IBM

System i

System i 接続
System i ナビゲーター プラグインの開発

バージョン 6 リリース 1

IBM

System i

System i 接続
System i ナビゲーター プラグインの開発

バージョン 6 リリース 1

ご注意

本書および本書で紹介する製品をご使用になる前に、 105 ページの『特記事項』に記載されている情報をお読みください。

本書は、i5/OS (5761–SS1) バージョン 6、リリース 1、モディフィケーション 0 に適用されます。また、改訂版で特に断りがない限り、それ以降のすべてのリリースおよびモディフィケーションに適用されます。このバージョンは、すべての RISC モデルで稼働するとは限りません。また CISC モデルでは稼働しません。

IBM 発行のマニュアルに関する情報のページ

http://www.ibm.com/jp/manuals/

こちらから、日本語版および英語版のオンライン・ライブラリーをご利用いただけます。また、マニュアルに関するご意見やご感想を、上記ページよりお送りください。今後の参考にさせていただきます。

(URL は、変更になる場合があります)

お客様の環境によっては、資料中の円記号がバックスラッシュと表示されたり、バックスラッシュが円記号と表示されたりする場合があります。

原　典： System i
Connecting to System i
Developing System i Navigator plug-ins
Version 6 Release 1

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 2008.2

この文書では、平成明朝体™W3、平成明朝体™W7、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™W7、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、平成角ゴシック体™W7

目次

System i ナビゲーターのプラグインの開発

System i™ ナビゲーターのプラグイン機能を使用することにより、システム管理タスクおよびクライアント/サーバー・プログラムを、単一のアプリケーション環境に統合することができます。

プラグインを使用することにより、C++、Visual Basic、または Java™ で作成されたサード・パーティーのアプリケーションと専用の機能を、System i ナビゲーター・インターフェースに統合することができます。ここにあるさまざまなトピックを活用して、プラグインとは何か、プラグインを開発してカスタマイズするにはどうすればよいか、プラグインをエンド・ユーザーに配布するにはどうすればよいか、などについて理解してください。

注: コードのサンプルを使用すると、 103 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

V6R1 の新機能

「System i ナビゲーター プラグインの開発」のトピックにおいて、新機能や大幅に変更された箇所に関する情報を紹介します。

setup.ini の新規フィールド

setup.ini ファイルの情報セクションに、EclipseHelp という新規フィールドが追加されています。このフィールドは、プラグイン・アプリケーションがヘルプの作成時に、Eclipse プラットフォームを使用するかどうかを表しています。

新機能または変更点を確認する方法

技術的変更が加えられた箇所を確認できるように、Information Center では以下のものを使用しています。

- » イメージ: 新規情報または変更された情報の開始場所のマークです。
- « イメージ: 新規情報または変更された情報の終了場所のマークです。

PDF ファイルでの新規情報および変更された情報については、左余白にあるリビジョン・バー (|) で確認できます。

本リリースの新機能または変更点についてのその他の情報を見るには、「プログラム資料説明書」を参照してください。

System i ナビゲーターのプラグイン開発に関する PDF ファイル

この情報の PDF ファイルを表示および印刷することができます。

この文書の PDF 版を表示またはダウンロードするには、「System i ナビゲーター プラグインの開発」を選択します。

PDF ファイルの保存

表示用または印刷用の PDF ファイルをワークステーションに保存するには、次のようにします。

1. ご使用のブラウザーで PDF のリンクを右クリックする。

2. ローカルに PDF を保存するオプションをクリックする。
3. PDF を保存したいディレクトリーに進む。
4. 「**保存**」をクリックする。

Adobe Reader のダウンロード

これらの PDF を表示または印刷するには、Adobe® Reader がシステムにインストールされている必要があります。Adobe Reader は、Adobe の Web サイト (www.adobe.com/products/acrobat/readstep.html) から無償でダウンロードすることができます。

System i ナビゲーターのプラグイン・サポート

System i ナビゲーターのプラグイン・サポートでは、独自の機能とアプリケーションを、System i ナビゲーターと呼ばれる単一のユーザー・インターフェースに統合する便利な方法が用意されています。

これらの新機能やアプリケーションは、単純なものから複雑なものまで多岐にわたります。プラグインが提供する特定の新しい機能の内容にかかわらず、それを System i ナビゲーターに統合することにより、多大な恩恵を受けることができます。例えば、共通のシステム・タスクを System i ナビゲーターの単一の場所にバンドルすることにより、共通の管理と操作機能を大幅に単純化することができます。また、System i ナビゲーターのグラフィカル・インターフェースを使用すれば、最低限のスキルで、統合化された機能を簡単に完成させることができます。

プラグインを使ってできること

プラグインは、System i ナビゲーターが特定のユーザー・アクションに応答して呼び出す、定義済みのクラスとメソッドのセットです。

プラグインを使用すると、ツールとアプリケーションを表す System i ナビゲーター階層のオブジェクトやフォルダーを追加したり、修正したりすることができます。以下の項目を追加あるいは修正することにより、独自のフォルダーおよびオブジェクトのサポートを完全にカスタマイズすることができます。

コンテキスト・メニュー
コンテキスト・メニューは、アプリケーションの立ち上げ、新しいダイアログの表示、および振る舞いの追加または変更に使用します。

プロパティー・ページ
プロパティー・ページは、カスタマイズされた属性 (例: 追加のセキュリティー設定) をサポートするために使用します。 プロパティー・シートを持つものであれば、どのオブジェクトまたはフォルダーにもプロパティー・ページを追加することができます。

ツールバー
ツールバーとボタンを完全にカスタマイズすることができます。

カスタム・フォルダーおよびオブジェクト
カスタマイズした独自のフォルダーおよびオブジェクトを、 System i ナビゲーターのツリー階層に追加することができます。

プラグインの仕組み

Windows® のレジストリーに対する新規のプラグインが識別されると、System i ナビゲーターがその新規プラグインを検出し、新規構成の中にインストールします。その後、System i ナビゲーター階層に新規コンテナーが表示されます。ユーザーがコンテナーを選択すると、プラグインのコードが呼び出され、コンテナーの内容が取得されます。

System i ナビゲーターは、ListManager インターフェースに定義されたメソッドを呼び出すことによって、プラグインと通信します。このインターフェースを使用することにより、アプリケーションは System i ナビゲーターのツリー・ビューおよびリスト・ビューにリスト・データを提供できるようになります。アプリケーションを System i ナビゲーターに統合するには、このインターフェースをインプリメントする新規クラスを作成します。新規クラスのメソッドは、既存のアプリケーションを呼び出して、リスト・データを取得します。

図 1 は、System i ナビゲーターのツリーに新規コンテナーを追加する Java プラグインの動作を表しています。

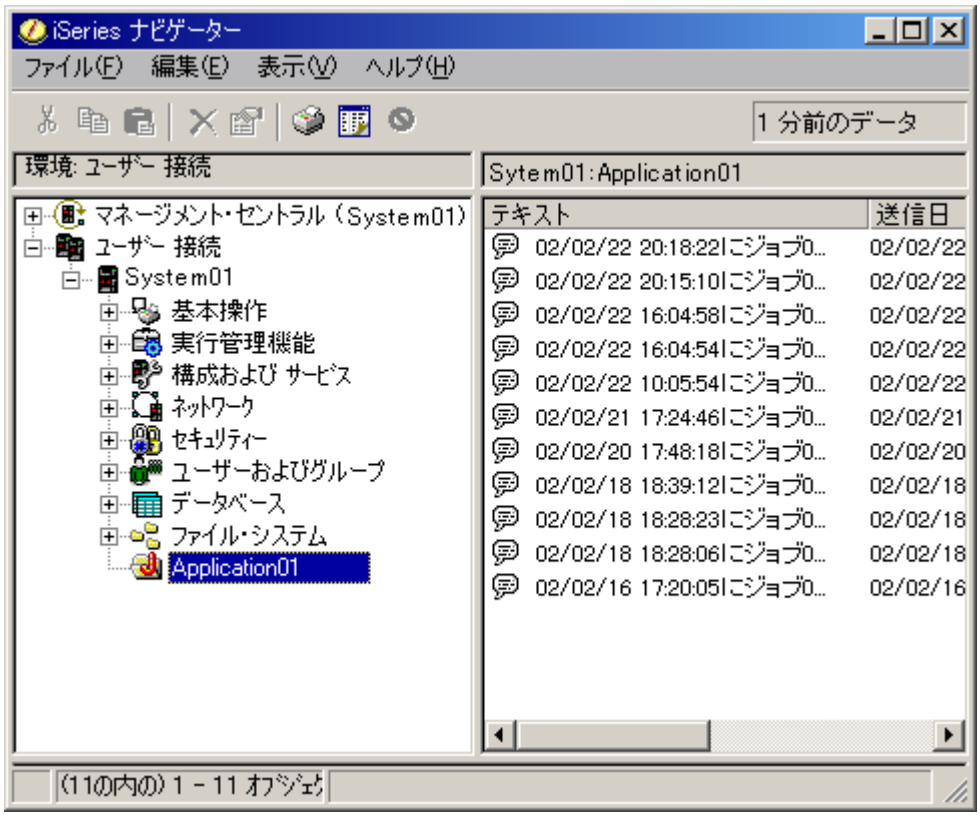

図 1. メッセージ待ち行列内のメッセージを表す、System i ナビゲーターのダイアログ

4 ページの図 2 は、System i ナビゲーターが、Java プラグインと通信してリスト・データを取得する様子を表しています。

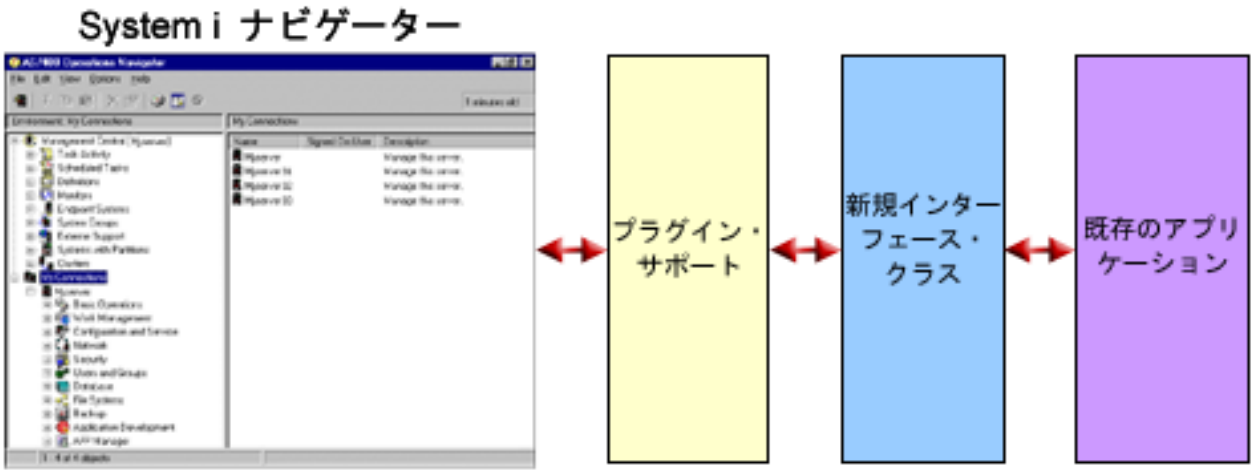

図 2. System i ナビゲーターがアプリケーションを呼び出してリスト・データを取得する方法

ActionsManager Java インターフェースを使用して、System i ナビゲーターを介してエンド・ユーザーがアプリケーションの特殊機能を使用できるようにしてください。ユーザーがメニュー項目を選択すると、System i ナビゲーターは別の ActionsManager メソッドを呼び出して、アクションを実行します (このインターフェースをインプリメントする新規の Java クラスを作成する必要があります)。 ActionsManager のインプリメンテーションが既存の Java アプリケーションを呼び出します。それによって、ユーザーによる特殊タスクの実行を支援する、確認ダイアログなどの複雑なユーザー・インターフェースが表示されます。

5 ページの図 3 は、ユーザーがメッセージ・オブジェクトを右クリックして、そのコンテキスト・メニューを表示する際の様子を表しています。

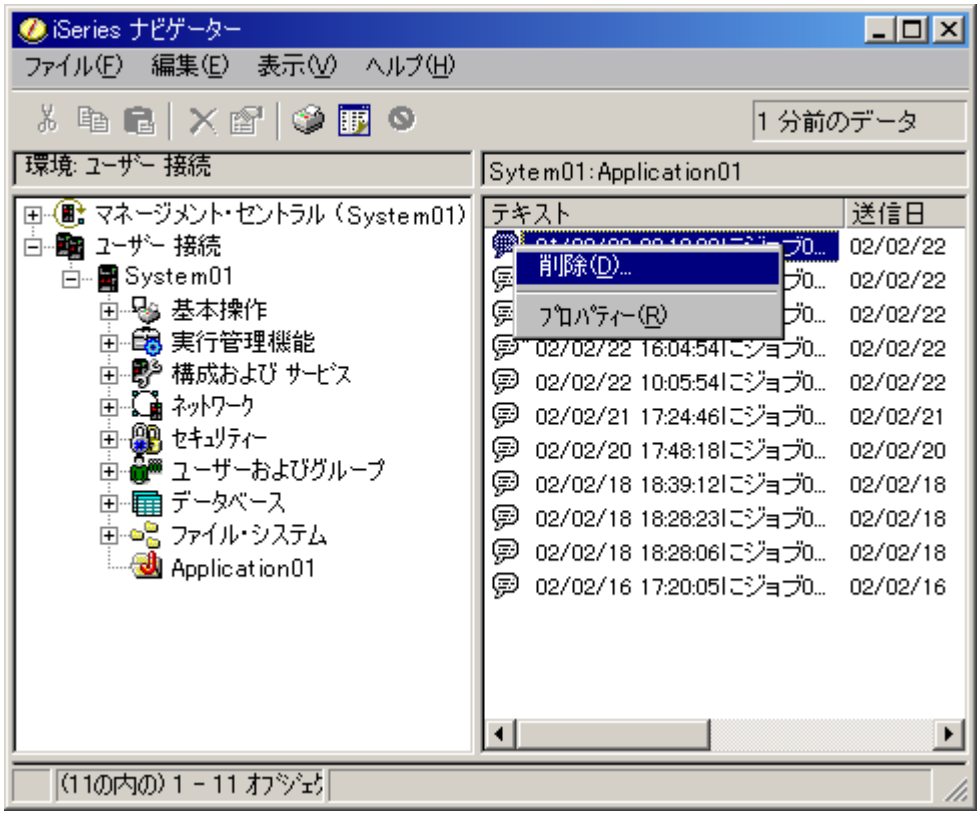

図 3. System i ナビゲーター・オブジェクトのコンテキスト・メニュー

ユーザーがメニュー項目を選択すると、System i ナビゲーターは別の ActionsManager メソッドを呼び出して、アクションを実行します。 System i ナビゲーターは、ActionsManager Java インターフェースの事前定義メソッドを呼び出します。このインターフェースは、メッセージ・オブジェクトでサポートされているメニュー項目のリストを取得します。 System i ナビゲーターのユーザー・インターフェースは、ユーザーがシステム・リソースを操作しやすくなるように設計されています。プラグイン機能のアーキテクチャーは、階層内のオブジェクトのリストを処理するためのインターフェースを定義するとともに、これらのオブジェクトへのアクションも定義することにより、このユーザー・インターフェースの設計を反映したものになっています。 3 番目のインターフェース DropTargetManager は、ドラッグ操作を扱います。

プラグインの要件

System i ナビゲーターのプラグイン要件は、使用するプログラム言語によって異なります。

C++ プラグイン

Microsoft の Visual C++ プログラム言語を使用してプラグインを開発する場合は、バージョン 4.2 以降で作成してください。

C++ プラグインには、以下の System i ナビゲーター API も必要です。

ヘッダー・ファイル	インポート・ライブラリー	ダイナミック・リンク・ライブラリー
cwbun.h	cwbunapi.lib	cwbunapi.dll
cwbunpla.h (アプリケーション管理 API)	cwbapi.lib	cwbunpla.dll

Java プラグイン

Java プラグインは、IBM® の Windows 用ランタイム環境である、Java Technology Edition 上で実行されます。以下の表は、System i Access for Windows ライセンス・プログラムとともにインストールされる Java のバージョンを示したものです。

リリース	JRE	Swing	JavaHelp
V6R1	5.0	N/A	1.1.1
V5R4	1.4.2	N/A	1.1.1
V5R3	1.4.1	N/A	1.1.1
V5R2	1.3.1	N/A	1.1.1

すべての Java プラグインには、そのプラグインに関する情報を含む Windows の資源 DLL が必要です。この DLL を使用すると、System i ナビゲーターは、プラグインのインプリメンテーションをロードしなくても、 System i ナビゲーターのオブジェクト階層に機能を表示することができます。サンプルの資源 DLL は、Microsoft Visual C++ バージョン 4.2 を使用して作成されたものですが、Windows の資源のコンパイルとリンクをサポートしている C コンパイラーであれば、どれを使用してもかまいません。

System i ナビゲーターには、デバッグの援助機能として Java コンソールが用意されています。 Java コンソールを活動化するには、必要なコンソール標識を Windows のレジストリーに書き込むためのレジストリー・ファイルを選択します。コンソールが活動化されると、JIT コンパイラーがオフになり、ソース・コードの行番号がスタック追跡に表示され、 System i ナビゲーターの Java インフラストラクチャーで発生する例外はすべてメッセージ・ボックスに表示されます。 System i Access for Windows Toolkit 内のサンプル Java プラグインには、コンソールを活動化および非活動化するためのレジストリー・ファイルが用意されています。

サンプルのユーザー・インターフェースは、IBM Toolbox for Java コンポーネントに含まれている Graphical Toolbox for Java を使用して開発されたものです。この Toolbox は、System i Access for Windows のオプションとしてインストール可能な機能です。 System i Access for Windows 製品の初期インストール時にインストールすることもできますし、System i Access for Windows の「コントロール パネル」にある「アプリケーションの追加と削除」を使用して、後から選択的にインストールすることもできます。

Visual Basic プラグイン

Visual Basic プラグインは、Visual Basic のバージョン 5.0 のランタイム環境で動作します。

関連概念

16 ページの『サンプル・プラグインのインストールと実行』
Programmer's Toolkit には、サポートされている各プログラム言語ごとに、サンプル・プラグインが用意されています。

プラグインの配布

i5/OS® アプリケーションにプラグイン・コードを組み込むことによって、System i ナビゲーターのユーザーにプラグイン・コードを配布することができます。

アプリケーションのインストール・プログラムによって、プラグインのコード・バイナリー、レジストリー・ファイル、および変換可能な資源が統合ファイル・システムのフォルダーに書き込まれます。このプロセスの完了後、エンド・ユーザーは、「**ユーザー接続 (My Connections)**」 → 「**インストール・オプション**

| **(Install Options)**」 → 「**インストール・プラグイン (Install Plug-ins)**」と右クリックして選択することで、
| プラグインをインストールすることができます。「プラグインのインストール (Install Plug-ins)」ウィザー
| ドでは、プラグインのコードをユーザーのワークステーションにコピーしてから、ユーザーのワークステー
| ションの言語設定に基づいて適切な変換可能資源をダウンロードし、レジストリー・ファイルを実行して、
| プラグインのレジストリー情報を Windows レジストリーに書き込みます。 System i Access for Windows
| がまだインストールされていない場合、エンド・ユーザーはカスタムのセットアップ・タイプを使用するこ
| とで、初期インストール時にプラグインをインストールできます。

プラグインのタイプ	インストール・ディレクトリー	組み込まれるファイル
C++	/QIBM/USERDATA/OpNavPlugin/ <*vendor*>.<*component*>	• プラグインのレジストリー・ファイル • プラグインの System i Access for Windows セットアップ・ファイル • プラグインの ActiveX サーバー DLL、および関連コード DLL
Java	/QIBM/USERDATA/OpNavPlugin/ <*vendor*>.<*component*>	• プラグインのレジストリー・ファイル • プラグインの System i Access for Windows セットアップ・ファイル • Java クラス、AUIML、HTML、GIF、PDML、PCML の各ファイル、および直列化ファイルがすべて含まれている、Java JAR ファイル
Visual Basic	/QIBM/USERDATA/OpNavPlugin/ <*vendor*>.<*component*>	• プラグインのレジストリー・ファイル • プラグインの System i Access for Windows セットアップ・ファイル • プラグインの ActiveX サーバー DLL、および関連コード DLL

注:

- <*vendor*>.<*component*> サブディレクトリーは、レジストリー・ファイルに指定されたものと同じでなければなりません。
| • System i ナビゲーターは、GUIPlugin の位置をサポートしていません。 GUIPlugin の位置から
| OpNavPlugin の位置に、プラグインを移行する必要があります。

また、すべてのプラグイン、<*vendor*>.<*component*> サブディレクトリーの下に、MRI29*XX* という名前のディレクトリーを少なくとも 1 つ作成しなければなりません。*XX* は、サポートされる言語を示します。例: MRI2924 (English)。このディレクトリーには、以下の項目についての適切な各国語バージョンが含まれなければなりません。

- プラグインの資源 DLL
- プラグインのヘルプ・ファイル
- プラグインの MRI セットアップ・ファイル

プラグインのアップグレードまたはアンインストール

| ユーザーがプラグインをインストールした後であれば、後日アップグレードを実行したり、バグ・フィック
| スを適用したりすることができます。システムでコードがアップグレードされた後、ユーザーは、System i

ナビゲーターの「**プラグインの更新または保守 (Update or Service Plug-ins)**」」オプションを使用して、プラグインの更新を手動で開始することができます。

System i Access for Windows はアンインストールをサポートしているため、エンド・ユーザーは、いつでもワークステーションからプラグインを完全に除去することができます。ワークステーションにインストールされているプラグインを確認するには、システムの System i ナビゲーターの「プロパティー (Properties)」ページにある「**プラグイン (Plug-ins)**」タブをクリックします。

アプリケーション管理によるプラグインへのアクセスの制限

System i ナビゲーターのシステム・ベースのアプリケーション管理サポートを使用して、プラグインにアクセス可能なユーザーおよびユーザー・グループを制御することができます。

Setup.ini ファイル

プラグインの setup.ini ファイルは、インストール・ウィザードに、クライアント・ワークステーションで System i ナビゲーター・プラグインをインストールする場合に必要な情報を提供します。また、このファイルには、プラグインをアップグレードする時期あるいはサービス・リリースを適用する時期を、チェック・サービス・レベル・プログラムが判別する場合に使用する情報も入っています。

セットアップ・ファイルの名前は、SETUP.INI でなければなりません。また、このファイルは、そのシステムにおけるプラグインの 1 次ディレクトリーである *<vendor>.<component>* に置かれている必要があります。

ファイルのフォーマットは、Windows の標準構成 (.INI) ファイルのフォーマットに準拠しています。ファイルは、以下の 4 つのパートに分けられます。

- プラグイン情報
- サービス
- クライアント・ワークステーションにインストールするファイルの識別を行うためのセクション
- クライアント・ワークステーションで実行する出口プログラムの識別を行うためのセクション

関連概念

19 ページの『サンプル Visual Basic プラグイン・ファイルのディレクトリー』
以下に示す表は、サンプル Visual Basic プラグインと共に組み込まれているすべてのファイルについて説明したものです。

例: setup.ini の情報セクション:

setup.ini ファイルの最初のセクション (Plug-in Info) には、プラグインに関するグローバル情報が含まれています。

```
[Plugin Info]
EclipseHelp=YES
ExpressMaxRelease=V6R1M0
ExpressMinRelease=V5R2M0
Name=Sample plug-in
NameDLL=sampmri.dll
NameResID=128
Description=Sample plug-in description
DescriptionDLL=sampmri.dll
DescriptionResID=129
Version=0
VendorID=IBM.Sample
JavaPlugin=YES
```

Setup.ini の [Plugin Info] セクションのフィールド	フィールドの説明
Name	プラグインの英語名です。プラグインのインストール時に、翻訳名を判別できない場合に、この名前が表示されます。
NameDLL	プラグインの翻訳名を含む資源 DLL の名前です。 この DLL は、プラグインの MRI のディレクトリーにあります。
NameResID	MRI DLL 内の翻訳名の資源識別コードです。このフィールドには、プラグインの 1 次レジストリー・キーで定義されている NameID フィールドと同じ値が含まれていなければなりません。
Description	プラグインの英語の説明です。この説明は、プラグインのインストール時に、翻訳された説明を判別できない場合に表示されます。
DescriptionDLL	プラグインの翻訳された説明を含む資源 DLL の名前です。この DLL は、プラグインの MRI のディレクトリーにあります。
DescriptionResID	MRI DLL 内の翻訳された説明の資源識別コードです。 このフィールドには、プラグインの 1 次レジストリー・キーで定義されている DescriptionID フィールドと同じ値が含まれていなければなりません。
Version	プラグインのリリース・レベルを示す数値です。クライアント・ワークステーションでプラグインをアップグレードする必要があるかどうかを判別する際に、この値が使用されます。プラグインのリリースごとに、この値は、前の値よりも大きい値になります。 この Version 値は、クライアント・ワークステーションにインストールされているプラグインの現行の Version 値と比較されます。この Version 値が、すでにクライアント・ワークステーションに存在している値よりも大きい場合、プラグインは新しいバージョンに更新されます。
VendorID	プラグインの識別に使用される *<vendor>.<component>* という形式のストリングです。プラグインのレジストリー・キーを System i Access for Windows のレジストリー・ツリーに作成する際に、このストリングが使用されます。 VendorID は、プラグインのインストール先となるシステム上のパスの *<vendor>.<component>* 部分と同じである必要があります。
JavaPlugin	このプラグインが Java プラグインであるかどうかを示すフィールドです。 Java プラグインの場合は、すべての JAR ファイルを ¥PLUGINS¥*<vendor>.<component>* ディレクトリーにインストールしなければならないため、この値を使用して、この処理をインストール・プロセスで実行するかどうかを決定します。 Java プラグインの場合、この値が NO に設定されているか、この値が存在しない時には、そのプラグインはインストール後に作動させることができません。

EclipseHelp	ヘルプを作成時する際、プラグイン・アプリケーションが Eclipse プラットフォームを使用するかどうかを示すフィールドです。 Eclipse のヘルプは、Java プラグインの場合にのみ使用します。 Eclipse 対応のプラグインのヘルプは、setup.ini ファイルに指定された圧縮ファイルの中に含まれています。言語ごとに適切な MRI29*xx* ディレクトリーから圧縮ファイルを入手し、[InstallDir]¥Eclipse¥Plugins ディレクトリーに解凍します。
	
	この項目が存在しない場合、デフォルト値は NO に設定されます。
	
	setup.ini ファイルに EclipseHelp=YES が指定されている場合、EclipseHelp セクションには以下の情報が入ります。
	`[Eclipse]`
	`EclipseDirName=com.ibm.iSeries.`*`help plug-in name`*`.help.doc`
	`EclipseZipFile=superzip.zip`

Setup.ini の [Plugin Info] セクションのフィールド	フィールドの説明
EclipseDirName	ヘルプ・ファイルの解凍先ディレクトリーです。圧縮ファイルにはすでにディレクトリー構造が含まれているため、このディレクトリー名はアンインストール時にのみ必要になります。
EclipseZipFile	ディレクトリーに解凍する圧縮ファイルの名前です。
ExpressMinRelease	そのプラグインをサポートするオペレーティング・システムの最小リリースです (例: V5R2M0)。
ExpressMaxRelease	そのプラグインをサポートするオペレーティング・システムの最大リリースです (例: V6R1M0)。

例: setup.ini のサービス・セクション:

setup.ini ファイルの 2 つ目のセクションである Service は、クライアント・ワークステーションにプラグインの新しい修正レベルを適用すべきかどうかを判別する際に、サービス・レベルの検査プログラムが必要とする情報を提供します。

```
[Service]
FixLevel=0
AdditionalSize=0
```

Setup.ini の [Service] セクションのフィールド	フィールドの説明
FixLevel	プラグインのサービス・レベルを示す数値です。 バージョンのサービス・リリースが改められた場合、この値には、それまでよりも大きな値を使用しなければなりません。 この FixLevel 値は、クライアントのワークステーションにインストール済みのプラグインの現行の FixLevel 値と比較されます。この FixLevel 値が、クライアント・ワークステーションにインストールされているプラグインの値よりも大きい場合、ユーザーが System i ナビゲーターの「**プラグインの更新または保守 (Update or Service Plug-ins)**」オプションを選択していれば、そのプラグインは新しい FixLevel にアップグレードされます。プラグインが新しいバージョンまたはリリース・レベルにアップグレードされた場合、この値をゼロにリセットしなければなりません。
AdditionalSize	サービス・リリースの適用時にプラグインに追加される、新規または追加の実行可能ファイルを保管するために必要なディスク・スペースの量です。インストールではこの値を使用して、ワークステーションにプラグイン用の十分なディスク・スペースがあるかどうかを判別します。

例: setup.ini のファイル識別セクション:

setup.ini ファイルのこの部分には、クライアント・ワークステーションにインストールされるファイルを識別する情報が含まれています。

ファイルが示されているセクションは、各ファイルのソースとターゲットの場所を識別します。これらのファイル・セクションは、初期インストール時、あるいは新しいバージョンまたはリリース・レベルへのアップグレード時に使用されます。

各ファイル・セクションのファイル項目の書式は `n=file.ext` とします。ここで、`n` はそのセクションのファイルの番号です。この番号は 1 から始まり、すべてのファイルが該当するセクションにリストされるまで 1 ずつ増加します。例えば、以下のようになります。

```
[Base Files]
1=file1.dll
2=file2.dll
3=file3.dll
```

