IBM Content Manager for iSeries

アプリケーション・プログラミングのガイドとリファレンス

バージョン 5 リリース 3

SC88-4007-01
(英文原典：SC27-1139-01)

IBM Content Manager for iSeries

アプリケーション・プログラミングのガイドとリファレンス

バージョン 5 リリース 3

SC88-4007-01
(英文原典：SC27-1139-01)

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、 337 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、IBM Content Manager for iSeries (プロダクト番号 5722-VI1) のバージョン 5、リリース 3 に適用されます。また、改訂版で断りがない限り、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。本書は SC88-4007-00 の改訂版です。

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

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

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

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

(URL は、変更になる場合があります)

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： SC27-1139-01
IBM Content Manager for iSeries
Application Programming Guide and Reference
Version 5 Release 3

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 2004.4

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

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

目次

本書について

本書では、イメージ、ワークフロー、およびその他のアプリケーションを作成して、これらを Content Manager for iSeries システムに組み込む方法を説明します。これらのアプリケーション・プログラミング・インターフェース (API) は、Content Manager for iSeries のクライアント・アプリケーション開発をサポートします。本書は、32 ビット Windows® プログラミング環境でのアプリケーション開発を対象としています。

本書では、以下のトピックについて説明します。

- Content Manager for iSeries の各種構成要素の使用法
- Content Manager for iSeries アプリケーションを作成する際に、アプリケーションの要件を判別するためのヒント
- API を使用して、Content Manager for iSeries API を使用するイメージ、ワークフロー、あるいはその他のアプリケーションを作成する方法
- Content Manager for iSeries で用いられる用語

本書の対象読者

読者が、イメージ、ワークフロー、またはその他のアプリケーションの開発を担当しているアプリケーション・プログラマーである場合には、本書を読むことによって、API を通じて利用可能な各機能について詳細な情報を入手できます。

読者が、Content Manager for iSeries システムやアプリケーションの設計を行うシステム設計者またはシステム・インテグレーターである場合には、Content Manager for iSeries の動作方法や、Content Manager for iSeries の新規のアプリケーションを作成する方法、または Content Manager for iSeries に既存のアプリケーションを統合する方法について理解する必要があります。本書は、各構成要素および対応する機能が、イメージ処理、ワークフロー、またはその他のアプリケーションに対する技術的要件、設計的要件、およびビジネス要件をどのように満たすかについて、説明しています。

Content Manager for iSeries のインプリメンテーションの運用管理およびサポートを担当するシステム管理者は、本書を参照資料として使用することができます。

Content Manager for iSeries でサーバー側プログラミングを適切に行なうには、C、COBOL、または RPG でのアプリケーション開発経験と、OS/400® 環境の経験が必要です。クライアント側プログラミングを適切に行なうには、OLE、VisualBasic、C++、C などでのアプリケーション開発経験と、 Windows 環境の経験が必要です。

本書の構成

本書には、以下の情報が収められています。

- 1 ページの『第 1 章 Content Manager for iSeries の紹介』は、Content Manager for iSeries のソフトウェアおよびハードウェア構成要素、および Content Manager for iSeries で使用可能な API について紹介しています。
- 5 ページの『第 2 章 Content Manager for iSeries の概念』は、Content Manager for iSeries の概念および機能について紹介しています。
- 193 ページの『第 5 章 OLE オートメーション・インターフェースの使用』は、Windows ベースの別のアプリケーションを Content Manager for iSeries にログオンさせ、OLE 2.0 Automation に基づく API を使用して Windows クライアントの中でさまざまなタスクを実行させる方法について説明しています。
- 227 ページの『Visual Basic 用高水準プログラミング・インターフェースのサンプル』は、Windows ベースの別のアプリケーションを Content Manager for iSeries にログオンさせ、OLE 2.0 Automation に基づく API を使用して Windows クライアントの中でさまざまなタスクを実行させる方法について説明しています。
- 11 ページの『第 3 章 アプリケーション・プログラミング・インターフェース』では、Content Manager for iSeries 共通アプリケーション・プログラミング・インターフェースについて説明します。
- 151 ページの『第 4 章 共通データ構造』では、オブジェクトやオブジェクトのクラスを操作し管理する際に使用できる、共通データ構造と共通データベース・テーブルについて説明します。
- 198 ページの『Windows 用 OLE オブジェクトのプロパティーおよびメソッド』では、すべてのクライアント・アプリケーション・オブジェクトに関連するプロパティーおよびメソッドについて説明します。
- 227 ページの『第 6 章 高水準プログラミング・インターフェースのサンプル』では、Windows 用のハイレベル API のサンプルを説明します。
- 271 ページの『第 7 章 Content Manager for iSeries プログラミング・インターフェースのサーバー API』では、Content Manager for iSeries サーバー版の API について説明します。
- 275 ページの『第 8 章 Content Manager for iSeries ユーザー出口』では、Content Manager for iSeries ユーザー出口について説明します。
- 325 ページの『付録 A. 検索式の指針』は、Windows クライアントを検索する際に従うべき指針を示しています。
- 329 ページの『付録 B. 事前定義済みコンテンツ・クラス』に、Content Manager for iSeries の事前定義済みコンテンツ・クラスをリストします。
- 333 ページの『付録 C. 外部参照』では、 Content Manager for iSeries Windows クライアントとプログラミング・インターフェースを使用して他のリポジトリー内のデータにアクセスする方法について説明しています。

バージョン 5.3 の新機能

本書「*IBM® Content Manager OnDemand for iSeries™ アプリケーション・プログラミングのガイドとリファレンス*」には新しい技術情報が収録されています。変更が行われたにもかかわらず、変更バーが付いていない箇所が存在する可能性があります。注意が必要な大きな変更点は、以下のとおりです。

10 文字のユーザー ID を保管できるように、機能が拡張されました。以前のリリースでは、ユーザー ID の最初の 8 文字のみが使用されていました。**重要:** 数多くのファイルが、10 文字のユーザー ID をサポートするように変更されました。外部参照をサポートして、EKD0314 ファイルの読み取り、またはこのファイルへの書き込みを行う場合は、そのファイル形式でユーザー ID フィールドの拡張をサポートするように、ご使用のカスタム・プログラムを再コンパイルする必要があります。

本書の使用方法

Content Manager for iSeries について理解を深めるには、1 ページの『第 1 章 Content Manager for iSeries の紹介』を参照してください。Content Manager for iSeries 構成要素の使用方法に関する概念の説明については、5 ページの『第 2 章 Content Manager for iSeries の概念』を参照してください。

表記に関する規則

テキストを理解しやすくするため、本書では以下の表記法を用いています。

表記規則	示す内容
大文字と小文字	ライブラリー・サーバー・データベース・テーブル中の列名 (例: Owner UserID)
大文字	オブジェクト・サーバー・データベース・テーブル中の列名 定数 データ構造名 データ型 データベース・テーブル名 関数呼び出しからの戻りコード
太字の大文字小文字混合	API 関数名 (例: **SimLibLogon**)
太字の大文字	指定するフィールド値 指定するパラメーター値
イタリックの大文字	フィールドの最大長
イタリック	データ構造内のフィールド名 参照資料名 API 関数のパラメーター名 本書中で初めて定義される用語

前提条件および関連情報

iSeries の技術的な情報については、まず、iSeries Information Center を参照してください。iSeries Information Center は、以下からご覧になれます。

- Web サイト : `http://www.ibm.com/eserver/iseries/infocenter`
- Content Manager for iSeries に同梱されている CD-ROM

  「*iSeries Information Center*」(SK88-8055-03)。このパッケージには、「*iSeries Information Center : 補足資料*」 (SK88-8056-01) に記載されている Content Manager for iSeries 資料の PDF 版も含まれています。これは、従来のソフトコピー・ライブラリー CD-ROM に代わるものです。

IBM iSeries Information Center には、CL コマンド、システム・アプリケーション・プログラミング・インターフェース (API)、論理区画、クラスター化、Java™、TCP/IP、Web サービス提供、および保護されたネットワークなどに関するアドバイスや重要なトピックが含まれています。また、関連する IBM Redbooks™ へのリンク、および Technical Studio や IBM ホーム・ページなどの他の IBM Web サイトへのインターネット・リンクも含まれています。

http://www-3.ibm.com/software/data/cm/cmgr/400/library.html を参照して、製品 Web サイトから Content Manager for iSeries 資料のページにアクセスしてください。資料のリストを、表 1 に示します。

表 1. IBM Content Manager for iSeries 5.3 の資料

資料名	資料番号
IBM Content Manager for iSeries 計画とインストール	SC88-4001
IBM Content Manager for iSeries Windows クライアント・スタートアップ・ガイド	GC88-4003
IBM Content Manager for iSeries システム管理ガイド	SC88-4004
IBM Content Manager for iSeries メッセージとコード	SC88-4005
IBM Content Manager for iSeries 拡張ワークフロー・ガイド	SC88-4006
IBM Content Manager for iSeries アプリケーション・プログラミングのガイドとリファレンス	SC88-4007

Web で入手可能なサポート

製品サポートは、IBM サポート (http://www-6.ibm.com/jp/software/data/support/) からご利用いただけます。

iSeries ナビゲーター

IBM iSeries ナビゲーターは、iSeries サーバーを管理するための、強力なグラフィカル・インターフェースです。iSeries ナビゲーターは、ユーザーのタスクをガイドするシステム・ナビゲーション、構成、計画、およびオンライン・ヘルプなどの機能を備えています。 iSeries ナビゲーターは、サーバーをより容易かつ生産的に運用

および管理するためのもので、 OS/400 オペレーティング・システムの新しい拡張機能を利用するための、唯一のユーザー・インターフェースとなっています。iSeries ナビゲーターには、中央のサーバーから複数のサーバーを管理するための Management Central も含まれています。

iSeries ナビゲーターについて詳しくは、Information Center を参照してください。

WWW で入手可能な情報

WWW では、iSeries のより詳しい情報をご覧になることができます。iSeries 全般的な情報については以下の Web サイトから入手することができます。

`http://www.ibm.com/jp/servers/eserver/iseries/`

iSeries の高度な機能をテーマにしたワークショップへアクセスするには、以下のアドレスにある Technical Studio を利用してください。

`http://www.iseries.ibm.com/tstudio`

第 1 章 Content Manager for iSeries の紹介

この章では、Content Manager for iSeries 構成要素をインプリメントする方法の概要について説明します。この情報は、アプリケーションを作成する際、Content Manager for iSeries API を最大限に活用する方法を決めるためのフレームワークになります。以下の Content Manager for iSeries 構成要素の概要も説明します。

クライアント・アプリケーション・プログラム
ユーザーが使用できるクライアント・アプリケーションには、Content Manager for iSeries とともに提供されるクライアント・アプリケーション・プログラムと、ユーザーが開発するアプリケーションがあります。

Content Manager for iSeries API
Content Manager for iSeries API は、ホスト・サーバーに格納されたデータをアクセスおよび操作できるようにする高水準プログラミング・インターフェースです。

クライアント・インターフェース (Windows 版)
Windows 用クライアント API は、Content Manager for iSeries に独自の Windows ベースのクライアント・アプリケーションを開発するために使用できるプログラミング・インターフェースを提供します。

Content Manager for iSeries を使用すると、複数のメディア・タイプ用の、ホスト・サーバーと情報処理機能を含む、カスタマイズされた文書管理ソリューションを開発することができます。Content Manager for iSeries により、イメージおよびその他のアプリケーションを作成し、企業が毎日処理する情報を自動化し、制御することが可能になります。Content Manager for iSeries により、生産性とセキュリティーを向上させ、ストレージ・コストを下げ、さらに、顧客サービスを改善することができます。

Content Manager for iSeries が提供する文書処理機能は、大規模な組織にも、小規模な部門にも適用できます。Content Manager for iSeries は、オンラインで文書を取り込み、保管し、検索することを可能にし、さらに、文書管理機能、フォルダー管理機能、およびワーク・マネージメント機能を提供しています。また、Content Manager for iSeries は、広範なデータ保全性とセキュリティー機能を備えています。

Content Manager for iSeries システムは、iSeries サーバーに接続された Windows クライアントによって構成されます。これにより、文書の処理、ストレージ、および管理の各機能への全社規模でのアクセスが可能になりました。つまり、Content Manager for iSeries により、1 つの企業の複数の部門から、部門レベルの文書はもちろん、企業全体で管理する文書にアクセスできるようになります。しかも、各部門は複数の場所に散在していてもかまいません。

Content Manager for iSeries の詳細

Content Manager for iSeries のクライアント/サーバー体系により、高度な文書管理システムが提供されます。クライアント/サーバーの概念を理解すると、Content Manager for iSeries を構成するキー構成要素のすべての詳細が分かります。

クライアント/サーバーの関係

Content Manager for iSeries は、1 つ以上のホスト・サーバーに接続しているクライアントで構成されています。ホスト・サーバーは、文書やフォルダーの索引情報、文書とフォルダーの関係、処理中作業の情報を保持しており、クライアントと対話を行います。

Content Manager for iSeries 構成要素

Content Manager for iSeries は、クライアント、クライアント・アプリケーション・プログラム、ホスト・サーバー、および Content Manager for iSeries API で構成されています。Content Manager for iSeries を使用して、追加のクライアントを開発することができます。

