086	10 01	L L
記述 値	QLEAPADJ 0	0088 0099	10 01	L L
記述 値	QCCSID 37	0101 0112	10 05	L L
記述 値	QTIMSEP :	0118 0129	10 01	L L
記述 値	QLANGID ENU	0131 0142	10 03	L L
記述 値	QCNTRYID US	0146 0157	10 02	L L
記述 値	QIGCCDEFNT *NONE	0160 0171	10 21	L L

日付形式: 日付表示形式に国際標準はありません。そこで、日付形式は、テキスト・データの一部として外部に保存しておく必要があります。OS/400 では、次の日付形式が有効です。

- *MDY (月、日、年)

- *DMY (日、月、年)
- *YMD (年、月、日)
- *JUL (yy/ddd)
- *ISO (YYYY-MM-DD)
- *USA (MM/DD/YYYY)
- *EUR (DD.MM.YYYY)
- *JIS (YYYY-MM-DD)

注: OS/400 には、上に示された日付形式の一部を使用できない機能があります。

データベース・ファイルでは、日付を次のように保存できます。
- 通常の数値データ・フィールド
- SAA日付データ・タイプ

日付を数値データで保存する場合は、日付の保存形式と表示形式をアプリケーションに指定する必要があります。

日付を DATE (L) のデータ・タイプで保存する場合は、データベース・ファイル上で DDS キーワード DATFMT を使用して形式を指定できます。この事前定義形式では、日付区切り記号も含めて、文字データとして日付が表示されます。

日付の保存やその他の処理が必要な場合は、日付を *ISO 形式 (YYYY-MM-DD) で保存してから、入出力時にほかの形式に変換します。日付を変換するには、高水準言語ルーチンを作成してください。

日付区切り記号: 次の日付区切り記号が有効です。
- / (スラッシュ)
- - (ダッシュ)
- . (ピリオド)
- , (コンマ)
- (ブランク)

表示用の日付区切り記号は、テキスト・データとして常に外部に保存します。

日付に 10 進数フィールドを使用した場合、アプリケーションは、形式を指定する以外に、入力操作や表示のときに日付区切り記号の処理も実行する必要があります。

日付タイプ・フィールドを使用すると、日付には常に日付区切り記号が組み込まれます。日付区切り記号を変更するには、高水準言語のルーチンを作成して日付を変換してください。

日付表示の編集: 表示装置ファイルと印刷装置ファイルは、日付の保存方法に従って、日付表示の処理が異なります。
- 通常の 10 進数データ・フィールドの場合

  日付の入力方法、保存方法、および表示方法は、使用するアプリケーション・プログラムで処理します。アプリケーションは、入力した日付形式が正しいかどうかを確認し、日付区切り記号を削除し、必要に応じて日付形式を変換し、さらに表示装置ファイルや印刷装置ファイル上で日付を編集する必要があります。

DDS キーワードの DATE は、出力専用フィールドとして使用します。DATE では、DATE、DATFMT、および DATSEP のジョブ属性を使用します。6 桁または 8 桁の日付フィールドでは、編集コード・キーワードの EDTCDE を使用して DATE を編集できます。

EDTCDE を使用して編集すると、指定する編集コードに従って、表示フィールドの表示方法を次のように変更できます。

- 先行ゼロを抑止する
- ゼロの値をゼロまたはブランクとして表示する
- ユーザー定義の編集コードを使用して、フィールドをさらに高度に編集する

EDTCDE Y キーワードを使用するその他のすべてのフィールドについては、プログラムが形式を指定する必要があります。これにより、装置ファイルを作成したジョブの日付区切り記号をシステムが使用します。日付区切り記号は、オブジェクト内に統合されるので、実行時に動的に変更することはできません。

- SAA データ・タイプ DATE (L) フィールドの場合

  DDS 日付形式の (DATFMT) キーワードを使用すると、デフォルトの日付区切り記号を含めて、データベース・フィールド・レベルで異なる日付形式を指定できます。*MDY、*DMY、*YMD、および *JUL のパラメーターについては、日付区切り記号の (DATSEP) キーワードを使用して、デフォルト日付区切り記号を変更できます。*ISO、*USA、*EUR、および *JIS の値の区切り記号は固定です。また、これらの値には DATSEP キーワードは使用できません。DATFMT と DATSEP のキーワードを使用すると、日付フィールドを保存するための形式と編集文字を指定できます。日付は、区切り記号を含めて、文字ストリングとして表示されます。

  入力された日付とデータベースが必要とする形式は、次の項目により変換されます。

  - アプリケーション・プログラムのルーチン
  - 異なる日付形式と日付区切り記号を定義する論理ファイルのフィールド・マッピング

  たとえば、次の CL プログラムを使用すると、実際のジョブ属性に依存する日付変換の処理を実行できます。

```
PGM       PARM(&fromfmt &fromfld  &tofld  );
DCL       VAR(&fromfmt); TYPE(*CHAR)  LEN(4)
DCL       VAR(&fromfld); TYPE(*CHAR)  LEN(10)
DCL       VAR(&tofld); TYPE(*CHAR)  LEN(10)
CVTDAT    DATE(&fromfld); TOVAR(&tofld);
FROMFMT(&fromfmt); TOFMT(*JOB)  TOSEP(*JOB)
ENDPGM
```

必要な日付形式と日付そのものをアプリケーション・プログラムから CL プログラムに渡す必要があります。CL プログラムは、ユーザーが期待する日付フィールドの編集方法をジョブ属性が表していると仮定します。これらの値を読み取って、値に従って変換を行い、その形式で日付を返します。
*ISO、*USA、*EUR、および *JIS の値の区切り記号は固定なので、変更はできません。TOFMT パラメーターにこれらのいずれかの値が含まれていると、TOSEP の値は無視されます。

時刻形式: OS/400 では、次の時刻形式を使用できます。

- *HMS (hh:mm:ss)
- *ISO (hh.mm.ss)
- *USA (hh:mm AM または hh:mm PM)
- *EUR (hh.mm.ss)
- *JIS (hh:mm:ss)

システム値の QTIME の形式は 1 つだけです (hhmmss)。時刻区切り記号値は、QTIMSEP システム値が決定します。

表示用の時刻形式は、テキスト・データとして常に外部に保存します。

データベース・ファイルでは、時刻を次のように保存できます。

- 通常の数値データ・フィールド
- SAA 時刻データ・タイプ

時刻を数値データで保存する場合は、日付の保存形式と表示形式をアプリケーションに指定する必要があります。

時刻を TIME (T) のデータ・タイプで保存する場合は、データベース・ファイル上で DDS キーワードの TIMFMT を使用して形式を指定できます。この事前定義形式では、時刻区切り記号も含めて、文字データとして時刻が保存されます。時刻フィールドの形式を変更するには、CL プログラムを作成するか、または高水準言語のルーチンを作成して変換してください。

時刻区切り記号: OS/400 では、次の日付区切り記号が有効です。

- : (コロン)
- . (ピリオド)
- (ブランク)
- , (コンマ)

表示用の時刻区切り記号は、テキスト・データとして常に外部に保存します。

タイム・フィールドに 10 進数データ・フィールドを使用すると、アプリケーションは、入力操作や表示操作のために、形式と時刻区切り記号を指定しなければなりません。

時刻タイプ・フィールドを使用すると、時刻フィールドに常に時刻区切り記号が含まれます。 時刻区切り記号を変更するには、CL プログラムを作成するか、または高水準言語のルーチンを作成して変換をしてください。

時刻表示の編集: 表示装置ファイルと印刷装置ファイルは、時刻の保存方法に従って、時刻表示の処理が異なります。

- 10 進数データ・フィールドの場合

  時刻の入力方法、保存方法、および表示方法は、使用するアプリケーション・プログラムで処理します。形式の確認、時刻区切り記号の削除、必要に応じた時刻形式の変換、および表示装置ファイルと印刷装置ファイルでの編集は、プログラムが実行します。

  編集は、フィールドに「単語の編集」 (EDTWRD) を指定して行います。TIME キーワードは、出力専用フィールドです。「単語の編集」および TIME キーワードは、ともに作成時の情報を使用します。時刻区切り記号は、装置ファイル・オブジェクトに統合されます。

  どちらの方法でも、さまざまな編集要件について、ソースとオブジェクトのコピーが必要になります。

- SAA データ・タイプ TIME (T) フィールド

  OS/400 プログラムでは、データベース・ファイル・レベルで、さまざまな時刻形式や時刻区切り記号を使用できます。TIME キーワードを使用すると、時刻フィールドを保存するときに、形式や編集文字を指定できます。時刻のタイプ・フィールドは、区切り記号を含む文字ストリングとして表示されます。

  表示装置ファイルまたは印刷装置ファイルに、そうした時刻フィールドを SAA データ・タイプの通常の文字フィールドとして指定できます。入力操作では、プログラムが入力値の形式や区切り記号が正し

いかどうかを確認して、データベース・フィールドに移動します。出力操作では、区切り記号を含めて文字ストリングをデータベース・ファイル・フィールドから装置ファイル・フィールドに移動します。入出力の時刻とデータベースが必要とする形式は、次のどちらかによって変換されます。

- アプリケーション・プログラムのルーチン
- 異なる時刻形式と時刻区切り記号を定義する論理ファイルのフィールド・マッピング

小数点形式: » QDECFMT システム値を使用して、それぞれの国や地域で使用されている小数点形式に応じて、小数点形式を変更できます。 «

分類順序: OS/400 では、分類順序がサポートされています。次のオプションを使用すると、文化に依存する個々のアプリケーションの要件に応じたデータの順序付けができます。

- 16 進法分類 (分類順序テーブルは使用しません)。これがデフォルトです。
- SRTSEQ パラメーターにより、ユーザー提供またはシステム提供の共用重みによる分類順序テーブルか、あるいは固有重みの分類順序テーブルが決定されます。