常に、ファイル名のみを指定します。ディレクトリー・パスの名前は指定しないでください。ファイル・セクションに項目がない場合、そのセクションは無視されます。

注: Programmer's Toolkit には、C++、Java、および Visual Basic の 3 つの異なるサンプル・プラグイン用のサンプル・セットアップ・ファイルが用意されています。

Setup.ini のセクション	**説明**
[Base Files]	Client Access のインストール・ディレクトリーの下にある ¥PLUGINS¥<*vendor*>.<*component*> にコピーされるファイルです。通常、プラグイン用の ActiveX サーバー DLL (およびそれに関連付けられているコード DLL) はここに指定します。
[Shared Files]	Client Access Shared ディレクトリーにコピーされるファイルです。
[System Files]	¥WINDOWS¥SYSTEM または ¥WINNT¥SYSTEM32 ディレクトリーにコピーされるファイルです。
[Core Files]	¥WINDOWS¥SYSTEM または ¥WINNT¥SYSTEM32 ディレクトリーにコピーされるファイルです。これらのファイルは、複数のアプリケーションに共通のファイルです。 それぞれの共通ファイルは、そのファイルを使用するアプリケーションの総数を示す数に関連付けられています。その共通ファイルを使用するアプリケーションが除去されると、その数が減分されます。共通ファイルは、そのファイルを使用する最後のアプリケーションがアンインストールされるまで除去されません。 通常、これらのファイルは再配布することができます。
[MRI Files]	システム上のプラグインの MRI ディレクトリーから、ワークステーション上の CLIENT ACCESS¥MRI29XX¥<*vendor*>.<*component*> ディレクトリーにコピーされるファイルです。通常、プラグインのロケール依存資源が、ここに置かれます。これには、資源の MRI DLL 名を含めます。
[Java MRI29xx] (29xx は、ファイルの NLV フィーチャー・コード)	システム上にあるプラグインの MRI29xx ディレクトリーから、基本ファイルのインストール先と同じディレクトリーにコピーされる Java ファイルです。通常、プラグインの JAR MRI29xx 資源が、ここに置かれます。 Java プラグインがサポートしている各 MRI29xx ディレクトリーの Java MRI29xx セクションで、これらのファイルをリストする必要があります。このセクションを使用するのは、Java プラグインだけです。
[Help files]	システム上のプラグインの MRI ディレクトリーから、ワークステーション上の CLIENT ACCESS¥MRI29XX¥<*vendor*>.<*component*> ディレクトリーにコピーされる .HLP および .CNT ファイルです。これらのファイルへのディレクトリー・パスは、Windows のレジストリーの HKEY_LOCAL_MACHINE¥SOFTWARE¥MICROSOFT¥WINDOWS¥HELP に書き込まれます。
[Registry files]	プラグインに関連付けられている Windows のレジストリー・ファイルです。

Setup.ini のセクション	説明

[Dependencies]	プラグインのインストールする前にインストールしておく必要がある、サブコンポーネントを定義するセクションです。 AS400_Client_Access_Express が必要となるのは、System i ナビゲーターの基本サポート・サブコンポーネント以外のサブコンポーネントのインストールを、プラグインが必要としている場合のみです。
	
	AS400_Client_Access_Express
	• サブコンポーネントは、コンマで区切られたリストで指定されます。1 つのサブコンポーネントは、1 つの番号として指定されます (AS400_Client_Access_Express=3) 。CWBAD.H ヘッダー・ファイルには、CWBAD_COMP_ という接頭部の付いた定数のリストが含まれています。これらの定数は、AS400_Client_Access_Express のコンマ区切りのリストで使用される数値を提供しています。これらの CWBAD_COMP_ 定数は、PC5250 のフォント・サブコンポーネントを指定するものです。AS400_Client_Access_Express 値では使用しないでください。
	//5250 表示およびプリンター・エミュレーターのサブコンポーネント
	`#define CWBAD_COMP_PC5250_BASE_KOREAN (150)`
	`#define CWBAD_COMP_PC5250_PDFPDT_KOREAN (151)`
	`#define CWBAD_COMP_PC5250_BASE_SIMPCHIN (152)`
	`#define CWBAD_COMP_PC5250_PDFPDT_SIMPCHIN (153)`
	`#define CWBAD_COMP_PC5250_BASE_TRADCHIN (154)`
	`#define CWBAD_COMP_PC5250_PDFPDT_TRADCHIN (155)`
	`#define CWBAD_COMP_PC5250_BASE_STANDARD (156)`
	`#define CWBAD_COMP_PC5250_PDFPDT_STANDARD (157)`
	`#define CWBAD_COMP_PC5250_FONT_ARABIC (158)`
	`#define CWBAD_COMP_PC5250_FONT_BALTIC (159)`
	`#define CWBAD_COMP_PC5250_FONT_LATIN2 (160)`
	`#define CWBAD_COMP_PC5250_FONT_CYRILLIC (161)`
	`#define CWBAD_COMP_PC5250_FONT_GREEK (162)`
	`#define CWBAD_COMP_PC5250_FONT_HEBREW (163)`
	`#define CWBAD_COMP_PC5250_FONT_LAO (164)`
	`#define CWBAD_COMP_PC5250_FONT_THAI (165)`
	`#define CWBAD_COMP_PC5250_FONT_TURKISH (166)`
	`#define CWBAD_COMP_PC5250_FONT_VIET (167)`
	
	注: AS400_Client_Access_Express の値は存在する場合にのみ使用されます。存在しない場合は、このセクションは無視されます。

[Service Base Files]	System i Access for Windows のインストール・ディレクトリーの下にある ¥PLUGINS¥<*vendor*>.<*component*> にコピーされるファイルです。
[Service Shared Files]	System i Access for Windows の Shared ディレクトリーにコピーされるファイルです。
[Service System Files]	¥WINDOWS¥SYSTEM または ¥WINNT¥SYSTEM32 ディレクトリーにコピーされるファイルです。
[Service Core Files]	¥WINDOWS¥SYSTEM または ¥WINNT¥SYSTEM32 ディレクトリーにコピーされるファイルです。これらのファイルは、複数のアプリケーションに共通のファイルです。 それぞれの共通ファイルは、そのファイルを使用するアプリケーションの総数を示す数に関連付けられています。その共通ファイルを使用するアプリケーションが除去されると、その数が減分されます。共通ファイルは、そのファイルを使用する最後のアプリケーションがアンインストールされるまで除去されません。 通常、これらのファイルは再配布することができます。
[Service Registry Files]	プラグインに関連付けられている Windows のレジストリー・ファイルです。

例: setup.ini の出口プログラム・セクション:

setup.ini ファイルの最終部分にあるセクションで、インストール、アップグレード、またはアンインストールの前後にクライアント・ワークステーションで実行されるプログラムを識別します。

こうしたプログラムを識別および実行するために、これらの出口プログラム・セクションで使用する構文について、以下の例で説明します。

例 1: 初期インストール時にファイルがインストールされる前に呼び出される、オプションのプログラム。

```
[PreInstallProgram]
Program=whatever.exe
CmdLine=
CheckReturnCode=
Wait=
```

[PreInstallProgram] のフィールド	**説明**
Program	必須。パスが指定されている場合、ファイル名のみが使用されます。プログラムは、インストール・ソースのプラグインの *<vendor>.<component>* パスに存在している必要があります。
CmdLine	オプション。特定のプログラムによりさまざまなコマンドが必要とされます。
CheckReturnCode	オプション。デフォルトは `N` です。この値を `Y` に設定した場合、戻りコードがゼロ以外だとプラグインのインストールは続行されません。プログラムがゼロ以外の戻りコードを返した場合、メッセージは表示されませんが、instlog.txt ファイルへの記録は行われます。
Wait	オプション。続行する前に、プログラムが終了するのを待ちます。デフォルトは、`Y` です。`CheckReturnCode=Y` の場合、ここで指定されているものに関係なく `Wait=Y` が使用されます。

例 2: 初期インストール時にファイルがインストールされた後に呼び出される、オプションのプログラム。

```
[PostInstallProgram]
Program=whatever.exe
CmdLine=
CheckReturnCode=
Wait=
```

[PostInstallProgram] のフィールド	**説明**
Program	必須。パスが指定されている場合、ファイル名のみが使用されます。ワークステーション上にあるプラグインの *<vendor>.<component>* ディレクトリーに、プログラムが存在している必要があります。
CmdLine	オプション。特定のプログラムによりさまざまなコマンドが必要とされます。
CheckReturnCode	オプション。`PostInstallProgram` セクションは、`PreInstallProgram` セクションと同じ値をサポートしています。 `CheckReturnCode` 値は、参考用に instlog.txt ファイルに記録されます。
Wait	オプション。続行する前に、プログラムが終了するのを待ちます。`PostInstallProgram` セクションは、`PreInstallProgram` セクションと同じ値をサポートしています。

例 3: ファイルのアンインストール前に呼び出されるオプションのプログラム。

```
[UninstallProgram]
Program=whatever.exe
CmdLine=
CheckReturnCode=
Wait=
```

[UninstallProgram] のフィールド	説明
Program	必須。パスが指定されている場合、ファイル名のみが使用されます。ワークステーション上にあるプラグインの *<vendor>.<component>* ディレクトリーに、プログラムが存在している必要があります。
CmdLine	オプション。特定のプログラムによりさまざまなコマンドが必要とされます。
CheckReturnCode	オプション。デフォルトは `N` です。この値を `Y` に設定した場合、戻りコードがゼロ以外だとプラグインのアンインストールは続行されません。戻りコードがゼロ以外の場合、メッセージは表示されませんが、instlog.txt ファイルへの記録は行われます。
Wait	オプション。続行する前に、プログラムが終了するのを待ちます。デフォルトは、`Y` です。`CheckReturnCode=Y` の場合、ここで指定されているものに関係なく `Wait=Y` が使用されます。

例 4: ファイルのアップグレード前に呼び出されるオプションのプログラム

```
[PreUpgradeProgram]
Program=whatever.exe
CmdLine=
CheckReturnCode=
Wait=
```

[PreUpgradeProgram] のフィールド	説明
Program	必須。パスが指定されている場合、ファイル名のみが使用されます。プログラムは、インストール・ソースのプラグインの *<vendor>.<component>* パスに存在している必要があります。
CmdLine	オプション。特定のプログラムによりさまざまなコマンドが必要とされます。
CheckReturnCode	オプション。デフォルトは `N` です。この値を `Y` に設定した場合、戻りコードがゼロ以外だとプラグインのインストールは続行されません。戻りコードがゼロ以外の場合、メッセージは表示されませんが、instlog.txt ファイルへの記録は行われます。
Wait	オプション。続行する前に、プログラムが終了するのを待ちます。デフォルトは、`Y` です。`CheckReturnCode=Y` の場合、ここで指定されているものに関係なく `Wait=Y` が使用されます。

例 5: ファイルのアップグレード後に呼び出されるオプションのプログラム

```
[PostUpgradeProgram]
Program=whatever.exe
CmdLine=
CheckReturnCode=
Wait=
```

[PostUpgradeProgram] のフィールド	説明
Program	必須。パスが指定されている場合、ファイル名のみが使用されます。プログラムは、インストール・ソースのプラグインの *<vendor>.<component>* パスに存在している必要があります。
CmdLine	オプション。特定のプログラムによりさまざまなコマンドが必要とされます。
CheckReturnCode	オプション。PostUpgradeProgram セクションは、PreUpgradeProgram セクションと同じ値をサポートしています。 CheckReturnCode 値は、参考用に instlog.txt ファイルに記録されます。
Wait	オプション。続行する前に、プログラムが終了するのを待ちます。デフォルトは、Y です。CheckReturnCode=Y の場合、ここで指定されているものに関係なく Wait=Y が使用されます。

MRI セットアップ・ファイル

MRI セットアップ・ファイルは、System i ナビゲーター・プラグインに関連付けられたロケール依存の資源をクライアント・ワークステーション上にインストールする際に、「プラグインのインストール (Install Plug-ins)」ウィザードが必要とする情報を提供します。

この MRI セットアップ・ファイルは、名前を MRISETUP.INI にしなければなりません。プラグインがサポートする各国語ごとに、いずれかのバージョンのこのファイルが System i ファイル・システム内のサブディレクトリー MRI29XX になければなりません。

ファイルのフォーマットは、Windows の標準構成 (.INI) ファイルのフォーマットに準拠しています。ファイルには、MRI Info セクションだけが含まれています。MRI Info セクションでは、プラグインの MRI のバージョン値が指定されています。プラグインの MRI には、特定の言語のヘルプ・ファイル (.HLP と .CNT) だけでなく、すべての資源 DLL も含まれています。例えば、以下のようになります。

```
[MRI Info]
Version=0
```

「プラグインのインストール (Install Plug-ins)」ウィザードは、MRI ファイルの Version 値を検査します。このファイル内の MRI のバージョン値は、プラグインの SETUP.INI ファイル内のバージョン値と一致していなければなりません。これらの値が一致しない場合、そのプラグインは「プラグインのインストール (Install Plug-ins)」ウィザードのリストに含められず、クライアント・ワークステーションにファイルはコピーされません。 Programmer's Toolkit には、サンプル・プラグインとサンプル MRI セットアップ・ファイルが用意されています。

関連概念

19 ページの『サンプル Visual Basic プラグイン・ファイルのディレクトリー』
以下に示す表は、サンプル Visual Basic プラグインと共に組み込まれているすべてのファイルについて説明したものです。

System i ナビゲーターに対するプラグインの識別

エンド・ユーザーのワークステーションにプラグイン・ソフトウェアがインストールされる際に、Windows レジストリーに情報を提供することによって、System i ナビゲーターでプラグインを識別できるようにします。

レジストリー項目によって、プラグイン・コードの位置を指定し、特殊な System i ナビゲーター・インターフェースをインプリメントするクラスを識別します。特定のシステムでプラグインの機能を活動化するか

どうかを System i ナビゲーターが判別する際に使用する、追加のレジストリー情報を指定することができます。例えば、プラグインが i5/OS の特定のリリースを最小要件として必要とする場合や、プラグインを機能させるために特定の製品をシステムにインストールするよう指定されている場合、などです。

プラグインのインストール後、ユーザーが System i ナビゲーター階層ツリーでシステムを選択すると、System i ナビゲーターはそのシステムを調べて、そのシステムが新規プラグインをサポート可能かどうかを判別します。プラグインのレジストリー項目に指定されたソフトウェア前提条件が、システムにインストールされたソフトウェアと比較されます。プラグインの要件が満たされている場合、新しい機能が階層ツリーに表示されます。要件が満たされていない場合、そのシステムにプラグインの機能は表示されません。

サンプル・プラグインのインストールと実行

Programmer's Toolkit には、サポートされている各プログラム言語ごとに、サンプル・プラグインが用意されています。

これらのサンプルは、プラグインの動作を学習するための優れた教材であり、独自のプラグインを開発する出発点として効果的です。Programmer's Toolkit をインストールしていない場合は、インストールしてからでないと、サンプル・プラグインを扱うことはできません。 System i Access for Windows では、「コントロール パネル」の「アプリケーションの追加と削除」を使用して、Toolkit をインストールすることができます。

注: サンプル・プラグインを扱う前に、これら 3 つの言語によるプラグインを開発するための、各言語に固有の要件を知っておいてください。

関連概念

5 ページの『プラグインの要件』
System i ナビゲーターのプラグイン要件は、使用するプログラム言語によって異なります。

サンプル C++ プラグインのセットアップ

このタスクでは、サンプル ActiveX サーバー DLL をビルドして実行します。

このサンプルは、実際に機能する Developer Studio のワークスペースを提供するものです。Developer Studio のワークスペースでは、ブレークポイントを設定して、一般的な System i ナビゲーター・プラグインの振る舞いを監視することができます。また、このサンプルを使用すると、プラグイン・コードのコンパイル用およびリンク用に Developer Studio 環境が正しくセットアップされているかどうかについて、検証することもできます。

ワークステーション上にサンプルの C++ プラグインをセットアップするには、以下のステップに従います。

C++ プラグインをダウンロードする	実行可能ファイル cppsmppq.exe をダウンロードしてください。このファイルを実行すると、プラグインに関連付けられているファイルがすべて解凍されます。新しいディレクトリー C:¥MyProject を作成し、解凍したファイルをすべてそこにコピーしてください。別のディレクトリーを作成した場合は、レジストリー・ファイルを修正して、プラグインの正しい場所を指定する必要があります。

ActiveX サーバー DLL のビルドの準備をする	1. 「MyProject」という名前の新しいディレクトリーをローカルのハード・ディスクに作成します。この例では、ローカル・ドライブとして C: ドライブを想定しています。 **注:** 新しいディレクトリーが C:¥MyProject ではない場合は、レジストリー・ファイルを変更する必要があります。 2. サンプル・ファイルをすべてこのディレクトリーにコピーします。 Programmer's Toolkit - System i ナビゲーター Plug-ins Web ページからサンプルをダウンロードできます。 3. Developer Studio の「**ファイル**」メニューをオープンし、「**ワークスペースを開く (Open Workspace)**」を選択します。 4. 「プロジェクトのワークスペースを開く (Open Project Workspace)」ダイアログで、「**MyProject**」ディレクトリーに移動し、「**ファイルの種類**」で「**Makefiles (*.mak)**」を選択します。 5. **sampext.mak** を選択し、「**オープン**」をクリックします。 6. 「**ツール**」メニューをオープンし、「**オプション**」を選択します。 7. 「**ディレクトリー (Directories)**」タブで、Client Access の Include ディレクトリーが、組み込みファイルの検索パスの先頭に表示されていることを確認します。 8. 「**表示するディレクトリー (Show directories for)**」で、「**ライブラリー・ファイル (Library files)**」を選択します。 Client Access の Lib ディレクトリーが、ライブラリー・ファイルの検索パスの先頭に表示されていることを確認します。 9. 「OK」をクリックして変更を保管し、Developer Studio をクローズした後、再オープンします。Developer Studio で検索パスの変更を強制的にハード・ディスクに保管するには、これ以外の方法はありません。
ActiveX サーバー DLL をビルドする	1. Developer Studio の「**ビルド (Build)**」メニューをオープンし、「**省略時構成の設定 (Set Default Configuration)**」を選択します。 2. 「省略時のプロジェクト構成 (Default Project Configuration)」ダイアログで、samptext の「**Win32 デバッグ構成 (Win32 Debug Configuration)**」を選択します。 3. 「**ビルド (Build)**」メニューをオープンし、「**すべて再ビルド (Rebuild All)**」を選択して DLL をコンパイルし、リンクします。 **注:** DLL が正常にコンパイルおよびリンクされない場合は、「ビルド (Build)」ウィンドウに表示されているエラー・メッセージをダブルクリックし、エラーの場所を見つけて修正します。次に、「**ビルド (Build)**」メニューをオープンし、「**sampext.dll**」を選択して再度ビルドを実行してください。
資源ライブラリーをビルドする	サンプルには、プラグイン用の翻訳可能なテキスト・ストリングとその他のロケール依存資源を含む資源 DLL が含まれています。したがって、この DLL は、独自に作成する必要はありません。プラグインがサポートする言語が 1 つだけであっても、プラグイン・コードは、テキスト・ストリングとロケール固有の資源をこの資源ライブラリーからロードしなければなりません。 資源 DLL をビルドするには、以下のステップを実行してください。 1. Developer Studio の「**ファイル**」メニューを開いて「**ワークスペースを開く (Open Workspace)**」を選択し、「**MyProject**」ディレクトリーを選択します。 2. 「**ファイルの種類**」で「**Makefiles (*.mak)**」を指定します。 3. **sampmri.mak** を選択し、「**オープン**」をクリックします。 4. 「**ビルド (Build)**」メニューをオープンし、「**すべて再ビルド (Rebuild All)**」を選択して DLL をコンパイルし、リンクします。

ActiveX サーバー DLL を登録する	MyProject ディレクトリーの SAMPDBG.REG ファイルに、サンプル・プラグインがあるワークステーション上の場所を System i ナビゲーターに認識させるレジストリー・キーが入っています。 ActiveX サーバー DLL を登録するには、Windows のエクスプローラで SAMPDBG.REG をダブルクリックします。レジストリー・ファイルが実行され、ファイル内の項目がワークステーション上の Windows レジストリーに書き込まれます。 C:¥MyProject 以外のディレクトリーを指定した場合は、レジストリー・ファイルを実行する前に、以下のステップを実行してください。 1. Developer Studio (または任意のテキスト・エディター) で SAMPDBG.REG ファイルを開きます。 2. C:¥¥MyProject¥¥ となっている箇所を、すべて *x*:¥¥<*dir*>¥¥ に置き換えます (*x* はそのディレクトリーがあるドライブ名、<*dir*> はそのディレクトリーの名前です)。 3. SAMPDBG.REG ファイルを保管します。
デバッガーで System i ナビゲーターを実行する	System i ナビゲーターを実行し、サンプル・プラグインの動作を確認するには、以下のステップを実行してください。 1. Developer Studio の「**ビルド (Build)**」メニューで、「**デバッグ (Debug)**」 → 「**実行 (Go)**」と選択します。 2. プロンプトが出されたら、ワークステーション上の System i Access for Windows のインストール・ディレクトリーにある、System i ナビゲーターの実行可能ファイルの絶対パスを入力します。このパスは、C:¥PROGRAM FILES¥IBM¥CLIENT ACCESS¥CWBUNNAV.EXE、あるいはこれに類似したものになります。 3. 「**OK**」をクリックします。 System i ナビゲーターのメイン・ウィンドウが開きます。 SAMPDBG.REG ファイルを実行して新規プラグインを登録したことで、System i ナビゲーターのダイアログに、新規プラグインを走査するためのプロンプトが出されます。進行状況表示が終了するとダイアログが表示されるので「**OK**」をクリックします。 System i ナビゲーター・ウィンドウが最新表示された後、最初に選択したシステムの下の階層に、新規のフォルダー (サード・パーティーのサンプル・フォルダー) が表示されます。これで、System i ナビゲーターでプラグインを操作することができ、その動作をデバッガーで確認することができます。

関連情報

IBM クライアント・アクセス Express ツールキット - System i ナビゲーター・プラグインの Web ページ (英語)

サンプル Visual Basic プラグインのセットアップ

サンプル Visual Basic プラグインでは、i5/OS ライブラリーのリストを提供する System i ナビゲーター階層にフォルダーを追加し、これらのライブラリー・オブジェクトでプロパティーおよびアクションをインプリメントする方法を示します。

インストールされるプラグイン・コードの他にも、サンプル・プラグインには、Readme.txt ファイルおよび 2 つのレジストリー・ファイルが含まれています。レジストリー・ファイルの 1 つは開発時に使用するためのもので、もう 1 つは市販バージョンで配布するためのものです。 Visual Basic プラグインに組み込まれているすべてのファイルの詳細な説明については、サンプル Visual Basic プラグイン・ファイルのディレクトリーを参照してください。

ワークステーション上にサンプル Visual Basic プラグインをセットアップするには、以下のステップに従います。

Visual Basic プラグインをダウンロードする	実行可能ファイル vbopnav.exe をダウンロードしてください。このファイルを実行すると、プラグインに関連付けられているファイルがすべて解凍されます。新しいディレクトリー C:¥VBSample を作成し、解凍したファイルをすべてそこにコピーしてください。別のディレクトリーを作成した場合は、レジストリー・ファイルを修正して、プラグインの正しい場所を指定する必要があります。
Visual Basic プロジェクトを作成する	Visual Basic で vbsample.vpb をオープンします。「参照」ダイアログで、「**IBM System i Access for Windows ActiveX オブジェクト・ライブラリー (IBM System i Access for Windows ActiveX Object Library)**」および「**System i ナビゲーター Visual Basic プラグイン・サポート (System i ナビゲーター Visual Basic Plug-in Support)**」を選択します。 **注:** 「参照」ダイアログにこれらの参照項目が表示されない場合は、「**参照**」を選択して、System i Access for Windows の共有ディレクトリーで cwbx.dll および cwbunvbi.dll を探してください。 IBM System i Access ActiveX オブジェクト・ライブラリーには、リモート・コマンド呼び出しをシステムに対して実行する際にサンプル・アプリケーションで必要となる、OLE 自動化オブジェクトが含まれています。 System i ナビゲーター Visual Basic プラグイン・サポートには、Visual Basic プラグインのディレクトリーを作成するために必要なクラスおよびインターフェースが含まれています。
ActiveX サーバー DLL をビルドする	Visual Basic の「ファイル」メニューの「**dll の作成 (Make)**」を選択して DLL をビルドします。 コンパイルまたはリンクされない場合は、エラーを見つけて修正し、DLL をもう一度ビルドしてください。
資源ライブラリーをビルドする	1. Microsoft® Developer Studio を開いて「ファイル」メニューを選択し、「ワークスペースを開く (Open Workspace)」を選択してから、「**VBSample¥win32**」ディレクトリーを選択します。 2. 「**ファイルの種類**」フィールドに、「**Makefiles**」(*.mak) を指定します。 3. vbsmpmri.mak を選択して「**オープン**」をクリックします。 4. 「ビルド (Build)」メニューをオープンし、「すべて再ビルド (Rebuild All)」を選択して DLL をコンパイルし、リンクします。 **注:** この DLL は、独自に作成する必要はありません。サンプルには、プラグイン用の翻訳可能なテキスト・ストリングとその他のロケール依存資源を含む資源 DLL が含まれています。プラグインがサポートする言語が 1 つだけであっても、プラグイン・コードは、テキスト・ストリングとロケール固有の資源をこの資源ライブラリーからロードしなければなりません。
プラグインを登録する	プラグインを登録するには、ファイル vbsmpdbg.reg をダブルクリックします。C:¥VBSample ディレクトリーを使用しなかった場合は、レジストリー・ファイルを編集して、C:¥¥VBSample¥¥ となっている箇所を、すべてプラグイン・コードの絶対パスに置き換えます。パス内の円記号 (¥) は、2 つ続けなければなりません。
System i ナビゲーターで、プラグインを実行します。	System i ナビゲーターで、走査するシステムを展開します。 System i ナビゲーターがレジストリーに対する変更を検出し、そのシステムが新規のプラグインをサポート可能であるかどうか、システム走査によって確認するよう促すプロンプトを出します。走査が完了すると、System i ナビゲーターのツリー階層に新規プラグインが表示されます。

関連情報

IBM クライアント・アクセス Express ツールキット - System i ナビゲーター・プラグインの Web ページ (英語)

サンプル Visual Basic プラグイン・ファイルのディレクトリー

以下に示す表は、サンプル Visual Basic プラグインと共に組み込まれているすべてのファイルについて説明したものです。

Visual Basic のプロジェクト・ファイル	説明
vbsample.vbp	Visual Basic 5.0 のプロジェクト・ファイル

Visual Basic のフォーム	説明
authorty.frm	「権限の設定 (Set authority)」フォーム
delete.frm	「削除の確認 (Confirm delete)」フォーム
propsht.frm	「プロパティー・シート (Property Sheet)」フォーム
sysstat.frm	「システム状況」フォーム
wizard.frm	「新しいライブラリーの作成ウィザード (Create new library wizard)」フォーム

Visual Basic のモジュール	説明
global.bas	グローバル宣言

Visual Basic のクラス・モジュール	説明
actnman.cls	SampleActions Manager クラス
dropman.cls	Sample Drop Target Manager クラス
library.cls	Library クラス
listman.cls	Sample List Manager クラス

Visual Basic のバイナリー・ファイル	説明
authorty.frx	「権限の設定 (Set authority)」フォーム・バイナリー
delete.frx	「削除の確認 (Confirm delete)」フォーム・バイナリー
propsht.frx	「プロパティー・シート (Property Sheet)」フォーム・バイナリー
sysstat.frx	「システム状況」フォーム・バイナリー
wizard.frx	「新しいライブラリーの作成ウィザード (Create new library wizard)」フォーム・バイナリー
vbsample.bin	Vbsample バイナリー

構成の設定値	説明
mrisetup.ini	プラグインの翻訳可能資源のインストール情報
setup.ini	プラグインの実行可能ファイルのインストール情報

レジストリー項目	説明
vbsmpdbg.reg	開発時に使用するレジストリー・ファイル
vbsmprls.reg	インストール時に使用されるレジストリー・ファイル

資源 DLL の構成のためのファイル	説明
vbsmpmri.mak	Make ファイル
vbsmpmri.rc	RC ファイル
vbsmpres.h	ヘッダー・ファイル

イメージ	説明
compass.bmp	System i ナビゲーターのアイコン
lib.ico	
vbsmpflr.ico	オープンおよびクローズ状態の Visual Basic のサンプル・プラグインのフォルダー
vbsmplib.ico	Visual Basic のサンプル・プラグインのライブラリーのアイコン

関連概念

15 ページの『MRI セットアップ・ファイル』
MRI セットアップ・ファイルは、System i ナビゲーター・プラグインに関連付けられたロケール依存の資源をクライアント・ワークステーション上にインストールする際に、「プラグインのインストール (Install Plug-ins)」ウィザードが必要とする情報を提供します。

8 ページの『Setup.ini ファイル』
プラグインの setup.ini ファイルは、インストール・ウィザードに、クライアント・ワークステーションで System i ナビゲーター・プラグインをインストールする場合に必要な情報を提供します。また、このファイルには、プラグインをアップグレードする時期あるいはサービス・リリースを適用する時期を、チェック・サービス・レベル・プログラムが判別する場合に使用する情報も入っています。

サンプル Java プラグインのセットアップ

サンプルの Java プラグインで、所定のシステムの QUSRSYS ライブラリーにあるメッセージ待ち行列を操作します。

最初のプラグインを使用すると、デフォルトのメッセージ待ち行列 (使用している System i のユーザー ID と同じ名前を持つ) にあるメッセージを表示、追加、および削除することができます。2 番目のプラグインは、複数のメッセージ待ち行列をサポートします。 3 番目のプラグインは、メッセージの待ち行列間のドラッグ機能を追加するものです。