次の図は、Content Manager for iSeries の主要な構成要素を示したものです。

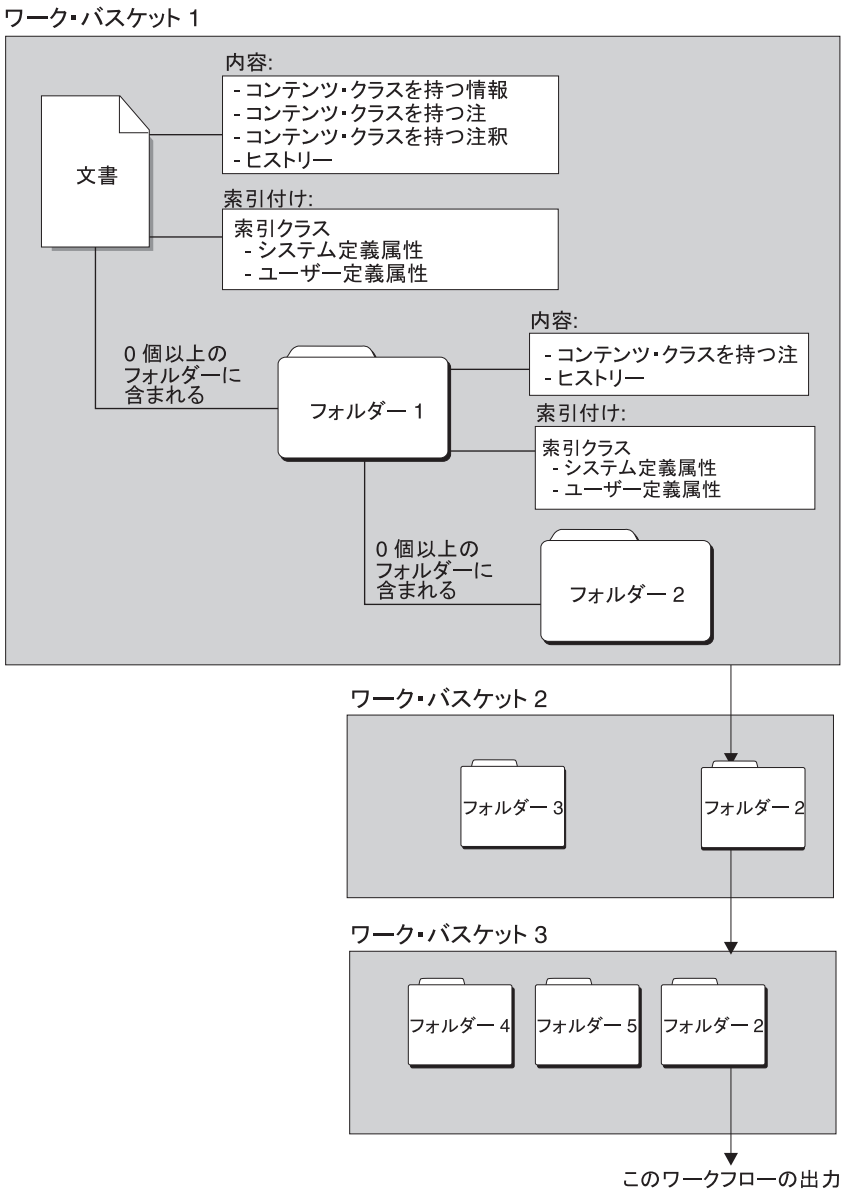

図 1. Content Manager for iSeries の主要な構成要素

クライアント・アプリケーション

Content Manager for iSeries クライアント・アプリケーションは、Content Manager for iSeries API に基づいて構築された、文書およびフォルダーの管理機能、スキャン・サポート機能、インポートおよびエクスポート機能、ワーク・マネージメント機能、および検索機能を提供します。

クライアント・アプリケーション・プログラムは、Content Manager for iSeries に対する完全なエンド・ユーザー・インターフェースを提供します。したがって、企業の具体的なニーズに合わせてクライアント・アプリケーション・プログラムを構成することができます。ユーザー出口は、クライアント・アプリケーション・プログラムをカスタマイズするためのアプリケーション特有の処理ルーチンを供給できるポイントを提供します。

クライアント・アプリケーション・プログラムは、フォルダー管理、ワーク・マネージメント、および文書管理を既存の情報システムに統合するための API を備えています。カスタム・ソフトウェアやその他のアプリケーションを、簡単にクライアント・アプリケーション・プログラムに組み込むことができます。

Content Manager for iSeries とともに提供されるクライアント・アプリケーション・プログラムを使用することも、独自のアプリケーションを作成することもでき、また、 IBM サービスやビジネス・パートナーが提供するアプリケーションを使用することもできます。

Content Manager for iSeries API

独自のアプリケーションを作成する場合には、Content Manager for iSeries ホスト・サーバーとアプリケーションとの間の基本インターフェースとして Content Manager for iSeries API を使用することができます。

Content Manager for iSeries データ・モデルにおいて、最も基本的な構成要素は文書、フォルダー、ワーク・バスケット、および作業パッケージです。*文書* は、紙の文書と同様です。*フォルダー* は、紙のファイリング・システムにおけるフォルダーと類似していて、別のフォルダーまたは文書を入れることができます。*ワーク・バスケット* は、1 名または複数の従業員が使用するように定義した、作業のキューです。これは、作業を取り出すインバスケットとほぼ同じです。*作業パッケージ* は、ワーク・マネージメントで使用するワーク・バスケットの中の 1 つの項目であり、これには文書またはフォルダーが入っています。

文書へのアクセス・レベルに応じて、これらの API を使用して、以下の操作を実行することができます。
- 文書の格納
- 文書あるいはフォルダーの索引付け
- 文書あるいはフォルダーの検索

API は、Content Manager for iSeries で使用可能な広範囲の関数をサポートします。これらの API を使用すると、Windows または OS/400 用のアプリケーションを作成することができます。

Content Manager for iSeries サーバー

Content Manager for iSeries サーバーは IBM のリレーショナル・データベース・テクノロジーを使用して、文書の内容を保守します。また、以下の機能を実行してデータ保全性を提供します。
- データの管理
- 索引情報の保守
- オブジェクト・サーバーに格納されている文書へのアクセス制御

複数の Content Manager for iSeries サーバーを参照するアプリケーションを開発することができます。

第 2 章 Content Manager for iSeries の概念

この章では、Content Manager for iSeries の概念 (論理データ・モデルを含む) について説明します。IBM Content Manager for iSeries ファミリーの他の製品では、『フォルダー・マネージャー・データ・モデル』 という用語はアプリケーション・プログラミング・インターフェース (API) のサブセットを示し、『共通アプリケーション・プログラミング・インターフェース』 (CAPI) という用語は *SimLib* インターフェースのサブセットを示します。Content Manager for iSeries では、すべての使用可能なプログラミング・インターフェースは Content Manager for iSeries API と呼びます。

論理データ・モデルの理解

Content Manager for iSeries は、項目、オブジェクト、フォルダー、索引クラス、および属性などの概念を含むフォルダー・マネージャー・データ・モデルをインプリメントしています。このデータ・モデルは、アプリケーションに、ビジネス・オブジェクトを管理するための多くの機能を提供します。Content Manager for iSeries 内の文書は、紙文書に類似しています。文書は、レターまたは報告書内のページのように、一連の密接に関連したオブジェクトから構成されます。文書は、1 つ以上のパーツからなります。これらの部分は、基本部分と呼ばれ、レター、報告書、またはその他の文書内のページまたは図で構成されます。文書に関連するその他の部分としては、注釈と注があります。

文書に関連した注釈の部分は、文書の節を強調表示することができます。文書に関連した注の部分は、他のユーザーに追加の情報を提供するために文書に付加するテキスト情報です。たとえば、読者の注意を引くために文書の一部に注を付加することがあります。文書に関連したイベント部分には、文書について実行した処理のヒストリーが記述されます。

Content Manager for iSeries におけるフォルダーは、紙のファイリング・システムのフォルダーに類似しています。各フォルダーには、1 つ以上の文書あるいは他のフォルダーを入れることができます。各フォルダーには、フォルダー内に含むすべての文書およびフォルダーをリストした目次が入っています。注の部分をフォルダーに関連付けることができます。

ワークフロー・ガイド

ワークフローは、作業の流れ、および作業の処理を説明したものです。ワークフロー およびワーク・マネージメント という用語は、同じ意味で使用されています。ワークフローは、作業の実行方法を管理するための定義および規則です。

以下の用語は、ワークフローの説明でよく使用されるものです。

アクション・リスト
スーパーバイザーによって定義された処置の承認リストであり、ユーザーがこれらの処置を作業パッケージに対して実行することができる。

特別処理　定義されたワークフロー処理ではない処理のこと。*特別処理* が開始されるのは、ユーザーが作業パッケージを作成し、それをワーク・バスケットに直接割り当てるときである。ユーザーは手操作で作業パッケージを割り当てなおすことにより、作業パッケージをあるワーク・バスケットから別のワーク・バスケットに経路を定める。ワークフロー処理において、値 *ADHOC を処理名の変わりに使用すると、作業パッケージが特別の方法で経路指定される。

コレクション・ポイント
作業パッケージが処理を続けるために特定のイベントが発生するかまたは同期をとるのを待つポイントのこと。

コレクション・ポイントは、プロセスの一部である。たとえば、「新規口座の開設」というプロセスの一部である作業パッケージの場合、コレクション・ポイントで、信用情報が検査されるまで待つ必要がある。

決定ポイント　各作業パッケージ内の特定の情報に応じて、作業パッケージが現行経路上で処理を続けるか、代替経路に切り替えるポイントのこと。決定ポイントは、変数名、値、および経路から構成されるテーブルである。

決定ポイントは、プロセスの一部である。たとえば、決定ポイントは、「新規口座の開設」の一部である作業パッケージが、信用情報に基づいて承認または否認を受け取る場合のポイント。

インスタンス　プロセス内での作業パッケージのオカレンス。このプロセスが複数の並列経路から構成されている場合には、1 つの作業パッケージについて複数のインスタンスが存在する。

プロセス　作業パッケージが順次流される一連のステップ、イベント、および規則。プロセスは、定義済みタイプの作業パッケージが流れる過程で通過する経路、コレクション・ポイント、および決定ポイントの組み合わせである。

例えば、「新規口座の開設」というプロセスの内容は、以下のようになる。

- 新規口座の開設に関連する作業パッケージが従う必要のあるステップ
- 新規口座の作業パッケージをシステム内の別のポイントへ経路指定するときの条件となるイベント (信用情報の検査など)
- その特定の顧客に関する情報 (信用格付けが高いか低いかなど) に基づいて、新規口座を開設するかどうかを決定する判断

中断　規定の基準が満たされるまで、作業パッケージをワーク・バスケットに保持しておくこと。作業パッケージを中断する基準は複数指定できるため、1 つの作業パッケージについて複数の中断要求が存在する場合もある。文書の作業パッケージは、特定の日付により中断することができる。フォルダーの作業パッケージは、特定の日付または索引クラスを条件に指定して中断することができる。

中断した作業パッケージがリリースされるのは、特定の条件が満たされたとき、あるいは適切な権限を持つユーザーが条件を変更して、中断要求を手操作でリリースしたときである。

作業パッケージ

ある場所から別の場所に経路指定される作業。作業パッケージは、文書、フォルダー、または顧客が定義した一連のオブジェクトから構成できる。作業パッケージは、定義済みのプロセスによって自動的に経路指定することができる。また、ユーザーが手操作で、特別な方法で、指定したワーク・バスケットに作業パッケージを経路指定することもできる。ユーザーは、ワーク・バスケットを介して、作業パッケージをアクセスおよび処理する。

ワーク・バスケット

作業パッケージを保持するコンテナー。ワーク・バスケットは、プロセス定義および随時経路の一部として使用できる。ワーク・バスケットの定義には、その作業パッケージ内容の表示、状況、およびセキュリティーを管理するルールが含まれている。

文書およびフォルダーについての情報の入手

文書またはフォルダーの属性を読み取るために、アプリケーションは項目をオープンし (**SimLibOpenItemAttr**)、一度に属性を 1 つ読み取り (**SimLibReadAttr**)、項目をクローズする **(SimLibCloseAttr)** ことができます。また、**SimLibGetItemSnapshot** を使用して、すべての属性とオプショナルの情報を検索することができます。この関数は、システム属性、ユーザー定義属性、ワークフロー情報、チェックアウト・ホルダー、およびフォルダーまたは文書についてのその他のデータを検索します。このすべての情報が必要で、後続の処理のために項目をオープンする必要がない場合は、この関数を使用してください。

SimLibSearch を使用して、事前に定義された検索基準と一致する項目のユーザー定義属性を検索することができます。

スナップショット・オプション・フラグにシステム属性 (SIM_SYSTEM_ATTR) がある場合、**SimLibGetItemSnapshot** は、ユーザー定義属性に加えて現行表示用の ATTRLISTSTRUCT 配列に次の 4 つの属性を戻します。

- OIM_ID_ITEM_NAME
- OIM_ID_CREATE_TIMESTAMP
- OIM_ID_MODSYS_TIMESTAMP
- OIM_ID_UID

アプリケーションは、属性が現れる順序、あるいは、ユーザー定義またはシステム定義のどちらが先に来るかの順序に依存する必要はありません。

SimLibGetItemSnapshot の代わりに、**SimLibGetTOCData** を使用して、項目の全リストのためのスナップショットを戻します。**SimLibGetTOC** によって戻される TOCENTRYSTRUCT 配列は、項目数が SIM_TOC_MAX_ENTRY_COUNT を超えていない場合、直接 **SimLibGetTOCData** に渡してグループとして処理することができます。カウントが最大値を超える場合は、項目は、一度に 1 つずつ、最大値にな

るまで渡されます。次に、TOCENTRYSTRUCT 配列にある次のバッチに進みます。**SimLibGetTOCData** へのリスト・ポインターは、配列内の項目を参照します。関数は、この項目で処理を開始します。

たとえば、アプリケーションは、以下に示すような基本ロジックを持つことができます。

```
ulRC = SimLibGetTOC(hSession,...);
if (ulRC != SIM_RC_OK) {
   // process errors
} else {
   ulCount = count returned by SimLibGetTOC
   pTOC = TOCENTRYSTRUCT array pointer returned by SimLibGetTOC
   while (ulCount > 0) {
       i = minimum of ulCount and SIM_MAX_TOC_ENTRY_COUNT
       ulRC = SimLibGetTOCData(hSession,pTOC,i,NULL,pRC);
       if (ulRC != SIM_RC_OK) {
          // process errors, possibly exit the loop
       } else {
          // process results
          call SimLibFree to release data returned
       }
       ulCount -= i;  // decrement number left to do
       pTOC += i;     // advance to next set, if any
   }
   close the TOC from SimLibGetTOC
}
```

ログオンしている場合、各項目ごとの属性を取得するための十分な特権を持っている必要があります。持っていない場合は、**SimLibGetTOC** 関数がエラーを戻します。

SimLibGetTOC から項目の完全なセットを処理せずに、**SimLibGetTOCData** の効率を引き続き利用することができます。**SimLibGetTOCData** は、NULL ストリングである TOCENTRYSTRUCT の項目 ID をスキップします。アプリケーションが **SimLibGetTOC** 関数によって戻された TOCENTRYSTRUCT 配列を修正しないため、TOCENTRYSTRUCT 配列を別のバッファーにコピーしてから項目 ID を NULL に設定します。また、必要なデータを一時的な TOCENTRYSTRUCT 配列にコピーして、それを **SimLibGetTOCData** に渡すことによって不要な項目をフィルターに掛けることができます。項目 ID が NULL の場合でも、**SimLibGetTOCData** はその項目について空の SNAPSHOTSTRUCT を戻します。