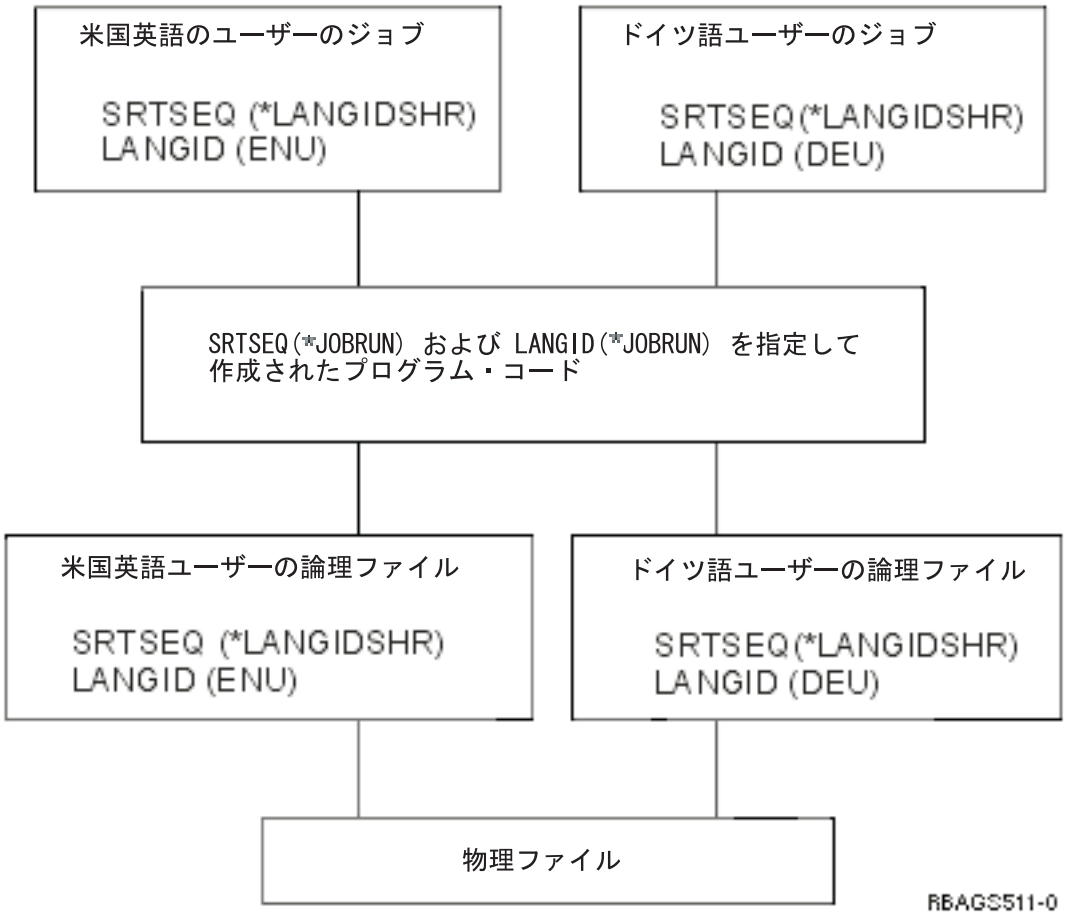
次の例は、1 つの DDS ソース・ファイルを使用して、異なる分類順序のデータベース・ファイルを作成する方法を示しています。次の操作で処理できます。

```
CRTxF     FILE(*CURLIB/NAME)
SRTSEQ(*JOB)
LANGID(*JOB)
```

次に、ジョブ属性を変更して、異なる分類順序のファイルを作成します。

分類順序の早期バインディングまたは実行時バインディングを指定して、CL プログラムおよび高水準言語のプログラムを作成できます。分類順序の早期バインディングを使用すると、使用する分類順序テーブルはコンパイル時に決定されます。分類順序の実行時バインディングを使用すると、使用する分類順序テーブルは実行時に決定されます。

実行時バインディングでは、異なる各国語環境で 1 組のプログラムを使用できます。次の図には、異なるジョブについて、物理ファイルとプログラム・コードの 1 つの組み合わせを使用したときのさまざまな分類順序を示しています。ジョブに定義する分類順序テーブルとプログラムが使用する分類順序テーブルは、ライブラリー・リストを通してアクセスする論理ファイルに割り当てられる分類順序テーブルと同一 (または互換) である必要があります。



さまざまな分類順序を使用するための設計

プログラムでさまざまな分類順序を使用する場合は、次のことを考慮してください。

- データを異なる順序で表示する。
- さまざまなレコードを処理する。

  「以下」や「以上」などの選択基準を指定すると、さまざまなレコードが選択される可能性があります。共用重み分類順序テーブルを使用している場合に、「等しい」の選択基準を使用すると、さまざまな数のレコードが選択されることがあります。
- 条件付き分岐の処理が異なる場合があります。

注: 分類順序を使用しても、システム・リスト (WRKOBJ コマンドの出力など) は影響を受けません。

順序付けテーブル、およびこのテーブルが含まれているライブラリーを指定するには、DDS ファイル・レベルのキーワード代替シーケンス (ALTSEQ) を使用します。 代替照合シーケンスを定義するには、共用および固有の重みが付けられたシステム提供の分類順序テーブルを使用します。

代替照合シーケンス・テーブルは、コンパイル時にファイルに挿入されるので、実行時には必要ありません。1 組の DDS を使用して、さまざまな照合シーケンスを含む複数のファイルを作成できます。

注: データベース・ファイルに定義した代替照合シーケンスは、アプリケーション・プログラムにも定義します。これを定義しないと、予期せぬ結果が生じる場合があります。

DDS キーワードの ALTSEQ の順序付け機能には制限があります。選択 / 省略の論理にはほとんど影響を与えません。ALTSEQ キーワードを使用できるのは、CRTPF および CRTLF コマンドで SRTSEQ(*SRC) パラメーターを使用したときだけです。

関連情報

分類順序について詳しくは、グラフィック文字 (データ) 分類機能の実装 を参照してください。

表示装置ファイルの設計: 通常のアプリケーションのパネルの主な要素は次のとおりです。

- 定数テキスト・ストリング
- 入出力フィールド
- フィールド編集指定項目
- カーソル位置指定
- 入力フィールド用のデフォルト値
- フィールド妥当性検査の指定項目
- エラー・メッセージ

以上は、DDS を使用して、プログラム記述のファイルまたは外部記述のファイルとして処理できます。上記のトピックでは、DDS を使用した外部記述の方法を説明します。

定数テキスト・ストリング: パネルを設計するときには、同じことを表現する場合でも、言語によってスペースの使用方法が異なることを考慮してください。1 つの行にたくさんのフィールドを置かないでください。ただし、フィールド・プロンプトの代わりに列のヘッディングがあるリスト・パネルは例外です。パネルには過多の情報を入れないようにしてください。パネルの作成には、 20 ページの『テキスト・データ・コードの設計』 に 示されているいずれかのテクニックを利用してください。

入出力フィールド: フィールドを定義するときは、アプリケーションの使用対象となる言語、国、文化、通貨、法規制などの要件に従ってください。たとえば、イタリアのリラと日本の円をアメリカのドルと同じフィールドに保存するとします。この場合、フィールド・サイズは、桁数が最も長くなるイタリアのリラに合わせる必要があります。

フィールド編集指定項目: 数値、日付、時刻などのフィールドで指定項目を編集するときは、対象となるユーザーの国別情報を考慮してください。アプリケーション・プログラムで形式をコード化したり、命令を編集するときには、ほかの規則が必要になったときにプログラムの変更が必要にならないように注意してください。詳しくは、文化依存型の設計 を参照してください。

カーソル位置の指定: カーソル位置については、言語によりスペース要件が異なるので、画面上の固定位置に指定しないでください。それぞれの表示装置ファイルを処理するときに、翻訳処理に合わせて位置を調整できます。 フィールドとは独立したカーソル位置が必要な処理では、コードの外に位置情報を保存し、プログラム内でキーワードの可変値を読み取ってください。

たとえば、次のようにします。

```
A     record-name      CSRLOC(field-name-1 field-name-2)
```

NLS 環境では、カーソル位置をフィールド・レベルにしたほうが便利です。フィールド・レベルにするには、通常のレコードの場合は、特定のフィールドで DSPATR(PC) キーワードを指定します。 サブファイルの場合は、特殊位置決めフィールドで SFLRCDNBR(CURSOR) を使用して、カーソル位置を指定します。さらに、形式を書き込む前に、サブファイル・レコードのレコード番号をそのフィールドに保存する必要があります。

たとえば、次のようにします。

```
A     field-name      4S 0B  line pos  SFLRCDNBR(CURSOR)
```

注: カーソルが置かれているレコードとフィールドの名前、サブファイルの相対レコード番号、およびサブファイルのフォールド / 切り捨て標識を、使用するアプリケーション・プログラムに戻すことができます。この機能は、DDS キーワードの RTNCSRLOC、 SFLCSRRRN、および SFLMODE 上の隠しフィールドが提供します。

入力フィールドのデフォルト値: 表示画面の入力フィールドにデフォルト値を設定するには 3 つの方法があります。デフォルト値は、ユーザーがオーバーライドできる値です。

- プログラムから情報を得る

  言語や文化に依存する値の場合は、値をリテラルとしてハードコーディングしないでください。システム日付やジョブ日付などのシステム提供情報の値を得るか、あるいは、プログラムの外のデータベース・ファイルやデータ域などのデータ・オブジェクトから値を得てください。

- DDS キーワードの DFT (デフォルト) または DFTVAL (デフォルト値) を使用する

  DDS 上でキーワードの後にデフォルト入力値を直接指定します。DDS キーワードの DFT は、入力専用 (I) フィールドに使用します。出力専用 (O) フィールドまたは入出力 (B) フィールドでは、DFTVAL キーワードを使用します。