インストールされるプラグイン・コードの他にも、サンプル・プラグインには、Java ドキュメント、Readme.txt ファイル、および 2 つのレジストリー・ファイルが含まれています。レジストリー・ファイルの 1 つは開発時に使用するためのもので、もう 1 つは市販バージョンで配布するためのものです。Java プラグインに組み込まれているすべてのファイルの詳細な説明については、サンプル Java プラグイン・ファイルのディレクトリーを参照してください。

これらのサンプル Java プラグインをワークステーションにセットアップするには、以下のステップに従います。

サンプル Java プラグインをダウンロードする	実行可能ファイル jvopnav.exe をダウンロードしてください。このファイルを実行すると、前述のファイルがすべて解凍されます。これらのファイルは、実行可能ファイルによりデフォルトのディレクトリー jvopnav¥com¥ibm¥as400¥opnav にインストールされるようにします。

System i ナビゲーターにプラグインを識別させる	1. jvopnav¥com¥ibm¥as400¥opnav¥MsgQueueSampleX にあるファイル MsgQueueSampleX.reg を編集します。(X=1、2 または 3。これはインストールするサンプルによって異なります。) 2. ″NLS″=″C:¥¥jvopnav¥¥win32¥¥mri¥¥MessageQueuesMRI.dll″ および ″JavaPath″=″C:¥¥jvopnav″ の行を見つけます。 3. 「C:¥¥」を、ワークステーションの jvopnav ディレクトリーへの完全修飾パスで置き換えます。パス内の円記号 (¥) は、すべて 2 つ続けなければなりません。 4. 変更を保存します。 5. MsgQueueSampleX.reg レジストリー・ファイルをダブルクリックします。レジストリー・ファイルが実行され、ファイル内の項目がワークステーション上の Windows レジストリーに書き込まれます。
サンプル Java プラグインを実行する	1. System i ナビゲーターで、走査するシステムを展開します。 2. System i ナビゲーターがレジストリーに対する変更を検出し、そのシステムが新規のプラグインをサポート可能であるかどうか、システム走査によって確認するよう促すプロンプトを出します。 「**今すぐ走査 (Scan Now)**」をクリックします。 3. System i ナビゲーターがシステムを走査します。走査が終了すると、System i ナビゲーターの階層ツリーに、Java Message Queue Sample 1、2、または 3 という新規フォルダーが表示されます。 新規フォルダーをダブルクリックします。 4. 最初のサンプル・プラグインでは、システム上の QUSRSYS ライブラリーにあるデフォルトのメッセージ待ち行列の内容が表示されます。 2 番目と 3 番目のサンプルは、メッセージ待ち行列のリストを表示します。 新規のメッセージを追加するには、メッセージ待ち行列フォルダーを右クリックし、「**新規**」 → 「**メッセージ**」と選択します。プラグインによって表示されたダイアログに、メッセージ・テキストを入力します。 メッセージを削除するには、メッセージを右クリックして、「**削除**」を選択します。 5. 3 番目のサンプル・プラグインを使用している場合は、メッセージを選択し、それを別の待ち行列にドラッグすることができます。この操作により、プラグインは、メッセージを別の待ち行列に移動します。

関連情報

IBM クライアント・アクセス Express ツールキット - System i ナビゲーター・プラグインの Web ページ (英語)

サンプル Java プラグイン・ファイルのディレクトリー

以下に示す表は、サンプル Java プラグインと共に組み込まれているすべてのファイルについて説明したものです。

詳しくは、プラグインの javadoc 文書をお読みください。javadoc 文書は jvopnav¥com¥ibm¥as400¥opnav¥MsgQueueSample1¥docs ディレクトリーにインストールされています。Package-com.ibm.as400.opnav.MsgQueueSample1.html ファイルから読み始めてください。サンプルのパッケージ名は、com.ibm.as400.opnav.MsgQueueSample1 です。他のパッケージの類似したクラス名と区別するた

めに、すべてのクラス名に Mq という接頭部が付けられています。

Java ソース・コード・ファイル (最初のサンプル・プラグイン)	**説明**
MqActionsManager.java	プラグイン用のすべてのコンテキスト・メニューを扱う ActionsManager のインプリメンテーションです。
MqDeleteMessageBean.java	「削除確認」ダイアログの UI DataBean のインプリメンテーションです。
MqMessage.java	システムのメッセージを表すオブジェクトです。
MqMessageQueue.java	メッセージ待ち行列上の、システムのメッセージ・オブジェクトのコレクションです。
MqMessagesListManager.java	メッセージのリスト用の ListManager です。
MqNewMessageBean.java	「新しいメッセージ (New Message)」ダイアログの UI DataBean のインプリメンテーションです。

Java ソース・コード・ファイル (2 番目のサンプル・プラグイン)	**説明**
MqActionsManager.java	プラグイン用のすべてのコンテキスト・メニューを扱う ActionsManager のインプリメンテーションです。
MqDeleteMessageBean.java	「削除確認」ダイアログの UI DataBean のインプリメンテーションです。
MqListManager.java	プラグインのマスターの ListManager のインプリメンテーションです。
MqMessage.java	システムのメッセージを表すオブジェクトです。
MqMessageQueue.java	特定の待ち行列上の、システムのメッセージ・オブジェクトのコレクションです。
MqMessageQueueList.java	システムのメッセージ待ち行列のコレクションです。
MqMessageQueuesListManager.java	メッセージ待ち行列リスト用のスレーブの ListManager です。
MqMessagesListManager.java	メッセージ・リスト用のスレーブの ListManager です。
MqNewMessageBean.java	「新しいメッセージ (New Message)」ダイアログの UI DataBean のインプリメンテーションです。

Java ソース・コード・ファイル (3 番目のサンプル・プラグイン)	**説明**
MqActionsManager.java	プラグイン用のすべてのコンテキスト・メニューを扱う ActionsManager のインプリメンテーションです。
MqDeleteMessageBean.java	「削除確認」ダイアログの UI DataBean のインプリメンテーションです。
MqDropTargetManager.java	プラグインのドラッグ・アンド・ドロップを扱う DropTargetManager のインプリメンテーションです。
MqListManager.java	プラグインのマスターの ListManager のインプリメンテーションです。
MqMessage.java	システムのメッセージを表すオブジェクトです。
MqMessageQueue.java	特定の待ち行列上の、システムのメッセージ・オブジェクトのコレクションです。
MqMessageQueueList.java	システムのメッセージ待ち行列のコレクションです。
MqMessageQueuesListManager.java	メッセージ待ち行列リスト用のスレーブの ListManager です。
MqMessagesListManager.java	メッセージ・リスト用のスレーブの ListManager です。
MqNewMessageBean.java	「新しいメッセージ (New Message)」ダイアログの UI DataBean のインプリメンテーションです。

PDML ファイル	説明
MessageQueueGUI.pdml	プラグインのすべての Java UI パネル定義を含んでいます。
MessageQueueGUI.java	関連付けられている Java 資源バンドルです (サブクラス java.util.ListResourceBundle)。

オンライン・ヘルプ・ファイル	説明
IDD_MSGQ_ADD.html	「新しいメッセージ (New Message)」ダイアログのオンライン・ヘルプのスケルトンです。
IDD_MSGQ_CONFIRM_DELETE.html	「削除確認」ダイアログのオンライン・ヘルプのスケルトンです。

シリアライズされたファイル	説明
IDD_MSGQ_ADD.pdml.ser	「新しいメッセージ (New Message)」ダイアログのシリアライズされたパネル定義です。
IDD_MSGQ_CONFIRM_DELETE.pdml.ser	「削除確認」ダイアログのシリアライズされたパネル定義です。 **注:** MessageQueueGUI.pdml を変更した場合、これらのファイルの名前を変更してください。変更しない場合、変更がパネルに反映されません。

レジストリー項目	説明
MsgQueueSample1.reg MsgQueueSample2.reg MsgQueueSample3.reg	このプラグインが存在すること、およびこのプラグインによって Java インプリメンテーション・クラスを識別することを System i ナビゲーターに知らせる、Windows のレジストリー項目です。
MsgQueueSample1install.reg MsgQueueSample2install.reg MsgQueueSample3install.reg	プラグインの市販用バージョンで配布するためのレジストリー・ファイルです。Windows オペレーティング・システムでは、このバージョンのレジストリー・ファイルを直接読み取ることはできません。このレジストリー・ファイルには、System i Access for Windows のインストール・ディレクトリーのディレクトリー・パスを表す置換変数が含まれています。ユーザーが「プラグインのインストール (Install Plug-ins)」ウィザードを開始して、プラグインをシステムからインストールする場合、ウィザードはこのレジストリー・ファイルを読み取って正しいディレクトリー・パスを入力し、ユーザーのワークステーションのレジストリーに項目を書き込みます。したがって、このファイル内の項目は、開発時に使用されたレジストリー・ファイルと同期を保っている必要があります。

プラグインのプログラミング・リファレンス

System i ナビゲーターは、プログラム言語ごとに異なる方法でプラグインを処理します。

以下のトピックを利用して、各言語に固有のインターフェースに関する特定の参照情報だけでなく、プラグインのタイプごとに System i ナビゲーターの制御のフローについて学ぶことができます。

各言語に固有の参照情報の他に、各プラグインでは、Windows のレジストリー・ファイルをカスタマイズする必要があります。

C++ のリファレンス

C++ プラグインには、System i ナビゲーターにおける固有の制御フローがあります。さまざまな System i ナビゲーター API を使用して、C++ プラグインを開発することができます。各プラグインでは、Component Object Model (COM) インターフェースを 1 つ以上インプリメントすることができます。

System i ナビゲーターの構造および C++ プラグインの制御のフロー

System i ナビゲーター製品の内部アーキテクチャーは、System i プラットフォームの拡張可能な幅広い操作インターフェースの統合ポイントとして機能することを目的としています。

インターフェースの各機能コンポーネントは、ActiveX サーバー DLL としてパッケージされています。System i ナビゲーターは、Microsoft の Component Object Model (COM) テクノロジーを使用することにより、現時点でユーザー要求にサービスを提供する必要があるコンポーネント・インプリメンテーションのみを活動化します。こうすることによって、始動時に製品全体がロードされてしまい Windows リソースの大半を消費した結果、システム全体のパフォーマンスに影響が出る、という問題が回避されます。複数のシステムが、System i ナビゲーター階層の所定のオブジェクト・タイプに対してメニュー項目やダイアログを追加する要求を登録することがあります。

プラグインは、ユーザーのアクションに応答して生成される System i ナビゲーターからのメソッド呼び出しに応答することにより機能します。例えば、ユーザーが System i ナビゲーター階層中のオブジェクトを右クリックした場合、 System i ナビゲーターはそのオブジェクトのコンテキスト・メニューを構成して画面にメニューを表示します。 System i ナビゲーターは、選択されたオブジェクト・タイプに対しコンテキスト・メニュー項目を提供するよう登録されている各プラグインを呼び出して、メニュー項目を取得します。

プラグインによって論理的にインプリメントされた機能は、インターフェースとしてグループ化されます。インターフェースは、System i ナビゲーターが特定の機能を実行するために呼び出すことのできるクラスの、論理的に関連するメソッドのセットです。 Component Object Model は、一連の純粋な仮想関数を定義する抽象クラスの宣言を行うことにより、C++ でのインターフェースの定義をサポートします。インターフェースを呼び出すクラスは、インプリメンテーション・クラスと呼ばれます。インプリメンテーション・クラスは、抽象クラス定義をサブクラス化して、インターフェースに定義されたそれぞれの関数について C++ コードを提供します。

指定したインプリメンテーション・クラスには、開発者が任意の数のインターフェースをインプリメントすることができます。Developer Studio において ActiveX サーバー DLL 用の新規プロジェクト・ワークスペースを作成する場合、AppWizard がマクロを生成して、インターフェースのインプリメンテーションを容易にします。インターフェースを含むインプリメンテーション・クラスでは、インターフェースをネストされたクラスとして宣言します。ネストされたクラスはメンバー・データを持たず、インターフェースに定義された関数以外の関数は使用しません。通常、そのメソッドは、インプリメンテーション・クラスの関数を呼び出して状態データを取得および設定し、インターフェース仕様により定義された実際の操作を実行します。

C++ 用 System i ナビゲーター COM インターフェース

プラグインによって論理的にインプリメントされた関数は、 Component Object Model (COM) インターフェースとしてグループ化されます。

インターフェースは、System i ナビゲーターが特定の機能を実行するために呼び出すことのできるクラスの、論理的に関連するメソッドのセットです。プラグインは、開発者が提供しようとする機能のタイプに応じ、1 つ以上の COM インターフェースをインプリメントすることができます。例えば、ツリー階層中のオブジェクトをユーザーが右クリックした場合、System i ナビゲーターがそのオブジェクトのコンテキス

ト・メニューを作成して、画面に表示します。 System i ナビゲーターは、選択されたオブジェクト・タイプに対してコンテキスト・メニュー項目を提供するよう登録されている各プラグインを呼び出すことで、メニュー項目を取得します。プラグインは、ナビゲーターが **IContextMenu インターフェース**の **QueryContextMenu** メソッドのインプリメンテーションを呼び出すと、 System i ナビゲーターにそのメニュー項目を渡します。

インターフェース	メソッド	説明
IContextMenu	QueryContextMenu	ユーザーがオブジェクトを右クリックした際に、コンテキスト・メニュー項目を提供します。
	GetCommandString	コンテキスト・メニューのヘルプ・テキストを提供し、オブジェクトの状態に応じて、項目が使用可能であるか、使用可能でないかを指示します。
	InvokeCommand	適切なダイアログを表示して、要求されたアクションを実行します。ユーザーが、指定されたメニュー項目をクリックした際に、呼び出されます。
IPropSheetExt	AddPages	標準の Windows API を使用して追加されるプロパティー・ページを作成します。その上で、パラメーターとして渡された関数を呼び出して、ページを追加します。
IDropTarget	DragEnter	ユーザーがドロップ域にオブジェクトをドラッグすると、活動状態になります。
	DragLeave	ユーザーがドロップ域の外にオブジェクトをドラッグすると、活動状態になります。
	DragOver	ユーザーがドロップ域上にある間、活動状態になります。
	Drop	ユーザーがオブジェクトをドロップすると、活動状態になります。
IPersistFile	Load	呼び出されると、選択されたフォルダーの完全修飾オブジェクト名を使って拡張を初期化します。
IA4SortingHierarchyFolder	IsSortingEnabled	フォルダーでソートを使用可能にするかどうかを指定します。
	SortOnColumn	指定されたリスト・ビュー列でリストをソートします。
IA4FilteringHierarchyFolder	GetFilterDescription	現在の組み込み基準のテキスト記述を戻します。
IA4PublicObjectHierarchyFolder	GetPublicListObject	別のプラグインもリスト・オブジェクトを使用できるようにしたい場合にプラグインによってインプリメントされます。

インターフェース	メソッド	説明
IA4ListObject	GetAttributes	サポートされている属性 ID と、各属性 ID に関連付けられているデータのタイプのリストを戻します。
	GetValue	属性 ID が指定されると、属性の現行値を戻します。
IA4TasksManager	QueryTasks	このオブジェクトでサポートされているタスクのリストを戻します。
	TaskSelected	特定のタスクがユーザーによって選択されたことを IA4TasksManager インプリメンテーションに通知します。

IA4 インターフェース

Microsoft の COM インターフェースの他に、IBM 提供の IA4HierarchyFolder および IA4PropSheetNotify インターフェースがあります。

IA4PropSheetNotify インターフェースは、メイン・ダイアログが閉じるとサード・パーティーのプロパティー・ページを通知します。また、プラグインに情報を伝えるメソッドも定義します。このメソッドで、例えば、プロパティーが表示されているユーザーが既存または定義中であることや、変更を保管するか破棄するかなどを伝えます。

IA4HierarchyFolder インターフェースを使用することにより、プラグインは System i ナビゲーター階層に新規フォルダーを追加することができます。このインターフェースの目的は、プラグインが System i ナビゲーター階層に追加した新規のフォルダーの内容を取り込むときに使用されるリスト・データを提供することです。また、リスト・ビューの列と見出しを指定するためのメソッドや、フォルダーに関連したカスタム・ツールバーを定義するためのメソッドを定義します。

IA4HierarchyFolder インターフェースについて:

IA4HierarchyFolder インターフェースは、ISV (independent software vendor) がインプリメントする一連の関数を記述します。IA4HierarchyFolder は、IBM が定義した Component Object Model (COM) インターフェースで、System i ナビゲーター階層にサード・パーティーが新規のフォルダーおよびオブジェクトを追加できるようにするものです。

Microsoft COM については、Microsoft Web サイトを参照してください。

System i ナビゲーター・プログラムは、サード・パーティー・プラグインとの通信が必要になると、IA4HierarchyFolder インターフェースのメソッドを呼び出します。このインターフェースの主な目的は、プラグインにより定義されたフォルダーの内容を System i ナビゲーターが表示する際に使用されるリスト・データを System i ナビゲーターに提供することです。インターフェースのメソッドにより、 System i ナビゲーターは特定のサード・パーティー・フォルダーにバインドして、その内容をリストすることができます。詳細ビューの列数とその関連見出しを戻すためのメソッドもあります。また、カスタム・ツールバーをフォルダーに関連付けるための仕様を指定する、追加メソッドもあります。

通常、インターフェースのインプリメンテーションはコンパイルされて、ActiveX サーバー DLL (ダイナミック・リンク・ライブラリー) にリンクされます。System i ナビゲーターは、Windows レジストリーの項目によって、新規 DLL の存在を認識します。これらの項目は、ユーザーの PC 上の DLL の位置と、

新規フォルダーの挿入先となるオブジェクト階層内の接合ポイントを指定します。 System i ナビゲーターは適宜 DLL をロードして、必要に応じて IA4HierarchyFolder インターフェースのメソッドを呼び出します。

ヘッダー・ファイル CWBA4HYF.H には、インターフェース・プロトタイプおよび関連するデータ構造体と戻りコードが宣言されています。

関連情報

Microsoft Web サイト

IA4HierarchyFolder インターフェース仕様のリスト:

項目識別コード・データ・エンティティーは、Windows のネーム・スペースにあるすべてのフォルダーおよびオブジェクトを識別します。項目識別コードは、階層ファイル・システムにおけるファイル名のようなものです。 Windows のネーム・スペースは、実際には、Windows のエクスプローラの「デスクトップ」の下にある階層ネーム・スペースになります。

項目識別コードは、2 バイトのカウント・フィールドと、これに続く可変長のバイナリー・データ構造から構成されます (Microsoft ヘッダー・ファイル SHLOBJ.H の SHITEMID 構造を参照)。この項目識別コードは、オブジェクトを、その親フォルダーとの相対的な関係において一意的に記述するものです。

System i ナビゲーターは、以下の構造をとる項目識別コードを使用します。これは IA4HierarchyFolder::ItemAt によって戻さなければなりません。

```
<cb><item name>¥x01<item type>¥x02<item index>
```

ここで、

<cb> は、カウント・フィールド自体を含めた、項目識別コードのバイト単位のサイズです。

<item name> は、ユーザーに表示するために適切な形に変換されたオブジェクト名です。

<item type> は、オブジェクト・タイプを識別する言語依存の固有のストリングです。最小の長さは 4 文字です。

<item index> は、親フォルダー・オブジェクト内でのオブジェクトの位置を識別するゼロ・ベースの索引です。

IA4HierarchyFolder::Activate:

このように指定することで、IA4HierarchyFolder インスタンスが活動化された状態になります。また、クライアント上のフォルダー・オブジェクトのキャッシュを準備するためのシステムの呼び出しなど、列挙用のフォルダーを準備するために必要な処理もすべて実行します。

この関数は、長時間の操作によりユーザー・インターフェースのパフォーマンスが劣化することがないよう、データ・スレッドから呼び出されます。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE Activate();
```

パラメーター

なし。

戻りコード

正常な場合は NOERROR を、フォルダーの内容を取得できない場合は E_FAIL を戻します。

コメント

ユーザーが最初にフォルダーを選択または拡張したときに、 System i ナビゲーターによってこの関数が呼び出されます。ユーザーがフォルダー内容の最新表示操作を要求すると、クローズを呼び出した後に、再度この関数が呼び出されます。

フォルダー・インターフェースへのポインターを再設定する必要がある場合 (例: ユーザーが 2 回目にフォルダーを選択した場合) は常に、この関数を呼び出すことができます。別のフォルダーが選択された後の場合は、関連処理が実行済みであれば TRUE が戻されます。

極端に大きなリストの場合は、最初にワーカー・スレッドを作成してリストの作成が続行されるようにしてから、リストを完全に構成する前に Activate メソッドから戻るようにすることができます。この場合、GetListSize のインプリメンテーションが、リストが完全に構成されたかどうかの適切な指示を必ず戻すようにしてください。

IA4HierarchyFolder::BindToList:

このように指定すると、System i ナビゲーター階層の特定のフォルダーに対応する、IA4HierarchyFolder のインスタンスが返されます。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE BindToList(
    HWND hwnd,
    LPCITEMIDLIST pidl,
    REFIID riid,
    LPVOID* ppvOut
    );
```

パラメーター

hwnd リスト (ツリー・コントロールまたはリスト・コントロールのいずれでも可) を表示する、ビュー・ウィンドウのハンドル。このビューのオブジェクトのリストがクライアント上のキャッシュに入っているかどうかを判別するために、コンポーネントはこのハンドルを使用する必要があります。

pidl 列挙するフォルダーを一意的に識別する ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

riid 戻されるインターフェースの識別コード。このパラメーターは、IID_IA4HierarchyFolder インターフェースの識別コードを指します。

ppvOut
インターフェース・ポインターを受け取るアドレス。エラーが発生した場合は、このアドレスに NULL ポインターが戻されなければなりません。

戻りコード

正常な場合は NOERROR を、一般エラーが発生した場合は E_FAIL を戻します。

コメント

指定されたフォルダーに IA4HierarchyFolder のインスタンスが既に存在する場合、このメンバー関数は、別個のインスタンスをインスタンス化して初期化するのではなく、キャッシュにあるインスタンスを返します。ただし、キャッシュ内のオブジェクトに関連付けられたウィンドウ・ハンドルが、hwnd パラメーターに指定された値と同じでない場合は、新しいインスタンスが作成されます。

この関数は、提供されたパラメーターからインプリメンテーション・クラスのメンバー変数を初期化する必要があります。

IA4HierarchyFolder::DisplayErrorMessage:

Activate メソッドがエラーを返した場合、このような指定が呼び出されて、エンド・ユーザーに対してエラー・メッセージが表示されます。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE DisplayErrorMessage();
```

パラメーター

なし。

戻りコード

正常な場合は NOERROR を、表示するメッセージがない場合は E_FAIL を戻します。

コメント

なし。

IA4HierarchyFolder::GetAttributesOf:

このように指定すると、System i ナビゲーター階層にある特定のフォルダーの属性が返されます。属性標識は、Microsoft のインターフェース・メソッド IShellFolder::GetAttributesOf に定義されているものと同じです。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE GetAttributesOf(
    LPCITEMIDLIST pidl,
    ULONG* ulfInOut
    );
```

パラメーター

pidl 検索する属性を持つオブジェクトを一意的に識別する ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

ulfInOut

戻されるオブジェクト属性。入力時、このパラメーターは検索するオブジェクト属性を示すよう設定されます。

戻りコード

正常な場合は NOERROR を、オブジェクト属性を見付けられない場合は E_FAIL を戻します。

コメント

ビット・フラグを定義する定数については、Windows 組み込みファイル shlobj.h を参照してください。

この関数は、ツリーまたはリスト・ビューを配置する際に、System i ナビゲーターにより繰り返し呼び出されます。したがって、長時間の実行操作は避けるようにしてください。

IA4HierarchyFolder::GetColumnDataItem:

このように指定すると、System i ナビゲーターのリスト・ビューの列に表示される、フォルダーまたはオブジェクトのデータ・フィールドが返されます。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE GetColumnDataItem(
    LPCITEMIDLIST pidl,
    LPARAM lParam,
    char * lpszColumnData,
    UINT cchMax
    );
```

パラメーター

pidl 取得する列データを持つオブジェクトを一意的に識別する ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

lParam
コンポーネントによってデータが要求される列に、以前関連のあった値 (GetColumnInfo を参照)。

lpszColumnData
NULL 終了のデータ・ストリングを受け取るバッファーのアドレス。

cchMax
NULL 終了のデータ・ストリングを受け取るバッファーのサイズ。

戻りコード

正常な場合は NOERROR を、列データを検索できなかった場合は E_FAIL を戻します。

コメント

この関数は、リスト・ビューを配置する際に、System i ナビゲーターにより繰り返し呼び出されます。したがって、長時間の実行操作は避けるようにしてください。

IA4HierarchyFolder::GetColumnInfo:

このように指定すると、詳細ビューにある特定のフォルダーの内容表示に必要な列を記述する、データ構造体が返されます。これは、オプションのメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE GetColumnInfo(
    LPVOID* ppvInfo
    );
```

パラメーター

ppvInfo

戻されるデータ構造体。戻される構造体は、A4hyfColumnInfo 構造体のインスタンスから構成されている必要があります。この構造体は、リスト・ビューの列ごとに 1 つずつ A4hyfColumnItem 構造体の配列を含みます。

それぞれの列項目構造体は、列見出し用に変換されたストリング、デフォルトの列幅、および列にデータを与えるデータ・フィールドを一意的に識別する整数値を提供します。CWBA4HYF.H を参照してください。

戻りコード

正常な場合は NOERROR を、関数をインプリメントできない場合は E_NOTIMPL を戻します。

コメント

System i ナビゲーターは、Open 呼び出しが戻った後にこの関数を呼び出し、詳細ビューの列見出しを作成します。

この関数がインプリメントされていない場合、System i ナビゲーターは名前と説明の 2 つの列を挿入します。 GetColumnDataItem 関数によって、0 および 1 という整数値で指定される、これら 2 つのフィールドにデータを返す必要があります。

戻された構造体に記憶域を割り振るには、Windows IMalloc インターフェースを使用します。この記憶域の削除は、System i ナビゲーターにより行います。

IA4HierarchyFolder::GetIconIndexOf:

このように指定すると、階層フォルダーのアイコンをロードする際に使用可能なコンポーネント資源 DLL に、索引が返されます。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE GetIconIndexOf(
    LPCITEMIDLIST pidl,
    UINT uFlags,
    int* piIndex
    );
```

パラメーター

pidl 検索されるアイコン索引を持つオブジェクトを一意的に識別する ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

uFlags 検索するアイコン索引タイプの指定。このパラメーターはゼロか、GIL_OPENICON (提供されるアイコンが開いたフォルダーであることを指す) にすることができます。
GIL_OPENICON は、Windows 組み込みファイル SHLOBJ.H に定義されています。

piIndex
アイコン索引を受け取る整数へのポインター。

戻りコード

正常な場合は NOERROR を、索引を判別できない場合は E_FAIL を戻します。

コメント

この関数は、ツリーまたはリスト・ビューを配置する際に、System i ナビゲーターにより繰り返し呼び出されます。したがって、長時間の実行操作は避けるようにしてください。

IA4HierarchyFolder::GetItemCount:

このように指定すると、System i ナビゲーター階層の特定のフォルダーに含まれているオブジェクトの総数が返されます。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE GetItemCount(
    ULONG* pCount
    );
```

パラメーター

pCount
リスト内の項目の数を受け取る長整数へのポインター。

戻りコード

- リストが完全に作成されて項目の合計数が判別された場合、A4HYF_OK_LISTCOMPLETE が返されます。
- リストが構成中の場合は A4HYF_OK_LISTNOTCOMPLETE が戻されます。この場合、項目数は部分的に構成されたリストの項目数を示します。

- リストの構成中にエラーが発生した場合は、A4HYF_E_LISTDATAERROR が返されます。この場合の項目数は、すでにクライアントのキャッシュに入れられた項目のみを表します。

コメント

Activate メソッドからの戻り値が正常な場合、System i ナビゲーターはこの関数を呼び出して、取り込もうとしているフォルダーのオブジェクトの数を取得します。この関数の呼び出し後、System i ナビゲーターはフォルダー内にあるオブジェクトの項目識別コードを取得するため、ItemAt を繰り返し呼び出します。

極端に大きなリストの場合は、リスト全体がクライアントのキャッシュに入れられる前に、Activate 関数から戻るようにすることができます。その場合は、GetItemCount 関数から A4HYF_OK_LISTNOTCOMPLETE を返す必要があります。その時点から System i ナビゲーターは、A4HYF_OK_LISTCOMPLETE または A4HYF_E_LISTDATAERROR が返されるまで、GetItemCount 関数を 10 秒ごとに呼び出します。

IA4HierarchyFolder::GetToolBarInfo:

このように指定すると、System i ナビゲーター階層にある指定のフォルダーに関連付けられたカスタム・ツールバーを記述する、構造体が返されます。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE GetToolBarInfo(
    LPCITEMIDLIST pidl,
    LPVOID* ppvInfo
    );
```

パラメーター

pidl ツールバー情報の検索対象となるオブジェクトを一意的に識別する ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

ppvInfo

戻されるデータ構造体。このポインターにおいては A4hyfToolBarInfo のインスタンスが戻される必要があります。この構造体は、オブジェクトのツールバー・ボタンの数、各ボタンの属性を含む TBBUTTON 構造体の配列のアドレス、およびプラグインのインスタンス・ハンドルを提供します。ヘッダー・ファイル CWBA4HYF.H を参照してください。

戻りコード

正常な場合は NOERROR を、関数をインプリメントしないよう選択する場合は E_NOTIMPL を戻します。

コメント

この関数は、ユーザーが System i ナビゲーターのプラグインに属するフォルダーまたはオブジェクトを選択するたびに呼び出されます。

戻された構造体に記憶域を割り振るには、Windows IMalloc インターフェースを使用します。この記憶域の削除は、System i ナビゲーターにより行います。

このメンバー関数がインプリメントされていない場合は、デフォルトの System i ナビゲーター・ツールバーが使用されます。このツールバーには、「コピー」、「貼り付け」、「削除」、「プロパティー」、4 つのリスト・ビューのボタン、および「最新表示」が含まれます。System i ナビゲーターは、製品内の IContextMenu::GetCommandString (GCS_VALIDATE フラグを設定) のインプリメンテーションを呼び出し、オブジェクトに対して使用可能にするツールバー・ボタンを判別します。

IA4HierarchyFolder::GetListObject:

完全修飾オブジェクト名を指定すると、この関数は、キャッシュ内の (プラグインが作成した) プロキシー・オブジェクトへのポインターを返します。これは、オプションのメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE GetListObject(
    const char * lpszObjectName,
    LPVOID* ppvObj
    );
```