SimLibGetTOC によって戻されなかった場合であっても、項目のブロックの処理について同じアプローチを使用することができます。アプリケーションは、同じ形式のアプリケーション独自のリストを生成し、そのリストを **SimLibGetTOCData** に渡すことができます。例として、検索 (**SimLibSearch**) の結果を用いて、項目 ID リストから TOCENTRYSTRUCT 配列を作成します。**SimLibGetTOCData** は、事前に各項目の索引クラスを必要とします。**SimLibSearch** は索引クラスを戻しませんが、検索を単一索引クラスに限定する場合、アプリケーションは検索によって戻された各項目の索引クラスを既に認識しています。

また、SIM_SEARCH_USER_ATTR オプションまたは SIM_SEARCH_USER_SYSTEM_ATTR オプションを使用することにより、**SimLibSearch** を使ってユーザー定義属性、あるいはユーザー定義属性とシステム定義属性の両方を直接検索することもできます。これは、**SimLibSearch** を呼び出

して項目 ID を取得し、続いて **SimLibGetTOCData** などの他の API を呼び出して属性情報を検索するよりも効率的です。

SimLibGetTOC からの配列のように見える TOCENTRYSTRUCT 配列を作成しても、シミュレートされた TOC 上の **Ip2TOCUpdates** などの目次関数を使用することはできません。目次関数は、**SimLibGetTOC** によって戻されたハンドルを必要とします。

大文字小文字の区別のサポート

Content Manager for iSeries は、アプリケーションによって提供されたとおり、正確に文字ストリング属性を格納します。Content Manager for iSeries では常にユーザー ID を大文字に変換します。

フォルダーの命名

Content Manager for iSeries 用のフォルダー・データ・モデルには、フォルダー名は含まれていません。顧客名、顧客番号、事例名、その他の認識可能テキストといったフォルダー名は、フォルダー名を使用するクラスの索引クラス属性です。したがって、フォルダーを名前で検索するには、アプリケーションがフォルダー名に関連した索引クラスを認識し、適切な検索を構成する必要があります。

項目の索引クラスの変更

項目を作成すると、その項目は索引クラスに関係づけられます。アプリケーションが項目の索引クラスを変更すると、この項目は更新され、変更部分が反映されます。この項目には、常に、項目が属する現行の索引クラスが含まれます。多くの Content Manager for iSeries の API (**SimLibGetItemInfo** と **SimLibGetItemSnapshot** を含む) は、アプリケーションにこの情報を戻します。アプリケーションではこの索引クラスを使用します。

項目へのアクセスの制限

アクセス制御には、ユーザーに対して定義される特権、およびアクセス・リストに対して定義される特権という、2 つの層があります。通常、ユーザー特権は、一般特権と呼ばれます。アクセス・リストは、索引クラス、ワーク・バスケット、およびプロセスへのアクセスを確立するために使用されます。アクセス・リストは、ユーザー・リストと一連の特権を組み合わせたものです。アクセス・リストは、一般特権に対して権限を**追加**するものであり、権限の除去を行うものではありません。

権限制御の最も簡単な例では、すべてのユーザーがライブラリーのすべての項目のアクセス権を持っています。このタイプの権限制御を実行するには、すべてのユーザーに最大限の権限を与えます。アクセス・リストは権限を追加するので、この例では、索引クラス、ワーク・バスケット、またはプロセスにアクセス・リストをインプリメントする必要はありません。ただし、制限されたアクセスの中には多くの使用可能レベルがあります。

制限の 1 つとして、一部のユーザーにしか、特定のフォルダーおよび文書へのアクセスを許可しないというものがあります。これを行なうには、まず、項目の索引ク

ラスへのアクセスを最小限に指定して、すべてのユーザーに一般特権を定義します。次に、索引クラスを使用できるユーザーおよびグループからなるリストを定義します。このユーザー・リストは、索引クラスの機能を許可する特権セットに関連付けられます。

ユーザー・リストと特別な特権設定を組み合わせることでアクセス・リストが生成され、これが索引クラスに使用されます。このようにして、アクセス・リストに含まれないユーザーは索引クラスの使用が拒否され、アクセス・リストに含まれるユーザーは、特権セットに指定されている機能を実行することができます。

SimLibLogon は、一般特権を戻します。**Ip2QueryClassPriv** は、索引クラスに対する権限を戻します。同様に、**SimWmGetWorkBasketInfo** および **SimWmGetProcessInfo** は、ワーク・バスケットまたはプロセスに対する特権を戻します。アプリケーションはこれらの特権ストリングを使用して、ユーザーに対して特定の関数オプションを提供するかどうかを事前に設定します。たとえば、アプリケーションにより、削除オプションを提供しなくても、ユーザーに削除権限のない項目を、ユーザーが表示できるようにすることができます。

オブジェクトのマイグレーション

Content Manager for iSeries ストレージ管理機能を使用すると、管理者が設定する制御に基づいて、オブジェクトをあるメディアから別のメディアに (たとえば、磁気ディスクから光ディスク記憶に) 移動することができます。コレクション名は、システム内で作成された各オブジェクトに割り当てられます。コレクションは、一般的に同様の性能、使用可能度、バックアップ、および保存の特性を持つオブジェクト・グループに関連付けられているストレージ管理制御を定義するためのものです。アプリケーションは、 **SimLibChangeObjectSMS** API を使用して、オブジェクトを別のコレクションに割り当てることができます。

第 3 章 アプリケーション・プログラミング・インターフェース

この章では、Content Manager for iSeries アプリケーション・プログラミング・インターフェース (API) の形式とパラメーターについて説明します。これらの API は **SimLib**、**SimWm**、**Sim400**、および **Ip2** の各接頭部で認識することができます。

これらの API のデータ構造の詳細については、151 ページの『第 4 章 共通データ構造』を参照してください。

Content Manager for iSeries アプリケーションのコンパイルおよびリンク

Content Manager for iSeries には、Content Manager for iSeries API を介してアクセスすることができます。Content Manager for iSeries にアクセスするアプリケーションを作成および実行するためには、次のファイルが必要です。

EKDVIAPI.H Content Manager for iSeries API の構造、マクロ、および関数のプロトタイプ。EKDVIAPI.H には、次のヘッダー・ファイルが含まれています。

- **EKDVIERR.H** エラー番号および記述名。検出されるすべてのエラーの名前が Content Manager for iSeries に記録されます。
- **EKDVILIB.H** ライブラリー API の定義。
- **EKDVITYP.H** 定数および共通の型定義。
- **EKDVIWM.H** ワークフロー API プロトタイプ。

EKDWS.LIB EKDWS.DLL とのリンクに必要な LIB ファイル。

EKDWS.DLL すべての API 関数。

EKDWS35I.DLL IBM VisualAge ランタイム DLL。

これらのファイルは、IBM Content Manager for iSeries Windows Client Toolkit のインストール時にインストールされます。

アプリケーションは、次のように、ヘッダーにアクセスする必要があります。

```
#include "EKDVIAPI.H"
```

VisualAge を使用していない場合には、ILIB コマンドまたは同等のコマンドを使用して LIB ファイルを再生成する必要があります。

Content Manager for iSeries API は、VisualAge からのコード・ページ変換テーブルを使用します。インストール・プログラムは、特定のインストール・システムに対して使用するコード・ページの必須ファイルをインストールする必要があります。コード・ページ変換ファイルは、FRNROOT¥ICONV および FRNROOT¥UCONVTAB ディレクトリーに入っています。

LOCPATH 環境変数を上記のディレクトリー (FRNROOT) に設定する必要があります。AUTOEXEC.BAT またはレジストリーでこの設定を行うか、あるいはアプリケーションで **SimLibLogon** の呼び出し前にこの設定を行うことができます。この設定により、変数が常に設定されるようになります。これにより、他のプロダクトとの競合が防止されます。

問題判別を支援するために、クライアントのトレースおよびロギングを使用可能にすることができます。下記の環境変数は、トレースを制御するために任意の値に設定することができます。結果は、VI400_LOG_PATH 環境変数に指定されている作業ディレクトリーまたはパスの VI400.LOG ファイルに記録されます。このファイルは、最初の呼び出し (たとえば、**SimLibLogon** への呼び出し) がなされたときに上書きされます。

VI400_LOG_PATH
VI400.LOG のパス

VI400_LOG_TRACE
関数の入り口および出口

VI400_LOG_PERFORMANCE
トレースおよびデータ伝送の時間

VI400_LOG_DATA
iSeries システムとの間で送受信されるデータ

VI400_LOG_STORAGE
Content Manager for iSeries オブジェクト・ストレージの割り振りおよび割り振り解除

VI400_LOG_LOCKS
API ごとのログのロックおよびアンロック操作

VI400_LOG_ALL
すべてのトレース・レベル

FRNOLINT.TBL ファイルは、Content Manager for iSeries サーバーを定義する項目を入れるために使用されます。このファイルは、プログラムが開始されたパス、または VI400_CONFIG_PATH 環境変数に含まれているパスの中に入っていなければなりません。以下に、APPC および TCP/IP の例を示します。

```
SERVER: MYVI400 REMOTE APPC
        LU_NAME     = USIBMNR.AS400DS1
        TP          = EKDCS01P.EKDCS01P.QVI
        MODE        = QPCSUPP
        SERVER_TYPE = FRNLS400
SERVER: MYVI400  REMOTE TCPIP
        HOSTNAME     AS400DS1
        PORT         31098
        SERVER_TYPE = FRNLS400
```

この例では、**SimLibLogon** に渡されたデータベース名が MYVI400 である場合には、上記の項目は iSeries システムへの接続に使用されることになります。VI400_CONFIG_PATH 環境変数のパスは、FRNOLINT.TBL にアクセスするため、ネットワーク・ドライブ上、あるいは Client Access かその同等のプロダクトを介してアクセスされる iSeries 上のディレクトリー内に置くことができます。この環境変

数が設定されない場合には、このファイルは現行ディレクトリー (すなわち、アイコン用に **Properties** の **Shortcut** ページに指定されている **Start in** ディレクトリー) 内でアクセスされます。

EKDVIERR.H は、VI400_CONFIG_PATH に定義されているパス内に入れてください。このファイルを使用して、Content Manager for iSeries の各戻りコードの記述名を記録します。

アプリケーション・プログラミング・インターフェース

SimLibAddFolderItem (フォルダーへの項目の追加)

形式

SimLibAddFolderItem(*hSession, pszFolderID, pszItemID, pAsyncCtl, pRC* **)**

目的

SimLibAddFolderItem 関数を使用して、文書またはフォルダー項目を既存のフォルダーに追加します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszFolderID PITEMID - 入力

フォルダーの ID。文書またはフォルダー項目を追加したい既存フォルダーの項目 ID を使用します。このフォルダーをオープンする必要はありません。

pszItemID PITEMID - 入力

項目の ID。フォルダーに追加する文書またはフォルダー項目の項目 ID を使用します。この項目が既にフォルダー内に存在しないようにしてください。*pszFolderID* パラメーターに指定したものと同じフォルダーの ID を使用しないようにしてください。あるフォルダーにそのフォルダー自身を追加することはできません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常に完了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_ITEM_OR_FOLDER
- SIM_RC_INVALID_PITEMIDFOLDER_PTR
- SIM_RC_INVALID_PITEMIDFOLDER_VALUE
- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PITEMIDFOLDER_NOT_A_FOLDER
- SIM_RC_PITEM_NOT_FOLDER_OR_DOCUMENT
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

準備:

- フォルダーを作成するには、**SimLibCreateItem** 関数を使用します。
- 文書またはフォルダーは、同時に複数のフォルダー内に存在することができます。
- フォルダーおよびそれに含まれる項目は、それぞれ異なる索引クラスを持つことができます。

制限:

- あるフォルダーにそのフォルダー自身を追加することはできません。
- この関数は、フォルダーの目次の一時コピーを自動的に更新しません。フォルダーの目次の一時コピーを更新するためには、 **lp2GetTOCUpdates** または **lp2GetTOC** 関数を使用する必要があります。

例

```
#include <windows.h>                    /* Main Windows header files     */
#include <sys¥types.h>
#include <stdio.h>                      /* Standard I/O header files     */
#include <stdlib.h>                     /* Standard library header files */
#include <stdarg.h>
#include <stddef.h>
#include <io.h>
#include "ekdviapi.h"                   /* Content Manager for iSeries   */
```

```
main ()
{

  HSESSION     hSession;               /* Product session handle        */
  PITEMID      pszFolderID;            /* ID of the folder              */
  PITEMID      pszItemID;              /* ID of the item to be added    */
  RCSTRUCT     RCStruct;               /* RC data structure             */
  USHORT       sResult;                /* return codes                  */

    /*****************************************************************/
    /*Initialize folderID and itemID                                 */
    /*****************************************************************/
    memset  (pszFolderID, '¥0', DOC_ID_SIZE); /* set to null         */
    strcpy  ((CHAR *)pszFolderID, (CHAR *) "F000000001");

    memset  (pszItemID, '¥0', DOC_ID_SIZE); /* set to null           */
    strcpy  ((CHAR *)pszItemID, (CHAR *) "DA97220AA.AAB");

   /*****************************************************************/
   /* Call SimLibAddFolderItem to place a new document in a folder   */
   /*****************************************************************/

    sResult = SimLibAddFolderItem(
            hSession,                    /* ses'n handle from SimLibLogon */
            pszFolderID,                 /* add item to this folder       */
            pszItemID,                   /* add this item to above folder */
            (PASYNCCTLSTRUCT) NULL,      /* Request SYNCHRONOUS processing*/
            (PRCSTRUCT) &RCStruct        /* Pointer to RC data structure  */
            );

    if (sResult != SIM_RC_OK) {
        printf("Add folder item failed ¥n");
    }
}
```

関連関数

- **SimLibGetTOCData**
- **Ip2GetTOCUpdates**
- **Ip2TOCCount**
- **SimLibGetTOC**
- **SimLibRemoveFolderItem**

SimLibCatalogObject (オブジェクトのカタログ作成)

形式

SimLibCatalogObject(*hSession, hObj, ulConCls, pSMS, pszFullFileName, ulPriority, fCreateControl, ulVersion, lSeqAfterPart, ulAffiliatedType, pAffiliatedData, pAsyncCtl, pRC* **)**

目的

SimLibCatalogObject 関数を使用して、ユーザーが指定したファイルから新しいオブジェクトを作成します。データがメモリー内ではなく、ファイル内に既に存在する場合、この関数を使用してください。

アプリケーションは、以下の一連の Content Manager for iSeries 関数の代りにこの関数を使用することができます。

- **SimLibCreateObject**
- **SimLibOpenObject**
- **SimLibWriteObject**
- **SimLibCloseObject**