  たとえば、次のようにします。

  ```
  A     field-name   length type I   line pos  DFT('default    ')
  または
  A     field-name   length type O/B line pos  DFTVAL('default value  ')
  ```

- DDS キーワード MSGID (メッセージ識別) を使用する

  「メッセージ識別」キーワード (MSGID) を使用すると、プログラム実行中に指定したメッセージ記述の内容を読み取ったり、表示画面のファイル・フィールドにデフォルト値を入力できます。この方法を使用するには、フィールドを入出力可能 (B) にしておく必要があります。

  たとえば、次のようにします。

  ```
  A   field-name  length type B line pos  MSGID(message-id [*libl/message-file)
  ```

  これで、プログラム実行中に、設定したライブラリー・リストに従って、各国語バージョンごとに複数のメッセージ・ファイルを使用できます。

フィールド妥当性検査の指定項目: 次の DDS キーワードは、表示画面上の入力可能なフィールドで妥当性検査を実行します。

- RANGE (範囲検査)
- VALUES (値検査)
- CMP および COMP (比較)
- CHECK (妥当性、キーボード制御、およびカーソル制御の検査)

言語、国、または文化に依存するハードコーディング値に DDS キーワードを使用する場合は、DDS およびアプリケーション・プログラムを複写して変更を加える必要があります。

例: 妥当性検査

VALUES、COMP、および CHECK の DDS キーワードを使用した入力可能フィールドのフィールド妥当性検査の例を以下に示します。

```
A     field-name   length type usage line pos  VALUES('Y' 'N')
または
A     field-name   length type usage line pos  COMP(EQ 'US$')
または
A     field-name   length type usage line pos  CHECK(M10 or M11)
(Modulus checking)
または
A     field-name   length type usage line pos  CHECK(RL)
(Right-to-left support)
```

妥当性検査は、表示装置ファイル作成時に定義した分類順序に従って行われます。同じ DDS ソース・ファイルを使用して、異なる言語のオブジェクトを作成できます。たとえば、次のコマンドは、Latin 1 分類順序テーブルにタグ付けされたディスプレイ・オブジェクトを作成します。

```
CRTDSPF FILE(name) SRTSEQ(*LANGIDSHR) LANGID(DEU)
```

次のように指定すると、

```
A     field-name   length type usage line pos  COMP(EQ 'a')
```

Latin 1 分類順序の共用重みの定義に従って、小文字、大文字、およびアクセント記号付きの文字をすべて受け入れます。

また、これらの DDS キーワードを使用して指定した検査は、OS/400 プログラムのデータ管理機能が実施します。ユーザーの入力間違いや処理間違いなどによるエラー・メッセージは、OS/400 プログラムの言語で表示されます。表示される言語は、ジョブのライブラリー・リストのセットアップに従って、1 次言語または 2 次言語となります。

DDS キーワードのCHKMSGID (メッセージ識別コードのチェック) を使用すると、この機能をオーバーライドできます。このキーワードを使用すると、OS/400 プログラムのルーチンの検査に、カスタマイズしたメッセージやメッセージ・ファイルを使用するように指定できます。

たとえば、次のようにします。

```
A     field-name   length type usage RANGE(1 999)
A                                    CHKMSGID(USR1234 [*libl/]APPMSGF
[&MSGFLD1])
A     MSGFLD1      length type   P   TEXT('Message data field')

および
ADDMSGD   MSGID(USR1234) MSGF(APPTXDENU/APPMSGF)
          MSG('Value &1; is out of range 1 to 999')
および

ADDMSGD   MSGID(USR1234) MSGF(APPTXDDEU/APPMSGF)
          MSG('Wert &1; ist ausserhalb des gltigen Bereichs 1 bis 999')
```

異なるライブラリー名の異なるメッセージ・ファイルを使用する場合は、固定のライブラリー名を指定しないでください。プログラムを実行するときに、ライブラリー・リストを設定すると、異なる言語のメッセージ・ファイルを使用できます。

エラー・メッセージ: 表示装置ファイルにエラー・メッセージを送る方法は 2 つあります。

- ERRMSG または SLFMSG のキーワードにテキストを定数として指定する方法

  DDS キーワードにテキストを定数として直接指定します。複数の言語を使用する場合は、DDS 仕様の中で DDS ソース・コードのコピーを作成し、定数を変換する必要があります。次に、それぞれの言語について、個別の表示装置ファイルを作成します。
- ERRMSGID または SLFMSGID のキーワードの事前定義メッセージを使用する方法

  定数の代わりに事前定義メッセージを使用するには、複数の表示装置ファイルが必要になります。

複数の異なる表示装置ファイルを使用せずに、使用する言語に従ってライブラリーを設定し、使用するメッセージ・ファイルだけを交換します。

たとえば、次のようにします。

```
A     field-name   length type usage EDTCDE(x)
A 61                                 ERRMSGID(USR3456 [*libl/]APPMSGF
[&MSGFLD2])
A     MSGFLD2      length type   P   TEXT('Message data field')
```

および

```
ADDMSGD   MSGID(USR3456) MSGF(APPTXDENU/APPMSGF)
          MSG('Delivery date &1; is earlier than production end date &2')
```

および

```
ADDMSGD   MSGID(USR3456) MSGF(APPTXDDEU/APPMSGF)
          MSG('Lieferdatum &1; ist . . .')
.
.
.
```

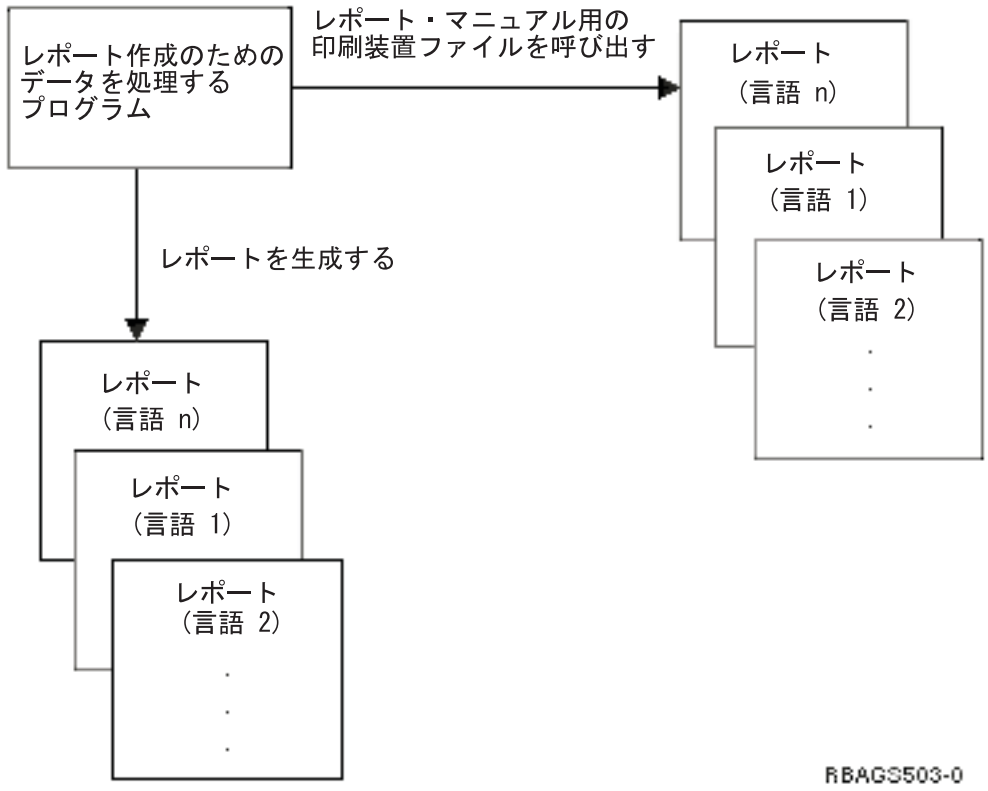
印刷装置ファイルの設計と翻訳: 次の印刷装置ファイルがあります。

- プログラム記述の印刷装置ファイル

  プログラム記述ファイルは、高水準言語を使用して、印刷するレコードやフィールドを定義します。
- 外部記述の印刷装置ファイル

  外部記述の印刷装置ファイルは、高水準言語の代わりに DDS を使用して、印刷するレコードやフィールドを定義します。

次の図は、ほかの各国語バージョン向けのレポートを作成するときに、外部記述の印刷装置ファイルがどのように使用されるかを示しています。



印刷装置ファイルの翻訳

各国語バージョンに翻訳する予定の印刷装置ファイルを設計する場合は、次のようにしてください。

- 印刷するレコードやフィールドを定義するには、外部記述の印刷装置ファイルを使用してください。プログラム記述の印刷装置ファイルを使用するのは避けます。プログラム記述の印刷装置ファイルは、高水準言語プログラムの中に記述されます。翻訳者がプログラム内に埋め込まれたテキストを翻訳すると、プログラムの部分であるリテラルを間違って翻訳することがあります。
- 国別グラフィック文字セット内のデータを印刷するときは、文字セットとコード・ページが対応する装置を使用して印刷してください。すべてのプリンターがすべての CHRID パラメーターをサポートしているわけではありません。
- メッセージ・ファイルに記述されている定数にアクセスするには、MSGCON キーワードを使用してください。印刷装置ファイルに MSGID キーワードはありません。ただし、無名の入力フィールド (リテラル) としてテキストを直接コーディングする技法や、テキストをデータベース・ファイルに保管する技法を使用して、印刷装置ファイル内で定数テキストを指定できます。 20 ページの『テキスト・データ・コードの設計』 を参照してください。
- 印刷装置ファイルでバーコードを記述する場合は、国別情報を考慮してください。バーコードは国によって規格が異なります。
- データを入力するときには、「印刷装置ファイルの作成」 (CRTPRTF) コマンドに使用する次のパラメーターを考慮してください。
  - PAGESIZE (ページ・サイズ)