パラメーター

lpszObjectName
リスト・オブジェクトが戻される完全修飾オブジェクト名。

ppvObj
インプリメンテーション定義のオブジェクトを指す戻りポインターです。呼び出しルーチンは、適切なオブジェクト・タイプにこのポインターをキャストする必要があります。

戻りコード

正常な場合は NOERROR を、関数をインプリメントしないよう選択する場合は E_NOTIMPL を戻します。

コメント

Activate メソッドによってインスタンス化されたプロキシー・オブジェクトを取得するために、プラグイン・コードが cwbUN_GetListObjectFromName または cwbUN_GetListObjectFromPidl API を呼び出すたびに、この関数への呼び出しが行われます。プラグインはこのプロキシー・オブジェクトを使用して、システム上のデータにアクセスしたり、システム上でアクションを実行したりします。 IA4HierarchyFolder をインプリメンテーションした場合、プロキシー・オブジェクトのキャッシュは保持されるため、呼び出し側プログラムではオブジェクトを削除しないようにしてください。

IA4HierarchyFolder::ItemAt:

このように指定すると、フォルダー内容のリストの指定された位置にあるフォルダー・オブジェクトの SHITEMID (項目識別コード) 構造体が返されます。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE ItemAt(
    ULONG ulIndex,
    LPITEMIDLIST* ppidl
    );
```

パラメーター

ulIndex
項目識別コードが要求される項目のゼロ・ベースの索引。

ppidl 要求された項目識別コードを受け取るポインターのアドレス。

戻りコード

正常な場合は NOERROR を、項目が使用できない場合は E_FAIL を戻します。項目識別コードに使用する記憶域が不足している場合は、E_OUTOFMEMORY を戻します。

コメント

この関数は、リアルタイムでフォルダーを配置する際に、System i ナビゲーターにより繰り返し呼び出されます。したがって、長時間の実行操作は避けるようにしてください。 System i ナビゲーターの項目識別コードの形式については、CWBA4HYF.H を参照してください。項目識別コードに記憶域を割り振るには、Windows IMalloc インターフェースを使用します。

IA4HierarchyFolder::ProcessTerminating:

ユーザーが System i ナビゲーター・ウィンドウを閉じる際、この関数が呼び出されます。この関数を使用すると、プラグインが永続的なデータを保存できるようになります。これは、オプションのメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE ProcessTerminating();
```

戻りコード

正常な場合は NOERROR を、関数をインプリメントしないよう選択する場合は E_NOTIMPL を戻します。戻りエラーは無視されます。

コメント

なし

IA4HierarchyFolder::Refresh:

このように指定すると、キャッシュに入れられたフォルダー・オブジェクトはすべて破棄され、システムから取得した新規のデータを使用してキャッシュが再作成されます。これは、必須のメンバー関数です。

構文

```
HRESULT STDMETHODCALLTYPE Refresh();
```

戻りコード

正常な場合は NOERROR を、フォルダーのオブジェクトへのアクセス時にエラーが発生した場合は A4HYF_E_LISTDATAERROR を戻します。

コメント

メインの System i ナビゲーター・ウィンドウのグローバル最新表示が行われると、 System i ナビゲーターはこの関数を呼び出します。

IA4PropSheetNotify インターフェースについて:

IA4PropSheetNotify インターフェースは IA4HierarchyFolder インターフェースと同じく、ISV (independent software vendor) がインプリメントする一連の関数を記述します。 IA4PropSheetNotify は、IBM が定義した Component Object Model (COM) インターフェースで、System i ナビゲーターがユーザー用に定義したプロパティー・シートに、サード・パーティーが新規のプロパティー・ページを追加できるようにするためのものです。

System i ナビゲーター・プログラムは、サード・パーティー・プラグインとの通信が必要になると、IA4PropSheetNotify インターフェースのメソッドを呼び出します。このインターフェースの目的は、ユーザーのメイン・プロパティー・ダイアログが閉じる際に通知を行うことです。通知には、ユーザーが行った変更を保管するか廃棄するかが示されます。 IPropSheetExt で使用するものと同じインプリメンテーション・クラスにインターフェースを追加することが意図されています。

インターフェース・インプリメンテーションは、コンパイルされ、プラグインの ActiveX サーバー DLL にリンクされます。System i ナビゲーターは、Windows レジストリーの項目によって、新規 DLL の存在を認識します。これらの項目は、ユーザーの PC 上の DLL の位置を指定します。 System i ナビゲーターは適宜 DLL をロードして、必要に応じて IA4PropSheetNotify インターフェースのメソッドを呼び出します。

CWBA4HYF.H には、インターフェース・プロトタイプおよび関連するデータ構造体と戻りコードの宣言が含まれています。

IA4PropSheetNotify インターフェース仕様のリスト:

IA4PropSheetNotify インターフェースは、IShellPropSheetExt のインプリメンテーションへの通知を提供します。 Users および Groups のプロパティー・シートのいずれかにプロパティー・ページを追加する場合に、これらの通知が必要になります。

ユーザーがメインのプロパティー・ダイアログで「**OK**」をクリックする前に、Users および Groups プロパティー・シートの作成および破棄が頻繁に行われることがあるため、これらの通知が必要になります。IA4PropSheetNotify インターフェースは、ユーザーが行った変更を保管する必要がある場合に IShellPropSheetExt インプリメンテーションに通知をします。

System i ナビゲーターは、System i ナビゲーターのプラグインに定義されている通常のレジストリー項目を使用して、IA4PropSheetNotify インプリメンテーションに関する情報を取得します。また、Users および Groups の各コンポーネントのプロパティー・シート・ハンドラーが登録されると、ページが追加されるプロパティー・シートをプラグインが指定できるようにする、特殊なレジストリー値がサポートされます。

関連概念

95 ページの『プロパティー・シート・ハンドラーのプロパティー・ページ』
Microsoft Foundation Class (MFC) ライブラリーのクラスでは、プロパティー・シート・ハンドラーでのプロパティー・ページの作成をサポートしていません。ただし、MFC のクラス CPropertyPage の代わりに、IBM が提供している CExtPropertyPage を使用することができます。

IA4PropSheetNotify::ApplyChanges:

ユーザーに属するデータを保存する必要があることをインプリメンテーションに通知するために、この関数が呼び出されます。

構文

```
HRESULT STDMETHODCALLTYPE ApplyChanges(
    const char * pszNewUserName
    );
```

パラメーター

pszNewUserName
新規ユーザーを初めて作成する場合 (例えば、InformUserState が IUS_USEREXISTS 以外の値を指定する場合) のユーザー名。

戻りコード
正常な場合は NOERROR を、一般エラーが発生した場合は E_FAIL を戻します。

コメント
なし

IA4PropSheetNotify::GetErrorMessage:

ApplyChanges でエラーが返された際に、インプリメンテーションのエラー・メッセージ・テキストを取り出すために、この関数が呼び出されます。

構文

```
HRESULT STDMETHODCALLTYPE GetErrorMessage(
    char * pszErrMsg,
    UINT cchMax
    );
```

パラメーター

pszErrMsg
NULL 終了のエラー・メッセージを受け取るバッファーのアドレス。

cchMax
NULL 終了のエラー・メッセージを受け取るバッファーのサイズ。

戻りコード
正常な場合は NOERROR を、メッセージ・テキストを検索できない、またはメッセージ・テキストが大きすぎてバッファーに収まらない場合は E_FAIL を戻します。

コメント
なし

IA4PropSheetNotify::InformUserState:

IShellPropSheetExt インスタンスの作成直後に、この関数が呼び出されます。このユーザーがシステムにすでに存在しているか、または初めて作成されたのかについて、この関数によってインプリメンテーションに通知されます。

構文

```
HRESULT STDMETHODCALLTYPE InformUserState(
    UINT wUserState
    );
```

パラメーター

wUserState
ユーザーの現在の状態。システムは、以下の相互排他値を提供します。

- IUS_NEWUSER

  System i ナビゲーターのユーザーにより提供される属性に基づきユーザーを作成します。

- IUS_NEWUSERBASEDON

  既存のユーザーの属性に基づきユーザーを作成します。

- IUS_USEREXISTS

  ユーザーがシステム上に存在します。

戻りコード
正常な場合は NOERROR を、一般エラーが発生した場合は E_FAIL を戻します。

コメント
なし

System i ナビゲーター API

System i ナビゲーター API は、プラグインの開発者が特定のタイプのグローバル情報を取得および管理する場合に役立ちます。

System i ナビゲーター API のリスト:

この表は、System i ナビゲーター API を機能別にグループ化してリストしたものです。

機能	System i ナビゲーター API
システム値: この API を使用することにより、プラグイン開発者はシステム値の現行値を取得することができます。	39 ページの『cwbUN_GetSystemValue』
システム・ハンドル: これらの API を使用することにより、プラグイン開発者は、指定されたシステムに使用する Secure Sockets Layer (SSL) 設定などの接続プロパティーを含むシステム・オブジェクト・ハンドルの現行値を、取得および解放することができます。	40 ページの『cwbUN_GetSystemHandle』 41 ページの『cwbUN_ReleaseSystemHandle』
ユーザー入力の検証: これらの API を使用することにより、プラグイン開発者は、現行ユーザーが特定の System i オブジェクトに対する権限を持っているかどうかを検査することができます。また、この API により、開発者はユーザーが 1 つ以上の特殊権限を持っているかどうかも判別することができます。	42 ページの『cwbUN_CheckObjectAuthority』 42 ページの『cwbUN_CheckSpecialAuthority』
ユーザー権限の検査: この API を使用することにより、プラグイン開発者は、特定のタイプのユーザー提供ストリングをシステムに送信する前に、その妥当性を検査することができます。	43 ページの『cwbUN_CheckAS400Name』
ユーザー・プロファイル属性: この API を使用することにより、プラグイン開発者は、現行の System i ナビゲーター・ユーザーについて、任意のユーザー・プロファイル属性値を取得することができます。	44 ページの『cwbUN_GetUserAttribute』
データ管理: ユーザーが選択したオブジェクトは、サード・パーティー・プラグインに対して 2 つのデータ・エンティティー (項目識別コード・リストおよびオブジェクト名) によって識別されます。データ管理 API を使用することにより、プラグイン開発者はこれらの構造体から情報を取り出すことができます。	45 ページの『cwbUN_ConvertPidlToString』 45 ページの『cwbUN_GetDisplayNameFromItemId』 46 ページの『cwbUN_GetDisplayNameFromName』 47 ページの『cwbUN_GetDisplayPathFromName』 47 ページの『cwbUN_GetIndexFromItemId』 48 ページの『cwbUN_GetIndexFromName』 48 ページの『cwbUN_GetIndexFromPidl』 49 ページの『cwbUN_GetListObject』 49 ページの『cwbUN_GetParentFolderNameFromName』 50 ページの『cwbUN_GetParentFolderPathFromName』 50 ページの『cwbUN_GetParentFolderPidl』 51 ページの『cwbUN_GetSystemNameFromName』 51 ページの『cwbUN_GetSystemNameFromPidl』 52 ページの『cwbUN_GetTypeFromName』 53 ページの『cwbUN_GetTypeFromPidl』

機能	System i ナビゲーター API
System i ナビゲーター・ウィンドウ最新表示: これらの API は、ユーザーに代わって操作を完了した後、プラグインによる要求を実行し、ツリーまたはリスト・ビューを最新表示、または System i ナビゲーターのステータス・バーにメッセージを表示することができます。	53 ページの『cwbUN_RefreshAll』 54 ページの『cwbUN_RefreshList』 54 ページの『cwbUN_RefreshListItems』 54 ページの『cwbUN_UpdateStatusBar』
ODBC 接続: これらの API を使用することにより、プラグイン開発者は、System i ナビゲーターのデータベース・コンポーネントにより取得された ODBC 接続のハンドルを、再使用および終了することができます。	55 ページの『cwbUN_GetODBCConnection』 55 ページの『cwbUN_EndODBCConnections』
System i ナビゲーター・アイコンへのアクセス: これらの API を使用することにより、プラグイン開発者は、System i ナビゲーターのオブジェクト階層に表示されるオブジェクトのアイコン・イメージ・リストにアクセスすることができます。	56 ページの『cwbUN_GetIconIndex』 56 ページの『cwbUN_GetSharedImageList』
アプリケーション管理: これらの API を使用することにより、プラグイン開発者は、ユーザーが管理機能の使用を許可されているかどうかを方針に基づき判別することができます。*管理機能*とは、System i ナビゲーターのアプリケーション管理サブコンポーネントによって使用が制御される機能のことです。	59 ページの『cwbUN_GetAdminCacheState』 60 ページの『cwbUN_GetAdminCacheStateEx』 57 ページの『cwbUN_GetAdminValue』 58 ページの『cwbUN_GetAdminValueEx』
インストール: この API を使用することにより、プラグイン開発者は、System i ナビゲーターのサブコンポーネントがインストールされているかどうかを判別することができます。	61 ページの『cwbUN_IsSubcomponentInstalled』
ディレクトリー・サービス: これらの API は、System i プラットフォーム上の Lightweight Directory Access Protocol (LDAP) サーバーに関する情報を提供します。これらの API は、サーバーへの接続機能も備えています。この接続機能により、識別名やパスワードなど、System i Access for Windows がキャッシュに入れる情報を使用して、サーバーに接続することができます。この接続機能では、System i Access for Windows (LDAP.LIB および LDAP.DLL) に付属の LDAP クライアントを使用するため、アプリケーションでそのクライアントを使用する必要があります。 ストリングを使用する機能は、米国規格協会 (ANSI) および Unicode のバージョンで使用可能です。 LDAP クライアント API で使用するための識別名およびその他のストリングを戻す機能は、 LDAP バージョン 3 サーバー用に UTF-8 バージョンで提供されています。	66 ページの『cwbUN_FreeLdapPublishing』 65 ページの『cwbUN_OpenLdapPublishing』 66 ページの『cwbUN_GetLdapPublishCount』 69 ページの『cwbUN_GetLdapPublishParentDn』 68 ページの『cwbUN_GetLdapPublishPort』 67 ページの『cwbUN_GetLdapPublishServer』 66 ページの『cwbUN_GetLdapPublishType』 63 ページの『cwbUN_GetLdapSvrPort』 63 ページの『cwbUN_GetLdapSvrSuffixCount』 64 ページの『cwbUN_GetLdapSvrSuffixName』 62 ページの『cwbUN_FreeLocalLdapServer』 61 ページの『cwbUN_OpenLocalLdapServer』

cwbUN_GetSystemValue:

この API は、システム値を含んだストリングを返します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetSystemValue(
    USHORT usSystemValueId,
    const char * szSystemName,
    char * szSystemValue,
    UINT cchMax
    );
```

パラメーター

const char * szSystemValueId - input
検索するシステム値を識別する数値。システム値定数の定義については、ヘッダー・ファイル CWBA4SVL.H に記されています。

char * szSystemValue - output
NULL 終了のシステム値ストリングを受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了の値ストリングを受け取るバッファーのサイズ。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_INTERNAL_ERROR
システム値を検索できませんでした。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 この API により戻される値は、NLS (各国語サポート) ストリングではなく、変換もされません。例えば、'*NONE' の代わりに 'None' が戻されます。

cwbUN_GetSystemHandle:

この API は、システムで使用するセキュリティー、ユーザー ID、およびパスワードの設定を含んだシステム・ハンドルを返します。このシステム・ハンドルの設定は、System i ナビゲーターで入力システム名として構成されたものです。

アプリケーション名が NULL に設定されている場合、戻されるシステム・ハンドルは固有のものになります。アプリケーション名が設定されている場合、このアプリケーション名に一致するシステム・ハンドルが戻されます。

アプリケーションがシステムの固有な i5/OS ジョブを必要とする場合、アプリケーション名として NULL または固有名を渡す必要があります。

アプリケーションが i5/OS ジョブの共用を必要とする場合は、この関数のすべての呼び出し元で同じアプリケーション名を渡す必要があります。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetSystemHandle(
    char * szSystemName,
    char * szAppName,
    cwbCO_SysHandle * systemHandle
    );
```

パラメーター

char * szSystemName - input
システム・ハンドルを作成するシステムの名前を含む ASCIIZ ストリングへのポインター。

char * szAppName - input
12 文字以内の ASCIIZ ストリングへのポインター。これにより、単一のシステム・ハンドルを共用するアプリケーションを一意的に識別します。

cwbCO_SysHandle * systemHandle - output
このシステム名のシステム・ハンドルへのポインター。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_NULL_PARM
システム名が NULL でした。

CWBUN_INVALID_NAME_PARM
システム名が無効です。

CWB_NON_REPRESENTABLE_UNICODE_CHAR
1 つ以上の入力 UNICODE 文字が、使用中のコード・ページにおける表記を持ちません。

CWB_API_ERROR
システム・ハンドルを戻すことができませんでした。

使用法

System i Access for Windows API を使用した SSL をサポートするサード・パーティー・アプリケーションすべてにおいて、この関数を使用する必要があります。例えば、すべての System i Access for Windows の通信 API では、SSL をサポートするためのシステム・ハンドルを必要とします。

この関数の呼び出し元が、通信においてシステム・ハンドルを必要としなくなった場合、関数 **cwbUN_ReleaseSystemHandle** を呼び出してハンドルを解放することができます。

System i ナビゲーター・アプリケーション (cwbunnav.exe) が終了すると、すべてのハンドルが解放されます。

cwbUN_ReleaseSystemHandle:

この API は、システムで使用するセキュリティー設定を含んだシステム・ハンドルを解放します。システム・ハンドルは、cwbUN_GetSystemHandle 関数を使用して取得されます。この関数の呼び出し元がハンドルへの最後の参照を持つ場合、ハンドル資源が破棄されます。

構文

```
CWBAPI unsigned int WINAPI cwbUN_ReleaseSystemHandle(
    cwbCO_SysHandle * systemHandle
    );
```

パラメーター

cwbCO_SysHandle * systemHandle - input
cwbUN_GetSystemHandle 呼び出しで取得されたシステム・ハンドルへのポインター。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_API_ERROR
システム・ハンドルを解放できませんでした。

使用法 この関数の呼び出し元が、通信においてシステム・ハンドルを必要としなくなった場合、ハンドルを解放することができます。

cwbUN_CheckObjectAuthority:

この API は、システム上の特定のオブジェクトに対する権限がSystem i ナビゲーター・ユーザーにあるかどうかを返します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_CheckObjectAuthority(
    const char * szObjectPath,
    const char * szObjectType,
    const char * szAuthorityType,
      const char * szSystemName
    );
```

パラメーター

const char * szObjectPath - input
権限を検査する対象の System i オブジェクト・パス。

const char * szObjectType - input
権限検査の対象となるオブジェクトの System i オブジェクト・タイプ (例: *DTAQ)。

const char * szAuthorityType - input
検査する System i オブジェクト権限。

複数の権限を検査する場合、権限を連結する必要があります (例、*OBJMGT*OBJEXIST)。単一の呼び出しで、最大 11 の権限タイプを指定することができます。オブジェクトに対し、指定された権限をユーザーがすべて所有している場合にのみ、この関数は CWB_OK を戻します。

const char * szSystemName - input
検査を実行するシステムの名前。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
オブジェクトに対し、ユーザーは指定された権限を所有しています。

CWBUN_USER_NOT_AUTHORIZED
ユーザーは指定された権限を所有していません。

CWBUN_OBJECT_NOT_FOUND
指定されたオブジェクトを検査できませんでした。

CWBUN_INTERNAL_ERROR
オブジェクト権限を検査できませんでした。

使用法 *EXCLUDE が権限として指定されている場合は、その他の権限タイプを指定することはできません。*AUTLMGT は、 szObjectType が *AUTL の場合にのみ有効です。

cwbUN_CheckSpecialAuthority:

この API は、システムに対する特定の特殊権限がSystem i ナビゲーター・ユーザーにあるかどうかを返します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_CheckSpecialAuthority(
    const char * szSpecialAuthority,
    const char * szSystemName
    );
```

パラメーター

const char * szSpecialAuthority - input
検査する System i 特殊権限。

const char * szSystemName - input
検査を実行するシステムの名前。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
ユーザーは指定された特殊権限を所有しています。

CWBUN_USER_NOT_AUTHORIZED
ユーザーは指定された権限を所有していません。

CWBUN_INTERNAL_ERROR
特殊権限を検査できませんでした。

使用法 なし

cwbUN_CheckAS400Name:

この API は、指定のストリングがシステム上で有効な名前パラメーターであるかどうかを返します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_CheckAS400Name(
    const char * szAS400Name,
      const char * szSystemName,
    USHORT usTypeId
    );
```

パラメーター

const char * szAS400Name - input
妥当性を検査するシステム名。

const char * szSystemName - input
検査を実行するシステムの名前。

USHORT usTypeId - input
入力ストリングの解釈方法を指示する数値。解釈の方法には、長いオブジェクト名、短いオブジェクト名、通信名、またはストリングがあります (通常、型や定数定義のインクルード・ファイルへのリンクが記述される)。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_NAME_TOO_LONG
名前が長すぎます。

CWBUN_NAME_NULLSTRING
ストリングが空です。文字がありません。

CWBUN_NAME_INVALIDCHAR
無効な文字です。

CWBUN_NAME_STRINGTOOLONG
ストリングが長すぎます。

CWBUN_NAME_MISSINGENDQUOTE
終了の引用符がありません。

CWBUN_NAME_INVALIDQUOTECHAR
引用符ストリングには無効な文字です。

CWBUN_NAME_ONLYBLANKS
ブランクのみのストリングが検出されました。

CWBUN_NAME_STRINGTOOSHORT
ストリングが短すぎます。

CWBUN_NAME_TOOLONGFORIBM
ストリングには問題がありませんが、IBM コマンドとして長すぎます。

CWBUN_NAME_INVALIDFIRSTCHAR
最初の文字が無効です。

使用法 なし

cwbUN_GetUserAttribute:

この API は、現行の System i ナビゲーター・ユーザーのユーザー・プロファイル属性の値を含むストリングを戻します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetUserAttribute(
    USHORT usAttributeId,
    const char * szSystemName,
    char * szValue,
    UINT cchMax
    );
```

パラメーター

USHORT usAttributeId - input
検索するユーザー属性値を識別する数値。ユーザー属性定数の定義については、ヘッダー・ファイル CWBA4USR.H に記されています。

const char * szSystemName - input
ユーザー属性を検索するシステムの名前。

char * szValue - output
NULL 終了の属性値ストリングを受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了の値ストリングを受け取るバッファーのサイズ。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_INTERNAL_ERROR
属性値を検索できませんでした。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 この API により戻される値は、NLS ストリングではなく、変換もされません。例えば、'*NONE' の代わりに 'None' が戻されます。

cwbUN_ConvertPidlToString:

この API は、System i ナビゲーターの項目識別コード・リストを、完全修飾オブジェクト名に変換します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_ConvertPidlToString(
    LPCITEMIDLIST pidl,
    char * szObjectName,
    UINT cchMax
    );
```

パラメーター

LPCITEMIDLIST pidl - input
変換する ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

char * szObjectName - output
NULL 終了のオブジェクト名を受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了のオブジェクト名を受け取るバッファーのサイズ。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定された項目識別コード・リストが無効です。

WB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_GetDisplayNameFromItemId:

この API は、Unity 項目識別コードから項目名フィールドを取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetDisplayNameFromItemId(
    const char * szItemId,
    char * szItemName,
    UINT cchMax
    );
```

パラメーター

const char * szItemId - input
項目名を取り出す Unity 項目識別コード。

char * szItemName - output
NULL 終了の項目名を受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了の項目名を受け取るバッファーのサイズ。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定された項目識別コードが無効です。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_GetDisplayNameFromName:

この API は、完全修飾 Unity オブジェクト名から項目名フィールドを取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetDisplayNameFromName(
    const char * szObjectName,
    char * szItemName,
    UINT cchMax
    );
```

パラメーター

const char * szObjectName - input
項目名を取り出す Unity オブジェクト名。

char * szItemName - output
NULL 終了の項目名を受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了の項目名を受け取るバッファーのサイズ。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定されたオブジェクト名が無効です。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_GetDisplayPathFromName:

この API は、完全修飾 Unity オブジェクト名を、ユーザーへの表示に適している完全修飾パス名に変換します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetDisplayPathFromName(
    const char * szObjectName,
    char * szPathName,
    UINT cchMax
    );
```

パラメーター

const char * szObjectName - input
パス名の派生元の Unity オブジェクト名。

char * szPathName - output
NULL 終了のパス名を受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了のパス名を受け取るバッファーのサイズ。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定されたオブジェクト名が無効です。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_GetIndexFromItemId:

この API は、Unity 項目識別コードから項目索引フィールドを取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetIndexFromItemId(
    const char * szItemId,
    ULONG* piIndex
    );
```

パラメーター

const char * szItemId - input
項目索引を取り出す Unity 項目識別コード。

ULONG* piIndex - output
項目索引を受け取る符号なし長整数のアドレス。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定された項目識別コードが無効です。

使用法 なし

cwbUN_GetIndexFromName:

この API は、完全修飾 Unity オブジェクト名から項目索引フィールドを取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetIndexFromName(
    const char * szObjectName,
    ULONG* piIndex
    );
```

パラメーター

const char * szObjectName - input
項目索引を取り出す Unity オブジェクト名。

ULONG* piIndex - output
項目索引を受け取る符号なし長整数のアドレス。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定されたオブジェクト名が無効です。

使用法 なし

cwbUN_GetIndexFromPidl:

この API は、完全修飾の Unity 項目識別コード・リストから項目索引フィールドを取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetIndexFromPidl(
    LPCITEMIDLIST pidl,
    ULONG* piIndex
    );
```

パラメーター

LPCITEMIDLIST pidl - input
項目索引が取り出される ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

ULONG* piIndex - output
項目索引を受け取る符号なし長整数のアドレス。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定された項目識別コード・リストが無効です。

使用法 なし

cwbUN_GetListObject:

この API は、指定されたリスト・オブジェクト名に関連したオブジェクトへのポインターを取得します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetListObject(
    const char * szFileName,
    LPVOID *pListObject
    );
```

パラメーター

const char * szFileName - input
オブジェクト・ポインターを検出して戻す対象の Unity オブジェクト名。

LPVOID pListObject - output
要求 Unity オブジェクトへのポインターのアドレス。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

使用法 なし

cwbUN_GetParentFolderNameFromName:

この API は、完全修飾 Unity オブジェクト名からオブジェクトの親フォルダーの名前を取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetParentFolderNameFromName(
    const char * szObjectName,
    char * szParentFolderName,
    UINT cchMax
    );
```

パラメーター

const char * szObjectName - input
親フォルダー名を取り出す Unity オブジェクト名。

char * szParentFolderPath - output
NULL 終了の親フォルダー名を受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了の親フォルダー名を受け取るバッファーのサイズ。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定されたオブジェクト名が無効です。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_GetParentFolderPathFromName:

完全修飾 Unity オブジェクト名を指定することにより、この API は、オブジェクトの親フォルダーの完全修飾オブジェクト名を戻します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetParentFolderPathFromName(
    const char * szObjectName,
    char * szParentFolderPath,
    UINT cchMax
    );
```

パラメーター

const char * szObjectName - input
親フォルダー・オブジェクト名を取り出す Unity オブジェクト名。

char * szParentFolderPath - output
NULL 終了の親フォルダー・オブジェクト名を受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了の親フォルダー・オブジェクト名を受け取るバッファーのサイズ。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定されたオブジェクト名が無効です。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_GetParentFolderPidl:

完全修飾 Unity 項目識別コード・リストを指定することにより、この API は、オブジェクトの親フォルダーの完全修飾項目識別コード・リストを戻します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetParentFolderPidl(
    LPCITEMIDLIST pidl,
    LPITEMIDLIST *ppidl
    );
```

パラメーター

LPCITEMIDLIST pidl - input
親フォルダー項目識別コード・リストを取り出す ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

LPITEMIDLIST* ppidl - output
親フォルダー項目識別コード・リストを受け取る項目識別コード・リスト・ポインターのアドレス。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定された項目識別コード・リストが無効です。

使用法 なし

cwbUN_GetSystemNameFromName:

この API は、完全修飾 Unity オブジェクト名からシステム名を取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetSystemNameFromName(
    const char * szObjectName,
    char * szSystemName,
    UINT cchMax
    );
```

パラメーター

const char * szObjectName - input
システム名を取り出す Unity オブジェクト名。

char * szSystemName - output
NULL 終了のシステム名を受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了のシステム名を受け取るバッファーのサイズ。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定されたオブジェクト名が無効です。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_GetSystemNameFromPidl:

この API は、完全修飾の Unity 項目識別コード・リストからシステム名を取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetSystemNameFromPidl(
    LPCITEMIDLIST pidl,
    char * szSystemName,
    UINT cchMax
    );
```

パラメーター

LPCITEMIDLIST pidl - input
システム名が取り出される ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

char * szSystemName - output
NULL 終了のシステム名を受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了のシステム名を受け取るバッファーのサイズ。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定された項目識別コード・リストが無効です。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_GetTypeFromName:

この API は、完全修飾 Unity オブジェクト名から項目タイプ・フィールドを取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetTypeFromName(
    const char * szObjectName,
    char * szType,
    UINT cchMax
    );
```