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObj HOBJ - 入力

HOBJ データ構造内のオブジェクト・ハンドル・ブロックへのポインター。HOBJ データ構造の詳細については、 161 ページの『HOBJ (記憶オブジェクトの照会のためのハンドル)』を参照してください。「使用の手引き」で、このデータ構造に入力した結果について説明します。

ulConCls ULONG - 入力

オブジェクトのコンテンツ・クラス ID (329 ページの『付録 B. 事前定義済みコンテンツ・クラス』を参照)。このパラメーターの値は、カタログ作成しようとしているオブジェクト内のデータの種類を示します。

未定義のコンテンツ・クラスを指示するには、このパラメーターに値 SIM_CC_UNKNOWN を指定してください。ただし、定義済みのコンテンツ・クラスを使用しない場合には、他のアプリケーションは、Content Manager for iSeries コンテンツ・クラス・サービスを使用して、ユーザーが格納するオブジェクトの内容を操作する方法を判別することはできません。

pSMS PSMS - 入力

オブジェクトのシステム管理ストレージ (SMS) 構造へのポインター。この構造では *szCollectionName* のみを使用します。

pszFullFileName

PSZ - 入力

完全修飾ディレクトリー・パスおよびファイル名へのポインター。

ulPriority USHORT - 入力

サポートされていません。

fCreateControl BITS - 入力

カタログ作成操作の制御オプション・ビットです。有効な値は、次のとおりです。

SIM_CLOSE
要求が完了するとオブジェクトをクローズします。

SIM_OPEN
更新モードでオブジェクトをオープンしたままにしておきます。

ulVersion ULONG - 入力

サポートされていません。

lSeqAfterPart LONG - 入力

サポートされていません。

ulAffiliatedType LONG - 入力

系列下にあるオブジェクトのタイプ。定義済みの値は次のとおりです。

SIM_ANNOTATION
オブジェクトが、フォルダーまたは文書に関する注釈であることを示します。

SIM_BASE
そのオブジェクトが、混合オブジェクト文書コンテンツ・アーキテクチャー (MO:DCA) またはタグ・イメージ・ファイル・フォーマット (TIFF) ファイルのような基本オブジェクトであることを示します。

SIM_EVENT
オブジェクトがフォルダーまたは文書に関するイベントであることを示します。

SIM_MGDS
オブジェクトがフォルダーまたは文書に関する MGDS (マシン生成データ・ストリーム) であることを示します。

SIM_NOTE
オブジェクトがフォルダーまたは文書に関する注であることを示します。

pAffiliatedData PVOID - 入力

ANNOTATIONSTRUCT タイプのデータ構造へのポインター。*ulAffiliatedType* パラメーターに値 SIM_ANNOTATION が含まれる場合、*pAffiliatedData* は、オブジェクト系列下に追加データが含まれるこの構造を指します。それ以外の場合は、Content Manager for iSeries システムはこのパラメーターを無視します。
ANNOTATIONSTRUCT 構造の詳細については、153 ページの『ANNOTATIONSTRUCT (注釈情報構造)』を参照してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。

ulParam1 *hObj* が入ります。hObj は、オブジェクト・ハンドル・ブロックへの HOBJ ポインターです。

ulParam2 *fCreateControl* パラメーターのフラグとして SIM_OPEN を指定し、かつフィールドが NULL ではない場合、フィールドにはオブジェクト・アクセス・ハンドルが含まれます。このハンドルはデータ型 HOBJACC を持っています。このフィールド内の値は、アクセスしたオブジェクトの現行インスタンスを識別します。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_INVALID_FOPTIONS
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_LOCAL_STORAGE_MODE
- SIM_RC_INVALID_OBJECT_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_SMS_PTR
- SIM_RC_NOT_SUPPORTED
- SIM_RC_OBJECT_ALREADY_EXISTS
- SIM_RC_OPEN_FAILED
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

準備:

- カタログの作成対象のオブジェクトは、ファイルとして存在する必要があります。
- *ulConCls* パラメーターに定義されている値を取得するには、**lp2ListContentClasses** 関数を使用してください。

結果:

- この関数は、オブジェクトを作成し、ユーザーが指定したファイルの内容をそのオブジェクトに書き込みます。
- この関数が正常に完了すると、オブジェクトへのアクセスに使用できるオブジェクト処理が戻されます。

  HOBJ データ構造内の入力値は、この関数の結果に影響を与えます。この構造の *szItemID*、*ulPart*、および *chRepType* フィールドに入力する値はオプショナルです。

  この部分番号に 0 が指定されている場合には、次の順次部分番号が作成されます。部分番号がゼロ以外の場合には、その部分番号が既存のものでなければ、その部分番号が使用されます。その部分番号が既に存在する場合には、最初の使用可能な番号が戻されます。部分番号 1 は通常、基本部分です。この API により、部分番号 1 を作成する前に、部分番号 2 (たとえば、注) を作成することができます。

- *fCreateControl* パラメーターに SIM_OPEN フラグを指定しない場合、オブジェクトはクローズされますが、**SimLibOpenObject** 関数を使用してオープンすることができます。そうすると、この関数によって戻されるオブジェクト・アクセス・ハンドルを使用してオブジェクトにアクセスすることができます。このオブジェクトを参照する場合にはオブジェクト・ハンドルを使用する必要があります。
- アプリケーションが系列下のタイプを格納することができても、他のアプリケーションがこれらのオブジェクトを処理できない場合があります。

例外: コンテンツ・クラス・パラメーターは、定義済みで、既知のコンテンツ・クラスと見なされ、妥当性検査はされません。

後続作業:

- SIM_OPEN を指定した場合には、**SimLibCloseObject** 関数を使用して、終了時にそのオブジェクトをクローズしてください。
- オブジェクト・ハンドル・ブロックのポインターの使用が終了したら、**SimLibFree** 関数を使用してそのスペースを解放してください。

例

```
   #include <stdio.h>                     /* Standard I/O header files         */
   #include <string.h>                    /* Standard string header file       */
   #include "ekdviapi.h"                  /* Content Manager for iSeries       */

main ()
{
   HSESSION hSession;                           // from logon
   HOBJ hObj;
   HOBJ hObj2;                                  //get pointer from catalog
   ULONG ulConCls = SIM_CC_MODCA_IS2;            // mod:ca object
   SMS  sms;
   CHAR  pszFullFileName[45];
   UCHAR ulPriority = 0;                        // not supported
   BITS fCreateControl = SIM_OPEN;              //leave open-get hobjacc
   ULONG ulVersion =  0;                        // not supported
   LONG lSeqAfterPart = 0;                      // take default
   ULONG ulAffiliatedType =  SIM_BASE;          // base part
   PVOID pAffiliatedData = NULL;                // no affil data for base part
   RCSTRUCT RC;
   PRCSTRUCT pRC = &RC;
   POBJ          pObj;                          // Created object  handle
   HOBJACC       hObjAcc;                       // object access handle
   USHORT        sResult;                       // return codes
   // create hobj
   if(0==(  pObj=malloc(sizeof(OBJ)))) {
      return(1);
   }
   (  pObj)->ulStruct = sizeof(OBJ);
   strcpy((  pObj)->szItemID,"");
   strcpy((  pObj)->chRepType,"");
   (  pObj)->ulPart = 0;
   hObj = pObj;

   strcpy(pszFullFileName, "d:¥¥spid¥¥modca.mda");
   memset(SMS,0, sizeof(sms));                  // null out struct to get defaults
   strcpy(SMS.szCollectionName, "*DFT");

   sResult = SimLibCatalogObject(
           hSession,
           hObj,
           ulConCls,
```

```
            SMS,
            pszFullFileName,
            ulPriority,
            fCreateControl,
            ulVersion,
            lSeqAfterPart,
            ulAffiliatedType,
            pAffiliatedData,
            0,
            pRC);
        if (pRC ->ulRC == SUCCESS) {
          // When only HOBJ is returned, it is in ulParam1
          hObj2 = (HOBJ)pRC->ulParam1;
          // Free memory allocated for HOBJ
          SimLibFree(hSession, (PVOID)(hObj2), pRC);
          // Mem containing the HOBJACC struct is freed by SimLibCloseObject.
          hObjAcc = pRC->ulParam2; // object access handle
        }
}
```

関連関数

- **Ip2ListContentClasses**
- **SimLibCloseObject**
- **SimLibCreateItem**
- **SimLibCreateObject**
- **SimLibFree**
- **SimLibOpenObject**
- **SimLibWriteObject**

SimLibChangeIndexClass (項目の索引クラスの変更)

形式

SimLibChangeIndexClass(*hSession, hItem, usClassId, pAsyncCtl, pRC* **)**

目的

SimLibChangeIndexClass 関数を使用して、ある項目の索引クラスを、ユーザーが指定した索引クラスに変更します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hItem HITEM - 入力

仮想項目のハンドル。**SimLibOpenItemAttr** 関数がこのハンドルを戻します。

usClassId USHORT - 入力

変更後の索引クラスの ID。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HITEM_VALUE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PASSED_ATTRIBUTE_DATA
- SIM_RC_INVALID_PATTRIBUTE_PTR
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USATTRIBUTEID_VALUE
- SIM_RC_INVALID_USCLASSID_VALUE
- SIM_RC_NO_WRITE_ACCESS
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

準備: この関数を使用するためには、まず **SimLibOpenItemAttr** 関数を使用して書き込みアクセス用の項目をオープンする必要があります。

結果:

- この関数は、項目の索引クラスを変更して、別のユーザー定義の属性セットをその項目と結び付けます。
- 項目が書き込みアクセスのためにオープンしていない場合、この関数はエラー SIM_RC_NO_WRITE_ACCESS を戻します。
- この関数の使用が失敗した場合には、Content Manager for iSeries システムは、この項目の現行の属性セットを維持します。
- 索引クラスの属性が、この項目の元の索引クラスと、ユーザーが指定した新しい索引クラスに共通する場合、この関数はそれらの属性を新しい索引クラスにコピーします。ここで、アプリケーションは **SimLibWriteAttr** 関数を使用して、新しい索引クラスの属性を必要な値に設定することができます。新しい索引クラスに必要な属性値をすべて指定してしまうと、 **SimLibSaveAttr** または **SimLibCloseAttr** を使用して項目への変更を保管し、それらの値を永続的なものにすることができます。

- **SimLibGetClassInfo** を使用して索引クラスに関連づけられている属性を判別し、さらに **SimLibGetAttrInfo** を使用して属性に関する詳細を取得してください。
- **SimLibOpenItemAttr** では、ユーザーが SIM_ACCESS_READ_WRITE 権限を持っているかどうかについての妥当性検査は行われません。この権限の妥当性検査は、SIM_OPT_SAVE パラメーターによって **SimLibCloseAttr** を呼び出したときに行われます。

関連関数

- **SimLibCloseAttr**
- **SimLibGetAttrInfo**
- **SimLibGetClassInfo**
- **SimLibOpenItemAttr**
- **SimLibSaveAttr**
- **SimLibWriteAttr**

SimLibChangeObjectSMS (オブジェクトの SMS 基準の変更)

形式

SimLibChangeObjectSMS(*hSession, hObj, pSMS, fChangeControl, pAsyncCtl, pRC* **)**

目的

SimLibChangeObjectSMS 関数を使用して、オブジェクトのシステム管理ストレージ (SMS) 基準を変更します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObj HOBJ - 入力

HOBJ データ構造内のオブジェクト・ハンドル・ブロックへのポインター。HOBJ 構造の詳細については、 161 ページの『HOBJ (記憶オブジェクトの照会のためのハンドル)』を参照してください。

pSMS PSMS - 入力

オブジェクトのシステム管理ストレージ (SMS) 構造へのポインター。この構造では *szCollectionName* のみを使用します。

fChangeControl BITS - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_FOPTIONS
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_ITEMID
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_PSMS_VALUE
- SIM_RC_INVALID_SMS_PTR
- SIM_RC_NEW_COLLECTION_NOT_FOUND
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PART_NOT_FOUND
- SIM_RC_PRIVILEGE_ERROR

関連関数

- **SimLibCreateObject**
- **SimLibQueryObject**

SimLibCloseAttr (属性セットのクローズ)

形式

SimLibCloseAttr(*hSession, hItem, ulDisposition, pAsyncCtl, pRC* **)**

目的

SimLibCloseAttr 関数を使用して、アプリケーションが指定されたフォルダーまたは文書に対して持っているアクセス権限を解放します。この関数を使用して、データベース内の項目の永続的な属性を、仮想項目に行われた修正と置換することができます。さらにこの関数を使用して、永続的な属性を更新せずに、仮想項目に対する修正を廃棄することもできます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hItem HITEM - 入力

仮想項目のハンドル。**SimLibOpenItemAttr** 関数がこのハンドルを戻します。

ulDisposition ULONG - 入力

項目の修正に関してとる処置。このパラメーターの値は、Content Manager for iSeries システムが仮想項目の属性に対する修正を保管または廃棄するかどうかを決定します。項目へのアクセスが読み取りのみの場合、または項目の属性がまったく変更されていない場合、Content Manager for iSeries システムはこのパラメーターを無視します。有効な値は、次のとおりです。

SIM_OPT_SAVE
仮想項目の属性の現行の設定値を使用して、データベース内の項目の永続的な属性を更新します。索引クラスに必要な属性は、クローズする前に書き込まれる必要があります。書き込まれないと、この関数はエラー SIM_RC_REQUIRED_ATTRIBUTE_MISSING を戻します。この値は、項目が更新のためにオープンしている場合のみ有効です。

SIM_OPT_DISCARD
データベース内の項目の永続的な属性を更新せずに、仮想項目の属性の設定値の修正を廃棄します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_FOPTIONS
- SIM_RC_INVALID_HITEM_VALUE
- SIM_RC_INVALID_HSESSION

- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USACCESSLEVEL_VALUE
- SIM_RC_INVALID_USCLASSID_VALUE
- SIM_RC_INVALID_USDISPOSITION_VALUE
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR
- SIM_RC_REQUIRED_ATTRIBUTE_MISSING

使用の手引き

結果: この関数は仮想属性セットをクローズするので、アクセス・ハンドルを使用できなくなります。また、この関数はアクセス・ハンドルによって使用されていたスペースを解放します。

関連関数

-
- **SimLibChangeIndexClass**
- **SimLibOpenItemAttr**
- **SimLibSaveAttr**
- **SimLibWriteAttr**

SimLibCloseObject (オブジェクトのクローズ)