    ページ・サイズは、国により規格が異なります。
  - OVRFLW (オーバーフロー行数)

    オーバーフロー行数は、ページ長さ以下にしてください。
  - CHRID (文字セットとコード・ページ)

    印刷装置ファイルの CHRID パラメーターを *DEVD に設定すると、プリンターは、コントロール・パネルまたは装置記述に指定されている文字識別コードを使用します。

    印刷装置ファイルの CHRID パラメーターを特定の値に指定すると、データを印刷するときのコード・ページと文字セットはこの値が決定します。外部記述印刷装置ファイルの場合は、CHRID パラメーターが使用されるのは、同じ CHRID DDS キーワードが指定されているフィールドだけです。その他のすべてのフィールドについては、*DEVD が指定されている場合と同様に、同じコード・ページと文字セットが使用されます。

    印刷装置ファイルの CHRID パラメーターを *JOBCCSID に設定すると、外部記述の印刷装置ファイルの定数テキストは、ジョブの CCSID に変換されます。プリンターのデータ・ストリームは、ジョブの CCSID から獲得した CHRID がタグ付けされ、この CHRID 値を使用してデータを印刷します。CHRID パラメーターの *JOBCCSID 値を使用すると、CHRID DDS キーワードは無視されます。

    注: それぞれのプリンターが処理できるコード・ページと文字セットは限られています。

ソース・ファイルの設計: データベースのソース・ファイルには、「物理ファイルの作成」 (CRTPF) コマンドまたは「ソース・ファイルの作成」 (CRTSRCPF) コマンドに CCSID パラメーターを指定して CCSID を明示的に指定しない限り、データベースを作成するときに暗黙的にジョブの CCSID が割り当てられます。ジョブの CCSID が 65535 の場合は、暗黙的に割り当てられる CCSID には、ジョブのデフォルト CCSID (DFTCCSID) が使用されます。ジョブのデフォルト CCSID は、システムの言語識別コード値とジョブの DBCS 対応標識により決定されます。

文字データ表現体系 (CDRA) の設計: アプリケーションを多国語環境向けに作成するときは、次のことを考慮してください。

- 物理ファイルの CCSID 値を直接 DDS にコーディングしないでください。複数の言語向けに複数の物理ファイルを作成する場合は、ジョブの CCSID を変更してください (CHGJOB コマンドを使用します)。これで管理すべき DDS のソース・コードが 1 組だけになります。

  すべての CCSID の間で変換することは意味をなさない場合があります。たとえば、ジョブの CCSID が 00273 のドイツ語のディスプレイ装置を使用して、CCSID が 00875 のギリシャ語のデータベースにアクセスすると、ディスプレイ装置の文字は化けてしまいます。

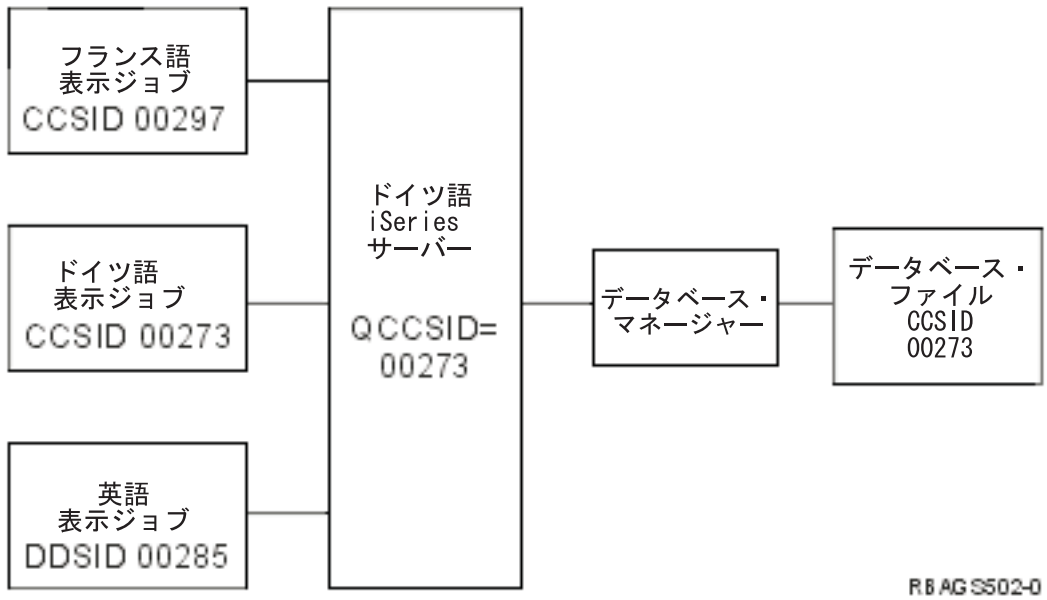
  文字セットが Latin 1 の国以外では、ローマ字以外の文字セットも含めて使用します。非ローマ字のコード・ポイントとローマ字のコード・ポイントの間で、意味のない変換が行われる可能性があります。アラビア語、ギリシャ語、ヘブライ語、およびトルコ語は、非ローマ字の SBCS です。
- データベースを共有する場合は、ファイルには 1 次言語の CCSID を定義してください。各ユーザーは、使用する言語の CCSID をそれぞれのユーザー・プロファイルに定義してください。

CDRA については、次の情報も参照してください。

- CCSID の処理
- SNDNETF コマンドの使用方法
- シナリオ: 多国語の単一システム
- シナリオ: 多国語ネットワーク

「ネットワーク・ファイルの送信」コマンドの使用方法: 「ネットワーク・ファイルの送信」 (SNDNETF) コマンドを使用すると、データ (メンバーだけを送信する場合) の CCSID は、コマンドを実行中のジョブの CCSID であると見なされるので、変換は行われません。データを受信したときには、発信元のファイルと同じ CCSID でメンバーをファイルに保存するように注意してください。受信者が着信ファイル・メンバーの CCSID を知らないときは、CCSID を 65535 にしてファイルに受信すれば、変換は行われません。

シナリオ: 多国語の単一システム: 次の図は、1 次言語にドイツ語、2 次言語に英語とフランス語を使用した多国語の単一システムの例を示しています。すべてのユーザーは、1 つのデータベース・ファイルにデータを入力します。

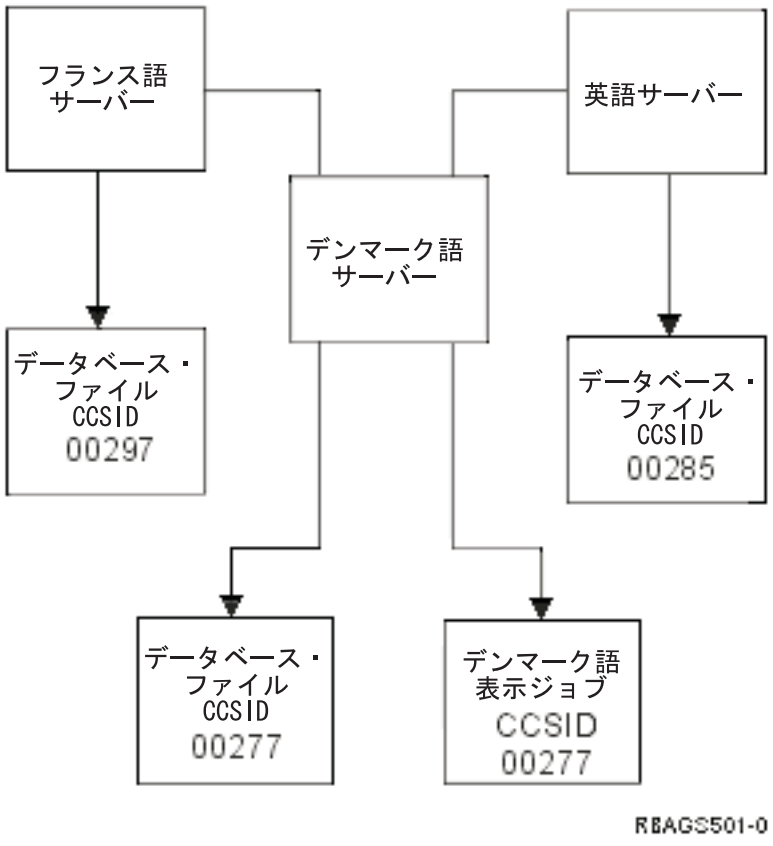


この多国語システムでは、すべてのユーザーは、CCSID が 00273 (ドイツ語) の 1 つのファイルに文字データを入力します。英語およびフランス語のディスプレイ装置で入力した文字データは、ドイツ語のファイルにマップされます。

正しいマッピングを得るために、文字フィールドとして定義されたフィールドは、実際の文字フィールドである必要があります。フィールドにアプリケーションの開発値が含まれている場合は (たとえば、制御文字、あるいは実際の文字フィールドとして使用されていないフィールドなど)、そのフィールドは 16 進数フィールドとして指定するか、CCSID 値を 65535 に割り当てる必要があります。

CCSID を使用すると、異なるコード・ページの間で変換できない文字は、置換コードに置換されます。画面の形式やレイアウトの設定に (DDS ではなく) ユーザー定義のデータ・ストリーム (UDDS) を使用している場合は、ユーザー定義のデータ・ストリームのデータをシステムが読み込みや挿入を行ったときに、置換コードが戻される場合があります。置換コードは、画面上で予期せぬ結果をもたらすことがあります。

シナリオ: 多国語ネットワーク: 次の図は、3 台の iSeries サーバーを言語の異なる 3 か国に置いた多国語ネットワークの例です。この例では、デンマーク語のシステムで、アプリケーションが分散リレーショナル・データベースを使用します。それぞれの言語の文字は、(データを保存する言語とは無関係に) デンマーク語のディスプレイで正しく表示できます。データベースの保全性は、言語の CCSID をデータベースで使用して維持します。異なるコード・ページ間のデータの変換処理については、OS/400 データベース管理機能により完全自動で行われます。