パラメーター

const char * szObjectName - input
項目索引を取り出す Unity オブジェクト名。

char * szType - output
NULL 終了の項目タイプを受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了の項目タイプを受け取るバッファーのサイズ。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定されたオブジェクト名が無効です。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_GetTypeFromPidl:

この API は、完全修飾の Unity 項目識別コード・リストから項目索引フィールドを取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetTypeFromPidl(
    LPCITEMIDLIST pidl,
    char * szType,
    UINT cchMax
    );
```

パラメーター

LPCITEMIDLIST pidl - input
項目索引が取り出される ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

char * szType - output
NULL 終了の項目タイプを受け取るバッファーのアドレス。

UINT cchMax - input
NULL 終了の項目タイプを受け取るバッファーのサイズ。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_FORMAT_NOT_VALID
指定された項目識別コード・リストが無効です。

CWB_BUFFER_OVERFLOW
バッファーが小さすぎるため、戻りストリングを含むことができません。

使用法 なし

cwbUN_RefreshAll:

この API は、System i ナビゲーターのツリー・ウィンドウおよびリスト・ウィンドウの内容を最新表示します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_RefreshAll(
    const char * pszStatusText
    );
```

パラメーター

const char * pszStatusText - input
完了時に、ステータス・バー・ウィンドウに配置される NULL 終了ストリング。このパラメーターは、NULL にすることができます。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_WINDOW_NOTAVAIL
ビュー・ウィンドウを検出できませんでした。

使用法 ユーザーから要求されたアクションをシステムが実行した後に、System i ナビゲーターの内容全体を最新表示する場合に、この関数を使用します。

cwbUN_RefreshList:

この API は、System i ナビゲーターのリスト・ビュー・ウィンドウの内容を最新表示します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_RefreshList(
    const char * pszStatusText
    );
```

パラメーター

const char * pszStatusText - input
完了時に、ステータス・バー・ウィンドウに配置される NULL 終了ストリング。このパラメーターは、NULL にすることができます。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_WINDOW_NOTAVAIL
リスト・ビュー・ウィンドウを検出できませんでした。

使用法 この関数は、ユーザーにより要求されたアクションの実行後に、リスト・ウィンドウの内容を最新表示するために使用します。

cwbUN_RefreshListItems:

この API は、System i ナビゲーターのリスト・ビュー・ウィンドウで現在選択されている項目 (複数可) を最新表示します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_RefreshListItems(
    const char * pszStatusText
    );
```

パラメーター

const char * pszStatusText - input
完了時に、ステータス・バー・ウィンドウに配置される NULL 終了ストリング。このパラメーターは、NULL にすることができます。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_WINDOW_NOTAVAIL
リスト・ビュー・ウィンドウを検出できませんでした。

使用法 この関数は、ユーザーにより要求されたアクションの実行後に、リスト・ウィンドウ内の選択項目を最新表示するために使用します。

cwbUN_UpdateStatusBar:

この API は、System i ナビゲーター・ウィンドウのステータス・バーにテキスト・ストリングを挿入します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_UpdateStatusBar(
    const char * pszStatusText
    );
```

パラメーター

const char * pszStatusText - input
完了時に、ステータス・バー・ウィンドウに配置される NULL 終了ストリング。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_WINDOW_NOTAVAIL
ステータス・バー・ウィンドウを検出できませんでした。

使用法 この関数は、ダイアログの「OK」ボタンをクリックして要求されたアクションが正常に完了したことをユーザーに通知するために使用します。

cwbUN_GetODBCConnection:

この API は、指定のシステム上にある Open Database Connectivity (ODBC) 接続に対するハンドルを返します。指定されたシステムに対する接続が存在しない場合、API は新しいハンドルを取得します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetODBCConnection(
    const char * szSystemName,
    HDBC *phDBC
    );
```

パラメーター

const char * szSystemName - input
ODBC 接続を検索するシステムの名前。

HDBC *phDBC - output
ODBC 接続ハンドルを戻すアドレス。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

使用法 なし

cwbUN_EndODBCConnections:

この API は、以前に cwbUN_GetODBCConnection API が開いたすべての Open Database Connectivity (ODBC) 接続を終了します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_EndODBCConnections(
    );
```

パラメーター

なし

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
ハンドルが作成されませんでした。

使用法

EndODBCConnections 関数は、**GetODBCConnection** 関数を使用して開いた接続のみを閉じることに注意してください。 **EndODBCConnections** 関数は、直接またはその他のインターフェースを使用して開いた ODBC 接続は認識しません。

また、拡張機能中のコードで **GetODBCConnection** を使用する場合、アプリケーション拡張機能のフォルダーのデストラクターが必ず **EndODBCConnections** を呼び出すようにしてください。

cwbUN_GetODBCConnection も参照してください。

cwbUN_GetIconIndex:

この API は、指定されたアイコンのイメージ・リストの索引を取得します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetIconIndex(
    LPCITEMIDLIST pidl,
    UINT uFlags,
    int* piIndex
    );
```

パラメーター

LPCITEMIDLIST pidl - input
参照するアイコンの識別に使用される ITEMIDLIST (項目識別コード・リスト) 構造体へのポインター。

UINT uFlags - input
検索するアイコン索引タイプの指定 (上記に定義)。

int * piIndex - output
アイコン索引を受け取る整数のアドレス。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_INVALID_FLAG_VALUE
サポートされる有効なフラグ値ではありません。

使用法 なし

cwbUN_GetSharedImageList:

この API は、System i ナビゲーターに関連付けられているアイコン・イメージ・リストを取り出します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetSharedImageList(
    UINT uFlags,
    HIMAGELIST *phImageList
    );
```

パラメーター

UINT uFlags - input
取り出すイメージ・リストのタイプ (上記で定義) の指定。

HIMAGELIST* phImageList -
イメージ・リスト・ハンドルを受け取る変数のアドレス。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWBUN_INVALID_FLAG_VALUE
サポートされる有効なフラグ値ではありません。

CWBUN_CANT_GET_IMAGELIST
アイコン・イメージ・リストの取得を試みた際に障害が発生しました。

使用法 なし

cwbUN_GetAdminValue:

この API は、指定されたシステム上の現行の System i ナビゲーター・ユーザーが特定の管理機能の使用を許可されているかどうかを戻します。*管理機能*とは、System i ナビゲーターのアプリケーション管理サブコンポーネントによって使用が制御される機能のことです。

例えば、管理者はアプリケーション管理を使用して、ユーザーが System i ナビゲーターのさまざまな機能にアクセスできるかどうかを制御できます。これらの機能の 1 つにジョブ管理があります。cwbUN_GetAdminValue API を使用することにより、System i ナビゲーターの現行ユーザーが、ジョブ管理に対応する管理機能の名前を指定してジョブ管理機能を使用できるかどうかについて、プログラマチックに判別することができます。 System i ナビゲーターでサポートされている管理機能の名前のリストについては、 CWBUNPLA.H ヘッダー・ファイルを参照してください。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetAdminValue(
    const char * szSystemName,
    char*   adminFunction,
    cwbUN_Usage& usageValue);
```

パラメーター

const char * szSystemName
検査を実行するシステムの名前。

char* adminFunction
管理機能の名前を含む ASCII ストリングへのポインター。ストリングは NULL 終了である必要があり、最大長は 30 バイト + 1 バイト (NULL 終了文字) です。サポートされる入力値については、 cwbunpla.h を参照してください。

cwbUN_Usage & usageValue
この値は、CWB_OK の戻りコードが戻される場合にのみ有効です。以下の 2 つの値のいずれかが戻されます。

- cwbUN_granted -- ユーザーは機能の使用を許可されます。
- cwbUN_denied -- ユーザーは機能の使用を拒否されます。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
API は正常です。

CWBSY_USER_CANCELLED
ユーザーが、API により示されたユーザー ID とパスワードを取り消しました。

使用法

この API は、指定したシステムの System i ナビゲーターの現行ユーザーが、指定の機能を使用することを許可されているかどうかを判別します。指定されたシステムに、現在、ユーザーがサインオンしていない場合、 API によりユーザーがサインオンされ、ユーザー ID とパスワード・プロンプトが表示されます。

この API は、System i ナビゲーターまたはクライアント・アプリケーションの機能カテゴリーにある管理機能の検査にのみ使用することができます。

cwbUN_GetAdminValueEx:

この API は、指定されたシステム上の現行のユーザーが特定の管理機能の使用を許可されているかどうかを戻します。*管理機能*とは、System i ナビゲーターのアプリケーション管理サブコンポーネントによって使用が制御される機能のことです。

注: System i ナビゲーターのプラグインは、cwbUN_GetAdminValueEx ではなく cwbUN_GetAdminValue API を使用する必要があります。

管理者はアプリケーション管理を使用して、ユーザーが System i ナビゲーターのさまざまな機能にアクセスできるかどうかを制御できます。これらの機能の 1 つにジョブ管理があります。
cwbUN_GetAdminValueEx API を使用することにより、ジョブ管理に対応する管理機能の名前を現行ユーザーが指定して、ジョブ管理機能を使用できるかどうかを、プログラマチックに判別することができます。
System i ナビゲーターでサポートされている管理機能の名前のリストについては、 CWBUNPLA.H ヘッダー・ファイルを参照してください。

この API は cwbUN_GetAdminValue と同じ機能を提供しますが、システム名の代わりにシステム・オブジェクト・ハンドルを受け入れる点が異なります。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetAdminValueEx(
    cwbCO_SysHandle* pSysHandle,
    char*   adminFunction,
    cwbUN_Usage& usageValue);
```

パラメーター

cwbCO_SysHandle* pSysHandle
システム・オブジェクト・ハンドルへのポインター。この API を呼び出す前に、システ

ム・オブジェクト中にシステム名を指定する必要があります。 cwbUN_GetAdminValueEx API の動作は、システム・オブジェクトがシステムへのサインオンを取得しているかどうかにより異なります。

サインオンしていない場合 ->

cwbUN_GetAdminValueEx によって、システムにサインオンします。ユーザーの最新のアプリケーション管理設定が、まだクライアント・ワークステーションのキャッシュに入れられていない場合は、システムからダウンロードされます。

サインオンしている場合 ->

System i のユーザー ID およびパスワードの検証を行うように指定している (検証モード) システムにシステム・オブジェクトがサインオンした場合、
cwbUN_GetAdminValueEx API は、そのサインオンが実行された時点の正確なアプリケーション管理設定のスナップショットを使用します。ユーザー ID とパスワードの検証を行わずにサインオンした場合、cwbUN_GetAdminValueEx はアプリケーション管理設定のコピー (24 時間以内のもの) を使用します。

char* adminFunction

管理機能の名前を含む ASCII ストリングへのポインター。ストリングは NULL 終了である必要があり、最大長は 30 バイト + 1 バイト (NULL 終了文字) です。サポートされる入力値については、 CWBUNPLA.H を参照してください。

cwbUN_Usage& usageValue

この値は、CWB_OK の戻りコードが戻される場合にのみ有効です。以下の 2 つの値のいずれかが戻されます。

cwbUN_granted

ユーザーは機能の使用を許可されます。

cwbUN_denied

ユーザーは機能の使用を拒否されます。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK

API は正常です。

CWBSY_USER_CANCELLED

ユーザーが、API により示されたユーザー ID とパスワードを取り消しました。

使用法

この API は、入力システム・オブジェクトによって定義された現行のシステム・ユーザーが指定された機能の使用を許可されているかどうかを判別します。指定されたシステムに、現在、ユーザーがサインオンしていない場合、 API によりユーザーがサインオンされ、ユーザー ID とパスワード・プロンプトが表示されます。

この API は、System i ナビゲーターまたはクライアント・アプリケーションの機能カテゴリーにある管理機能の検査にのみ使用することができます。

cwbUN_GetAdminCacheState:

この API は、長時間かかるであろう cwbUN_GetAdminValue API の次回の呼び出しを行うかどうかを指示します。cwbUN_GetAdminValue API は、ワークステーションのキャッシュにデータを入れます。キャッシュが現行のものでない場合、cwbUN_GetAdminValue はキャッシュを更新するためサインオン・プロンプトを表示するか、その他の処理を実行します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetAdminCacheState(
    const char * szSystemName,
    cwbUN_State& adminState);
```

パラメーター

const char * szSystemName
検査を実行するシステムの名前。

cwbUN_State& adminState
cwbUN_GetAdminValue API の次回の呼び出しについて、長時間実行のものであるかどうか、またはホスト・システムにアクセスせずに戻るために、内部キャッシュを使用するかどうかを示すパラメーター。

以下の値のいずれかが戻されます。

cwbUN_logon
指定のシステムに現行ユーザーは存在しません。 cwbUN_GetAdminValue API はサインオン・プロンプトを示すことができます。

cwbUN_refresh
cwbUN_GetAdminValue はシステムにアクセスし、内部キャッシュを更新します。

cwbUN_cache
cwbUN_GetAdminValue は現行のキャッシュを持っており、長時間実行することはできません。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
API は正常です。

使用法 cwbUN_GetAdminValue のユーザーは、この API を使用して、cwbUN_GetAdminValue の次の呼び出しが長時間実行のものであるかどうかを判別できます。

cwbUN_GetAdminCacheStateEx:

この API は、長時間かかるであろう cwbUN_GetAdminValueEx API の次回の呼び出しを行うかどうかを指示します。cwbUN_GetAdminValueEx API は、ワークステーションのキャッシュにデータを入れます。キャッシュが現行のものでない場合、cwbUN_GetAdminValueEx API はキャッシュを更新するためサインオン・プロンプトを表示するか、その他の処理を実行します。

構文

```
CWBAPI unsigned int WINAPI cwbUN_GetAdminCacheStateEx(
    cwbCO_SysHandle* pSysHandle,
    cwbUN_State& adminState);
```

パラメーター

cwbCO_SysHandle* pSysHandle - input
システム・オブジェクト・ハンドルへのポインター。この API への呼び出しの前に、システム・オブジェクト中にシステム名を指定する必要があります。

cwbUN_State& adminState

cwbUN_GetAdminValue API の次回の呼び出しについて、長時間実行のものであるかどうか、またはホスト・システムにアクセスせずに戻るために、内部キャッシュを使用するかどうかを示すパラメーター。

以下の値のいずれかが戻されます。

cwbUN_logon

指定のシステムに現行ユーザーは存在しません。 cwbUN_GetAdminValue API はサインオン・プロンプトを示すことができます。

cwbUN_refresh

cwbUN_GetAdminValue はシステムにアクセスし、内部キャッシュを更新します。

cwbUN_cache

cwbUN_GetAdminValue は現行のキャッシュを持っており、長時間実行することはできません。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK

API は正常です。

使用法 cwbUN_GetAdminValueEx のユーザーは、この API を使用して、cwbUN_GetAdminValueEx の次の呼び出しが長時間実行のものであるかどうかを判別できます。

cwbUN_IsSubcomponentInstalled:

この API は、System i ナビゲーターのサブコンポーネントが PC にインストールされているかどうかを判別します。

構文

```
CWBAPI BOOL WINAPI cwbUN_IsSubcomponentInstalled(
    UNIT uOption);
```

パラメーター

UNIT uOption

このパラメーターは、検査する System i ナビゲーターのサブコンポーネントを指定します。サポートされる値のリストについては、cwbun.h にある API のまえがきを参照してください。

戻りコード

ブール値を戻します。

TRUE サブコンポーネントがインストールされています。

FALSE

サブコンポーネントがインストールされていません。

使用法 なし。

cwbUN_OpenLocalLdapServer:

この API は、システム上の Lightweight Directory Access Protocol (LDAP) サーバーに関する構成情報にアクセスする際に、使用可能なハンドルを作成します。

構文

```
int cwbUN_OpenLocalLdapServerW
    ( LPCWSTR              system,
      cwbUN_ldapSvrHandle  *pHandle
    );

int cwbUN_OpenLocalLdapServerA
    ( LPCSTR               system,
      cwbUN_ldapSvrHandle  *pHandle
    );
```

パラメーター

LPCSTR system - input
システム名へのポインター。

cwbUN_ldapSvrHandle *pHandle - output
戻される際に、以下の API で使用可能なハンドルが含まれます。

- cwbUN_FreeLocalLdapServer
- cwbUN_GetLdapSvrPort
- cwbUN_GetLdapSvrSuffixCount
- cwbUN_GetLdapSuffixName

注: このハンドルは、cwbUN_FreeLocalLdapServer への呼び出しにより開放する必要があります。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_PARAMETER
無効なパラメーターが指定されています。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

CWBUN_LDAP_NOT_AVAIL
ディレクトリー・サービスがインストールされていないか、サーバーが構成されていません。

使用法 なし

cwbUN_FreeLocalLdapServer:

この API は、入力ハンドルに関連付けられているリソースを解放します。

構文

```
int cwbUN_FreeLocalLdapServer
    ( cwbUN_ldapSvrHandle  handle
    );
```

パラメーター

cwbUN_ldapSvrHandle handle - input
資源を解放する対象となるハンドル。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
cwbUN_OpenLocalLdapServer() によりハンドルが作成されませんでした。

使用法 ハンドルは、cwbUN_OpenLocalLdapServer 呼び出しを実行して取得します。

cwbUN_GetLdapSvrPort:

この API は、Lightweight Directory Access Protocol (LDAP) サーバーが使用するポート番号を返します。

構文

```
int cwbUN_GetLdapSvrPort
    ( cwbUN_ldapSvrHandle  handle,
      int                  *port,
      int                  *sslPort
    );
```

パラメーター

cwbUN_ldapSvrHandle handle - input
cwbUN_OpenLocalLdapServer() への呼び出しにより以前に取得されたハンドル。

int * port - output
LDAP 接続に使用するポート番号。

int * sslPort - output
SSL 接続に使用するポート番号。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
無効なハンドル。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

使用法 なし

cwbUN_GetLdapSvrSuffixCount:

この API は、このサーバーに対して構成されたサフィックスの数を戻します。サフィックスは、ディレクトリー・ツリー内の開始点の識別名 (DN) です。

構文

```
int cwbUN_GetLdapSvrSuffixCount
    ( cwbUN_ldapSvrHandle  handle,
      int                  *count
    );
```

パラメーター

cwbUN_ldapSvrHandle handle - input
cwbUN_OpenLocalLdapServer() への呼び出しにより以前に取得されたハンドル。

int * count - output
サーバー上に存在するサフィックスの数。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
無効なハンドル。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

使用法 なし

cwbUN_GetLdapSvrSuffixName:

この API は、サフィックスの識別名を戻します。

構文

```
int cwbUN_GetLdapSuffixNameA
    ( cwbUN_ldapSvrHandle      handle,
      int                      index,
      LPSTR                    suffix,
      int                      *length
    );

int cwbUN_GetLdapSuffixNameW
    ( cwbUN_ldapSvrHandle      handle,
      int                      index,
      LPWSTR                   suffix,
      int                      *length
    );

int cwbUN_GetLdapSuffixName8 /* returns suffix in UTF-8 */
    ( cwbUN_ldapSvrHandle      handle,
      int                      index,
      LPSTR                    suffix,
      int                      *length
    );
```

パラメーター

cwbUN_ldapSuffixHandle handle - input
cwbUN_OpenLocalLdapServer() への呼び出しにより以前に取得されたハンドル。

int index - input
サフィックスのゼロ・ベースの索引。この値は、cwbUN_GetLdapSvrSuffixCount() が戻すカウントより小さくなければなりません。

LPSTR suffix - output
サフィックスの識別名を含むバッファーへのポインター。

int * length - input/output
サフィックス・バッファーの長さへのポインター。バッファーが小さいためにストリング (NULL を含む) を保持できない場合、必要なバッファーのサイズがこのパラメーターに設定されます。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
無効なハンドル。

CWB_INVALID_API_PARAMETER
無効な索引。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

CWB_BUFFER_OVERFLOW
サフィックス・バッファーが小さいため、結果全体を保持できません。

使用法 なし

cwbUN_OpenLdapPublishing:

この API は、Lightweight Directory Access Protocol (LDAP) ディレクトリーにシステムが公開する情報についての構成情報にアクセスする際に、使用可能なハンドルを作成します。

構文

```
int cwbUN_OpenLdapPublishingW
    ( LPCWSTR              system,
      cwbUN_ldapPubHandle  *pHandle
    );

int cwbUN_OpenLdapPublishingA
    ( LPCSTR               system,
      cwbUN_ldapPubHandle  *pHandle
    );
```

パラメーター

LPCSTR system - input
システム名へのポインター。

cwbUN_ldapSvrHandle *pHandle - output
戻される際に、API で使用可能なハンドルが含まれます。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_PARAMETER
無効なパラメーターが指定されています。

CWB_INVALID_API_HANDLE
無効なハンドル。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

CWBUN_LDAP_NOT_AVAIL
ディレクトリー・サービスがインストールされていないか、サーバーが構成されていません。

使用法 なし

cwbUN_FreeLdapPublishing:

この API は、入力ハンドルに関連付けられているリソースを解放します。

構文

```
int cwbUN_FreeLdapPublishing
    ( cwbUN_ldapPubHandle  handle
    );
```

パラメーター

cwbUN_ldapPubHandle handle - input
資源を解放する対象となるハンドル。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
cwbUN_OpenLdapPublishing() によりハンドルが作成されませんでした。

使用法 ハンドルは、cwbUN_OpenLdapPublishing() 呼び出しを実行して取得します。

cwbUN_GetLdapPublishCount:

この API は、サーバー用に構成された公開レコードの数を戻します。公開レコードは、公開する情報のカテゴリー、およびこれを公開する方法と場所を識別します。

構文

```
int cwbUN_GetLdapPublishCount
    ( cwbUN_ldapPubHandle  handle,
      int                  *count
    );
```

パラメーター

cwbUN_ldapPubHandle handle - input
cwbUN_OpenLdapPublishing() への呼び出しにより以前に取得されたハンドル。

int * count - output
サーバーに構成されている公開レコードの数。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
無効なハンドル。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

使用法 なし

cwbUN_GetLdapPublishType:

この API は、公開レコードの情報のタイプを返します。

構文

```
int cwbUN_GetLdapPublishType
    ( cwbUN_ldapPubHandle      handle,
      int                      index,
      cwbUN_LdapPubCategories  *information
    );
```

パラメーター

cwbUN_ldapPubHandle handle - input
cwbUN_OpenLdapPublishing() への呼び出しにより以前に取得されたハンドル。

int index - input
公開レコードのゼロ・ベースの索引。この値は、cwbUN_GetLdapPublishCount() が戻すカウントより小さくなければなりません。

cwbUN_LdapPubCategories * information - output
この公開レコードが対象とする情報のタイプ。以下の値が使用可能です。

CWBUN_LDAP_PUBLISH_USERS
ユーザー情報。

CWBUN_LDAP_PUBLISH_COMPUTERS
System i プラットフォーム。

CWBUN_LDAP_PUBLISH_NETWORK_INVENTORY
NetFinity。

CWBUN_LDAP_PUBLISH_PRINTERS
System i プラットフォームに接続されたプリンター。

戻りコード

以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
無効なハンドル。

CWB_INVALID_API_PARAMETER
無効な索引。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

使用法 なし

cwbUN_GetLdapPublishServer:

この API は、この情報の公開先のサーバーの名前を戻します。

構文

```
int cwbUN_GetLdapPublishServerW
    ( cwbUN_ldapPubHandle  handle,
      int                  index,
      LPWSTR               server,
      int                  *length
    );

int cwbUN_GetLdapPublishServerA
    ( cwbUN_ldapPubHandle  handle,
```

```
    int                 index,
    LPSTR               server,
    int                 *length
  );
```

パラメーター

cwbUN_ldapPubHandle handle - input
cwbUN_OpenLdapPublishing() への呼び出しにより以前に取得されたハンドル。

int index - input
公開レコードのゼロ・ベースの索引。この値は、cwbUN_GetLdapPublishCount() が戻すカウントより小さくなければなりません。

LPSTR server - output
サーバーの名前を含むバッファーへのポインター。

int * length - input/output
サーバー・バッファーの長さへのポインター。バッファーが小さいためにストリング (NULL を含む) を保持できない場合、必要なバッファーのサイズがこのパラメーターに設定されます。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
無効なハンドル。

CWB_INVALID_API_PARAMETER
無効な索引。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

CWB_BUFFER_OVERFLOW
サフィックス・バッファーが小さいため、結果全体を保持できません。

使用法 なし

cwbUN_GetLdapPublishPort:

この API は、この情報を公開するために使用されるサーバーのポート番号を戻します。

構文

```
int cwbUN_GetLdapPublishPort
    ( cwbUN_ldapPubHandle      handle,
      int                      index,
      int                      *port,
      cwbUN_LdapCnnSecurity    *connectionSecurity
    );
```

パラメーター

cwbUN_ldapPubHandle handle - input
cwbUN_OpenLdapPublishing() への呼び出しにより以前に取得されたハンドル。

int index - input
公開レコードのゼロ・ベースの索引。この値は、cwbUN_GetLdapPublishCount() が戻すカウントより小さくなければなりません。

int * port - output
サーバーへの接続に使用されるポート番号。

cwbUN_LdapCnnSecurity * connectionSecurity - output
サーバーへの接続に使用される接続タイプ。これは、関連ポートを介して確立することのできる接続タイプを示します。このパラメーターでは、以下の値を使用することができます。

CWBUN_LDAPCNN_NORMAL
通常接続が使用されます。

CWBUN_LDAPCNN_SSL
SSL 接続が使用されます。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
無効なハンドル。

CWB_INVALID_API_PARAMETER
無効な索引。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

使用法 なし

cwbUN_GetLdapPublishParentDn:

この API は、公開されたオブジェクトの親識別名を戻します。

例えば、公開ユーザーの parentDN が cn=users,o=ace industry,c=us で、 John Smith のユーザー情報が公開された場合、公開されたオブジェクトの DN は、 cn=john smith,cn=users,ou=ace industry,c=us になります。

構文

```
int cwbUN_GetLdapPublishParentDnW
    ( cwbUN_ldapPubHandle  handle,
      int                  index,
      LPWSTR               parentDn,
      int                  *length
    );

int cwbUN_GetLdapPublishParentDnA
    ( cwbUN_ldapPubHandle  handle,
      int                  index,
      LPSTR                parentDn,
      int                  *length
    );

int cwbUN_GetLdapPublishParentDn8 /* return parentDn in UTF-8 */
    ( cwbUN_ldapPubHandle  handle,
```

```
    int                 index,
    LPSTR               parentDn,
    int                 *length
  );
```

パラメーター

cwbUN_ldapPubHandle handle - input
cwbUN_OpenLdapPublishing() への呼び出しにより以前に取得されたハンドル。

int index - input
公開レコードのゼロ・ベースの索引。この値は、cwbUN_GetLdapPublishCount() が戻すカウントより小さくなければなりません。

LPSTR parentDn - output
parentDn の名前を含むバッファーへのポインター。

int * length - input/output
parentDn バッファーの長さへのポインター。バッファーが小さいためにストリング (NULL を含む) を保持できない場合、必要なバッファーのサイズがこのパラメーターに設定されます。

戻りコード
以下のリストは共通の戻り値を示します。

CWB_OK
正常終了。

CWB_INVALID_API_HANDLE
無効なハンドル。

CWB_INVALID_API_PARAMETER
無効な索引。

CWB_INVALID_POINTER
NULL ポインターが指定されました。

CWB_BUFFER_OVERFLOW
サフィックス・バッファーが小さいため、結果全体を保持できません。

使用法 なし

System i ナビゲーター API に固有の戻りコード

System i ナビゲーターには、固有の戻りコード・セットがあります。各コードには、それぞれ関連付けられた独自の意味があります。

```
6000    CWBUN_BAD_PARAMETER
          入力パラメーターが無効です。
6001    CWBUN_FORMAT_NOT_VALID
          入力オブジェクト名が無効です。
6002    CWBUN_WINDOW_NOTAVAIL
          ビュー・ウィンドウが見つかりません。
6003    CWBUN_INTERNAL_ERROR
          処理エラーが発生しました。
6004    CWBUN_USER_NOT_AUTHORIZED
          ユーザーには指定された権限がありません。
6005    CWBUN_OBJECT_NOT_FOUND
          オブジェクトが iSeries にありません。
6006    CWBUN_INVALID_ITEM_ID
          項目識別コード・パラメーターが無効です。
6007    CWBUN_NULL_PARM
          NULL パラメーターが渡されました。
```

6008 CWBUN_RTN_STR_TOO_LONG
ストリングが長すぎるため、戻りバッファーに挿入できません。
6009 CWBUN_INVALID_OBJ_NAME
オブジェクト名パラメーターが無効です。
6010 CWBUN_INVALID_PIDL
PIDL パラメーターが無効です。
6011 CWBUN_NULL_PIDL_RETURNED
親フォルダー PIDL が NULL です。
6012 CWBUN_REFRESH_FAILED
リストの最新表示に失敗しました。
6012 CWBUN_UPDATE_FAILED
ツールバーの更新に失敗しました。
6013 CWBUN_INVALID_NAME_TYPE
iSeries の名前タイプが無効です。
6014 CWBUN_INVALID_AUTH_TYPE
権限タイプが無効です。
6016 CWBUN_HOST_COMM_ERROR
iSeries 通信エラーです。
6017 CWBUN_INVALID_NAME_PARM
名前パラメーターが無効です。
6018 CWBUN_NULL_DISPLAY_STRING
NULL の表示ストリングが戻されました。
6019 CWBUN_GENERAL_FAILURE
一般の iSeries 操作障害です。
6020 CWBUN_INVALID_SYSVAL_ID
無効なシステム値識別コードです。
6021 CWBUN_INVALID_LIST_OBJECT
名前からリスト・オブジェクトを取得できません。
6022 CWBUN_INVALID_IFS_PATH
無効な IFS パスが指定されました。
6023 CWBUN_LANG_NOT_FOUND
拡張機能は、インストールされている言語をどれもサポートしていません。
6024 CWBUN_INVALID_USER_ATTR_ID
無効なユーザー属性識別コードです。
6025 CWBUN_GET_USER_ATTR_FAILED
ユーザー属性を取り出すことができません。
6026 CWBUN_INVALID_FLAG_VALUE
無効なフラグ・パラメーター値が設定されました。
6027 CWBUN_CANT_GET_IMAGELIST
アイコン・イメージ・リストを取得できません。