形式

SimLibCloseObject(*hSession, hObjAcc, fCommit, pAsyncCtl, pRC* **)**

目的

SimLibCloseObject 関数を使用して、オープンされているオブジェクトをクローズし、そのオブジェクトへのアクセスを終了します。

以下のいずれかの関数を使用してオープンしたオブジェクトをクローズするには、この関数を使用する必要があります。

- **SimLibCatalogObject**
- **SimLibCreateObject**
- **SimLibOpenObject**

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObjAcc HOBJACC - 入力

オブジェクト・アクセス・ハンドル。このパラメーターの値は、アクセスされたオブジェクトの現行インスタンスを識別します。

fCommit BOOL - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_FOPTIONS
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_OBJECT_ACCESS_HANDLE
- SIM_RC_INVALID_OBJECT_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果: この関数が正常に完了すると、ユーザーはアクセス・ハンドルを使用できなくなります。また、この関数はアクセス・ハンドルによって使用されていたスペースを解放するので、**SimLibFree** 関数を呼び出す必要はありません。

SIM_RC_PRIVILEGE_ERROR が戻った場合は、項目のロックが解除されている状態を保証するために、SIM_OPT_DISCARD を指定して **SimLibCloseAttr** を呼び出す必要があります。

例

```
   #include <stdio.h>                 /* Standard I/O header files         */
   #include "ekdviapi.h"              /* Content Manager for iSeries       */

main ()
{
   HSESSION hSession;                       // get from logon
   HOBJACC hObjAcc;                         // get from catalog, open, or create
   BOOL fCommit = TRUE;                     // keep the changes
   RCSTRUCT RC;
   PRCSTRUCT pRC = &RC;
```

```
   USHORT        sResult;                    // return codes

  /*Call the function  */

   sResult = SimLibCloseObject(
           hSession,
           hObjAcc,
           fCommit,
           0,
           pRC);
}
```

関連関数

- **SimLibCatalogObject**
- **SimLibCreateObject**
- **SimLibOpenObject**

SimLibCopyObject (オブジェクトのコピー)

形式

SimLibCopyObject(*hSession, hDestObj, hSrcObj, pSMS, ulPriority, fDelete, pAsyncCtl, pRC* **)**

目的

SimLibCopyObject 関数を使用すると、あるオブジェクト全体をソース・オブジェクトのロケーションからターゲット・オブジェクトのロケーションにコピーし、既存のターゲット・オブジェクトを置換することができます。ただし、ソース・オブジェクト、ターゲット・オブジェクトの一方または両方がオープンしていると、この関数は使用できません。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hDestObj HOBJ- 入力

宛先オブジェクト・ハンドル。このパラメーターの値は、ターゲット・オブジェクトを識別します。

hSrcObj HOBJ- 入力

ソース・オブジェクト・ハンドル。このパラメーターの値は、関数によってコピーされるソース・オブジェクトを識別します。

pSMS PSMS- 入力

サポートされていません。

ulPriority ULONG - 入力

サポートされていません。

fDelete BOOL - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。戻りコードが SIM_RC_ITEM_CHECKEDOUT の場合には、このフィールドには値 1 が入ります。この値は、ulParam1 にポインターが入っていることを示しています。Content Manager for iSeries システムが別のエラーを戻す場合、このフィールドには値 NULL が入ります。

ulParam1 戻りコードが SIM_RC_ITEM_CHECKEDOUT の場合、このフィールドには値 NULL が入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INUSE
- SIM_RC_INVALID_FOPTIONS
- SIM_RC_INVALID_HSESSION
- SIM_RC_NOT_SUPPORTED_
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

関連関数

- **SimLibLogon**

SimLibCreateItem (項目の作成)

形式

SimLibCreateItem(*hSession, usItemType, usIndexClass, usNumOfAttrs, pAttributeList, ulAccessControl, pAsyncCtl, pRC* **)**

目的

SimLibCreateItem 関数を使用して、指定された索引クラスに新しい文書または新しいフォルダーを作成します。その索引クラスに必要な属性を指定する必要があります。その項目にオプショナルの属性を指定することもできます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

usItemType USHORT - 入力

作成する項目のタイプ。有効な値は、次のとおりです。

SIM_DOCUMENT
項目が文書であることを示します。

SIM_FOLDER
項目がフォルダーであることを示します。

usIndexClass USHORT - 入力

ユーザー定義の属性セットをこの項目に関連づけるための索引クラス ID。ログオンするときにはこの索引クラスが存在している必要があります。

ユーザー定義の属性を必要としない場合には、項目がまだ索引付けされていないことを指示するために、SIM_INDEX_NOINDEX を使用してください。SIM_INDEX_NOINDEX は、インストール時に作成され、ユーザー定義属性を用いて事前設定された特別な索引クラスです。「使用の手引き」では、事前定義の索引クラスを使用することが重要である理由について説明します。

usNumOfAttrs USHORT - 入力

pAttributeList パラメーター配列内のデータ構造の数。

pAttributeList PATTRLISTSTRUCT - 入力

ATTRLISTSTRUCT データ構造の配列へのポインター。このデータ構造には、この文書またはフォルダーに関連づけるための属性が含まれます。配列内の各データ構造は、それぞれ 1 つの属性を指定します。このパラメーターを NULL に設定すると、項目には属性が関連づけられません。ATTRLISTSTRUCT データ構造の詳細については、155 ページの『ATTRLISTSTRUCT (属性リスト・データ構造)』を参照してください。

後に項目に属性を追加するには、ユーザー・アプリケーションでまずその項目をオープンし、別の関数を使用して項目に属性を書き込む必要があります。

ulAccessControl ULONG - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。エラーが発生すると、このフィールドには値 0 が入ります。

ulParam1 新しい項目に対する項目 ID (*pszItemID*) が付いたバッファーへの PITEMID ポインターが入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_ATTR_NOT_FOUND
- SIM_RC_ATTRIBUTE_READ_ONLY
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_INDEX_CLASS
- SIM_RC_INVALID_MSGID
- SIM_RC_INVALID_PASSED_ATTRIBUTE_DATA
- SIM_RC_INVALID_PATTRIBUTELIST_PTR
- SIM_RC_INVALID_PATTRIBUTELIST_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USATTRIBUTEID_VALUE
- SIM_RC_INVALID_USITEMTYPE_VALUE
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

準備:

- 事前定義索引クラスの使用は、各項目を見つけるための **SimLibSearch** 関数を使用できるようにするのに重要です。
- 新しく作成された索引クラスに項目を追加するには、この関数を使用する前に一度ログオフしてから再度ログオンし、索引クラスがログオン時に存在するようにしてください。
- **SimLibCatalogObject** または **SimLibCreateObject** を使用して、項目を自動的に作成することもできます。属性値を持つ索引クラスがある場合は、**SimLibCreateItem** を使用してください。その後で **SimLibCatalogObject**、**SimLibCreateObject**、**SimLibStoreNewObject** のいずれかを使用して、新しい項目にオブジェクトを入れてください。

後続作業: 関数が項目の ID を取得したら、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用してバッファーを解放してください。

例

```
#include <windows.h>                /* Main Windows header files    */
#include <sys¥types.h>
#include <stdio.h>                  /* Standard I/O header files    */
#include <stdlib.h>                 /* Standard library header files*/
#include <stdarg.h>
#include <stddef.h>
#include <io.h>
#include "ekdviapi.h"               /* Content Manager for iSeries   */

main ()
{

  HSESSION     hSession;            /* Product session handle        */
  ITEMID       FolderItemID;        /* ItemID of new folder          */
  USHORT       usFoldAttrs;         /* Number of ATTRLISTSTRUCTs     */
  ATTRLISTSTRUCT Folder [ 1 ] = {
               sizeof(Folder),      /* structure size                */
               "SourceName",        /* attribute value               */
               SIM_ATTR_READWRITE,  /* attribute flags               */
               140,                 /* attribute ID                  */
               SIM_ATTR_FSTRING     /* attribute type                */
               };

  USHORT       usIndexClass;        /* Index class for folder        */
  RCSTRUCT     RCStruct;            /* RC data structure             */
  USHORT       sResult;             /* return codes                  */


    /*****************************************************************/
    /* Initialize SimLibCreateItem Parameters.                       */
    /*****************************************************************/

    /* We will create an item in the SIM_INDEX_NOINDEX Index Class.  */
    /* This index has three optional attributes.  We will provide a  */
    /* value for only one of these attributes.  This is done by      */
    /* initializing the attribute array "Folder" above.              */

    usIndexClass = SIM_INDEX_NOINDEX;/* Index Class of the folder    */
    usFoldAttrs  = 1;                /* # of attrs for the folder    */

   /*****************************************************************/
   /* Call SimLibCreateItem to create a new folder                  */
   /*****************************************************************/

    sResult = SimLibCreateItem(
            hSession,                /* session handle from SimLibLogon*/
            SIM_FOLDER,              /* Create a folder              */
            usIndexClass,            /* Index class of folder        */
            usFoldAttrs,             /* Number of attribute lists    */
            &Folder,                 /* Pointer to attribute list    */
            NULL,                    /* Reserved for future use      */
            NULL,                    /* Request SYNCHRONOUS processing*/
            &RCStruct                /* Pointer to RC data structure*/
            );

   /*****************************************************************/
   /* If successful, copy the itemID                                */
   /*****************************************************************/

    if (sResult == SIM_RC_OK) {
        strcpy (FolderItemID, (char*)RCStruct.ulParam1;
        printf("New Folder ItemID         = %s¥n¥n", FolderItemID);
    }
```

```
    else {
     /* ..... exception processing ..... */
    }
}
```

関連関数

- **SimLibChangeIndexClass**
- **SimLibFree**
- **SimLibGetAttrInfo**
- **SimLibGetClassInfo**
- **SimLibSearch**

SimLibCreateObject (オブジェクトの作成)

形式

SimLibCreateObject(*hSession, hObj, ulConCls, pSMS, ulPriority, fCreateControl, ulVersion, lSeqAfterPart, ulAffiliatedType, pAffiliatedData, pAsyncCtl, pRC* **)**

目的

SimLibCreateObject 関数を使用して、新しい空のオブジェクトを作成します (たとえば、データがファイル内ではなくメモリー内にある場合)。

また、**SimLibCatalogObject** 関数を使用してオブジェクトを作成することもできます。これは、**SimLibCreateObject**、**SimLibWriteObject**、および **SimLibCloseObject** 関数をともに使用することと同等です。
SimLibStoreNewObject 関数を使用してオブジェクトを作成することもできます。そうすると複数の関数を組み合わせて使用するよりも、速く簡単に作成することができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObj HOBJ - 入力

HOBJ データ構造内のオブジェクト・ハンドル・ブロックへのポインター。HOBJ データ構造の詳細については、 161 ページの『HOBJ (記憶オブジェクトの照会のためのハンドル)』を参照してください。「使用の手引き」で、このデータ構造に入力した結果について説明します。

ulConCls ULONG - 入力

オブジェクトのコンテンツ・クラス ID。このパラメーターの値は、作成しようとしているオブジェクト内のデータの種類を示します (329 ページの『付録 B. 事前定義済みコンテンツ・クラス』を参照)。未定義のコンテンツ・クラスを指示するには、このパラメータ

ーに値 **SIM_CC_UNKNOWN** を指定してください。ただし、定義済みのコンテンツ・クラスを使用しない場合には、他のアプリケーションは、Content Manager for iSeries コンテンツ・クラス・サービスを使用して、ユーザーが格納するオブジェクトの内容を操作する方法を判別することはできません。

pSMS PSMS - 入力

オブジェクトのシステム管理ストレージ (SMS) 構造へのポインター。この構造では *szCollectionName* のみを使用します。

ulPriority ULONG - 入力

サポートされていません。

fCreateControl BITS - 入力

作成操作の制御オプション・ビット。有効な値は、次のとおりです。

SIM_CLOSE
要求が完了するとオブジェクトをクローズします。これがデフォルトです。

SIM_OPEN
更新モードでオブジェクトをオープンしたままにしておきます。

このフラグを指定しない場合、作成されたオブジェクトはクローズされます。

ulVersion ULONG - 入力

サポートされていません。

lSeqAfterPart LONG - 入力

サポートされていません。

ulAffiliatedType ULONG - 入力

系列下にあるオブジェクトのタイプ。定義済みの値は次のとおりです。

SIM_ANNOTATION
オブジェクトが、フォルダーまたは文書に関する注釈であることを示します。

SIM_BASE
オブジェクトが MO:DCA または TIFF ファイルなどの基本オブジェクトであり、フォルダーまたは文書に関する注釈、注またはイベントではないことを示します。

SIM_EVENT
オブジェクトがフォルダーまたは文書に関するイベントであることを示します。

SIM_MGDS
オブジェクトがフォルダーまたは文書に関する MGDS (マシン生成データ・ストリーム) であることを示します。

SIM_NOTE
オブジェクトがフォルダーまたは文書に関する注であることを示します。

pAffiliatedData PVOID - 入力

ANNOTATIONSTRUCT タイプのデータ構造へのポインター。*ulAffiliatedType* パラメーターに値 SIM_ANNOTATION が含まれる場合、*pAffiliatedData* は、オブジェクト系列下に追加データが含まれるこの構造を指します。それ以外の場合は、Content Manager for iSeries システムはこのパラメーターを無視します。
ANNOTATIONSTRUCT 構造の詳細については、153 ページの『ANNOTATIONSTRUCT (注釈情報構造)』を参照してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。

ulParam1 *hObj* が入ります。hObj は、オブジェクト・ハンドル・ブロックへの HOBJ ポインターです。

ulParam2 *fCreateControl* パラメーター・フラグが SIM_OPEN に既に設定されていて、このフィールドが空ではない場合、このフィールドには、オブジェクト・アクセス・ハンドル *hobjacc* が入ります。このハンドルはデータ型 HOBJACC を持っています。このフィールド内の値は、アクセスしたオブジェクトの現行インスタンスを識別します。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_OBJECT_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_SMS_PTR
- SIM_RC_OPEN_FAILED
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR
- SIM_RC_INVALID_USCLASSID_VALUE

使用の手引き

準備: *ulConCls* パラメーターでサポートされている値を取得するには、**lp2ListContentClasses** 関数を使用してください。

結果:

- この関数は、**SimLibWriteObject** を使用して書き込むことができる空のオブジェクトを作成します。
- この関数が正常に完了すると、オブジェクトへのアクセスに使用できるオブジェクト処理が戻されます。
- ユーザーは指定された項目内に新しいオブジェクトを作成したり、その項目内に項目とオブジェクトの両方を作成することができます。項目を作成したら、属性を指定することはできません。この項目は SIM_INDEX_NOINDEX 索引クラスに入れられます。属性値の指定は、後で **SimLibOpenItemAttr**、**SimLibWriteAttr**、および **SimLibCloseAttr** を使用して行う必要があります。
- アプリケーションが系列下のタイプを格納することができても、他のアプリケーションがこれらのオブジェクトを処理できない場合があります。
- HOBJ データ構造内の入力値は、この関数の結果に影響を与えます。この構造内の *szItemID*、*ulPart*、および *chRepType* フィールドへの入力値はオプショナルです。