NLV サポート対象外の言語の処理: 各国語バージョンのサポート対象外の言語を使用する場合は、次の一般的な操作を実行してください。

1. 使用可能な各国語バージョンを調査します。文字の表示について、使用する言語に最も類似する各国語バージョンを探します。

2. 1 次言語として最も適切な各国語バージョンをインストールします。
3. システム値を変更して、文化要件を満たします。たとえば、サポート対象とする文化に応じた日時の形式を設定します。文化要件に関するシステム値の設定については、ほかの言語に関するシステム値 を参照してください。
4. ワークステーションおよびプリンターを 1 次言語に従って構成します。次に、インストールした NLV と使用言語の間の矛盾を処理します。

   注: ワークステーションのカスタマイズ機能を使用できるのは、ハードウェアに組み込まれた機能だけです。ハードウェアがサポートしていない機能をワークステーションのカスタマイズで処理することはできません。
5. 「テーブルの作成」(CRTTBL) コマンドを使用して、使用する言語に適した分類に最も近い既存のテーブルを基に、分類順序テーブルを作成します。
6. 使用言語が DBCS 言語の場合は、インストールした NLV に対応するコード・ページに存在しない文字について、独自の文字 (UDC) を作成してください。UDC は、ユーザー定義文字 (user-defined character) の略語です。文字作成プログラム (CGU) を使用して作成します。 CGU は、ユーザーが特別に定義する表意文字、記号、およびロゴで、コード・ページの外字となります。

グローバル・アプリケーション設計のプログラミングに関する考慮事項

グローバル・アプリケーションを開発するときに、各国語バージョン環境では、コードの作成およびコンパイル方法に特別に注意を払うことが必要です。以下のトピックは、これらの要件について説明し、問題を最小化するためのガイドラインを提供します。

- 高水準言語を使用したグローバル・アプリケーションのコーディング
- 両方向データを使用したグローバル・アプリケーションのコーディング
- メッセージ・カタログの使用方法

高水準言語を使用したグローバル・アプリケーションのコーディング

すべての言語バージョンに共通な汎用実行コードを 1 つだけ作成し、プログラムをできるだけテーブル駆動型にするようにしてください。次のようにします。

- 基本妥当性検査は、ハードコーディングのリテラルやテーブルではなく、データベース・アクセスやメッセージ・ファイルで実行する。
- 可変要素に対する基本計算は、インラインでコード化するのではなく、ファイルから読み取る。
- 文化依存型機能は、柔軟なコード化ができない場合は、アプリケーションの個別のモジュールに入れて呼び出すようにする。

ハードコーディング値は、比較、スキャン、置換、または呼び出しの操作では、言語や文化にまったく依存しない場合を除いて、使用しないでください。 また、大文字小文字の区別が必要な値を使用しないでください。たとえば、プログラムの中で「Yes」や「No」 (Y または N) などの応答をハードコーディングしないようにします。これらの値は、言語により異なるので、テキスト・データの一部とします。

ソース・コード内のリテラルや定数については、不変文字セットの文字だけを使用してください。プログラム内で入力データの妥当性を検査する場合は、検査対象を不変文字セットだけにします。このようにしないと、キーボードに存在しない文字を入力するように指示される場合があります。たとえば、アラビア語のキーボードには、左中括弧 ({) と右中括弧 (}) はありません。不変文字セットについては、不変文字セットのリストを参照してください。

コンパイル時配列を使用して、メッセージあるいは言語や文化に依存するデータを保管しないでください。

外部の NLS 依存型モジュールを呼び出すときにパフォーマンスを改善するためには、プログラム名を含む変数フィールドではなく、(ライブラリー・リストに従った) リテラルとしての固定名で呼び出してください。 これで、アプリケーションは、関連ライブラリー・リストに従って、異なるライブラリーのモジュールを呼び出すことができます。

ユーザーが自分の使用する言語や文化に応じてアプリケーションを操作できるようにするため、編集値 (日付、時刻、日付区切り記号など) を言語や国または地域に依存させて指定してください。 これで、ユーザー・プロファイル内の情報に従って、これらの値を読み取ることができます。パラメーターは、LANGID (言語識別コード) と CNTRYID (国別または地域別識別コード) です。文化依存型の情報を読み取る操作は、プログラムの初期化のときに一度実行するだけです。 これは、初期 CL プログラムまたは高水準言語プログラムを使用して、次のように準備します。

- 呼び出し操作のパラメーターとする。
- ローカル・データ域 (LDA) のパラメーターとする。
- プログラム・ロード・テーブルとする。

初期プログラムを使用して、ユーザーのジョブ属性を設定すれば、 OS/400 プログラムやその他のライセンス・プログラムなど、一貫したアプリケーションを提供できます。

高水準言語については、次のトピックにその他の情報があります。

- 言語コンパイラー CCSID
- セッション・マネージャー
- ILE C/400(R) に関する考慮事項
- ILE RPG の分類順序
- ILE COBOL の分類順序
- DB2 および SQL の分類順序
- iSeries Access の分類順序

言語コンパイラー CCSID: 一部の言語コンパイラーは、ソース・コードの構文演算子と命名規則の CCSID が 00037 であることを前提とします。(使用する言語コンパイラーの資料を参照してください。)そうしたコンパイラーでは、ソースを 00037 または 65535 以外の CCSID でコンパイルした場合にマッピング間違いが生じます。これらのコンパイラーでは、言語構文の可変文字の CCSID を 00037 にしてください。

ILE 言語のコンパイラー

ILE C/400 for iSeries、ILE RPG for iSeries、または ILE COBOL for iSeries プログラムをコンパイルするときは、データベース・ソース・ファイルからのソースは、1 次 ソース・ファイルの CCSID に変換されます。

これらの言語のコンパイラーは、ほとんどの CCSID で構文演算子を処理します。これらのコンパイラーは、ほとんどの CCSID で、ソース・コードの命名規則も処理します。

非 ILE 言語のコンパイラー

非 ILE CL、RPG、または COBOL プログラムをコンパイルするときには、データベース・ソース・ファイルのソースは、ジョブの CCSID に変換されます。

名前、定数、あるいはリテラルをジョブの CCSID に変換したくない場合は、ジョブの CCSID を 65535 に変更できます。これで、定数、リテラル、および名前は変更されません。

注: REXX/400 プロシージャーとそこにコード化されたリテラル・データは、ジョブの CCSID には変換されません。

例 1

次の例は、非 ILE RPG プログラムの例を示しています。この例では、アメリカ合衆国のシステムにある英語ソースを示しています。

```
*  RPG Source  (Source file created using CCSID 00037 but tagged
*                with CCSID 65535)
FFILE1  IF  E                    DISK                          80
C                    READ FILE1
C* Test char
C*
C           FLD1     IFEQ '$'
C                    ...
C* Move char
C*
C                    MOVE FLD1         FLD$
C                    ...
C*
C                    SETON                                   LR
```

例 2

フィールド名の FLD$ に可変文字 (ドル記号) が含まれているので、最初の例のプログラムはフィンランドではコンパイルできません。可変文字は、00037 以外のコード・ページのさまざまなコード・ポイントを示します。この図は、フィンランドのシステムで、英語 (米国英語) のソースによる 非 ILE RPG プログラムの例を示しています (CCSID 278)。

```
*  RPG Source  (Source file created with CCSID 00037, but tagged
*                with 65535)
FFILE1  IF  E                    DISK                          80
C                    READ FILE1
C* Test char
C*
C           FLD1     IFEQ ''
C                    ...
C* Move char
C*
C                    MOVE FLD1         FLD
C                     ...
C*
C                    SETON                                   LR
```

例 3

ファイルの CCSID を 00037 に変更し、ジョブの CCSID を 00278 (フィンランド向け) に設定すると、このエラーは訂正できます。次の例は、フィンランドの英語ソース向けに変更したファイルです。

```
*  RPG Source  (Source file created using CCSID 00037 and tagged
*                with CCSID 00037)
FFILE1  IF  E                    DISK                          80
C                    READ FILE1
C* Test char
C*
C           FLD1     IFEQ '$'
C                    ...
C* Move char
C*
C                    MOVE FLD1         FLD$
```

```
C                   ...
C*
C                   SETON                                     LR
```

セッション・マネージャー: セッション・マネージャーを使用するすべてのアプリケーションについて、出力データ・ストリームには X'3F' 値を含めないでください。OS/400 は、画面を空白にするときに X'3F' 値を使用します。

一般的な分類順序

プログラムが使用する分類順序がプログラム・ロジックに影響を与えることがあります。この例を次の図に示します。

Latin 1 の共用重みによる分類順序を使用すると、文字テスト 3 は、文字テスト 4 と同等になります (すべての文字は表示してありません)。 16 進数または固有の分類を使用すると、この 2 つはまったく別のものになります。 次の例では、RPG プログラムでさまざまな分類順序を使用します。

```
*  RPG Source  (Program created with Latin 1 sort sequence)
*
C* Test char 3
C*
C           FLD1      IFEQ 'a'
C                     ...
C* Test char 4
C*
C           FLD1      IFEQ 'a'
C           FLD1      OREQ 'A'
C           FLD1      OREQ ''
C           FLD1      OREQ ''
C                     ...
C*
C                     SETON                                     LR
```

SRTSEQ パラメーターに *JOBRUN、LANGID パラメーターに *JOBRUN を指定してプログラムをコンパイルすると、コンパイル時には、実行時に使用する分類順序は分かりません。

ILE C/400 用および DB2 Query Manager and SQL Development Kit for iSeries のライセンス・プログラムには、このほかの特別な考慮事項の説明があります。