以下に示す戻りコードは、名前検査の API 用です。

6050 CWBUN_NAME_TOO_LONG
名前が長すぎます。
6051 CWBUN_NAME_NULLSTRING
空のストリングです - 文字が含まれていません。
6054 CWBUN_NAME_INVALIDCHAR
無効な文字です。
6055 CWBUN_NAME_STRINGTOOLONG
ストリングが長すぎます。
6056 CWBUN_NAME_MISSINGENDQUOTE
終了の引用符がありません。
6057 CWBUN_NAME_INVALIDQUOTECHAR
文字が引用符付きストリングに対して無効です。
6058 CWBUN_NAME_ONLYBLANKS
ブランクのみのストリングが検出されました。
6059 CWBUN_NAME_STRINGTOOSHORT
ストリングが短すぎます。
6060 CWBUN_NAME_TOOLONGFORIBM
ストリングには問題がありませんが、IBM(R) コマンドとしては長すぎます。
6011 CWBUN_NAME_INVALIDFIRSTCHAR
最初の文字が無効です。
6020 CWBUN_NAME_CHECK_LAST
予約済みの範囲です。

以下に示す戻りコードは、LDAP 関連の API 用です。

6101 CWBUN_LDAP_NOT_AVAIL
LDAP がインストールされていないか、構成されていません。
6102 CWBUN_LDAP_BIND_FAILED
LDAP のバイン下に失敗しました。

以下に示す戻りコードは、iSeries(TM) 名の検査の API 用です。

1001 CWBUN_NULLSTRING
ストリングが空です。
1004 CWBUN_INVALIDCHAR
無効な文字です。
1005 CWBUN_STRINGTOOLONG
ストリングが長すぎます。
1006 CWBUN_MISSINGENDQUOTE
引用符付きストリングの終了の引用符がありません。
1007 CWBUN_INVALIDQUOTECHAR
文字が引用符付きストリングに対して無効です。
1008 CWBUN_ONLYBLANKS
ブランクのみのストリングです。
1009 CWBUN_STRINGTOOSHORT
定義されている最小値よりも短いストリングです。
1011 CWBUN_TOOLONGFORIBM
ストリングには問題がありませんが、IBM コマンドとして長すぎます。
1012 CWBUN_INVALIDFIRSTCHAR
最初の文字が無効です。
1999 CWBUN_GENERALFAILURE
指定されていないエラーです。

Visual Basic のリファレンス

Visual Basic プラグインには、System i ナビゲーターにおける固有の制御フローがあります。また、Visual Basic プラグインは、少なくとも 1 つの System i ナビゲーター・インターフェース・クラスにインプリメントされていなければなりません。

System i ナビゲーターの構造および Visual Basic プラグインの制御フロー

System i ナビゲーターは、Visual Basic プラグインに対して、System i ナビゲーターとプラグインの間の通信を管理する、組み込みの ActiveX サーバーを提供します。

System i ナビゲーターのプラグインを開発している Visual Basic プログラマーは、Microsoft の Visual Basic 5.0 に用意された機能を使用して、プラグインのクラスを作成したり、それらを ActiveX サーバー DLL にパッケージしたりすることができます。

プラグインは、ユーザーのアクションに応答して生成される System i ナビゲーターからのメソッド呼び出しに応答することにより機能します。例えば、ユーザーが System i ナビゲーター階層中のオブジェクトを右クリックした場合、System i ナビゲーターはそのオブジェクトのコンテキスト・メニューを構成して画面にメニューを表示します。System i ナビゲーターは、選択されたオブジェクト・タイプに対しコンテキスト・メニュー項目を提供するよう登録されている各プラグインを呼び出して、メニュー項目を取得します。

プラグインによってインプリメントされた関数は、**インターフェース**として論理的にグループ化されます。インターフェースは、System i ナビゲーターが特定の機能を実行するために呼び出すことのできるクラスの、論理的に関連するメソッドのセットです。 Visual Basic プラグインの場合、以下の 3 つのインターフェースが定義されています。

- ListManager
- ActionsManager
- DropTargetManager

Visual Basic プラグイン用の System i ナビゲーター・データ

System i ナビゲーターがプラグインによりインプリメントされた関数を呼び出す場合、通常、その要求には、ユーザーが System i ナビゲーターのメイン・ウィンドウで選択したオブジェクト (複数の場合もある) が含まれます。プラグインは、どのオブジェクトが選択されているのかを判別できなければなりません。プラグインは、この情報を完全修飾オブジェクト名のリストとして受け取ります。Visual Basic プラグインの場合、選択されたオブジェクトに関する情報を提供する ObjectName クラスが定義されています。 オブジェクト階層にフォルダーを追加するプラグインは、フォルダー内の項目を項目識別コードの形で System i ナビゲーターに戻さなければなりません。 Visual Basic プラグインの場合、ItemIdentifier クラスが定義されて、要求された情報を返すためにプラグインによって使用されます。

Visual Basic プラグインにおける System i ナビゲーター・サービス

System i ナビゲーターのプラグインでは、メインの System i ナビゲーター・ウィンドウの動作に影響を与える場合があります。例えば、ユーザー操作の実行後に、System i ナビゲーターのリスト・ビューを最新表示する必要がある場合や、System i ナビゲーターの状況域にテキストを挿入する必要がある場合などです。 Visual Basic 環境では、UIServices という名前のユーティリティー・クラスが用意されており、これによって必要なサービスが提供されます。 Visual Basic プラグインは、同様の結果を得るために、cwbun.h ヘッダー・ファイル内の C++ API を使用することもできます。このクラスとそのメソッドの詳細については、 System i ナビゲーターの Visual Basic プラグイン・サポート DLL で提供されるオンライン・ヘルプ (cwbunvbi.dll および cwbunvbi.hlp) を参照してください。

関連概念

74 ページの『System i ナビゲーターの ListManager インターフェース・クラス』
ListManager インターフェース・クラスは、System i ナビゲーターにおけるデータの提供に使用されます。例えば、リスト・ビューを作成して、それにオブジェクトを設定する必要がある場合、 System i ナビゲーターが ListManager クラスのメソッドを呼び出して、それを実行します。

74 ページの『System i ナビゲーターの ActionsManager インターフェース・クラス』
ActionsManager インターフェース・クラスは、コンテキスト・メニューを構成し、コンテキスト・メニューのアクション・コマンドをインプリメントするために使用します。例えば、ユーザーが System i ナビゲーターで Visual Basic のリスト・オブジェクトを右クリックすると、ActionsManager インターフェース・クラスの queryActions メソッドが呼び出され、コンテキスト・メニュー項目のストリングが戻されます。

74 ページの『System i ナビゲーターの DropTargetManager インターフェース・クラス』
DropTargetManager インターフェース・クラスは、System i ナビゲーターでのドラッグ・アンド・ドロップ操作の処理に使用します。

System i ナビゲーターの Visual Basic インターフェース

Visual Basic プラグインは、開発者が System i ナビゲーターに提供する予定の機能のタイプに応じて、System i ナビゲーターのインターフェース・クラスを 1 つ以上インプリメントしなければなりません。

Programmer's Toolkit に、Visual Basic インターフェース定義のヘルプ・ファイルへのリンクが含まれています。

System i ナビゲーターには、以下に示す 3 つのインターフェース・クラスがあります。

- System i ナビゲーターの ActionsManager インターフェース・クラス
- System i ナビゲーターの DropTargetManager インターフェース・クラス
- System i ナビゲーターの ListManager インターフェース・クラス

注: アプリケーションで、これら 3 つのインターフェース・クラスすべてをインプリメントする必要はありません。

System i ナビゲーターの ListManager インターフェース・クラス:

ListManager インターフェース・クラスは、System i ナビゲーターにおけるデータの提供に使用されます。例えば、リスト・ビューを作成して、それにオブジェクトを設定する必要がある場合、 System i ナビゲーターが ListManager クラスのメソッドを呼び出して、それを実行します。

Visual Basic のサンプル・プラグインの listman.cls ファイルには、このクラスの例が提供されています。プラグインが System i ナビゲーターのコンポーネント・リストを生成する必要がある場合は、ListManager クラスが必要です。

このクラスとそのメソッドの詳細については、 System i ナビゲーターの Visual Basic プラグイン・サポート DLL で提供されるオンライン・ヘルプ (cwbunvbi.dll ファイルおよび cwbunvbi.hlp) を参照してください。

関連概念

72 ページの『System i ナビゲーターの構造および Visual Basic プラグインの制御フロー』
System i ナビゲーターは、Visual Basic プラグインに対して、System i ナビゲーターとプラグインの間の通信を管理する、組み込みの ActiveX サーバーを提供します。

System i ナビゲーターの ActionsManager インターフェース・クラス:

ActionsManager インターフェース・クラスは、コンテキスト・メニューを構成し、コンテキスト・メニューのアクション・コマンドをインプリメントするために使用します。例えば、ユーザーが System i ナビゲーターで Visual Basic のリスト・オブジェクトを右クリックすると、ActionsManager インターフェース・クラスの queryActions メソッドが呼び出され、コンテキスト・メニュー項目のストリングが戻されます。

Visual Basic のサンプル・プラグインの **actnman.cls** ファイルには、このクラスの例が提供されています。ActionsManager インターフェース・クラスは、プラグインがサポートする固有のオブジェクト・タイプごとに定義しなければなりません。異なるオブジェクト・タイプに同じ ActionsManager インターフェース・クラスを指定することはできますが、コード・ロジックで、複数のタイプのオブジェクトでの呼び出しを処理しなければなりません。

このクラスとそのメソッドの詳細については、 System i ナビゲーターの Visual Basic プラグイン・サポート DLL で提供されるオンライン・ヘルプ (cwbunvbi.dll ファイルおよび cwbunvbi.hlp ファイル) を参照してください。

関連概念

72 ページの『System i ナビゲーターの構造および Visual Basic プラグインの制御フロー』
System i ナビゲーターは、Visual Basic プラグインに対して、System i ナビゲーターとプラグインの間の通信を管理する、組み込みの ActiveX サーバーを提供します。

System i ナビゲーターの DropTargetManager インターフェース・クラス:

DropTargetManager インターフェース・クラスは、System i ナビゲーターでのドラッグ・アンド・ドロップ操作の処理に使用します。

ユーザーが Visual Basic のリスト・オブジェクトを選択し、そのオブジェクトに対してマウスでドラッグ・アンド・ドロップを行うと、このクラスのメソッドが呼び出され、ドラッグ・アンド・ドロップ操作が実行されます。

このクラスとそのメソッドの詳細については、 System i ナビゲーターの Visual Basic プラグイン・サポート DLL で提供されるオンライン・ヘルプ (cwbunvbi.dll ファイルおよび cwbunvbi.hlp) を参照してください。

関連概念

72 ページの『System i ナビゲーターの構造および Visual Basic プラグインの制御フロー』
System i ナビゲーターは、Visual Basic プラグインに対して、System i ナビゲーターとプラグインの間の通信を管理する、組み込みの ActiveX サーバーを提供します。

Java のリファレンス

Java プラグインには、System i ナビゲーターにおける固有の制御フローがあります。

System i ナビゲーターの構造および Java プラグインの制御のフロー

System i ナビゲーターは、Java プラグインに対して、System i ナビゲーターとプラグインの Java クラスとの間の通信を管理する、組み込みの ActiveX サーバーを提供します。

このサーバー・コンポーネントは、Java Native Interface (JNI) API を使用してプラグインのオブジェクトを作成し、そのメソッドを呼び出します。したがって、System i ナビゲーターのプラグインを開発する Java プログラマーは、 ActiveX サーバーのインプリメンテーションの詳細に煩わされることがありません。

ユーザーが System i ナビゲーターの Java プラグインを操作する場合には、特定の要求をインプリメントするためのさまざまな登録済み Java インターフェース・クラスが呼び出されます。

プラグインは、ユーザーのアクションに応答して生成される System i ナビゲーターからのメソッド呼び出しに応答することにより機能します。例えば、ユーザーが System i ナビゲーター階層中のオブジェクトを右クリックした場合、System i ナビゲーターはそのオブジェクトのコンテキスト・メニューを構成して画面にメニューを表示します。System i ナビゲーターは、選択されたオブジェクト・タイプに対しコンテキスト・メニュー項目を提供するよう登録されている各プラグインを呼び出して、メニュー項目を取得します。

プラグインによって論理的にインプリメントされた機能は、インターフェースとしてグループ化されます。インターフェースは、System i ナビゲーターが特定の機能を実行するために呼び出すことのできるクラスの、論理的に関連するメソッドのセットです。 Java プラグインの場合、以下の 3 つの **Java インターフェース**が定義されています。

- ActionsManager
- DropTargetManager
- ListManager

System i ナビゲーターにおけるプラグインの製品アーキテクチャー

System i ナビゲーター製品の内部アーキテクチャーは、System i プラットフォームの拡張可能な幅広い操作インターフェースの統合ポイントとして機能する、という目的を反映したものになっています。インターフェースの各機能コンポーネントは、ActiveX サーバーとしてパッケージされています。 System i ナビゲーターは、Windows レジストリー項目によって、特定のサーバー・コンポーネントの存在を認識します。複数のサーバーが、System i ナビゲーター階層の所定のオブジェクト・タイプに対してメニュー項目やダイアログを追加する要求を登録することがあります。

注: System i ナビゲーターのユーザーがサード・パーティーの Java プラグインを使用できるようにするには、System i Access for Windows ユーザーの使用しているパーソナル・コンピューターに、System i Access for Windows のバージョン 4 リリース 4 モディフィケーション・レベル 0 がインストールされている必要があります。

Java プラグインにおける System i ナビゲーター・データ

System i ナビゲーターがプラグインによりインプリメントされた関数を呼び出す場合、通常、その要求には、ユーザーが System i ナビゲーターのメイン・ウィンドウで選択したオブジェクト (複数の場合もある) が含まれます。プラグインは、どのオブジェクトが選択されているのかを判別できなければなりません。プラグインは、この情報を完全修飾オブジェクト名のリストとして受け取ります。Java プラグインの場合、ObjectName クラスが定義されています。これによって、選択したオブジェクトに関する情報が提供されます。オブジェクト階層にフォルダーを追加するプラグインは、フォルダー内の項目を項目識別コードの形で System i ナビゲーターに戻さなければなりません。 Java プラグインの場合、 ItemIdentifier クラスが定義されています。これは、要求された情報を返すためにプラグインが使用します。

System i ナビゲーターのプラグインでは、メインの System i ナビゲーター・ウィンドウの動作に影響を与える場合があります。例えば、ユーザー操作の実行後に、System i ナビゲーターのリスト・ビューを最新表示する必要がある場合や、System i ナビゲーターの状況域にテキストを挿入する必要がある場合などです。必要なサービスを提供するユーティリティー・クラスが、com.ibm.as400.opnav パッケージに用意されています。

プラグイン・レジストリー・ファイルのカスタマイズ

サンプル・プラグインには 2 つのレジストリー・ファイル (開発中に使用する Windows 読み取り可能のコピー、およびシステム上の配布用コピー) が含まれています。独自のプラグインを開発するには、これらのレジストリー・ファイルを変更する必要があります。このトピックでは、レジストリー・ファイルの概要および各レジストリー・ファイルの必須セクションの詳細について説明します。

System i ナビゲーターは、レジストリー・ファイルを使用してプラグインの有無、要件、および機能を識別します。この情報を提供するため、各プラグインでは少なくとも以下の情報を指定する必要があります。

- プラグインに関するグローバル情報を提供する 1 次レジストリー・キー。

  このセクションには、プログラム識別コード (ProgID) が組み込まれています。このコードは、プラグインのベンダー名およびコンポーネント名を指定するとともに、プラグインが置かれるシステム上のフォルダーの名前も指定します。 ProgID は、<*vendor*>.<*component*> という形式 (例えば、IBM.Sample) にする必要があります。
- System i ナビゲーター階層にあるオブジェクト・タイプのうち、プラグインが追加機能を提供するオブジェクト・タイプを識別するレジストリー・キー。
- プラグインがオブジェクト階層に追加するオブジェクトの各サブツリーのルート用の個別のレジストリー・キー。

  このキーには、サブツリーのルート・フォルダーに関する情報が含まれています。

C++ レジストリー値のカスタマイズ

このサンプル・プラグインには、開発時に使用するレジストリー・ファイルである SAMDBG.REG と、システムでの配布を行うためのレジストリー・ファイルである SAMPRLS.REG の、2 つのレジストリー・ファイルが含まれています。どちらのファイルも Windows オペレーティング・システムで読み取ることができます。このレジストリー・ファイルのサンプルは、独自のプラグインに合わせてカスタマイズすることができます。

プラグインのレジストリー・ファイルは、複数のセクションから構成されています。独自のプラグインを開発する際には、この情報で説明されているとおりに各セクションをカスタマイズする必要があります。

1 次レジストリー・キー:

1 次レジストリー・キーは、プラグインのグローバル情報を指定する一連のフィールドを定義します。この情報は必須です。

```
;-------------------------------------------------------------------
; プラグインの 1 次レジストリー・キーの定義
; 注: NLS および ServerEntryPoint DLL 名には、修飾ディレクトリー・パスを含めないでください。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY plug-inS¥IBM.Sample]
"Type"="PLUGIN"
"NLS"="sampmri.dll"
"NameID"=dword:00000080
"DescriptionID"=dword:00000081
"MinimumIMPIRelease"="NONE"
"MinimumRISCRelease"="030701"
"ProductID"="NONE"
"ServerEntryPoint"="sampext.dll"
```

1 次レジストリー・キー・フィールド	フィールドの説明
Type	プラグインが System i ナビゲーター階層に新しいフォルダーを追加する場合、このフィールドの値は PLUGIN に設定します。それ以外の場合は、EXT にします。
NLS	プラグインのロケール依存資源を含む資源 DLL の名前を識別します。開発バージョンのレジストリー・ファイルでは、この名前は完全修飾パス名であることがあります。
NameID	System i ナビゲーターのユーザー・インターフェースでプラグインを識別するために使用される、資源 DLL 内のテキスト・ストリングの資源識別コードを含むダブルワードです。
DescriptionID	資源 DLL 内のテキスト・ストリングの資源識別コードを含むダブルワードです。この資源 DLL は、System i ナビゲーターのユーザー・インターフェースでプラグインの機能を説明するために使用されます。
MinimumIMPIRelease	プラグインが要求する IMPI ハードウェア上で実行される i5/OS の最小リリースを識別する 6 文字の文字ストリングです。このストリングは、vvrrmm という形式でなければなりません。ここで、vv は i5/OS のバージョン、rr はリリース、そして mm はモディフィケーション・レベルです。例えば、プラグインに、バージョン 3 リリース 2 モディフィケーション・レベル 0 が必要な場合は、このフィールドの値は ″030200″ となります。 IMPI ハードウェアで実行される i5/OS のリリース (バージョン 3 リリース 6 よりも前のリリース) をプラグインがどれもサポートしていない場合は、このフィールドの値を ″NONE″ に設定します。 IMPI ハードウェアで実行されるリリースをプラグインがすべてサポートしている場合は、このフィールドの値を ″ANY″ に設定します。

1 次レジストリー・キー・フィールド	フィールドの説明
MinimumRISCRelease	プラグインが要求する RISC ハードウェア上で実行される i5/OS の最小リリースを識別する 6 文字の文字ストリングです。このストリングは、vvrrmm という形式でなければなりません。ここで、vv は i5/OS のバージョン、rr はリリース、そして mm はモディフィケーション・レベルです。例えば、プラグインに、バージョン 3 リリース 7 モディフィケーション・レベル 1 が必要な場合は、このフィールドの値は ″030701″ となります。 RISC ハードウェアで実行される i5/OS のリリース (バージョン 3 リリース 6 以降) をプラグインがどれもサポートしていない場合は、このフィールドの値を ″NONE″ に設定します。RISC ハードウェアで実行されるリリースをプラグインがすべてサポートしている場合は、このフィールドの値を ″ANY″ に設定します。
ProductID	プラグインが要求する、前提条件の System i ライセンス・プログラムのプロダクト識別コードを指定する 7 文字の文字ストリングです。システム上に特定のライセンス・プログラムがインストールされていることをプラグインが要求しない場合は、このフィールドの値を NONE に設定します。 同一プロダクトに複数のプロダクト識別コードがある場合は、それらの識別コードをコンマで区切って複数指定することができます。
ServerEntryPoint	サーバーのエントリー・ポイントをインプリメントするコード DLL の名前です。プラグインが特定のシステムでサポートされているかどうかを iSeries ナビゲーターが判別する必要が生じると、このエントリー・ポイントが System i ナビゲーターから呼び出されます。プラグインがエントリー・ポイントをインプリメントしていない場合は、このフィールドの値を ″NONE″ に設定します。開発バージョンのレジストリー・ファイルでは、この名前は完全修飾パス名であることがあります。
JavaPath	プラグインの Java クラスの場所を識別するクラスパスのストリングです。プラグインの開発中は、このフィールドには、クラス・ファイルが存在するディレクトリーのディレクトリー・パスが含まれます。実動バージョンのレジストリー・ファイルでは、System i Access for Windows のインストール・パスを基準とした相対 JAR ファイル名を指定する必要があります。各ファイル名の前には、インストール・パスを表す System i Access for Windows の置換変数が置かれます。

1 次レジストリー・キー・フィールド	フィールドの説明
JavaMRI	プラグインのロケール依存資源を含む JAR ファイルのベース名です。System i ナビゲーターは、まず、その名前の接尾部として、該当する Java 言語識別コードと国別識別コードを付けてから、各 JAR ファイルを検索します。指定されたロケールの MRI JAR ファイルが存在しない場合、 System i ナビゲーターは、基本ロケール (通常は米国英語) の MRI がコード JAR ファイル内にあると見なします。

データ・サーバーのインプリメンテーション:

このセクションでは、System i ナビゲーター階層に追加されたそれぞれの新規フォルダーに対する、IA4HierarchyFolder のインプリメンテーションを登録します。

```
--------------------------------------------------------------------------------
; このセクションでは、System i ナビゲーター階層に追加された新規のフォルダーごとに
; IA4HierarchyFolder のインプリメンテーションを登録します。

[HKEY_CLASSES_ROOT¥CLSID¥{D09970E1-9073-11d0-82BD-08005AA74F5C}]
    @="AS/400 Data Server - Sample Data"

[HKEY_CLASSES_ROOT¥CLSID¥{D09970E1-9073-11d0-82BD-08005AA74F5C}¥InprocServer32]
    @="%CLIENTACCESS%¥Plugins¥IBM.Sample¥sampext.dll"
    "ThreadingModel"="Apartment"
```

プラグインがこの階層に新しいフォルダーを複数追加する場合は、レジストリー・ファイルのこのセクションを、追加するフォルダーごとに複写しなければなりません。各フォルダーに個別の大域固有識別コード (GUID) を作成するようにしてください。プラグインがフォルダーを追加しない場合は、このセクションを除去することができます。

以下の方法で SAMPDATA.CPP を複写すると、すべての新規フォルダーが、初期状態でライブラリー・オブジェクトを持つようになります。

1. 新しいプロジェクト・ワークスペースによって生成される DLL の名前に一致するように、この DLL の名前を変更します。
2. 新規 GUID を生成し、コピーします。 82 ページの『C++ プラグイン・レジストリー・ファイルの一括変更』を参照。
3. レジストリーのこのセクションのクラス ID (CLSID) を両方とも、前のステップで生成した新規 GUID のストリングで置き換えます。
4. 複写した方の SAMPDATA.CPP ファイルで、ストリング IMPLEMENT_OLECREATE を探します。
5. コメント行にすでにある CLSID に新しい GUID を貼り付け、次に新しい GUID の 16 進値に一致するよう、IMPLEMENT_OLECREATE マクロ呼び出しの CLSID を変更します。 Sample という語を、新しいフォルダーの名前で置き換えます。
6. 名前変更された SAMPDATA.H と SAMPDATA.CPP のコピーを基にして、新しい GUID ごとに 2 つの新しいソース・ファイルを作成します。

   注: ヘッダー・ファイル (.H) には、新しいインプリメンテーション・クラスのクラス宣言が含まれています。インプリメンテーション・ファイル (.CPP) には、新しいフォルダーのデータを取得するコードが含まれています。
7. 2 つのソース・ファイル内の CSampleData というクラス名をすべて、プラグインのコンテキスト内で意味のあるクラス名に置き換えます。

8. 新しいインプリメンテーション・ファイルをプロジェクト・ワークスペースに追加するために、「**挿入**」メニューをオープンして、「**ファイルをプロジェクトに挿入 (Files Into Project)**」を選択します。

シェル・プラグインのインプリメンテーション・クラス:

このセクションでは、シェル・プラグインのインプリメンテーション・クラスを登録します。すべての C++ プラグインで、このセクションを使用しなければなりません。

```
;------------------------------------------------------------------------
; このセクションでは、シェル・プラグインのインプリメンテーション・クラスを登録します。
; シェル・プラグインは、階層内の新しいオブジェクトまたは既存のオブジェクトの
; コンテキスト・メニュー項目またはプロパティー・ページ (あるいはその両方) を追加します。

[HKEY_CLASSES_ROOT¥CLSID¥{3D7907A1-9080-11d0-82BD-08005AA74F5C}]
    @="AS/400 Shell plug-ins - Sample"

[HKEY_CLASSES_ROOT¥CLSID¥{3D7907A1-9080-11d0-82BD-08005AA74F5C}¥InprocServer32]
    @="%CLIENTACCESS%¥Plugins¥IBM.Sample¥sampext.dll"
    "ThreadingModel"="Apartment"

;------------------------------------------------------------------------
; シェル・プラグインの承認 (Windows NT(R) では必須)

[HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Shell Extensions¥Approved]
    "{3D7907A1-9080-11d0-82BD-08005AA74F5C}"="AS/400 Shell plug-ins - Sample"
```

独自のプラグイン用に、このセクションをカスタマイズするには、以下のステップに従います。

1. 新しいプロジェクト・ワークスペースによって生成された DLL の名前に一致するように、この DLL の名前を変更します。
2. 新規の大域固有識別コード (GUID) を生成して、コピーします。 82 ページの『C++ プラグイン・レジストリー・ファイルの一括変更』を参照。
3. 項目内のクラス ID (CLSID) を、生成した新規の GUID ですべて置き換えます。
4. 独自のファイル EXTINTFC.CPP でストリング IMPLEMENT_OLECREATE を探します。
5. コメント行にすでにある CLSID に新しい GUID を貼り付け、次に新しい GUID の 16 進値に一致するよう、IMPLEMENT_OLECREATE マクロ呼び出しの CLSID を変更します。

オブジェクトのシェル・プラグインのインプリメンテーション:

レジストリーの最後のセクションでは、プラグインのインプリメンテーションによって影響を受ける、System i ナビゲーター階層のオブジェクトを指定します。

```
;------------------------------------------------------------------------
; 新しいフォルダーとそのオブジェクトのコンテキスト・メニュー・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY plug-inS¥IBM.Sample¥shellex¥Sample¥*
¥ContextMenuHandlers¥{3D7907A1-9080-11d0-82BD-08005AA74F5C}]

;------------------------------------------------------------------------
; 新しいフォルダーとそのオブジェクトのプロパティー・シート・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.Sample¥shellex¥Sample¥*
¥PropertySheetHandlers¥{3D7907A1-9080-11d0-82BD-08005AA74F5C}]

;------------------------------------------------------------------------
; 新しいフォルダーとそのオブジェクトの「自動最新表示」
; プロパティー・シート・ハンドラーを登録します。
; (これにより、System i ナビゲーターの自動最新表示機能を
; フォルダーで利用することができます。)

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY plug-inS¥IBM.Sample¥shellex¥Sample¥*
```

```
¥PropertySheetHandlers¥{5E44E520-2F69-11d1-9318-0004AC946C18}]

;--------------------------------------------------------------------------

; ドラッグ・アンド・ドロップのコンテキスト・メニュー・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY plug-inS¥IBM.Sample¥shellex¥Sample¥*
¥DragDropHandlers¥{3D7907A1-9080-11d0-82BD-08005AA74F5C}]

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY plug-inS¥IBM.Sample¥shellex¥File Systems¥*
¥DragDropHandlers¥{3D7907A1-9080-11d0-82BD-08005AA74F5C}]

;--------------------------------------------------------------------------

; オブジェクトのドロップを受け入れるためのドロップ・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY plug-inS¥IBM.Sample¥shellex¥Sample¥*¥DropHandler]
        @="{3D7907A1-9080-11d0-82BD-08005AA74F5C}"

;--------------------------------------------------------------------------

; このプラグインが Secure Socket Layer (SSL) 接続をサポートしていることを登録します。
; 注:  "Support Level"=dword:00000001 は、プラグインが SSL をサポートしていることを示しています。
; 注:  "Support Level"=dword:00000000 は、プラグインが SSL をサポートしていないことを示しています。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.Sample¥SSL]
 "Support Level"=dword:00000001
```