  この部分番号に 0 が指定されている場合には、次の順次部分番号が作成されます。部分番号がゼロ以外の場合には、その部分番号が既存のものでなければ、その部分番号が使用されます。その部分番号が既に存在する場合には、最初の使用可能な番号が戻されます。部分番号 1 は通常、基本部分です。この API により、部分番号 1 を作成する前に、部分番号 2 (たとえば、注) を作成することができます。
- この関数がオブジェクトをクローズした場合、**SimLibOpenObject** 関数を使用してそれをオープンすることができます。
- この関数がオブジェクト・アクセス・ハンドルを戻す場合、このハンドルはオープン・オブジェクトへのアクセスの現行インスタンスを識別します。このハンドルは、格納されているオブジェクトを参照するのに通常使用するハンドルとは異なります。以下の関数では、オブジェクト処理 (*hObj*) ではなく、オブジェクト・アクセス・ハンドル (*hObjAcc*) を使用してください。
  - **SimLibCloseObject**
  - **SimLibReadObject**
  - **SimLibResizeObject**
  - **SimLibSeekObject**
  - **SimLibWriteObject**

例外:

- コンテンツ・クラス・パラメーターは、定義済みで、既知のコンテンツ・クラスと見なされ、妥当性検査はされません。

後続作業:

- アプリケーションがオブジェクト処理 *hObj* の使用を終了した後で、**SimLibFree** 関数を使用してそのスペースを解放してください。

- アプリケーションは、オブジェクト・アクセス・ハンドル *hObjAcc* が使用しているスペースを解放しないようにしてください。これ以降に、**SimLibCloseObject** への呼び出しによってそのスペースが解放されるためです。

例

```
   #include <stdio.h>                       /* Standard I/O header files         */
   #include <string.h>                      /* Standard string header file       */
   #include "ekdviapi.h"                    /* Content Manager for iSeries       */

 main()
{
   HSESSION hSession;                      // get from logon
   HOBJ hObj, hObj2;
   ULONG ulConCls = SIM_CC_MODCA_IS2;      // mod:ca object
   SMS  sms;
   ULONG ulPriority = 0;                   // not supported
   BITS fCreateControl = SIM_OPEN;         //leave open-get hobjacc
   ULONG ulVersion = 0;                    // not supported
   LONG lSeqAfterPart = 0;                 // not supported
   ULONG ulAffiliatedType = SIM_BASE;
   PVOID pAffiliatedData = NULL;           // no affiliated data
   RCSTRUCT RC;
   PRCSTRUCT pRC = &RC;
   POBJ          pObj;                     // Created object  handle
   USHORT        sResult;                  // get rc back
   HOBJACC       hObjAcc;                  // object access handle

                                           // create hobj
   if(0==(  pObj=(POBJ) malloc(sizeof(OBJ)))) {
      return(1);
   }
   (  pObj)->ulStruct = sizeof(OBJ);
   strcpy((  pObj)->szItemID,"");
   strcpy((  pObj)->chRepType,"");
   (  pObj)->ulPart = 0;
   hObj = pObj;

   memset(SMS,0, sizeof(sms));             // null out struct to get defaults
   strcpy(SMS.szCollectionName, "*DFT");

   /*Call the function*/

   sResult = SimLibCreateObject(
           hSession,
           hObj,
           ulConCls,
           SMS,;
           ulPriority,
           fCreateControl,
           ulVersion,
           lSeqAfterPart,
           ulAffiliatedType,
           pAffiliatedData,
           0,
           pRC);
      if (pRC ->ulRC == SUCCESS) {
        // When only HOBJ is returned, it is in ulParam1
        hObj2 = (HOBJ)pRC->ulParam1;
        // Free memory allocated for HOBJ
        SimLibFree(hSession, (PVOID)(hObj2), pRC);
```

```
        // Mem containing the HOBJACC struct is freed by SimLibCloseObject.
        hObjAcc = pRC->ulParam2;            // object access handle
      }
}
```

関連関数

- **Ip2ListContentClasses**
- **SimLibCatalogObject**
- **SimLibCloseObject**
- **SimLibCreateObject**
- **SimLibFree**
- **SimLibOpenObject**
- **SimLibReadObject**
- **SimLibResizeObject**
- **SimLibSeekObject**
- **SimLibWriteObject**

SimLibDeleteItem (項目の削除)

形式

SimLibDeleteItem(*hSession, pszItemID, pAsyncCtl, pRC* **)**

目的

SimLibDeleteItem 関数を使用して、システムからフォルダーまたは文書を削除します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszItemID PITEMID - 入力

削除したい項目の ID。この ID は、項目 ID です。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。項目がサーバー上でロックされている場合には、

このフィールドには値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。

ulParam1 *usParam* が 1 の場合、このフィールドには USERACCESSSTRUCT データ構造を持つバッファーへのポインターが入ります。このデータ構造には、項目をロックしたユーザーを示すユーザー ID が含まれます。他のなんらかのエラーが戻された場合、このフィールドには値 NULL が入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INUSE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_ITEM_CHECKEDOUT
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PARENT_CHECKEDOUT
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果:

- この関数は、指定した文書またはフォルダーをデータベースから削除します。この関数が完了すると、その項目と関連する項目 ID (*pszItemID*) は無効になります。
- この関数は、フォルダーまたはワーク・バスケットの目次から、削除された項目の参照を自動的に削除します。
- フォルダーまたは文書のいずれかの場合、Content Manager for iSeries システムは、項目に関連づけられているすべてのオブジェクトを削除します。
- フォルダーが削除される場合、そのフォルダー内の文書またはフォルダーは削除されません。

例外:

- この関数は、項目またはその項目を含むフォルダーが、**SimLibLogon** 関数を使用してこの Content Manager for iSeries セッションを開始したときに *pszUserID* パラメーターに指定したユーザー ID 以外のユーザー ID によって現在ロックされている場合には、その項目を削除することはできません。

  1 つのフォルダーが複数の親フォルダーを持つことが可能です。親フォルダーがロックされており、**SimLibDeleteItem** 関数が SIM_RC_PARENT_CHECKEDOUT を戻す場合には、この関数は、ロックされているフォルダーを識別しません。

後続作業: アプリケーションでユーザー・アクセス情報が不要になった場合は、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、USERACCESSSTRUCT データ構造が入っているバッファーを解放してください。

例

```
#include <windows.h>                /* Main Windows header files    */
#include <sys¥types.h>
#include <stdio.h>                  /* Standard I/O header files    */
#include <stdlib.h>                 /* Standard library header files*/
#include <stdarg.h>
#include <stddef.h>
#include <io.h>
#include "ekdviapi.h"               /* Content Manager for iSeries   */
main ()
{

  HSESSION     hSession;            /* Product session handle        */
  PITEMID      pszItemID;           /* Pointer to an item ID.        */
  RCSTRUCT     RCStruct;            /* RC data structure             */
  USHORT       sResult;             /* return codes                  */

    /*****************************************************************/
    /*Initialize the itemID to prepare for a call to SimLibDeleteItem*/
    /*****************************************************************/
    memset  (pszItemID, '¥0', DOC_ID_SIZE); /* set to null           */
    strcpy  ((CHAR *)pszItemID, (CHAR *) "DA97220AA.AAB");

   /*****************************************************************/
   /* Call SimLibDeleteItem to delete a document from the system     */
   /*****************************************************************/

    sResult = SimLibDeleteItem(
            hSession,                 /* session handle from SimLibLogon */
            pszItemID,                /* itemID to be deleted          */
            (PASYNCCTLSTRUCT) NULL,   /* Request SYNCHRONOUS processing*/
            (PRCSTRUCT) &RCStruct     /* Pointer to RC data structure  */
            );

    if (sResult != SIM_RC_OK) {
        printf("Item %s cannot be deleted", pszItemID);
    }
}
```

関連関数

- **SimLibAddFolderItem**
- **SimLibCloseAttr**
- **SimLibCreateItem**
- **SimLibFree**
- **SimLibGetItem**
- **SimLibOpenItemAttr**

SimLibDeleteObject (オブジェクトの削除)

形式

SimLibDeleteObject(*hSession, hObj, ulDeleteOption, pAsyncCtl, pRC* **)**

目的

SimLibDeleteObject 関数を使用して、指定したオブジェクトを削除します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObj HOBJ - 入力

HOBJ データ構造内のオブジェクト・ハンドル・ブロックへのポインター。HOBJ 構造の詳細については、 161 ページの『HOBJ (記憶オブジェクトの照会のためのハンドル)』を参照してください。

ulDeleteOption ULONG - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。戻りコードが SIM_RC_ITEM_CHECKEDOUT の場合には、このフィールドには値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。Content Manager for iSeries システムが別のエラーを戻す場合、このフィールドには値 NULL が入ります。

ulParam1 *usParam* が 1 の場合、このフィールドには USERACCESSSTRUCT データ構造へのポインターが入ります。このデータ構造には、項目をロックしたユーザーのユーザー ID が入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_OBJECT_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_ITEM_CHECKEDOUT
- SIM_RC_ITEM_NOT_FOUND

- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PART_NOT_FOUND
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果: 項目内の最後のオブジェクトが削除されると、その項目も削除されます。1回の操作ですべてのオブジェクトを削除するには、**SimLibDeleteItem** 関数を使用してください。この関数は、項目とその項目内のすべてのオブジェクトを削除します。

例外:

- オブジェクトを含んでいる項目が別のユーザーによってロックされている場合には、そのオブジェクトを削除することはできません。
- そのオブジェクトだけがその項目に入っている場合は、その項目も削除されます。

SimLibFree (メモリーの解放)

形式

SimLibFree(*hSession, pBuffer, pRC* **)**

目的

SimLibFree 関数を使用して、Content Manager for iSeries システムによって割り振られ戻されたすべてのメモリーを解放します。アプリケーションがメモリーを割り振った場合は、この関数を呼び出さないでください。指示された方法でのみこの関数を使用してください。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pBuffer PVOID - 入力

不確定タイプのデータ構造へのポインター。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常に完了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC

例

```
ULONG          ulRC;
HSESSION       hsession;
RCSTRUCT       RC;

ulRC = SimLibListClasses(hSession, 0, NULL, &RC);
if (ulRC == SIM_RC_OK) {
   // process list of classes
   SimLibFree(hSession, (PVOID)RC.ulParam1, &RC);
}
```

関連関数
- **SimLibLogon**

SimLibGetAttrInfo (属性情報の取得)

形式

SimLibGetAttrInfo(*hSession, usAttributeId, pAsyncCtl, pRC* **)**

目的
SimLibGetAttrInfo 関数を使用して、システム内の特定の属性に関する詳細情報を戻します。この関数は、システム定義の属性およびユーザー定義の索引属性の両方に関する情報を戻すことができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

usAttributeId USHORT - 入力

属性に割り当てられた固有 ID。索引クラスの ID または以下の Content Manager for iSeries システム定義属性のいずれかを渡すことができます。

OIM_ID_ITEM_CREATE_TIMESTAMP
項目が作成された時間を示します。

OIM_ID_ITEM_NAME
項目の名前を示します。この属性はオプショナルです。

OIM_ID_SYS_MOD_TIMESTAMP
項目が最後に変更された時間を示します。

OIM_ID_UID
項目 ID を示します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。正常に完了しなかった場合、フィールドには値 0 が入ります。

ulParam1 ATTRINFOSTRUCT データ構造が、指定された属性に関する情報を提供するバッファーへのポインターが入ります。
ATTRINFOSTRUCT データ構造の詳細については、153 ページの『ATTRINFOSTRUCT (属性情報構造)』を参照してください。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USATTRIBUTEID_VALUE
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

後続作業: アプリケーションで ATTRINFOSTRUCT データが不要になった場合は、**SimLibFree(** *hSession,*(PVOID)*ulParam1, pRC* **)** 関数を使用して、その構造を含むバッファーを解放してください。

関連関数

- **Ip2ListAttrs**
- **SimLibFree**
- **SimLibGetClassInfo**

SimLibGetClassInfo (索引クラス情報の取得)

形式

SimLibGetClassInfo(*hSession, usClassType, usID, pAsyncCtl, pRC* **)**

目的

SimLibGetClassInfo 関数を使用して、システム内で定義されている特定の索引クラスに関する詳細情報を戻します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

usClassType USHORT - 入力

usID パラメーターに入っている情報のタイプ。有効な値は、次のとおりです。

SIM_INDEXCLASSID
usID パラメーターに、索引クラス ID が入っていることを示します。

usID USHORT - 入力

索引クラスの ID。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 1 の値が入り、*ulParam1* にデータ域を指すポインターが入っていることを示します。

ulParam1 CLASSINFOSTRUCT データ構造を持つバッファーへのポインターを含みます。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_CLASS_TYPE

- SIM_RC_INVALID_FOPTIONS
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USCLASSID_VALUE
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

例外: この関数によって戻される情報は、アクセス制御制限の対象です。索引クラスにアクセスできない場合には、この関数の使用は失敗し、SIM_RC_INVALID_USCLASSID_VALUE が戻されます。

後続作業: アプリケーションで CLASSINFOSTRUCT データが不要になった場合、**SimLibFree(** *hSession,*(PVOID)*ulParam1, pRC* **)** 関数を使用してバッファーを解放してください。

SimLibGetItemAffiliatedTOC (項目系列の目次の取得)

形式

SimLibGetItemAffiliatedTOC(*hSession, pszItemID, usAffiliatedType, pAsyncCtl, pRC* **)**

目的

SimLibGetItemAffiliatedTOC() 関数を使用して、ある項目の系列下にあるオブジェクトをリストした目次を取得します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszItemID PITEMID - 入力

系列オブジェクトをリストした目次を必要とする場合の項目の ID。この ID は、項目 ID です。

usAffiliatedType USHORT - 入力

目次にリストする系列オブジェクトのタイプ。有効な値は、次のとおりです。

SIM_ANNOTATION
フォルダーまたは文書に関する注釈をリストします。

SIM_BASE
フォルダーまたは文書に関する注釈、注、またはイベントではない、MO:DCA または TIFF ファイルなどの基本オブジェクトをリストします。

SIM_EVENT
フォルダーまたは文書に関するイベントをリストします。

SIM_MGDS
フォルダーまたは文書に関する MGDS (マシン生成データ・ストリーム) をリストします。

SIM_NOTE
フォルダーまたは文書に関する注をリストします。

SIM_ALL
フォルダーまたは文書に関するすべてのタイプのオブジェクトをリストします。

基本オブジェクト以外のオブジェクトを戻す必要があることを指定する場合には、そのオブジェクトはゼロ以外の長さでなければなりません。基本オブジェクトは、その長さに関係なく、常に組み込まれます。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。その他の場合、このフィールドには値 0 が入ります。

ulParam1 AFFTOCENTRYSTRUCT データ構造の配列を持つバッファーへのポインターが入ります。*usAffiliatedType* フィルターを満たす関連オブジェクトがない場合は、このフィールドには値 NULL が入ります。AFFTOCENTRYSTRUCT データ構造の詳細については、151 ページの『AFFTOCENTRYSTRUCT (関連目次項目構造)』を参照してください。

ulParam2 *ulParam1* によって参照される AFFTOCENTRYSTRUCT 配列内の項目の数が入ります。*usAffiliatedType* フィルターを満たす関連オブジェクトがない場合は、このフィールドには値 NULL が入ります。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR

- SIM_RC_COMPLETION_ERROR
- SIM_RC_COMPLETION_MSG_NOT_POSTED
- SIM_RC_COMPLETION_SEM_ALREADY_POSTED
- SIM_RC_COMPLETION_SEM_TOO_MANY_POSTS
- SIM_RC_DOCSS_ERROR
- SIM_RC_ERROR_RELEASING_SEMAPHORE
- SIM_RC_ERROR_REQUESTING_SEMAPHORE
- SIM_RC_FUNC_NOT_IN_TRANS
- SIM_RC_GETRESPONSE_TIMEOUT
- SIM_RC_INVALID_AFFILIATEDTYPE_VALUE
- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_PLATSESSION_TYPE
- SIM_RC_INVALID_PRC
- SIM_RC_ITEM_NOT_FOUND
- SIM_RC_NOT_SUPPORTED
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

後続作業: TOC 情報を取得したら、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、AFFTOCENTRYSTRUCT データ構造を含むバッファーを解放してください。

関連関数

- **SimLibFree**
- **SimLibLogon**

SimLibGetItemInfo (項目情報の取得)

形式

SimLibGetItemInfo(*hSession, pszItemID, usClassId, pAsyncCtl, pRC* **)**

目的

SimLibGetItemInfo 関数を使用して、文書またはフォルダーに関する下記の情報をアプリケーションに戻します。

- 項目のタイプ
- 項目名
- 項目の索引クラス
- ワークフロー情報
- 項目をロックしたユーザーのユーザー ID

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszItemID PITEMID - 入力

情報が必要な項目の ID。この ID は、項目 ID です。

usClassId USHORT - 入力

索引クラスの ID。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはデータ域へのポインターが入ることを示します。

ulParam1 項目情報を提供する ITEMINFOSTRUCT データ構造へのポインターを含みます。このデータ構造の詳細については、163 ページの『ITEMINFOSTRUCT (項目情報構造)』を参照してください。

ulParam2 値 1 が入り、*ulParam1* によって参照されるバッファーに 1 つの項目が含まれることを示します。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_ITEM_TYPE
- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

例外: この関数を使用してワーク・バスケットに関する情報を戻さないでください。ワーク・バスケット情報を戻すには、**SimWmGetWorkBasketInfo** を使用してください。

後続作業: アプリケーションで項目情報が不要になった場合、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用してバッファーを解放してください。

関連関数

- **SimWmGetWorkBasketInfo**
- **SimLibListClasses**

SimLibGetItemSnapshot (項目属性のスナップショットの取得)

形式

SimLibGetItemSnapshot(*hSession, pszItemID, fReadAttrInd, pAsyncCtl, pRC* **)**

目的

SimLibGetItemSnapshot 関数を使用して、文書またはフォルダーに関連づけられている属性のコピーを戻します。アプリケーションは、以下の一連の Content Manager for iSeries 関数の代りにこの関数を使用することができます。

- **SimLibGetItemType**
- **SimLibOpenItemAttr**
- **SimLibReadAttr**
- **SimLibCloseAttr**

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszItemID PITEMID - 入力

項目の ID。この ID は、項目 ID です。

fReadAttrInd BITS - 入力

戻される属性値のタイプ。有効な値は次のとおりです。ビット単位の包含 OR 演算子 (|) を使用して、これらの値を組み合わせることができます。

SIM_SYSTEM_ATTR
文書またはフォルダーのシステム定義属性値を戻します。

SIM_USER_ATTR
文書またはフォルダーのユーザー定義属性値を戻します。

SIM_WORK_ATTR
文書またはフォルダーのワーク・マネージメント情報を戻します。

この関数は、現行のビューについての属性値を戻します。Content Manager for iSeries システムは、SNAPSHOTSTRUCT データ構造からシステム定義およびユーザー定義の属性値を取得し、それらを

ICVIEWSTRUCT データ構造の *pAttr* フィールドに戻します。システムは、優先順位属性およびワーク・マネージメント情報を、SNAPSHOTSTRUCT データ構造の *pWmSnapshot* フィールドに戻します。「使用の手引き」で詳しく説明します。ICVIEWSTRUCT および SNAPSHOTSTRUCT データ構造の詳細については、162 ページの『ICVIEWSTRUCT (索引クラスのビュー情報構造)』 と 175 ページの『SNAPSHOTSTRUCT (スナップショット情報構造)』を参照してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはデータ域へのポインターが入ることを示します。

ulParam1 戻された属性値を提供する SNAPSHOTSTRUCT データ構造へのポインターを含みます。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_ITEM_TYPE
- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_READATTRIND
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR
- SIM_RC_SESSION_DB_VIEW_MISMATCH

使用の手引き

例外: アプリケーションは、ASCII - 整数間ルーチンなどの変換ルーチンを使用して、属性値の文字表現をアプリケーションに適した形式に変更しなければならない場合があります。

後続作業: アプリケーションが、Content Manager for iSeries システムが SNAPSHOTSTRUCT データ構造に戻した情報を処理した後で、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、SNAPSHOTSTRUCT データ構造へのポインターを解放してください。

関連関数

- **SimLibCloseAttr**
- **SimLibFree**
- **SimLibGetItemType**
- **SimLibGetTOCData**
- **SimLibOpenItemAttr**
- **SimLibReadAttr**

SimLibGetItemType (項目のタイプの取得)

形式

SimLibGetItemType(*hSession, pszItemID, pAsyncCtl, pRC* **)**

目的

SimLibGetItemType 関数を使用して、指定した項目 ID に対応付けられている項目のタイプを戻します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszItemID PITEMID - 入力

タイプを戻したい項目の ID。この ID は、項目 ID です。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。

ulParam1 項目のタイプを示す値 (以下のうちのどれか) が入ります。

SIM_DOCUMENT
項目が文書であることを示します。

SIM_FOLDER
項目がフォルダーであることを示します。

SIM_WORKBASKET
項目がワーク・バスケットであることを示します。

SIM_WORKFLOW
項目がワークフローであることを示します。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY

使用の手引き

結果: この関数が正常終了した後で、他の Content Manager for iSeries 関数を使用して、項目に関する追加の詳細情報を取得することができます。追加情報を戻すには、以下のいずれかの関数を使用してください。

SimLibGetItemInfo
フォルダーまたは文書に関する情報を戻します。

SimWmGetWorkBasketInfo
ワーク・バスケットに関する情報を戻します。

関連関数
- **SimWmGetWorkBasketInfo**
- **SimLibGetItemInfo**

SimLibGetItemXREF (項目の相互参照の取得)

形式
SimLibGetItemXREF(*hSession, pszItemID, ulFilter, pAsyncCtl, pRC* **)**

目的
SimLibGetItemXREF 関数を使用して、指定した項目を含み、かつ指定した他の基準に合致するフォルダーをリストします。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszItemID PITEMID - 入力

ユーザーが相互参照を必要としている項目の ID。この ID は、項目 ID です。

ulFilter ULONG - 入力

相互参照に合致する基準。有効な値は次のとおりです。

SIM_XREF_FOLDERS_ONLY_FILTER
指定した項目を含むフォルダーだけを戻します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。ユーザーが指定した基準と一致する項目がない場合は、このフィールドには値 NULL が入ります。

ulParam1 ITEMID ストリングの配列を持つバッファーへのポインターを含みます。各ストリングには、指定した項目を含むフォルダーの項目 ID が入ります。ユーザーが指定した基準と一致する項目がない場合は、このフィールドには値 NULL が入ります。

ulParam2 *ulParam1* によって指し示される項目の数が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_ITEM_TYPE
- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USFILTER_VALUE
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

後続作業: 項目 ID 情報を取得したら、 **SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、相互参照情報を含むバッファーを解放してください。

SimLibGetSessionType (セッション・タイプの取得)

形式

SimLibGetSessionType(*hSession, pAsyncCtl, pRC* **)**

目的

SimLibGetSessionType 関数を使用して、現行セッションのプラットフォーム・タイプに関する情報を戻します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。 **SimLibLogon** 関数は、セッション情報を作成します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常に完了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。

ulParam1 現行セッション・タイプに対する PSZ が入ります。LAN ベースのライブラリー・セッションがある場合には、セッション・タイプは Ip2 です。その他の値はプラットフォームに依存します。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_INVALID_HSESSION
- SIM_RC_OUT_OF_MEMORY

使用の手引き

後続作業: アプリケーションでセッション・タイプ情報が不要になった場合、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用してバッファーを解放してください。

関連関数

- **SimLibLogon**

SimLibGetTOC (目次の取得)

形式

SimLibGetTOC(*hSession, pszItemID, usItemType, usWipFilter, usSuspendFilter, usNbrOfClasses, pusClassIdList, pLinkCriteria, pAsyncCtl, pRC* **)**

目的

SimLibGetTOC 関数を使用して、指定したワーク・バスケットまたはフォルダーの部分的な目次または完全な目次を戻します。目次には、そのワーク・バスケットあるいはフォルダーの文書リストおよびフォルダー・リストが含まれています。この関数のパラメーターに、さまざまな値を指定して、目次の項目を判別することができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszItemID PITEMID - 入力

目次を必要としているワーク・バスケットあるいはフォルダーの ID。この ID は、項目 ID です。

usItemType USHORT - 入力

目次に戻す項目のタイプ。有効な値は、次のとおりです。

SIM_DOCUMENT
文書を戻します。

SIM_FOLDER
フォルダーを戻します。

SIM_ALL
文書とフォルダーの両方を戻します。

usWipFilter USHORT - 入力

サポートされていません。

usSuspendFilter USHORT - 入力

サポートされていません。

usNbrOfClasses USHORT - 入力

pusClassIdList パラメーターの値として指定したリスト内の索引クラス ID の数。*usNbrOfClasses* パラメーターに値 0 を指定して、クラスは選択される項目の基準ではないことを指示してください。

pusClassIdList PUSHORT - 入力

目次に選択する項目を指示する索引クラス ID リストへのポインター。*usNbrOfClasses* パラメーターに値 0 を指定した場合にのみ、*pusClassIdList* パラメーターに値 NULL を指定することができます。

pLinkCriteria PVOID - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 目次の項目数が入ります。フィルターに適する項目がない場合、フィールドには値 NULL が入ります。

ulParam1 TOCENTRYSTRUCT データ構造配列のバッファーを指すポインターが入ります。フィルターに適する項目がない場合、フィールドには値 NULL が入ります。このデータ構造の詳細については、178 ページの『TOCENTRYSTRUCT (目次項目データ構造)』を参照してください。

制約事項: アプリケーションで、TOCENTRYSTRUCT データ構造配列を含むバッファーを変更する必要はありません。戻された情報をアプリケーションで更新する必要がある場合は、そのアプリケーションのメモリー・バッファーにこの情報をコピーする必要があります。

ulParam2 目次ハンドル (*hTOC*) が入ります。フィルターに適する項目がない場合、フィールドには値 NULL が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_ITEM_TYPE

- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_PUSCLASSIDLIST_PTR
- SIM_RC_INVALID_USITEMTYPE_VALUE
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果: この関数を使用するたびに、新規の目次ハンドルが作成されます。**SimLibGetTOCData** および **Ip2GetTOCUpdates** 関数とともにあとでこのハンドルを使用して、どの目次を処理するかを指定することができます。

例外: **SimLibGetTOC** 関数は、ワーク・バスケットまたはフォルダーの現行内容を示す目次を作成します。ただし、ワーク・バスケットあるいはフォルダーの内容は、この関数を使用した後も変更される可能性があります。変更内容のリストを戻すには、**Ip2GetTOCUpdates** 関数を使用します。変更された項目を示すために、TOCENTRYSTRUCT (*usItemStatus* を含む) を更新してください。

後続作業: 目次ハンドルが不要になった場合には、**Ip2CloseTOC** 関数を使用してその目次ハンドルを解放してください。この関数は、目次ハンドル (*hTOC*) と、PTOCENTRYSTRUCT ポインターが指すデータの両方を解放します。

例

```
#include "ekdviapi.h"              // Content Manager for iSeries
  HSESSION        hSession;        // Session handle
  PITEMID         pszItemID;       // Pointer to an item ID
  USHORT          usItemType;      // The item type
  USHORT          usWipFilter;     // WIP status of search items
  USHORT          usSuspendFilter; // Suspend status of search items
  USHORT          usNbrOfClasses;  // # of index class identifiers in
                                   // pusClassIdList
  PUSHORT         pusClassIdList;  // Pointer to list of index class IDs
                                   // that indicates TOC items.
  PVOID           pLinkCriteria;   // Not used
  PASYNCCTLSTRUCT pAsyncCtl;       // Pointer to asynchronous control block.
  RCSTRUCT        RC;              // Pointer to return data structure.
  USHORT            usNumRows = 0; // # of returned TOC entries
  PTOCENTRYSTRUCT   pTocEntry;     // pointer to TOC entries

  usItemType       = SIM_ALL;      // Set up item type filter.
  usWipFilter      = OIM_ALL;      // Set up Work-In-Process status filter
  usSuspendFilter  = OIM_ALL;      // Set up suspend status of search items.
  usNbrOfClasses   = 1;            // Set up index class filter
  usClassIdList[0] = NO_INDEX;

  ulRC = SimLibGetTOC(
            hSession,              // Handle to a Content Manager for iSeries.
            pfoldid,               // Pointer to folder or Workbasket ID.
            SIM_ALL,               // The item type filter.
            NULL,                  // WIP status of search items.
            NULL,                  // Suspend status of search items.
            usNbrOfClasses,        // # of index class IDs in pusClassIdList.
            usClassIdList,         // Pointer to index class identifiers list.
            NULL,                  // Not used; link criteria
            NULL,                  // asynch not supported
```

```
          &RC                   // pointer to return struct
          );
if (ulRC == SIM_RC_OK) {
   hTOC      = (HTOC)RC.ulParam2;// TOC handle
   usNumRows = RC.usParam;       // # of returned toc entries
   pTocEntry = RC.ulParam1;      // pointer to TOC entries.
}

/*********************************************************/
/* ... Call other Content Manager for iSeries by using the  ... */
/* ... session handle obtained by calling SimLibLogon ... */
/*********************************************************/

ulRC = Ip2CloseTOC(
          hSession,             // Handle to a Content Manager for iSeries
          hTOC,                 // TOC Handle from SimLibGetTOC
          NULL,                 //  by NULL, asynchronous call made
          &RC                   // pointer to return struct
          );
if (ulRC == SIM_RC_OK) {
   /* Ip2CloseTOC released all resource associated with hTOC */
}
```