ILE C に関する考慮事項: ILE C を使用してプログラムをコンパイルするときには、次のことを考慮してください。

- ソース・ファイルは、コード・ページ 00290 を除くその他のすべての EBCDIC コード・ページでコンパイルできます。
- 1 次ソース・ファイルの CCSID が 65535 の場合は、コード・ページを 00037 と想定します。
- すべての 2 次ソース・ファイルは、1 次ソース・ファイルの CCSID に変換されます。

  注: ほとんどの 2 次ソース・ファイルは、1 次ソース・ファイルの CCSID に変換されますが、サポート対象外の変換もあります。サポート対象外の CCSID 変換が必要な場合は、IBM サービス技術員にご連絡ください。
- 2 次ソース・ファイルの CCSID が 65535 の場合は、変換処理は実行されません。
- モジュールは、1 次ソース・ファイルのコード・ページに作成されます。 モジュールは、OS/400 のオブジェクトです。モジュールは、1 つまたは複数のプロシージャーの場合、あるいは 1 つまたは複数の外部または内部の変数定義の場合があります。モジュールは、ソース・コードからコンパイルします。
- 異なる CCSID のモジュールを結合すると、変換が行われて予期せぬ結果が生じることがあります。

- 一部のキーボードで使用できない C 文字については、三重音字サポートを使用できます。三重音字サポートでは、通常、可変文字を表記するのに不変文字を使用します。たとえば、左側のブラケット ([) は、??(のように表記します。

ILE C 実行時ライブラリーでは、可変文字を含む構文解析ストリングは、ジョブの CCSID に対応する可変文字コード・ポイント値を使用します。

ILE RPG の分類順序: ILE RPG for iSeries ライセンス・プログラムでは、ユーザーは、分類順序テーブルを指定して、非数値データの比較処理に使用できます。工場出荷時には、サーバーにはそれぞれのサポート対象言語に 2 つのテーブルが添付してあります (共用重みと固有重み)。分類順序サポートを使用すると、既存のテーブルを基に、分類順序テーブルを作成できます。

制御仕様は、ILE RPG for iSeries コンパイラーに使用するプログラムやサーバーの情報を提供するための仕様です。ILE RPG for iSeries プログラムが使用する分類順序は、次のすべての項目により制御されます。

- 制御仕様。
- 「RPG モジュールの作成」コマンドおよび「バインド済み RPG プログラムの作成」コマンドの SRTSEQ (分類順序テーブル) パラメーター。
- 「RPG モジュールの作成」コマンドおよび「バインド済み RPG プログラムの作成」コマンドの LANGID (言語識別コード) パラメーター。

制御仕様の代替照合シーケンス・フィールド (ALTSEQ) では、次の値を使用できます。

ブランク
RPG プログラムで代替照合シーケンスを使用しません。RPG プログラムで通常の照合シーケンスを使用します。コンパイル・オプションの SRTSEQ と LANGID は無視されます。

***NONE**
RPG プログラムで代替照合シーケンスを使用しません。RPG プログラムで通常の照合シーケンスを使用します。コンパイル・オプションの SRTSEQ と LANGID は無視されます。

***SRC** RPG プログラムの最後に入力したテーブルに従って、代替照合シーケンスを RPG プログラムで使用します。代替照合シーケンスをコンパイル時にロードして、そのテーブルに従って、配列、分類、比較、および突き合わせフィールド処理を実行します。

SORTA と LOOKUP の命令コードは、指定した代替照合シーケンス・テーブルを使用しません。

「RPG モジュールの作成」コマンドおよび「バインド済み RPG プログラムの作成」コマンドの SRTSEQ と LANGID のパラメーターは無視されます。

***EXT** 代替照合シーケンスは、RPG プログラムの外部に指定されます。「RPG モジュールの作成」コマンドおよび「バインド済み RPG プログラムの作成」コマンドの SRTSEQ および LANGID に従って、 RPG コンパイラーが外部の分類順序テーブルをインポートします。

コンパイル時および処理時の配列とテーブルに関する SORTA と LOOKUP の関数が有効になるのは、制御仕様に D を指定したときだけです。

プログラムが使用する分類順序テーブルは、コンパイル時またはジョブの実行時に決定できます。「RPG モジュールの作成」コマンドおよび「バインド済み RPG プログラムの作成」コマンドの SRTSEQ パラメーターの設定によって、次のようになります。

- *HEX を設定すると、分類順序を使用しません。

- テーブル名を指定すると、ジョブの実行時に使用するプログラム・オブジェクトとともにそのテーブルを保存します。サポート対象言語用にシステムが提供するデフォルトの分類順序テーブルについては、分類順序テーブル を参照してください。
- *LANGIDSHR または *LANGIDUNQ を設定すると、LANGID パラメーターが決定する言語の共用重みまたは固有重みがプログラム・オブジェクトとともに保存されます。有効な言語識別コードについては、言語識別コードと国別/地域別識別コードのリストを参照してください。
- *JOB を指定すると、コンパイル時のジョブの SRTSEQ パラメーターを使用して、分類順序を決定します。テーブルは、プログラム・オブジェクトとともに保存されます。
- *JOBRUN を指定すると、コンパイル済みプログラムを実行するジョブ属性が使用する分類順序を決定します。ジョブの SRTSEQ 属性が LANGID を参照すると、プログラム・オブジェクトとともに保存されている LANGID が使用されます。 プログラムとともに保存されている LANGID が *JOBRUN の場合は、実行時ジョブの LANGID が使用されます。

注:

1. コンパイル時にプログラム・オブジェクトとともに保存するテーブルが存在しない場合は、16 進数の分類順序を定義している、CCSID 値 65535 がタグ付けされたテーブルが使用されます。
2. 分類順序テーブルとプログラムを実行するジョブの CCSID が異なる場合は、テーブルはジョブの CCSID に変換されます。

SORTA と LOOKUP の命令コード

分類順序テーブルを処理する比較命令コード、突き合わせフィールド、および制御フィールドの実装は、代替照合シーケンスおよび分類順序のサポートの場合と同じです。比較命令コードは、
ANDxx、COMP、CABxx、CASxx、DOUxx、DOWxx、IFxx、ORxx、および WHxx です。このほか、
SORTA と LOOKUP の命令コードには、次の機能があります。

SORTA
配列データは、分類順序テーブルに従って分類されます。

LOOKUP
正しいテーブル検索を実行するために、配列とテーブルにある検索引き数に対して分類順序テーブルを使用します。

検索引き数およびテーブルまたは配列の要素は、分類順序テーブルを使用して比較されます。

配列とテーブルのデータは、昇順または降順の順序が指定されると、分類順序テーブルを使用して検査されます。SRTSEQ と LANGID のパラメーター値が解決して実行時に再び分類順序テーブルを読み取る場合、配列およびテーブルの要素は、コンパイル時にシーケンス検査なしでロードされます。シーケンス検査は、分類順序テーブルに従って、実行時に実行されます。

ILE COBOL の分類順序: ILE COBOL for iSeries ライセンス・プログラムでは、分類順序のサポートに次の方法を使用します。

- 「COBOL モジュールの作成」コマンド
- 「バインド済み COBOL プログラムの作成」コマンド
- PROCESS 文節
- ALPHABET 文節

ILE COBOL for iSeries ライセンス・プログラムでは、システム提供またはユーザー提供の分類順序テーブルを使用します。

「COBOL モジュールの作成」および「バインド済み COBOL プログラムの作成」コマンド

この CL コマンドには、分類順序サポートに関連する 2 つのコンパイラー・オプション、SRTSEQ パラメーターと LANGID パラメーターがあります。SRTSEQ パラメーターを使用すると、特定のライブラリーにあるシステム提供またはユーザー提供の分類順序テーブルを指定できます。分類順序テーブルをコンパイル時に使用するか、または実行時に使用するかを指定できます。 さらに、共用重みと固有重みのどちらかを選択できます。

LANGID パラメーターを使用すると、システム定義の言語識別コードを指定できます。または、パラメーターを実行時まで未定義のままにすることもできます。

「COBOL モジュールの作成」および「バインド済み COBOL プログラムの作成」コマンドの SRTSEQ と LANGID のパラメーターの意味は、 50 ページの『ILE RPG の分類順序』 で説明されている「RPG モジュールの作成」および「バインド済み RPG プログラムの作成」コマンドと同じです。

PROCESS ステートメント

PROCESS ステートメントには、分類順序サポート・オプションを使用できます。このコマンドの構文は、「COBOL モジュールの作成」および「バインド済み COBOL プログラムの作成」コマンドに似ています。 唯一の違いは、PROCESS ステートメントで事前定義値としてパラメーター値を入力するときに、アスタリスク (*) を使用しないことです。PROCESS ステートメントにオプションを指定すると、「COBOL モジュールの作成」および「COBOL プログラムの作成」コマンドの対応オプションがオーバーライドされます。

ALPHABET 文節

SPECIAL-NAMES 段落の ALPHABET 文節にあるアルファベット名では、NLSSORT オプションを使用できます。代替照合シーケンス・オプションについては、コンパイラーの SRTSEQ および LANGID のパラメーターを使用してください。このオプションを使用しない場合は、NATIVE オプションと同じになります。

次の COBOL 行は、NLSSORT オプションの影響を受けます。

- OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 句

  非数値の比較結果を評価するときに、指定された分類順序オプションをプログラムが使用するには、この句の中でアルファベット名を参照する必要があります。このオプションは、非数値の分類やマージにも使用されます。この操作を行わない場合は、16 進数の照合シーケンスが使用されます。
- SPECIAL-NAMES 段落の ALPHABET CLAUSE

  この文節で NLSSORT オプションを指定します。
- MERGE (または SORT) ステートメントの COLLATING SEQUENCE

  この句は、MERGE または SORT 処理の KEY データ名について非数値の比較に使用する照合シーケンスを指定します。省略すると、OBJECT-COMPUTER 段落の PROGRAM COLLATING SEQUENCE 文節が使用する照合シーケンスを定義します。どちらも指定しない場合は、16 進数の照合シーケンスが使用されます。
- 非数値の比較名と条件名