独自のプラグイン用に、このセクションをカスタマイズするには、以下のステップに従います。

1. このセクション内のクラス ID (CLSID) を、新規の大域固有識別コード (GUID) で置き換えます。
2. プラグインが追加のプロパティー・ページを、フォルダーまたはオブジェクトのプロパティー・シートに追加しない場合は、プロパティー・シート・ハンドラーのレジストリー項目を除去します。
3. プラグインがオブジェクトのドロップ・ハンドラーでない場合は、ドロップ・コンテキスト・メニュー・ハンドラーおよびハンドラーのレジストリー項目を除去します。
4. サブキー ¥Sample¥*¥ を編集します。詳しくは、『シェルのプラグイン』を参照してください。
5. 独自の EXTINTFC.CPP 内のコードを編集または除去します。このコードでは、サンプルで定義されたオブジェクト・タイプを検査します。

   サンプルでは、フォルダー、コンテキスト・メニュー項目、プロパティー・ページ、およびドロップ・アクションを扱っていますが、サンプルの機能をどれだけ残したかによって異なります。

   注: サンプル・ファイル EXTINTFC.CPP に基づくコード・ファイルには、コンテキスト・メニュー、プロパティー・ページ、およびドロップ・アクションに対して呼び出されるコードが含まれています。サンプルのコードには、サンプルで定義しているオブジェクト・タイプの検査が含まれています。このファイルを編集し、これらのテストを除去するか、新しい機能を与えようとしているオブジェクト・タイプをチェックするようテストを変更しなければなりません。

シェルのプラグイン:

これらのレジストリー・キーは、階層内の特定のノードまたはノードのセットを、プラグインが提供する機能のタイプと、その機能をインプリメントするインプリメンテーション・クラスの CLSID にマップします。

System i ナビゲーター階層の 1 つのオブジェクト・タイプに、任意の数のシェル・プラグインが機能を追加できるよう登録することができます。あるオブジェクト・タイプに機能を提供しているサーバー構成要素が、そのプラグインだけであると想定しないでください。 このことは、既存のオブジェクト・タイプだけ

ではなく、プラグインが定義する新しいオブジェクトに対しても適用されます。プラグインが広く使用されている場合は、そのプラグインで定義されているオブジェクト・タイプを、別のベンダーに拡張させないようにすることはできません。

オブジェクト・タイプの識別コード

オブジェクト・タイプの識別コードのペア、つまりサブキー ¥Sample¥*¥ は常に、サブキー階層のこのレベルに存在しているものと想定されます。

ペアの最初の識別コードは、System i ナビゲーター構成要素のルート・フォルダーを指定します。新しいフォルダーを追加するプラグインの場合、この識別コードは、常に、前のセクションで指定されたルート・フォルダーのレジストリー・キー名に一致しなければなりません。既存のオブジェクト・タイプに振る舞いを追加するプラグインの場合、通常このサブキーは、System i コンテナー・オブジェクトの下にある第 1 レベルのフォルダーのオブジェクト・タイプになります。これらのタイプのストリングは、レジストリー内の HKEY_CLASSES_ROOT¥IBM.AS400.Network¥TYPES の下に定義されています。

ペアの 2 番目の識別コードは、プラグインが操作対象にする特定のオブジェクト・タイプを識別します。* が指定されている場合は、親のサブキーで識別されるフォルダー・タイプ、およびそのフォルダーの下の階層に表示されるすべてのフォルダーとオブジェクトに対して、プラグインが呼び出されます。それ以外の場合は、特定のタイプの識別コードを指定しなければなりません。それにより、そのオブジェクト・タイプに対してのみプラグインが呼び出されます。

オブジェクト・タイプの検査

既存のオブジェクト・タイプを検査する場合は、レジストリーの HKEY_CLASSES_ROOT¥IBM.AS400.Network¥TYPES キーの下に定義されている 3 文字の識別コードを使用する必要があります。プラグインで定義された新しいオブジェクト・タイプを検査する場合は、レジストリー・キーを使用してください。プラグインによって定義されているフォルダーに対してデータを提供する場合は、接合点として指定したフォルダーを識別する、または System i ナビゲーターに戻されるタイプを識別するレジストリー・キーを使用してください。

C++ プラグイン・レジストリー・ファイルの一括変更:

独自のプラグインを開発する際、サンプルのプラグイン・レジストリー・ファイルに対して、何らかの一括変更を行わなければならない場合があります。プラグイン・レジストリー・ファイル全体を通して使用する、固有のプログラム識別コード (ProgID) および大域固有識別コード (GUID) を指定する必要があります。

プラグインの固有 ProgID の定義

ProgID のテキスト・ストリングは、<*vendor*>.<*component*> という形式に合わせる必要があります。*vendor* にはプラグインを開発したベンダーの名前が入り、*component* には提供される機能を記述します。サンプル・プラグインのストリング IBM.Sample の場合は、IBM がベンダーを表し、Sample がこのプラグインが提供する機能の説明になります。これは、レジストリー・ファイル全体を通して使用されます。これによって、システムとワークステーションの両方で、プラグインが置かれるディレクトリーが指定されます。レジストリー・ファイル内のすべての IBM.Sample をユーザーの ProgID で置き換えてください。

新規 GUID の作成およびレジストリー・ファイルでの CLSID 値の置き換え

System i ナビゲーターの C++ プラグインが正しく動作するためには、新規のレジストリー・ファイルにおける特定のクラス ID (CLSID) を、作成した GUID で置き換えなければなりません。

Microsoft の Component Object Model では、16 バイトの 16 進整数を使用して、ActiveX のインプリメンテーション・クラスとインターフェースを一意に識別します。これらの整数は GUID として知られます。インプリメンテーション・クラスを識別する GUID を CLSID と言います System i ナビゲーターは、Windows ActiveX ランタイム・サポートを使用して、プラグインのコンポーネントのロード、および特定インターフェースのプラグインのインプリメンテーション・インスタンスに対するポインターの取得を行います。レジストリー内の CLSID は、特定の ActiveX サーバー DLL にある特定のインプリメンテーション・クラスを一意的に識別します。このマッピングの最初の段階、つまり CLSID からサーバー DLL の名前と場所までは、レジストリー項目によって行われます。したがって、System i ナビゲーターのプラグインによって、自身が提供するインプリメンテーション・クラスごとに CLSID を登録する必要があります。

GUID を生成するには、以下のステップに従います。

1. Windows のタスクバーから「**スタート**」、「**ファイル名を指定して実行**」と順に選択します。
2. GUIDGEN と入力し、「**OK**」をクリックします。
3. 「Registry Format」が選択されていることを確認します。
4. 新しい GUID 値を生成する場合は、「**New GUID**」を選択します。
5. 新しい GUID 値をクリップボードにコピーする場合は、「**コピー**」を選択します。

Visual Basic プラグインのレジストリー値のカスタマイズ

このサンプル・プラグインには、開発時に使用するレジストリー・ファイルである VBSMPDBG.REG と、システムでの配布を行うためのレジストリー・ファイルである VBSMPRLS.REG の、2 つのレジストリー・ファイルが含まれています。どちらのファイルも Windows オペレーティング・システムで読み取ることができます。このレジストリー・ファイルのサンプルは、独自のプラグインに合わせてカスタマイズすることができます。

プラグインのレジストリー・ファイルは、複数のセクションから構成されています。独自のプラグインを開発する際には、この情報で説明されているとおりに各セクションをカスタマイズする必要があります。

1 次レジストリー・キー:

1 次レジストリー・キーは、プラグインのグローバル情報を指定する一連のフィールドを定義します。この情報は必須です。

注: サブキー名は、プラグインの ProgID に一致していなければなりません。

```
[HKEY_CLASSES_ROOT¥IBM.AS400.Network
¥3RD PARTY EXTENSIONS¥IBM.VBSample]

"Type"="Plugin"

"NLS"="vbsmpmri.dll"

"NameID"=dword:00000080

"DescriptionID"=dword:00000081

"MinimumIMPIRelease"="NONE"

"MinimumRISCRelease"="040200"

"ProductID"="NONE"

"ServerEntryPoint"="vbsample.dll"
```

1 次レジストリー・キー・フィールド	フィールドの説明
Type	プラグインが System i ナビゲーター階層に新しいフォルダーを追加する場合、このフィールドの値は PLUGIN に設定します。それ以外の場合は、EXT にします。
NLS	プラグインのロケール依存資源を含む資源 DLL の名前を識別します。開発バージョンのレジストリー・ファイルでは、この名前は完全修飾パス名であることがあります。
NameID	System i ナビゲーターのユーザー・インターフェースでプラグインを識別するために使用される、資源 DLL 内のテキスト・ストリングの資源識別コードを含むダブルワードです。
DescriptionID	資源 DLL 内のテキスト・ストリングの資源識別コードを含むダブルワードです。この資源 DLL は、System i ナビゲーターのユーザー・インターフェースでプラグインの機能を説明するために使用されます。
MinimumIMPIRelease	プラグインが要求する IMPI ハードウェア上で実行される i5/OS の最小リリースを識別する 6 文字の文字ストリングです。このストリングは、vvrrmm という形式でなければなりません。ここで、vv は i5/OS のバージョン、rr はリリース、そして mm はモディフィケーション・レベルです。例えば、プラグインに、バージョン 3 リリース 2 モディフィケーション・レベル 0 が必要な場合は、このフィールドの値は ″030200″ となります。 IMPI ハードウェアで実行される i5/OS のリリース (バージョン 3 リリース 6 よりも前のリリース) をプラグインがどれもサポートしていない場合は、このフィールドの値を ″NONE″ に設定します。 IMPI ハードウェアで実行されるリリースをプラグインがすべてサポートしている場合は、このフィールドの値を ″ANY″ に設定します。
MinimumRISCRelease	プラグインが要求する RISC ハードウェア上で実行される i5/OS の最小リリースを識別する 6 文字の文字ストリングです。このストリングは、vvrrmm という形式でなければなりません。ここで、vv は i5/OS のバージョン、rr はリリース、そして mm はモディフィケーション・レベルです。例えば、プラグインに、バージョン 3 リリース 7 モディフィケーション・レベル 1 が必要な場合は、このフィールドの値は ″030701″ となります。 RISC ハードウェアで実行される i5/OS のリリース (バージョン 3 リリース 6 以降) をプラグインがどれもサポートしていない場合は、このフィールドの値を ″NONE″ に設定します。RISC ハードウェアで実行されるリリースをプラグインがすべてサポートしている場合は、このフィールドの値を ″ANY″ に設定します。

1 次レジストリー・キー・フィールド	フィールドの説明
ProductID	プラグインが要求する、前提条件の System i ライセンス・プログラムのプロダクト識別コードを指定する 7 文字の文字ストリングです。システム上に特定のライセンス・プログラムがインストールされていることをプラグインが要求しない場合は、このフィールドの値を NONE に設定します。 同一プロダクトに複数のプロダクト識別コードがある場合は、それらの識別コードをコンマで区切って複数指定することができます。
ServerEntryPoint	サーバーのエントリー・ポイントをインプリメントするコード DLL の名前です。プラグインが特定のシステムでサポートされているかどうかを iSeries ナビゲーターが判別する必要が生じると、このエントリー・ポイントが System i ナビゲーターから呼び出されます。プラグインがエントリー・ポイントをインプリメントしていない場合は、このフィールドの値を ″NONE″ に設定します。開発バージョンのレジストリー・ファイルでは、この名前は完全修飾パス名であることがあります。
JavaPath	プラグインの Java クラスの場所を識別するクラスパスのストリングです。プラグインの開発中は、このフィールドには、クラス・ファイルが存在するディレクトリーのディレクトリー・パスが含まれます。実動バージョンのレジストリー・ファイルでは、System i Access for Windows のインストール・パスを基準とした相対 JAR ファイル名を指定する必要があります。各ファイル名の前には、インストール・パスを表す System i Access for Windows の置換変数が置かれます。
JavaMRI	プラグインのロケール依存資源を含む JAR ファイルのベース名です。System i ナビゲーターは、まず、その名前の接尾部として、該当する Java 言語識別コードと国別識別コードを付けてから、各 JAR ファイルを検索します。指定されたロケールの MRI JAR ファイルが存在しない場合、 System i ナビゲーターは、基本ロケール (通常は米国英語) の MRI がコード JAR ファイル内にあると見なします。

独自のプラグインの 1 次レジストリー・キーをカスタマイズするには、以下のステップに従います。

1. ServerEntryPoint キーの ″vbsample.dll″ という名前を、プラグインの ActiveX サーバー DLL の名前に一致するように変更します。
2. NLS キーの ″vbsmpmri.dll″ という名前を、プラグインの C++ MRI の資源 DLL の名前に一致するように変更します。各 Visual Basic プラグインには、固有の C++ MRI DLL 名がなければなりません。

注: これらの変更には、どちらの場合もパスを含めないでください。

関連概念

89 ページの『Visual Basic プラグイン・レジストリー・ファイルの一括変更』
独自のプラグインを開発する場合、そのプラグインに固有のプログラム識別コード (ProgID) を定義する必要があります。ファイル全体を通して使用する、固有の ProgID を指定しなければなりません。

Visual Basic プラグインのインプリメンテーション・クラス:

このセクションでは、System i ナビゲーター階層に追加される新しいフォルダーごとに、 Visual Basic プラグインの ListManager クラスのインプリメンテーションを登録します。

プラグインが System i ナビゲーター階層に新しいフォルダーを追加しない場合は、このセクションを飛ばしてください。

Visual Basic の ListManager クラスは、プラグイン・フォルダーにデータを提供するためのメイン・インターフェースです。

このサンプルでは、Sample Visual Basic Folder を、System i ナビゲーター階層の AS4 という名前のシステムのルート・レベルに配置します。フォルダーをこの階層の別のところに表示する場合は、Parent キー値を変更しなければなりません。

```
[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥

3RD PARTY EXTENSIONS¥IBM.VBSample¥

folders¥SampleVBFolder]

"Parent"="AS4"

"Attributes"=hex:00,01,00,20

"CLSID"="{040606B1-1C19-11d2-AA12-08005AD17735}"

"VBClass"="vbsample.SampleListManager"

"VBInterface"="{0FC5EC72-8E00-11D2-AA9A-08005AD17735}"

"NameID"=dword:00000082

"DescriptionID"=dword:00000083

"DefaultIconIndex"=dword:00000001

"OpenIconIndex"=dword:00000001
```

独自のプラグイン用に、このセクションをカスタマイズするには、以下のステップに従います。

1. レジストリー・ファイル内のすべての ″SampleVBFolder″ という名前を、フォルダー・オブジェクトを識別する固有の名前に変更します。レジストリー・ファイルで指定された名前は、Visual Basic の、ListManager および ActionsManager クラスで指定されたオブジェクト名に一致していなければなりません。サンプル・プラグインの場合、これらの Visual Basic のソース・ファイルは、**listman.cls** と **actnman.cls** です。
2. VBClass キーの ″vbsample.SampleListManager″ という名前を、ListManager クラスのプログラム識別コード名に一致するように変更します。例えば、ActiveX サーバー DLL が foo.dll という名前であり、ListManager のインプリメンテーション・クラスが MyListManager の場合、プログラム識別コードは ″foo.MyListManager″ になります。この名前は大文字小文字が区別されます。
3. ″VBInterface″ キーの値を、ListManager のインプリメンテーション・クラスのインターフェース識別コードに変更します。

Parent フィールドの値:

追加されるフォルダーの親を識別するために使用する、3 文字の識別コードです。 System i ナビゲーターには、親キー値用に一連の ID が用意されています。

以下のいずれかの ID を指定することができます。

AS4	システム・フォルダー
BKF	バックアップ・フォルダー
BOF	基本操作フォルダー
CFG	構成およびサービス・フォルダー
DBF	データベース・フォルダー
FSF	ファイル・システム・フォルダー
JMF	ジョブ管理フォルダー
MCN	マネージメント・セントラル・フォルダー
MCS	マネージメント・セントラルの構成およびサービス・フォルダー
MDF	マネージメント・セントラル定義フォルダー
MST	マネージメント・セントラル・スケジュール・タスク
MSM	マネージメント・セントラル・モニター
MTA	マネージメント・セントラル・タスク・アクティビティー
MXS	マネージメント・セントラル・エクストリーム・サポート
NSR	ネットワーク・サーバー・フォルダー
NWF	ネットワーク・フォルダー
SCF	セキュリティー・フォルダー
UGF	ユーザーおよびグループ・フォルダー

関連概念

『例: 新しいフォルダーのレジストリー・キー』
プラグインがオブジェクト階層に追加するオブジェクトの各サブツリーのルートに対して、個別のレジストリー・キーを定義しなければなりません。このキーには、サブツリーのルート・フォルダーに固有の情報が含まれています。このトピックでは、それぞれの新しいフォルダーにおけるレジストリー・キー・フィールドと指定可能な値について説明します。

例: 新しいフォルダーのレジストリー・キー:

プラグインがオブジェクト階層に追加するオブジェクトの各サブツリーのルートに対して、個別のレジストリー・キーを定義しなければなりません。このキーには、サブツリーのルート・フォルダーに固有の情報が含まれています。このトピックでは、それぞれの新しいフォルダーにおけるレジストリー・キー・フィールドと指定可能な値について説明します。

このレジストリー・キーには、少なくとも 4 文字から成る、意味のあるフォルダー名を割り当ててください。

```
;-------------------------------------------------------------------
; 新しいフォルダーの登録

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY plug-inS¥IBM.Sample¥folders¥Sample]
"Parent"="AS4"
"Attributes"=hex:00,01,00,20
"CLSID"="{D09970E1-9073-11d0-82BD-08005AA74F5C}"
"NameID"=dword:00000082
"DescriptionID"=dword:00000083
"DefaultIconIndex"=dword:00000000
"OpenIconIndex"=dword:00000001
"AdminItem"="QIBM_SAMPLE_SMPFLR"
```

Parent	追加されるフォルダーの親を識別する、3 文字の識別コードです。
Attributes	標識バイトを逆順にした、フォルダーの属性を含む 4 バイトの 2 進数フィールドです。Microsoft の組み込みファイル SHLOBJ.H 内で IShellFolder::GetAttributesOf メソッドに対して定義されているフォルダーの属性フラグを参照してください。

CLSID	System i ナビゲーターによって呼び出される IA4HierarchyFolder インプリメンテーションの CLSID です。 **Java プラグイン**の場合は、CLSID は常に 1827A856-9C20-11d1-96C3-00062912C9B2 でなければなりません。 **Visual Basic プラグイン**の場合は、CLSID は常に 040606B1-1C19-11d2-AA12-08005AD17735} でなければなりません。
JavaClass	フォルダーの内容を取得するために System i ナビゲーターによって呼び出される **ListManager** インプリメンテーションの完全修飾 Java クラス名です。プラグインが Java プラグインではない場合、このフィールドは省略します。
VBClass	フォルダーの内容を取得するために System i ナビゲーターによって呼び出される **ListManager** インプリメンテーション・クラスのプログラム識別コード (ProgID) です。
VBInterface	**ListManager** インプリメンテーション・クラスのインターフェースの GUID です。
NameID	System i ナビゲーター階層に、フォルダーの名前として表示されるストリングの資源 ID を含むダブルワードです。
DescriptionID	System i ナビゲーター階層に、フォルダーの説明として表示されるストリングの資源 ID を含むダブルワードです。
DefaultIconIndex	プラグインの NLS 資源 DLL に組み込まれた、 System i ナビゲーター階層のフォルダー用として表示されるアイコンの索引を含むダブルワードです。これは、資源 DLL に組み込まれたゼロ・ベースの索引であり、アイコンの資源識別コードではありません。索引付けが正常に行われるためには、アイコンの資源識別コードが順番に割り当てられている必要があります。
OpenIconIndex	プラグインの NLS 資源 DLL に組み込まれた、ユーザーがフォルダーを選択したときに System i ナビゲーター階層のフォルダー用として必ず表示されるアイコンの索引を含むダブルワードです。
AdminItem	フォルダーへのアクセスを制御するアプリケーション管理機能の機能 ID を含むストリングです。このフィールドを省略した場合、アプリケーション管理機能はフォルダーへのアクセスを管理しません。このフィールドを指定した場合は、これはグループ機能あるいは管理可能機能の機能識別コードでなければなりません。 これをプロダクト機能の機能識別コードにはできません。

関連概念

86 ページの『Parent フィールドの値』
追加されるフォルダーの親を識別するために使用する、3 文字の識別コードです。 System i ナビゲーターには、親キー値用に一連の ID が用意されています。

Visual Basic プラグインのインプリメンテーション・オブジェクト:

レジストリーの最後のセクションでは、Visual Basic プラグインのインプリメンテーションによって影響を受ける、 System i ナビゲーター階層のオブジェクトを指定します。

ActionsManager、ListManager および DropTargetManager クラスのメソッドを実行すると、多くの場合、項目とオブジェクトが渡されます。どのフォルダー・オブジェクトが参照されているのかを判別するには、Windows レジストリーで定義されているオブジェクト・タイプのストリングを使用します。

コンテキスト・メニュー項目を使用して、プロパティー・シートをプラグインに追加することもできます。C++ プラグインに使用されるメカニズムであるプロパティー・シートのレジストリー・キーは使用することができません。自動最新表示プロパティー・シート・ハンドラーを含め、プロパティー・シート・ハンドラーは、Visual Basic プラグインではサポートされていません。

```
;-------------------------------------------------------------------
; 新しいフォルダーとそのオブジェクトのコンテキスト・メニュー・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥

IBM.VBSample¥shellex¥SampleVBFolder¥*¥
ContextMenuHandlers¥{040606B2-1C19-11d2-AA12-08005AD17735}]
"VBClass"="vbsample.SampleActionsManager"
"VBInterface"="{0FC5EC7A-8E00-11D2-AA9A-08005AD17735}"

;-------------------------------------------------------------------
; ドラッグ・アンド・ドロップのコンテキスト・メニュー・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥

IBM.VBSample¥shellex¥SampleVBFolder¥*¥
DragDropHandlers¥{040606B2-1C19-11d2-AA12-08005AD17735}]
"VBClass"="vbsample.SampleActionsManager"
"VBInterface"="{0FC5EC7A-8E00-11D2-AA9A-08005AD17735}"

;-------------------------------------------------------------------
; オブジェクトのドロップを受け入れるためのドロップ・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.VBSample¥

shellex¥SampleVBFolder¥*¥
DropHandler]
@="{040606B2-1C19-11d2-AA12-08005AD17735}"
"VBClass"="vbsample.SampleDropTargetManager"
"VBInterface"="{0FC5EC6E-8E00-11D2-AA9A-08005AD17735}"
```

独自のプラグイン用に、このセクションをカスタマイズするには、以下のステップに従います。

1. 上記の項目のクラス ID (CLSID) に、常に {040606B2-1C19-11d2-AA12-08005AD17735} というストリングが含まれるようにします。
2. VBClass キーには、Visual Basic のインプリメンテーション・クラスのプログラム識別コード (ProgID) が含まれます。
3. VBInterface キーには、Visual Basic のインプリメンテーション・クラスのインターフェース識別コードが含まれます。
4. プラグインがオブジェクトのドロップ・ハンドラーでない場合は、ドロップ・コンテキスト・メニュー・ハンドラーおよびハンドラーのレジストリー項目を除去します。
5. サブキー ¥SampleVBFolder¥*¥ の名前を変更し、フォルダー・オブジェクトを識別する一意のストリングを使用します。この名前は、 System i ナビゲーターのこのフォルダーに対してアクションが取られたとき、 Visual Basic のソースで識別に使用されるオブジェクト・タイプです。
6. ActionsManager インターフェースに基づいて作成されたファイル内で、サンプルによって定義されたオブジェクト・タイプを検査するコードを、新しいフォルダー・オブジェクトの名前を反映するように編集します。サンプル ActionsManager インターフェースは、actnman.cls にあります。

Visual Basic プラグイン・レジストリー・ファイルの一括変更:

独自のプラグインを開発する場合、そのプラグインに固有のプログラム識別コード (ProgID) を定義する必要があります。ファイル全体を通して使用する、固有の ProgID を指定しなければなりません。

ProgID のテキスト・ストリングは、*<vendor>.<component>* という形式に合わせる必要があります。*vendor* にはプラグインを開発したベンダーの名前が入り、*component* には提供される機能を記述します。サンプル・プラグインのストリング IBM.Sample の場合は、IBM がベンダーを表し、Sample がこのプラグインが提供する機能の説明になります。これは、レジストリー・ファイル全体を通して使用されます。これによ

って、システムとワークステーションの両方で、プラグインが置かれるディレクトリーが指定されます。レジストリー・ファイル内のすべての IBM.Sample をユーザーの ProgID で置き換えてください。

IBM.VBSample となっている箇所を、新規の [vendor].ProgID ですべて置き換えてください。

注: System i ナビゲーターには、Java および Visual Basic で作成されたプラグインを管理する、組み込みの ActiveX サーバー DLL が用意されています。したがって、すべての Java および Visual Basic のプラグインは、それぞれ専用の CLSID を登録しています。プログラミング・サンプルで提供されているレジストリー・ファイルには、すでにこれらの定義済みの CLSID が含まれています。

関連概念

83 ページの『1 次レジストリー・キー』
1 次レジストリー・キーは、プラグインのグローバル情報を指定する一連のフィールドを定義します。この情報は必須です。

サンプル Java レジストリー・ファイル

Java で作成されたサンプル・プラグインには、それぞれ専用のレジストリー・ファイルがあります。

以下のセクションでは、レジストリー・ファイルの重要な部分を説明し、プラグイン用に適切な項目を作成する方法を示します。例は、説明する機能に適したサンプルに基づいています。

プログラム識別コード (ProgID)

作成したプラグインは、<*vendor*>.<*component*> という形式 (*vendor* はそのプラグインを開発したベンダーを表し、*component* は提供される機能を記述したものになります) のテキスト・ストリングによって、System i ナビゲーターで一意的に識別されます。以下の例にあるストリング IBM.MsgQueueSample3 の場合、IBM はベンダーを表し、MsgQueueSample3 はプラグインの提供する機能を記述したものになります。このストリングは、プログラム識別コード (ProgID) と呼ばれます。これは、プラグインが提供する機能を指定する際に、レジストリー・ファイル全体を通して使用されます。また、システムおよびクライアント・ワークステーションの両方におけるプラグイン配置先ディレクトリーも、これによって指定されます。

大域固有識別コード (GUID)

Microsoft の Component Object Model では、16 バイトの 16 進整数を使用して、ActiveX のインプリメンテーション・クラスとインターフェースを一意的に識別します。これらの整数は、*大域固有識別コード*、つまり *GUID* として知られます。インプリメンテーション・クラスを識別する GUID は、CLSID (「クラス ID」と発音します) と呼ばれます。

Java で作成された System i ナビゲーター・コンポーネントの場合は、新規の GUID を定義しないでください。 すべての Java プラグインは、Java プラグインを管理する組み込みの ActiveX サーバー構成要素を指定する標準の GUID のセットを使用します。使用する標準の CLSID は、以下の例のとおりです。

プラグインの 1 次属性の定義

```
;------------------------------------------------------------------
; Message Queue Sample 3 の 1 次レジストリー・キーの定義

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.MsgQueueSample3]
"Type"="PLUGIN"
"NLS"="MessageQueuesMRI.dll"
"NameID"=dword:00000001
"DescriptionID"=dword:00000002
"MinimumIMPIRelease"="NONE"
"MinimumRISCRelease"="ANY"
```

```
"ProductID"="NONE"
"ServerEntryPoint"="NONE"
"JavaPath"="MsgQueueSample3.jar"
"JavaMRI"="MsgQueueSample3MRI.jar"
```

Type プラグインが System i ナビゲーター階層に新しいフォルダーを追加する場合、このフィールドの値は PLUGIN に設定します。それ以外の場合は、EXT にします。

NLS プラグインのロケール依存資源を含む資源 DLL の名前を識別します。開発バージョンのレジストリー・ファイルでは、この名前は完全修飾パス名であることがあります。

NameID

System i ナビゲーターのユーザー・インターフェースでプラグインを識別するために使用される、資源 DLL 内のテキスト・ストリングの資源識別コードを含むダブルワードです。

DescriptionID

資源 DLL 内のテキスト・ストリングの資源識別コードを含むダブルワードです。この資源 DLL は、System i ナビゲーターのユーザー・インターフェースでプラグインの機能を説明するために使用されます。

MinimumIMPIRelease

IMPI ハードウェア上で実行される i5/OS のうち、プラグインで必要となる最小リリースを表す、6 文字のストリングです。このストリングは、vvrrmm という形式でなければなりません。ここで、vv は i5/OS のバージョン、rr はリリース、そして mm はモディフィケーション・レベルです。例えば、プラグインに、バージョン 3 リリース 2 モディフィケーション・レベル 0 が必要な場合は、このフィールドの値は 030200 となります。

IMPI ハードウェアで実行される i5/OS のリリース (バージョン 3 リリース 6 よりも前のリリース) をプラグインがどれもサポートしていない場合は、このフィールドの値を ″NONE″ に設定します。 IMPI ハードウェアで実行されるリリースをプラグインがすべてサポートしている場合は、このフィールドの値を ANY に設定します。

MinimumRISCRelease

プラグインが要求する RISC ハードウェア上で実行される i5/OS の最小リリースを識別する 6 文字の文字ストリングです。このストリングは、vvrrmm という形式でなければなりません。ここで、vv は i5/OS のバージョン、rr はリリース、そして mm はモディフィケーション・レベルです。例えば、プラグインに、バージョン 3 リリース 7 モディフィケーション・レベル 1 が必要な場合は、このフィールドの値は 030701 となります。

RISC ハードウェアで実行される i5/OS のリリース (バージョン 3 リリース 6 以降) をプラグインがどれもサポートしていない場合は、このフィールドの値を NONE に設定します。RISC ハードウェアで実行されるリリースをプラグインがすべてサポートしている場合は、このフィールドの値を ″ANY″ に設定します。

ProductID

プラグインが要求する、前提条件の System i ライセンス・プログラムのプロダクト識別コードを指定する 7 文字の文字ストリングです。システム上に特定のライセンス・プログラムがインストールされていることをプラグインが要求しない場合は、このフィールドの値を NONE に設定します。