  非数値の比較名や条件名を使用すると、選択した分類順序テーブルが特定のステートメントに影響を与えます: EVALUATE、IF、PERFORM...UNTIL、SEARCH、および START。非数値比較の真理値は、選択された分類順序テーブルの文字に与えられた重みに依存します。たとえば、フランス語 (Latin 1) で、固有重みのテーブル (LANGIDUNQ) を指定すると、変数 ITEM-1,e だけについては、次のステートメントが真となります。

```
IF ITEM-1 = "e"
```

フランス語 (Latin 1) で共用重みテーブル (LANGIDSHR) を指定すると、同じステートメントが変数 ITEM-1 の複数の値について真となります。共用重みは、すべてについて 77 となります。

```
lowercase e (e), uppercase e (E),
lowercase e acute (), uppercase e acute (),
lowercase e grave (), uppercase e grave (),
lowercase e caret (), uppercase e caret (),
lowercase e umlaut (), uppercase e umlaut ()
```

DB2 および SQL の分類順序:

対話式の SQL については、SRTSEQ および LANGID のパラメーターを STRSQL コマンドに指定できます。これらのパラメーターは、後に対話形式の表示に関するセッション・サービスを使用して変更できます。

分類順序テーブルは、すべてのストリング比較で使用します。ストリング比較は、次の SQL ステートメント内で実行します。

- ORDER BY 文節
- WHERE 文節
- GROUP 文節
- HAVING 文節
- UNION および UNION ALL 文節
- DISTINCT 文節
- BETWEEN 述部
- IN 述部
- LIKE 述部
- MIN および MAX スカラー関数
- MIN および MAX 列関数

さらに、CREATE INDEX または CREATE VIEW ステートメントを使用して作成する索引や表示は、分類順序テーブルを指定して作成できます。

DB2 Query Manager and SQL Development Kit for iSeries

DB2 Query Manager and SQL Development Kit for iSeries は、ソースをプリコンパイルするときに、特定の CCSID を前提とはしません。言語構文内の可変文字 (否定記号 ¬ など) は、ソース・ファイルの CCSID にコード化されていることを前提とします。

たとえば、ソース・ファイルの CCSID が 00037 の場合、否定記号 ¬ は、コード・ポイント X'5F' に正しく解釈されます。また、ソース・ファイルの CCSID が 00500 の場合、否定記号 ¬ は、コード・ポイント X'BA' に正しく解釈されます。

リテラルは、ソース・ファイルの CCSID に保存されます。

DB2 Query Manager and SQL Development Kit for iSeries は、SQL プログラムを作成するときに言語コンパイラーを呼び出すので、高水準言語に関する一般ガイドラインも考慮する必要があります。

iSeries Access の分類順序: iSeries Access 機能、リモート SQL、および転送機能では、分類順序を指定できます。サーバーのデータベースや SQL テーブルで照会操作を実行するときには、システム提供またはユーザー提供の分類順序テーブルを指定できます。

リモート SQL サポート

照会操作では、選択したデータの分類方法を指定できます。このためには、ORDER BY 文節に分類フィールドを指定する必要があります。次の文節でも指定した分類順序が使用されます。

- WHERE 文節
- GROUP BY 文節
- HAVING 文節
- JOIN BY 文節
- UNION 文節
- DISTINCT 文節
- IN 述部
- LIKE 述部
- BETWEEN 述部
- RANGE 述部
- MAX 関数
- MIN 関数

実際の分類順序テーブルは、ユーザーのジョブ属性から読み取られます。ユーザー・プロファイルやジョブ属性を変更すると、SRTSEQ および LANGID のパラメーターが影響を受けます。

転送機能サポート

iSeries サーバーからワークステーションにデータを転送するときには、選択したデータに使用する分類順序を指定できます。分類順序テーブルは、次のストリング比較演算でも使用されます。

- WHERE 文節
- GROUP BY 文節
- HAVING 文節
- JOIN BY 文節
- IN 述部
- LIKE 述部
- BETWEEN 述部
- MAX 関数
- MIN 関数

OPTION ステートメントには分類順序に関して次のパラメーターを指定できます。

- SRTSEQ (分類順序テーブル)
  - *JOB
  - *HEX
  - *LANGIDSHR
  - *LANGIDUNQ

- *LIBL/sort-seq-table-name
- *CURLIB/sort-seq-table-name
- library-name/sort-seq-table-name

- LANGID (言語識別コード)
  - *JOB
  - language-identifier

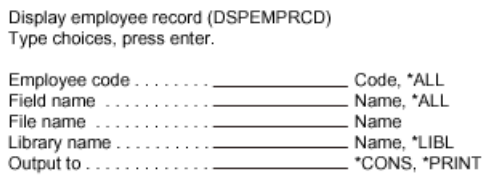
iSeries Access 画面のオプションで、正しい分類順序を選択できます。

両方向データを使用したグローバル・アプリケーションのコーディング

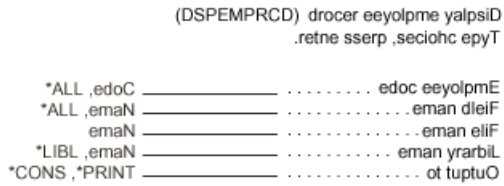
NLV 対応アプリケーションを開発するときは、以下の制約事項を考慮することが必要です。

- 両方向言語の表示レイアウト
  データを右から左方向に表示する必要があります。リテラルは、リテラルを記述するフィールドの右側に表示します。 次の例は、米国英語の左から右方向の表示、および同じ表示内容を右から左にした場合を示してあります。

  米国英語の左から右方向の表示

  米国英語の右から左方向の表示

- 両方向言語の長いフィールド
  入力フィールドの定義には、複数の行を使用しないでください。 フィールドを 1 つのエンティティーとして表示または印刷したときに、両方向言語では予期せぬ結果が生じます。
- 両方向言語での変数の位置
  アプリケーションで変数の順序を自由に変更できるようにする必要があります。たとえば、次の英語のメッセージの例を見ます。

  ```
  File &1 in library &2 not found
  ```

  別の言語に翻訳すると、メッセージは次のようになります。

  ```
  dnuof ton &2 yrarbil ni &1 eliF
  ```

  この場合、変数 2 は変数 1 の前に置かれます。
- 両方向言語での CHECK(RL) と CHECK(RB) のキーワード
  このオプションは、右から左への移動が可能なディスプレイ装置でのみ使用できます。次の制約があります。
  - オプション標識は、カーソル制御コードについては無効です。

– このキーワードでは、CHECK(RZ) と CHECK(RB) は無効になります。
– フィールドで改行すると警告が出ます。
– モジュラス検査のチェック・ディジットは、フィールド右端のバイトです。
– CHECK(RL) は、文字フィールドでのみ使用できます。

- 両方向言語のオンライン情報
  特殊両方向タグには制約があります。オンライン・ヘルプの情報が、異なる BIDI タグの値を使用する複数のパネル・グループから構成されている場合、エンド・ユーザーが逆方向のオンライン・ヘルプを読むときに、ホット・キー・シーケンスが必要になります。
- 両方向言語の CCSID
  両方向言語には、固有の特殊文字セットがあります。ほかの言語とのデータ交換はできません。ただし、EBCDIC と ASCII のデータ・ストリームの間では、データ・マッピングが必要になることがあります。たとえば、分散リレーショナル・データベース・アーキテクチャー (Distributed Relational Database Architecture(R) (DRDA(R)) を使用すると、EBCDIC と ASCII のデータ・ストリームの間でデータ・マッピングが必要になります。

  ローマ字を使用する言語でデータを交換する場合で、不変文字セット以外の特殊文字が必要な場合にデータ・マッピングを実行するには、ヘブライ語では CCSID 00424、アラビア語では CCSID 00420 を使用してください。 サポート対象 CCSID については、CCSID を参照してください。

両方向データについて詳しくは、両方向データの処理を参照してください。このトピックには、両方向サポートを使用してアプリケーションを作成する場合のガイドラインを示した 両方向データのチェックリストも含まれています。

メッセージ・カタログの使用方法

OS/400 では、メッセージ・カタログを使用してメッセージを保存します。メッセージは、メッセージ・カタログ内でセットにまとめられます。セット内のメッセージには、それぞれ固有の番号が付きます。メッセージ・カタログは、ストリーム・ファイルとして、ソース・ファイル・メンバーとして、あるいは 1 つ以上のソース・ファイルのユーザー・スペース・オブジェクト・タイプとして作成できます。

メッセージ・カタログはストリーム・ファイルとして保存できるので、プロダクトや各国語バージョンに応じて、ディレクトリーを使用してメッセージを分離できます。

GENCAT および MRGMSGCLG コマンドを使用したメッセージ・カタログの作成または更新

メッセージ・カタログを作成または更新するには、「メッセージ・カタログの作成」(GENCAT) コマンドと「メッセージ・カタログのマージ」(MRGMSGCLG) コマンドの両方を使用できます。メッセージ・カタログを作成した後は、これらのコマンドでオリジナルのメッセージとソースのメッセージを比較して、カタログを更新できます。セット内のほかのメッセージを変更せずに、特定のメッセージを新規メッセージ・テキストに置換できます。これらのコマンドを使用すると、既存のメッセージ・セットについて、メッセージの追加や削除ができます。メッセージ・セットは、既存のメッセージ・カタログから削除できます。

関連情報

メッセージ・カタログについて詳しくは、次のトピックを参照してください。

- メッセージ・カタログのソース
- メッセージ・カタログのオープン、抽出、およびクローズ

メッセージ・カタログのソース: メッセージ・カタログのソースは、ソース物理ファイル、ストリーム・ファイル、または複数のファイルのいずれかになります。ソースには、セット数、メッセージ番号、および

メッセージ・テキストを定義したり、または削除するセットを指定するためのフィールドがあります。次のトピックには、メッセージ・カタログに関するその他の情報があります。

メッセージ・カタログ・ソースの形式

メッセージ・カタログには、メッセージ・テキスト・ソース行のためのフィールドが 5 つあります。5 つのフィールドは、1 つのブランク文字で区切ります。その他のブランク文字は、フィールド・データの一部として扱われます。 追加情報として、特殊文字とエスケープ・シーケンス (59 ページを参照してください。) を参照してください。

注: キー・フィールドでは、ドル記号 ($) と小文字を使用して、下記の通りに正確に入力してください。最大値と最小値の定義は、QSYSINC/QSYS/LIMITS に保存されています。

- **$ comment**
  行を $ 記号で開始し、1 つ以上のブランク文字を続けると、その行はコメント行として扱われます。コメント行は、対象メッセージの直下に置きます。ソース・ファイルの $set ディレクティブの直下にセット全体のコメントを置きます。
- **$quote** *C*
  この行は、メッセージ・テキストを囲むのに使用するオプションの引用文字 *C* を指定します。この文字を使用すると、メッセージ・ソース行の末尾のスペースやヌル (空の) メッセージを表示します。デフォルト設定の場合、または空白の $quote ディレクティブがあると、メッセージ・テキストの引用は認識されません。
- **$set n comment**
  この行は、次の $set または end-of-file が現れるまで、メッセージのセット識別コードを指定します。*N* は、セット識別コードを表します。 1 ～ NL_SETMAX の間の数値で定義します。識別コードは、1 つソース・ファイル内で昇順にしてください。隣接する必要はありません。セット識別コードに続く文字ストリングは、コメントとして扱われて無視されます。
- **$delsetncomment**
  この行は、既存メッセージ・カタログのメッセージ・セット *n* を表します。*n* はセット番号を指定します。セット番号に続くデータは、コメントとして扱われます。$set および $delset の識別コードは、ともにメッセージ・カタログ・ソース内かフィールド・タグ内に置けます。
- **m message text**
  m は、メッセージ識別コードを指定します。1 ～ NL_SETMAX の間の数値で定義します。メッセージ・テキストは、最後の $set ディレクティブに指定されているセット識別コードが付いたメッセージ・カタログ識別コードの *m* とともに、メッセージ・カタログ内に保存されます。メッセージ・テキストが空白で、ブランク文字フィールド区切り記号が存在する場合は、メッセージ・カタログに空のストリングが保存されます。メッセージ行にフィールド区切り記号または MESSAGE TEXT がなく、メッセージ行の後に改行または復帰がある場合は、カタログの既存メッセージは削除されます。 メッセージ識別コードは、昇順、不連続で 1 セット内としてください。 MESSAGE TEXT の長さは、0 ～ NL_TEXTMAX の範囲にします。

  注: メッセージ・テキスト・ソース・ファイルの空白行は無視されます。

メッセージ・プログラミング形式

MESSAGES については、次の推奨項目に従ってください。

- メッセージの最後の行は、必ず ¥n で終了します。
- メッセージの 2 行目以降の行は、タブを示す ¥t で開始します。
- メッセージの行を次の行に続ける場合は、行末に ¥n¥ を置き、次の行に続くことを示します。

- 行頭または行末の引用符は省略します。引用符があると、メッセージの先頭と末尾が区切られてしまいます。

複数のソース・ファイルの使用

ソース・ファイル・パラメーターには、複数のソース・ファイルを指定できます。すべてのファイルに含まれているメッセージは、1 つのソース・ファイルに定義されているセットとメッセージに関する同じ規則に従う必要があります。 たとえば、最初のソース・ファイルに、セット 1 ～ 3 のメッセージがある場合、次のソース・ファイルは、セット 3 で開始して、最初のソース・ファイルの最終メッセージ番号よりも大きな数字のメッセージ番号で終了する必要があります。あるいは、2 番目のソース・ファイルのセットは、最初のソース・ファイルに含まれている番号 (セット 3) よりも大きな番号で開始する必要があります。

メッセージの置換

変更するメッセージ・テキストと同一のセット番号とメッセージ番号を含むソース・ファイルを指定すると、既存メッセージ・カタログ内のメッセージを置換できます。ソース・ファイル内にあるその他のメッセージは変更されません。カタログの $QUOTE 値を更新するには、その後に続くソース・ファイルと同じ $QUOTE 文字を使用します。

メッセージ・カタログのソースの例

メッセージ・カタログの作成に使用するソース形式の例を以下に示します。メッセージは、引用符で区切ることができます。メッセージ・カタログに保存されているメニュー・テキストは、余分なブランク文字が削除されています。ここでは、3 組のメッセージの例を示します。セット 2 は削除されますが、セット 1 と 3 はメッセージ・カタログに保存されます。

```
$ Messages for my new product
$quote "

$set 1

1 "Error occurred.¥n"
$ The next message is continued on the next line.
2 "This is a very long message ¥n¥
¥t that requires another line to display. ¥n"
3 "Specify a value greater than %d.¥n"
4 "File %c cannot be used at this time.¥n"

$set 2
1 "Error %d occurred. ¥n"
2 "Flag not set.¥n"
3 "Number of arguments must be %d.¥n"

$set 4
1 "Before using this command, you must ¥
set the correct values in the %c box.¥n"
2 "You have not properly NLS enabled this function.¥n"
10 "Messages should end with a %c.¥n"

$delset 2
```

注: セット 1 のメッセージ 2 は、2 行で表示されます。セット 4 のメッセージ 1 は、1 行のメッセージです。

次は、MRGMSGCLG コマンドを使用して、メッセージ・カタログを作成する例です。

```
MRGMSGCLG CLGFILE('/MYPRODUCT/MESSAGES?US')
     SRCFILE('QSYS.LIB/MYLIB.LIB/MYSOURCE.FILE/US.MBR')
     CLGCCSID(*SRCCCSID) SRCCCSID(*SRCFILE)
     TEXT('Message catalog for USA')
```

この例では、MYSOURCE ファイルの MYLIB ライブラリーとメンバー US を使用して、/MYPRODUCT/MESSAGES ディレクトリーのストリーム・ファイル US にメッセージ・カタログを作成しています。メッセージ・カタログ内のデータの CCSID は、ソース・ファイルの CCSID タグと同じです。

特殊文字とエスケープ・シーケンス

テキスト・ストリングには、次の表に定義されているように、特殊文字とエスケープ・シーケンスを含めることができます。

特殊文字の説明	シーケンス
¥	¥¥
バックスペース	¥b
復帰	¥r
用紙送り	¥f
水平タブ	¥t
改行	¥n
8 進数ビット・パターン	¥ddd **注:** エスケープ・シーケンスの ¥ddd は、逆スラッシュの後に目標の文字の値を指定する 3 つの 8 進数字を続けます。逆スラッシュの後の文字が 8 進数字以外の場合は、逆スラッシュとデータは、テキストの一部になります。

メッセージ・カタログのオープン、抽出、およびクローズ: メッセージ・カタログを作成すると、次の C 関数を使用できます。

CATOPEN()
: メッセージ・カタログを開きます。

CATGETS()
: セット識別コードとメッセージ識別コードを使用して、メッセージ・カタログのメッセージを抽出します。

CATCLOSE()
: メッセージ・カタログを閉じます。

C 関数の CATOPEN は、メッセージ・カタログを開きます。名前にスラッシュ文字 (/) が含まれていない場合は、NLSPATH 環境変数と LC_MESSAGES カテゴリーを使用して、指定されたメッセージ・カタログを見付けます。名前に 1 つ以上のスラッシュ文字 (/) があると、その名前は、カタログを開くためのパス名として解釈されます。

NLSPATH 環境変数がない場合、または NLSPATH に指定されたパスにメッセージ・カタログが存在しない場合は、デフォルト・パスが使用されます。oflag の値が NO_CAT_LOCALE の場合は、LC_MESSAGES を設定する環境変数がデフォルト・パスに影響することがあります。oflag の値がゼロの場合は、LANG 環境変数もデフォルト・パスに影響することがあります。

C 関数およびメッセージ・カタログについて詳しくは、ILE C/C++ ランゲージ・リファレンス  PDF を参照してください。

グローバル・アプリケーションの納入

グローバル・アプリケーションの納入の準備をするときには、グローバリゼーションの問題が、そのアプリケーションをお客様がインストールして使用する方法にどのような影響を与えるかを考慮することが必要です。以下のトピックは、これらの問題について簡単に説明しています。

多国語システムに対するハードウェア・サポート

ここでいうハードウェアとは、iSeries サーバーを構成する物理的なキーボード、ディスプレイ、プリンター、および制御装置を指します。各国語のサポートについては、ハードウェアがサポートする機能の制限により、作成するアプリケーションのサポートにも制限が生じます。IBM 以外のハードウェアについては、それぞれのマニュアルを参照して、ハードウェア上の制限を確認してください。

文字データの翻訳

翻訳は、人間の言語による概念、アイデア、ステートメントなどの組み合わせの文字データの意味を他の言語による文化的に類似する意味に変更することです。翻訳をスムーズに行うためには、いくつかの 基本的な規則に従ってください。これらの規則のサブセットは、 16 ページの『ユーザー・インターフェース』トピックで提供されています。

グローバル・アプリケーションのお客様への納入

作成したアプリケーションをお客様に納入する作業には、パッケージング、保守、サポート、アプリケーションに関するユーザー教育なども含まれます。 これらの作業を世界中の異なる国や異なる文化を対象に実行するには、さまざまなタスクを考慮する必要があります。作成したアプリケーションをお客様に納入するときの作業について詳しくは、 7 ページの『パッケージングとインストール作業』 を参照してください。

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12