同一プロダクトに複数のプロダクト識別コードがある場合は、それらの識別コードをコンマで区切って複数指定することができます。

ServerEntryPoint

サーバーのエントリー・ポイントをインプリメントするコード DLL の名前です。プラグインが特定のシステムでサポートされているかどうかを iSeries ナビゲーターが判別する必要が生じると、

このエントリー・ポイントが System i ナビゲーターから呼び出されます。プラグインがエントリー・ポイントをインプリメントしていない場合は、このフィールドの値を NONE に設定します。開発バージョンのレジストリー・ファイルでは、この名前は完全修飾パス名であることがあります。

JavaPath
プラグインの Java クラスの場所を識別するクラスパスのストリングです。プラグインの開発中は、このフィールドには、クラス・ファイルが存在するディレクトリーのディレクトリー・パスが含まれます。実動バージョンのレジストリー・ファイルでは、JAR ファイルを識別する必要があります。JAR ファイル名は、ディレクトリー名で修飾しないでください。System i ナビゲーターは、Java VM に渡すクラスパスのストリングを構成する際に、それらのファイル名を自動的に修飾します。

JavaMRI
プラグインのロケール依存資源を含む JAR ファイルのベース名です。System i ナビゲーターは、最初に各 JAR ファイルの名前に適切な Java 言語識別コードと国別識別コードを接尾部として付けてから、それぞれのファイルを検索します。開発バージョンのレジストリー・ファイルでは、このフィールドに空ストリングを含めることができます。基本ロケール (通常は米国英語) の資源が、コード JAR に存在するためです。

新しいフォルダーの定義

```
;-------------------------------------------------------------------
; 新しいフォルダーの登録

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.MsgQueueSample3¥folders¥Sample3]
"Parent"="AS4"
"Attributes"=hex:00,01,00,a0
"CLSID"="{1827A856-9C20-11d1-96C3-00062912C9B2}"
"JavaClass"="com.ibm.as400.opnav.MsgQueueSample3.MqListManager"
"NameID"=dword:0000000b
"DescriptionID"=dword:0000000c
"DefaultIconIndex"=dword:00000001
"OpenIconIndex"=dword:00000000
"AdminItem"="QIBM_SAMPLE_SMPFLR"
"TaskpadNameID"=dword:00000003
"TaskpadDescriptionID"=dword:00000004
```

Type プラグインが System i ナビゲーターの階層に追加する個々の新しいフォルダーには、固有の論理タイプがあります。上の例では、ストリング Sample3 が、実行時に制御がプラグインに渡されるときに、現在選択されているフォルダーを識別するために使用されるタイプです。

Parent 追加されるフォルダーの親を識別する、3 文字の識別コードです。以下の識別コードの 1 つを指定できます。

ADF	アプリケーション開発フォルダー
AS4	システム・フォルダー
BKF	バックアップ・フォルダー
BOF	基本操作フォルダー
CFG	構成およびサービス・フォルダー
DBF	データベース・フォルダー
FSF	ファイル・システム・フォルダー
MCN	マネージメント・セントラル・フォルダー
MCS	マネージメント・セントラルの構成およびサービス・フォルダー
MDF	マネージメント・セントラル定義フォルダー
MMN	マネージメント・セントラル・モニター
MST	マネージメント・セントラル・スケジュール・タスク
MTA	マネージメント・セントラル・タスク・アクティビティー

MXS	マネージメント・セントラル・エクストリーム・サポート
NSR	ネットワーク・サーバー・フォルダー
NWF	ネットワーク・フォルダー
SCF	セキュリティー・フォルダー
UGF	ユーザーおよびグループ・フォルダー
WMF	ワーク・マネージメント・フォルダー

Attributes

標識バイトを逆順にした、フォルダーの属性を含む 4 バイトの 2 進数フィールドです。Microsoft の組み込みファイル SHLOBJ.H 内で IShellFolder::GetAttributesOf メソッドに対して定義されているフォルダーの属性フラグを参照してください。フォルダーにタスクパッドがあることを示すには、0x00000008 を使用してください。

CLSID

System i ナビゲーターによって呼び出される IA4HierarchyFolder インプリメンテーションの CLSID です。Java プラグインの場合、この CLSID は、常に、{1827A856-9C20-11d1-96C3-00062912C9B2} でなければなりません。

JavaClass

フォルダーの内容を取得するために System i ナビゲーターによって呼び出される **ListManager** インプリメンテーションの完全修飾 Java クラス名です。

NameID

System i ナビゲーター階層に、フォルダーの名前として表示されるストリングの資源 ID を含むダブルワードです。

DescriptionID

System i ナビゲーター階層に、フォルダーの説明として表示されるストリングの資源 ID を含むダブルワードです。

DefaultIconIndex

プラグインの NLS 資源 DLL に組み込まれた、 System i ナビゲーター階層のフォルダー用として表示されるアイコンの索引を含むダブルワードです。これは、資源 DLL に組み込まれたゼロ・ベースの索引であり、アイコンの資源識別コードではありません。索引付けが正常に行われるためには、アイコンの資源識別コードが順番に割り当てられている必要があります。

OpenIconIndex

プラグインの NLS 資源 DLL に組み込まれた、ユーザーがフォルダーを選択したときに System i ナビゲーター階層のフォルダー用として必ず表示されるアイコンの索引を含むダブルワードです。これは、デフォルトのアイコン索引と同じであってもかまいません。

AdminItem

フォルダーへのアクセスを制御するアプリケーション管理機能の機能 ID を含むストリングです。このフィールドを省略した場合、アプリケーション管理機能はフォルダーへのアクセスを管理しません。このフィールドを指定した場合は、これはグループ機能あるいは管理可能機能の機能識別コードでなければなりません。 これをプロダクト機能の機能識別コードにはできません。

TaskpadNameID

System i ナビゲーター階層に、タスクパッドの名前として表示されるストリングの資源 ID を含むダブルワードです。

TaskpadDescriptionID

資源 DLL 内のテキスト・ストリングの資源識別コードを含むダブルワードです。この資源 DLL は、System i ナビゲーターのユーザー・インターフェースでタスクパッドの機能を説明するために使用されます。

コンテキスト・メニュー項目の追加

```
;------------------------------------------------------------------
; 新しいフォルダーとそのオブジェクトのコンテキスト・メニュー・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.MsgQueueSample3¥
   shellex¥Sample3¥*¥ContextMenuHandlers¥{1827A857-9C20-11d1-96C3-00062912C9B2}]
"JavaClass"="com.ibm.as400.opnav.MsgQueueSample3.MqActionsManager"

;------------------------------------------------------------------------------
; 新しいフォルダーとそのオブジェクトのドラッグ・アンド・ドロップの
; コンテキスト・メニュー・ハンドラーを登録します。
[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.MsgQueueSample3¥
   shellex¥Sample3¥*¥DragDropHandlers¥{1827A857-9C20-11d1-96C3-00062912C9B2}]
"JavaClass"="com.ibm.as400.opnav.MsgQueueSample3.MqActionsManager"
```

タスクパッドのタスクの追加

```
;------------------------------------------------------------------
; 新しいフォルダーとそのオブジェクトのタスク・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.MsgQueueSample5¥
   shellex¥Sample5¥*¥TaskHandlers¥{1827A857-9C20-11d1-96C3-00062912C9B2}]
"JavaClass"="com.ibm.as400.opnav.MsgQueueSample5.MqTasksManager"
"JavaClassType"="TasksManager"
```

ドラッグ・アンド・ドロップのサポート

```
;------------------------------------------------------------------
; 新しいフォルダーとそのオブジェクトのドロップ・ハンドラーを登録します。

[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.MsgQueueSample3¥
    shellex¥Sample3¥*¥DropHandler]
@="{1827A857-9C20-11d1-96C3-00062912C9B2}"
"JavaClass"="com.ibm.as400.opnav.MsgQueueSample3.MqDropTargetManager"
```

管理されるオブジェクトの指定

オブジェクト・タイプ識別コードのペアが shellex キーの下に必要です。ペアの最初の識別コードは、System i ナビゲーター構成要素のルート・フォルダーを指定します。プラグインが追加する新しいフォルダーの場合、この識別コードは、接合点として指定したフォルダーの論理タイプに一致する必要があります。既存のフォルダーの場合は、このサブキーは、通常、System i のコンテナー・オブジェクトの下の、最初のレベルのフォルダーのオブジェクト・タイプでなければなりません。これらのタイプのストリングは、レジストリー内の HKEY_CLASSES_ROOT¥IBM.AS400.Network¥TYPES の下に定義されています。

ペアの 2 番目の識別コードは、プラグインが操作対象にする特定のオブジェクト・タイプを識別します。「*」が指定されている場合は、最初の識別コードで識別されるフォルダー・タイプ、およびそのフォルダーの下の階層に表示されるすべてのフォルダーとオブジェクトに対して、プラグインが呼び出されます。それ以外の場合は、特定のタイプの識別コードを指定しなければなりません。それにより、そのタイプのオブジェクトに対してユーザーがアクションを行った場合のみ、そのプラグインが呼び出されます。

System i ナビゲーター階層の所定のオブジェクト・タイプに対して機能を追加できるように、任意の数のプラグインで登録することができます。あるオブジェクト・タイプに機能を提供しているシステム構成要素が、そのプラグインだけであると想定しないでください。 このことは、既存のオブジェクト・タイプだけ

ではなく、プラグインが定義する新しいオブジェクトに対しても適用されます。プラグインが広く使用されている場合は、そのプラグインで定義されているオブジェクト・タイプを、別のベンダーが拡張できないようにすることはできません。

CLSID

上の例で示されている CLSID は、Java プラグインを管理する組み込みの ActiveX サーバー構成要素を指定します。フォルダーに関係しないすべての機能に対しては、この CLSID は {1827A857-9C20-11d1-96C3-00062912C9B2} でなければなりません。

JavaClass

指定された機能をサポートするために、System i ナビゲーターによって呼び出されるインターフェースのインプリメンテーションの完全修飾 Java クラス名です。

SSL サポート

プラグインとシステムの間の通信を、ソケット API などといった低レベルの通信サービスを使用して行う場合、SSL が要求されたときに SSL をサポートするのは、プラグインの担当となります。これをサポートしていないプラグインでは、"Support Level"=dword:00000000 が指定されます。これは、プラグインが SSL をサポートしていないことを示します。これにより、ユーザーがセキュア接続を要求しても、プラグインの機能は使用不可になっています。

```
;-------------------------------------------------------------------------
; このプラグインが SSL をサポートしていることを示します。

 [HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.MsgQueueSample3¥SSL]
 "Support Level"=dword:00000001
```

Support Level

プラグインが SSL をサポートしている場合は、サポート・レベルの値を 1 に設定します。それ以外の場合は 0 に設定します。

プロパティー・シート・ハンドラーのプロパティー・ページ

Microsoft Foundation Class (MFC) ライブラリーのクラスでは、プロパティー・シート・ハンドラーでのプロパティー・ページの作成をサポートしていません。ただし、MFC のクラス CPropertyPage の代わりに、IBM が提供している CExtPropertyPage を使用することができます。

System i ナビゲーターのプラグインによってインプリメントされるプロパティー・ページには、サブクラス CExtPropertyPage がなければなりません。クラス宣言はヘッダー・ファイル PROPEXT.H に、インプリメンテーションはファイル PROPEXT.CPP にあります。どちらのファイルも、サンプル・プラグインのパーツとして提供されています。

注: プラグインのプロジェクト・ワークスペースに PROPEXT.CPP を組み込む必要があります。

1 つのプロパティー・シートをプラグインのオブジェクト・タイプの 1 つに関連付けることを、プラグインが要求している場合は、SFGAO_HASPROPSHEET フラグをそのオブジェクトの属性の一部として戻さなければなりません。このフラグがオンにされている場合にコンテキスト・メニュー項目が選択されると、System i ナビゲーターは、オブジェクトのコンテキスト・メニューに自動的に「プロパティー (Properties)」を追加し、登録済みのすべてのプロパティー・シート・ハンドラーを呼び出して、プロパティー・シートにページを追加します。

場合によっては、独自のオブジェクト・タイプの 1 つに定義された「プロパティー (Properties)」コンテキスト・メニュー項目が、プラグインによって、プロパティー・シートではなく、標準の Windows ダイアログとしてインプリメントされることがあります。このような場合、フラグが定義されます。IContextMenu::QueryContextMenu への呼び出し時に System i ナビゲーターに戻されます。このフラグが戻

されると、プロパティーに対する自動処理は実行されないため、コンテキスト・メニュー項目の追加と、関連付けられているダイアログのインプリメントは、プラグインが行います。このフラグについては、QueryContextMenu フラグについてで説明されています。

プラグインによって、ユーザー・プロパティー・シートにプロパティー・ページが追加される場合、プロパティー・シート・ハンドラーの CLSID を指定するキーには、PropSheet フィールドを指定しなければなりません。このフィールドでは、指定のハンドラーがページを追加する際の追加先となる、プロパティー・シートが識別されます。以下は、その一例です。

```
;-------------------------------------------------------------------
; System i ユーザー用に Network プロパティー・シートのプロパティー・シート・ハンドラーを登録する
[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY plug-inS¥IBM.Sample¥shellex¥Users
and Groups¥User¥PropertySheetHandlers¥{3D7907A1-9080-11d0-82BD-08005AA74F5C}]
"PropSheet"="Networks"
```

PropSheet フィールドの有効な値は以下のとおりです。

PropSheet フィールドの有効な値				
グループ	**個人**	**セキュリティーまたは機能**	**ジョブ**	**ネットワーク**
Groups-Before-All Groups-After-Info	Personal-Before-All Personal-After-Name Personal-After-Location Personal-After-Mail	Capabilities-Before-All Capabilities-After-Privileges Capabilities-After-Auditing Capabilities-Before-Other Capabilities-After-Other	Jobs-Before-All Jobs-After-General Jobs-After-Startup Jobs-After-Display Jobs-After-Output Jobs-After-International	Networks-Before-All Networks-After-Servers Networks-After-General

システム・ユーザーのプロパティー・シートにページを追加するには、プラグインが IA4PropSheetNotify インターフェースをインプリメントしていなければなりません (IA4PropSheetNotify インターフェース仕様のリストを参照)。

制約事項: System i ユーザー・オブジェクトのプロパティー・シートでは、現在このような制約事項が設けられています。 1 人のシステム・ユーザーに関連付けられている各種のプロパティー・シートに対して複数のプロパティー・シート・ハンドラーを同じインプリメンテーション・クラスでインプリメントすることはできません。各プロパティー・シートには、個別の CLSID が必要です。

関連概念

36 ページの『IA4PropSheetNotify インターフェース仕様のリスト』

IA4PropSheetNotify インターフェースは、IShellPropSheetExt のインプリメンテーションへの通知を提供します。 Users および Groups のプロパティー・シートのいずれかにプロパティー・ページを追加する場合に、これらの通知が必要になります。

QueryContextMenu フラグについて:

System i ナビゲーターの IContextMenu インターフェースのサポートが拡張されました。

コンテキスト・メニュー項目の順序

System i ナビゲーターは、IContextMenu インターフェースを拡張して、特定のフォルダーまたはオブジェクトのメニューにメニュー項目を追加する順序を、より詳細に制御しています。System i ナビゲーターでは、そのコンテキスト・メニューは 3 つのセクションに構造化されています。この構造により、オブジェクトのコンテキスト・メニューに複数のコンポーネントが項目を追加する場合でも、項目が Windows ユーザー・インターフェースに定義されている適切な順序で表示されるようになります。

最初のセクションには、データベース・テーブルの再編成など、オブジェクト・タイプに固有のアクションが含まれます。2 番目のセクションには、「オブジェクトの作成」項目が含まれます。これらの項目は、「新規作成 (New)」メニュー項目のカスケード・メニューとなるオブジェクト・タイプです。最後に、削除やプロパティーなど「標準の」Windows メニュー項目があります。コンテキスト・メニューのいずれのセクションにもメニュー項目を追加することができます。

System i ナビゲーターは、1 つのコンポーネントに対してメニューのセクションごとに 1 回ずつ、計 3 回連続して QueryContextMenu メソッドを呼び出します。コンテキスト・メニューのどのセクションでサービスが提供されているかを判別できるように、uFlags パラメーターに以下の追加フラグが定義されています。

UNITY_CMF_CUSTOM
このフラグは、メニューにオブジェクト固有のアクションを追加する必要があることを示します。

UNITY_CMF_NEW
このフラグは、メニューにオブジェクト作成項目を追加する必要があることを示します。

UNITY_CMF_STANDARD
このフラグは、メニューに標準のアクションを追加する必要があることを示します。

UNITY_CMF_FILEMENU

このフラグは、UNITY_CMF_STANDARD を変更します。これは、ユーザーがマウス・ボタンを右クリックした際に表示されるメニューとは対照的な、オブジェクトの「ファイル」プルダウン・メニュー構造を指し示します。

「ファイル」プルダウンの項目は、やや異なる方法で配置されます。メニューに「プロパティー」を追加する場合、通常この項目の前に挿入するセパレーターは挿入しないでください。また、「ファイル」メニューに「コピー」や「貼り付け」などの編集アクションを追加しないようにしてください。これらは「編集 (Edit)」プルダウンに表示されます。(System i ナビゲーターは、ユーザーのシェル・プラグインを適宜呼び出し、「編集 (Edit)」メニューの項目を取得します。UNITY_CMF_FILEMENU は設定しません。)

固有のプロパティー・ダイアログ

プラグインは、プロパティー・シートではなく、独自のオブジェクト・タイプの 1 つに定義された「プロパティー (Properties)」コンテキスト・メニュー項目を標準の Windows ダイアログとしてインプリメントすることがあります。 UNITY_CMF_STANDARD フラグが設定されている場合、このような状況で定義されたフラグは、IContextMenu::QueryContextMenu への呼び出し時に System i ナビゲーターに戻されます。この A4HYF_INFO_PROPERTIESADDED フラグは、QueryContextMenu により戻される HRESULT 値と論理和演算されます。

このフラグが戻るということは、「プロパティー (Properties)」の自動処理が行われないということです。この場合、プラグインはコンテキスト・メニュー項目を追加し、関連ダイアログを構成しなければなりません。

例: プロパティー・シート・ハンドラーの Visual Basic プロパティー・ページの作成

System i ナビゲーターの Visual Basic プラグインがインプリメントしたプロパティー・ページは、レジストリー・キーを使用して指定することができません。特定のプロパティー・ページ・コンテキスト・メニュー項目を ListManager クラスに追加して、プロパティー・ページをインプリメントする必要があります。既存のプロパティー・シート・オブジェクトには、プロパティー・ページを追加することはできません。

Visual Basic のサンプル・プラグインでは、 System i ナビゲーター・リストの Library に対するプロパティー・ページがサポートされています。これは、以下のステップで行います。

1. listman.cls 内で、Library オブジェクト・タイプが getAttributes メソッドのプロパティー・ページを指定します。

```
' Returns the attributes of an object in the list.
Public Function ListManager_getAttributes(ByVal item As Object) As Long
    Dim uItem As ItemIdentifier
    Dim nAttributes As ObjectTypeConstants

    If Not IsEmpty(item) Then
        Set uItem = item
    End If

    If uItem.getType = "SampleVBFolder" Then
        nAttributes = OBJECT_ISCONTAINER
    ElseIf item.getType = "SampleLibrary" Then
        nAttributes = OBJECT_IMPLEMENTSPROPERTIES
    Else
        nAttributes = 0
    End If

    ListManager_getAttributes = nAttributes
End Function
```

2. actnman.cls 内で、queryActions メソッドが Library オブジェクトのコンテキスト・メニューにプロパティーを表示するよう指定します。

```
Public Function ActionsManager_queryActions(ByVal flags As Long) As Variant
        .
        .

        ' Add menu items to a Sample Library
        If selectedFolderType = "SampleLibrary" Then
             ' Standard Actions
             If (flags And STANDARD_ACTIONS) = STANDARD_ACTIONS Then
                   ReDim actions(0)

                   ' Properties
                   Set actions(0) = New ActionDescriptor
                   With actions(0)
                         .Create
                         .setID IDPROPERTIES
                         .SetText m_uLoader.getString(IDS_ACTIONTEXT_PROPERTIES)
                         .setHelpText m_uLoader.getString(IDS_ACTIONHELP_PROPERTIES)
                         .setVerb "PROPERTIES"
                         .setEnabled True
                         .setDefault True
                   End With

                   ' Properties is only selectable if there is ONLY 1 object selected
                   If Not IsEmpty(m_ObjectNames) Then
                        If UBound(m_ObjectNames) > 0 Then
                             actions(2).setEnabled False
                        End If
                   End If
              End If
```

```
        End If
        .
        .
End Function
```

3. actnman.cls 内で、actionsSelected メソッドが、プロパティーのコンテキスト・メニューが選択されるとプロパティー・フォームを表示します。

```
Public Sub ActionsManager_actionSelected(ByVal action As Integer, ByVal owner As Long)
    .
    .
    Select Case action
        .
        .
        Case IDPROPERTIES
            If (Not IsEmpty(m_ObjectNames)) Then
                ' Pass the System Name into a hidden field on the form for later use
                frmProperties.lblSystemName = m_ObjectNames(0).getSystemName

                ' Pass the Display Name of the selected object into a hidden field on the form
                frmProperties.lblLibName = m_ObjectNames(0).getDisplayName

                ' Show the properties
                frmProperties.Show vbModal
            End If
        .
        .
        Case Else
            'Do Nothing
    End Select

  .
End Sub
```

注: プロパティー・シートを作成して表示するためのコードは、**propsht.frm** にあります。

Java におけるプロパティー・シートの処理

Java プラグインのプロパティー・シートに、プロパティー・ページを追加することができます。これにより、オブジェクト名の作成、プロパティーの表示、サード・パーティーとのオブジェクトの共用、および同一プラグインでの C++ と Java コードの混合が可能になります。

プロパティー・ページを使用するには、以下のメソッドを提供するプロパティー・マネージャー・インターフェースを構築しなければなりません。

- Initialize

  プロパティーのコンテナー・オブジェクトを識別します。

- getPages

  PanelManager オブジェクトのベクトルを構成し、提供します。

- CommitHandlers

  コミット時に呼び出されるハンドラーのベクトルを戻します。

- CancelHandlers

  取り消し時に呼び出されるハンドラーのベクトルを戻します。

次に、ListManager の getAttributes メソッドで ListManager.OBJECT_HASPROPERTIES を戻して、プロパティー・メニューを使用可能にします。

最後に、PopertiesManagerInterface を識別するレジストリー項目を作成します。例えば、以下のようになります。

```
[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥AS/400 Network¥*
¥shellex¥PropertySheetHandlers¥{1827A857-9C20-11d1-96C3-00062912C9B2}]
"JavaClass"="com.ibm.as400.opnav.TestPages.TestPropertiesManager"
"JavaClassType"="PropertiesManager"
```

注: 複数の PropertiesManager のインプリメンテーションが、ある 1 つのオブジェクト・タイプ用のプロパティー・ページを提供するように登録していることがあります。したがって、ユーザーのエンティティーだけがページを提供していると想定したり、ページが追加される順序を想定したりしないでください。

例: Java Properties Manager:

この例では、Java Properties Manager のサンプル・コードを紹介します。

注: コードのサンプルを使用すると、 103 ページの『コードに関するライセンス情報および特記事項』の条件に同意したものとみなされます。

```
package com.ibm.as400.opnav.Sample;

import com.ibm.as400.opnav.*;

import java.awt.Frame;

import com.ibm.as400.ui.framework.java.*;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class SamplePropertiesManager implements
PropertiesManager
{

// The list of selected objects.
ObjectName[] m_objectNames;

// Save the array of selected object names
//
public void initialize(ObjectName[] objectNames)
{

m_objectNames = objectNames;

}

// Return an array of Panel Managers
//
public PanelManager[] getPages()
{

// Instantiate the data beans
MyDataBean dataBean = new MyDataBean();
dataBean.load();
AnotherDataBean dataBean2 = new AnotherDataBean();
dataBean2.load();

DataBean[] dataBeans = { dataBean };
DataBean[] dataBeans2 = { dataBean2 };

// Create the panel
PanelManager pm = null;
PanelManager pm2 = null; try
```

```
{

pm = new PanelManager("com.ibm.as400.opnav.Sample.Sample",
"PAGE1",
dataBeans);

pm2 = new PanelManager("com.ibm.as400.opnav.Sample.Sample",
"PAGE2",
dataBeans2);


}

catch (com.ibm.as400.ui.framework.java.DisplayManagerException
e)
{

Monitor.logError("SamplePropertiesManager: Exception when
creating pages "+e);

}

pm.setTitle("First Java Page");
pm2.setTitle("Second Java Page");

PanelManager[] PMArray = {pm, pm2};

return PMArray;

}

// Return a list of ActionListener objects to be notified when
commit is processed

public ActionListener[] getCommitListeners()
{

ActionListener[] al = new ActionListener[1];
al[0] = new ActionListener()
{

public void actionPerformed(ActionEvent evt)
{

Monitor.logError("SamplePropertiesManager: Processing Commit
Listener");

}

};
return al;

}
// Return a list of ActionListener objects to be notified when
cancel is selected
public ActionListener[] getCancelListeners()
{

ActionListener[] al = new ActionListener[1];
al[0] = new ActionListener()
{

public void actionPerformed(ActionEvent evt)
{

Monitor.logError("SamplePropertiesManager: Processing Cancel
Listener");
```

```
}

};
return al;

}

}
```

Secure Sockets Layer のレジストリー項目

System i ナビゲーターのユーザーは、System i オブジェクトのプロパティー・シートの「接続」タブ付きページにある「**Secure Sockets Layer を使用する (Use Secure Sockets Layer)**」チェック・ボックスを選択することによって、システムに対するセキュア接続を要求することができます。これが選択されている場合、ユーザーは、Secure Sockets Layer (SSL) 通信をサポート可能な System i ナビゲーター・コンポーネントのみを活動化できます。

System i Access for Windows のシステム・ハンドルを使用するか (cwbCO_SysHandle と入力)、または com.ibm.as400.access.AS400 クラスを使用して (Java プラグインの場合)、プラグインとシステムの間の通信をすべて管理する場合、そのプラグインは、システムに対するセキュア接続をサポートしていることを示さなければなりません。 C++ プラグインの場合は、cwbCO_SysHandle は cwbUN_GetSystemHandle API を呼び出すことにより取得されます。ユーザーがセキュア接続を要求している場合、System i ナビゲーターは自動的に SSL を有効にします。 Java プラグインの場合、 com.ibm.as400.opnav.ObjectName クラスの getSystemObject メソッドを呼び出して取得した System i オブジェクトは、実際には com.ibm.as400.access.SecureAS400 のインスタンスとなります。

注: SSL で Java を実行し、独自の CA 証明書を作成している場合は、System i Access for Windows GA サービス・パックが必要になります。

プラグインとシステムの間の通信を、ソケット API などといった低レベルの通信サービスを使用して行う場合、SSL が要求されたときに SSL をサポートするのは、プラグインの担当となります。これをサポートしていないプラグインでは、以下の例にある方法で、SSL をサポートしていないことを示します。これにより、ユーザーがセキュア接続を要求しても、プラグインの機能は使用不可になっています。

例: SSL を使用可能にするためのレジストリー・キーを追加する

キーは、`[HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.Sample¥SSL] "Support Level"=dword:00000001` の下の SSL です。ここで、`IBM.Sample` はプラグインが提供するプロダクト・コンポーネントです。

注: `"Support Level"=dword:00000001` は、SSL をサポートしています。`"Support Level"=dword:00000000` は、SSL をサポートしていません。

```
 ;-----------------------------------------------------------------------------

; このプラグインが SSL をサポートしていることを示している
  レジストリー・キーの例
  {HKEY_CLASSES_ROOT¥IBM.AS400.Network¥3RD PARTY EXTENSIONS¥IBM.Sample¥SSL}
  "Support Level"=dword:00000001
```

コードに関するライセンス情報および特記事項

IBM は、お客様に、すべてのプログラム・コードのサンプルを使用することができる非独占的な著作使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

強行法規で除外を禁止されている場合を除き、IBM、そのプログラム開発者、および供給者は「プログラム」および「プログラム」に対する技術的サポートがある場合にはその技術的サポートについて、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。

いかなる場合においても、IBM および IBM のサプライヤーならびに IBM ビジネス・パートナーは、その予見の有無を問わず発生した以下のものについて賠償責任を負いません。

1. データの喪失、または損傷。
2. 直接損害、特別損害、付随的損害、間接損害、または経済上の結果的損害
3. 逸失した利益、ビジネス上の収益、あるいは節約すべかりし費用

国または地域によっては、法律の強行規定により、上記の責任の制限が適用されない場合があります。

付録. 特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービス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、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。 IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

本プログラムに関する上記の情報は、適切な使用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。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、IBM 機械コードのご使用条件、またはそれと同等の条項に基づいて、 IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次のように、著作権表示を入れていただく必要があります。

© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。 © Copyright IBM Corp. _年を入れる_. All rights reserved.

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は表示されない場合があります。

プログラミング・インターフェース情報

この「System i ナビゲーター プラグインの開発」資料には、プログラムを作成するユーザーが IBM i5/OS のサービスを使用するためのプログラミング・インターフェースが記述されています。

商標

以下は、International Business Machines Corporation の米国およびその他の国における商標です。

i5/OS
IBM
IBM (ロゴ)
iSeries
System i

Adobe、Adobe ロゴ、PostScript、PostScript ロゴは、Adobe Systems Incorporated の米国およびその他の国における登録商標または商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

使用条件

これらの資料は、以下の条件に同意していただける場合に限りご使用いただけます。

個人使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用: これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。 ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態で提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

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