関連関数

- **Ip2CloseTOC**
- **Ip2GetTOCUpdates**
- **Ip2TOCCount**
- **Ip2TOCStatus**
- **SimLibGetTOCData**

SimLibGetTOCData (項目のグループ属性のスナップショットの取得)

形式

SimLibGetTOCData(*hSession, pTOCEntries, ulEntryCount, fDataOptions, pAsyncCtl, pRC* **)**

目的

SimLibGetTOCData 関数を使用して、文書またはフォルダーのグループに関連づけられている属性のコピーを戻します。

アプリケーションは、**SimLibGetItemSnapshot** 関数への一連の呼び出しの代わりにこの関数を使用することができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pTOCEntries PTOCENTRYSTRUCT - 入力

属性をコピーする必要のある項目を識別する、TOCENTRYSTRUCT データ構造の配列へのポインター。このデータ構造の詳細については、178 ページの『TOCENTRYSTRUCT (目次項目データ構造)』を参照してください。

ulEntryCount ULONG - 入力

TOCENTRYSTRUCT 配列内の項目数。各項目で大量のデータが生じる可能性があるので、項目の数を制限する必要があります。

fDataOptions BITS - 入力

各項目について戻すデータのタイプ。このパラメーターには、値を少なくとも 1 つ指定する必要があります。有効な値は次のとおりです。これらの値を組み合わせるために、ビット単位の包含 OR 演算子 (|) を使用することができます。

SIM_TOC_SNAPSHOT_SYSTEM_ATTR
文書あるいはフォルダーのシステム定義の属性値を戻します。

SIM_TOC_SNAPSHOT_USER_ATTR
文書あるいはフォルダーのユーザー定義の属性値を戻します。

SIM_TOC_SNAPSHOT_WORK_ATTR
文書あるいはフォルダーのワーク・マネージメント情報を戻します。

SIM_TOC_SNAPSHOT_ALL
他のあらゆる値に指定した情報を戻します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはデータ域へのポインターが入ることを示します。エラーが発生すると、*usParam* には 0 が入ります。

ulParam1 戻された情報を供給する SNAPSHOTSTRUCT データ構造の配列へのポインターが入ります。

usParam が値 0 である場合、*ulParam1* にはエラー状況にある TOCENTRYSTRUCT 要素の配列索引が入ります。一部のエラー状態については、失敗した項目をこの関数で識別することができます。0 でない場合、このフィールドには *SIM_TOC_MAX_ENTRY_COUNT* が入ります。

ulParam2 戻された配列内の項目数が入ります。この数は、*ulEntryCount* パラメーターの値と同じです。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_BUFFER_NULL
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_INDEX_CLASS
- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_ITEM_TYPE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_READATTRIND
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果:

- この関数で、正しく識別したフォルダーあるいは文書の、どのグループのデータも検索できます。この関数は、**SimLibGetTOC** 関数によって戻される項目の情報を検索し、1 回の関数呼び出しでリスト全体を処理します。ワーク・マネージメント情報の検索は、属性の検索よりも相当に時間がかかります。
- *fDataOptions* パラメーターに指定するビット値によって結果が異なります。
  - SIM_TOC_SNAPSHOT_SYSTEM_ATTR を指定してシステム定義の属性を戻す場合は、項目が有効な文書あるいはフォルダーであれば、常にデータを取得します。
  - SIM_TOC_SNAPSHOT_WORK_ATTR を指定したが、項目がワーク・バスケット内にない場合には、正常な戻りコードを得ますが、WMSNAPSHOTSTRUCT データ構造は空 (NULL) になります。
  - 0 あるいは無効な組み合わせのビット値を指定した場合、この関数では SIM_RC_INVALID_DATA_OPTIONS を戻します。
- 戻されたデータはすべて、単一メモリー・ブロック内にあります。SNAPSHOTSTRUCT 構造は、 TOCENTRYSTRUCT 構造と同じ順序の配列で示されます。残りの情報は、個々の SNAPSHOTSTRUCT 構造内に発生したポインターで参照される、同じブロック内に続きます。

例外:

- この関数は、TOCENTRYSTRUCT のほとんどのフィールドを無視します。この関数は常に項目 ID フィールドを使用し、さらにユーザー定義属性の要求時に索引クラスを使用します。そのため、この関数を使用し、 TOCENTRYSTRUCT 構造を準備したり、 *fDataOptions* パラメーターの SIM_TOC_SNAPSHOT_SYSTEM_ATTR 値だけを使用して、フォルダー・リストおよび文書リストの項目タイプを検索できます。この関数は、SNAPSHOTSTRUCT 構造内の正しい項目タイプを戻します。

- アプリケーションは、ASCII - 整数間ルーチンなどの変換ルーチンを使用して、属性値の文字表現をアプリケーションに適した形式に変更しなければならない場合があります。

後続作業: Content Manager for iSeries システムが SNAPSHOTSTRUCT データ構造に戻す情報を、アプリケーションが処理した後で、 **SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、SNAPSHOTSTRUCT データ構造配列へのポインターを解放してください。

関連関数
- **SimLibCloseAttr**
- **SimLibFree**
- **SimLibGetItemSnapshot**
- **SimLibGetItemType**
- **SimLibGetTOC**
- **SimLibOpenItemAttr**
- **SimLibReadAttr**

SimLibListClasses (索引クラスのリスト)

形式

SimLibListClasses(*hSession, fClassOptions, pAsyncCtl, pRC* **)**

目的
SimLibListClasses 関数を使用して、Content Manager for iSeries データベース内のすべての既存の索引クラスをリストします。この関数は、このユーザーがアクセスし、属性を含むものに関するクラスのみをリストします。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。 **SimLibLogon** 関数は、セッション情報を作成します。

fClassOptions BITS - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値
この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。その他の場合、このフィールドには値 0 が入ります。

ulParam1 *ulParam2* に 0 より大きな値が入る場合、このフィールドにはバッファーへのポインターが入ります。バッファー内の NAMESTRUCT 配列は、索引クラス ID および関連名を提供します。このデータ構造の詳細については、168 ページの『NAMESTRUCT (名前データ構造)』を参照してください。

ulParam2 *ulParam1* が指す配列のフィールドの数が入ります。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果: この関数が戻す名前情報は、現行の Content Manager for iSeries セッションに対して定義された言語を反映します。

例外: この関数は、現行ユーザーがアクセスを許可されているシステム内の索引クラス ID のみを提供します。**SimLibGetClassInfo** 関数を使用して、索引クラスの索引属性を判別してください。

後続作業: アプリケーションで索引クラス ID のリストが不要になった場合、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、バッファーを解放してください。

SimLibLogoff (ログオフ)

形式

SimLibLogoff(*hSession, pAsyncCtl, pRC* **)**

目的

SimLibLogoff 関数を使用して、現行アプリケーションに関する Content Manager for iSeries 操作へのアクセスを終了します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC

使用の手引き

結果:

- アプリケーションがこの関数を使用した後では、同じセッション・ハンドルを使用する場合は、いかなる追加の Content Manager for iSeries 関数を使用しても失敗します。
- Content Manager for iSeries API が割り振り、 **SimLibFree** 関数を使用して解放されなかったすべての構造は、ログオフ時に解放されます。

例

```
#include <stdio.h>                    /* Standard I/O header files    */
#include "ekdviapi.h"                 /* Content Manager for iSeries  */

int main (void) {
   ULONG    ulRC;                     /* Return code                  */
   HSESSION hSession;                 /* Session handle               */
   PUSERLOGONINFOSTRUCT pUserLogonInfo; /* User logon info struct     */
   PSZ      pszDBName="VI400LIB";     /* Pointer to Database name     */
   PSZ      pszUserId="QVIADMIN";     /* Pointer to User Id (Name)    */
   PSZ      pszPassword="PASSWORD";   /* Pointer to User's Password   */
   BITS     fSessionType=1;           /* Product Session Type         */
   RCSTRUCT RC;                       /* RC data structure            */

   /************************************************/
   /* Logon to system, and establish a normal session */
   /************************************************/
   fSessionType = SIM_SS_NORMAL;
   ulRC = SimLibLogon(
             pszDBName,             // library database
             NULL,                  // not used; library tableset
```

```
            pszUserId,             // user ID
            pszPassword,           // user ID password
            NULL,                  // if any, new password
            NULL,                  // not used; proxy ID
            NULL,                  // not used; proxy scope
            fSessionType,          // session access
            NULL,                  // NULL = synchronous call
            &RC                    // pointer to return data struct
            );
   if (ulRC == SIM_RC_OK
      // hSession session handle and user logon info structure
      // returned through RC structure.
      hSession       = (HSESSION)RC.ulParam1;
      pUserLogonInfo = (PUSERLOGONINFOSTRUCT)RC.ulParam2;
   } else {
      printf("error -SimLibLogon failed with %ld.¥n",ulRC);
      exit(1);
   }

   /*********************************************************/
   /* Call other Content Manager for iSeries APIs by using the */
   /* session handle obtained by calling SimLibLogon           */
   /*********************************************************/

   /*******************************************************/
   /* Logoff from system, and end a normal session       */
   /*******************************************************/
   ulRC = SimLibLogoff(
            hSession,           // Session handle
            NULL,               // not supported
            &RC                 // pointer to return data struct
            );
   if (ulRC == SIM_RC_OK) {
      /******************/
      /* Logoff success */
      /******************/
   } else {
      printf("error - SimLibLogoff failed with %ld¥n.",ulRC);
      exit(1);
   }
   return (0);
}
```

関連関数

- **SimLibLogon**

SimLibLogon (ログオン)

形式

SimLibLogon(*pszDBName, pszApplicationName, pszUserID, pszPassword, pszNewPassword, pszProxyID, pszProxyScope, fSession, pAsyncCtl, pRC* **)**

目的

SimLibLogon 関数を使用して、アプリケーションが Content Manager for iSeries 操作にアクセスできるようにします。アプリケーションは、この関数を使用してから他の Content Manager for iSeries 関数を使用する必要があります。また、Content Manager for iSeries 操作の使用を終了する際には **SimLibLogoff** を使用する必要があります。

パラメーター

pszDBName PSZ - 入力

FRNOLINT.TBL に入っているシステム名。

pszApplicationName
PSZ - 入力

サポートされていません。

pszUserID PSZ - 入力

ログオンするユーザーのユーザー ID を指定する NULL 文字で終了する文字ストリング。大文字小文字の区別をしません。

pszPassword PSZ - 入力

ユーザー ID のパスワードを指定する NULL 文字で終了する文字ストリング。大文字小文字の区別は、 iSeries オペレーティング・システム定義のシステム値 QPWDLVL に基づいて行われます。

pszNewPassword
PSZ - 入力

ユーザー ID の有効なパスワードを新たに指定する NULL 文字で終了する文字ストリング。大文字小文字の区別は、 iSeries オペレーティング・システム定義のシステム値 QPWDLVL に基づいて行われます。既存のパスワードを保持するために、NULL を使用してください。

pszProxyID PSZ - 入力

サポートされていません。

pszProxyScope PSZ - 入力

サポートされていません。

fSession BITS - 入力

SIM_SS_NORMAL
ログオン・プロセスの一部として、索引クラスおよび属性情報が検索されます。これにより、後続の呼び出しのパフォーマンスが向上します。

SIM_SS_CONFIG
USERLOGONINFOSTRUCT のみがサーバーから戻されます。このデータ構造の情報の詳細については、 179 ページの『USERLOGONINFOSTRUCT (ユーザー・ログオン情報構造)』 を参照してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入り、*ulParam1* にはセッション・ハンドルが、 *ulParam2* にはバッファーへのポインターが入っていることを示します。

ulParam1 *hSession* パラメーターまたは NULL が入ります。

ulParam2 USERLOGONINFOSTRUCT データ構造へのポインターが入っています。このデータ構造の情報の詳細については、 179 ページの『USERLOGONINFOSTRUCT (ユーザー・ログオン情報構造)』 を参照してください。

ulRC 以下の戻りコードのうちのいずれかが入ります。「使用の手引き」に、詳しく説明されています。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_GRACE_PERIOD_ENDED
- SIM_RC_GRACE_PERIOD_OVER_LIMIT
- SIM_RC_INVALID_PASSWORD
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USERID
- SIM_RC_USERID_UNKNOWN

関数が正常に完了すると、ゼロの値 (SIM_RC_OK) を戻します。

使用の手引き

後続作業: アプリケーションが USERLOGONINFOSTRUCT データ構造から情報を得たあとは、 **SimLibFree(** *hSession,* (PVOID)*ulParam2, pRC* **)** 関数を使用して、メモリーを解放してください。

例

```
#include <stdio.h>                      /* Standard I/O header files    */
#include "ekdviapi.h"                   /* Content Manager for iSeries  */

int main (void) {
   ULONG    ulRC;                       /* Return code                  */
   HSESSION hSession;                   /* Session handle               */
   PUSERLOGONINFOSTRUCT pUserLogonInfo; /* User logon info struct*/
   PSZ      pszDBName="VI400LIB";       /* Pointer to Database name     */
   PSZ      pszUserId="QVIADMIN";       /* Pointer to User Id (Name)    */
   PSZ      pszPassword="PASSWORD";     /* Pointer to User's Password   */
   BITS     fSessionType=1;             /* Product Session Type         */
   RCSTRUCT RC;                         /* RC's data structure          */

   /************************************************/
   /* Logon to system, and establish a normal session */
   /************************************************/
   fSessionType = SIM_SS_NORMAL;
   ulRC = SimLibLogon(
             pszDBName,                // library database
             NULL,                     // not used; library tableset
             pszUserId,                // user ID
             pszPassword,              // user ID password
             NULL,                     // if any, new password
```

```
            NULL,                    // not used; proxy ID
            NULL,                    // not used; proxy scope
            fSessionType,            // session access
            NULL,                    // not supported
            &RC                      // pointer to return data struct
            );
   if (ulRC == SIM_RC_OK ||

      // hSession session handle and user logon info structure
      // returned through RC structure.
      hSession       = (HSESSION)RC.ulParam1;
      pUserLogonInfo = (PUSERLOGONINFOSTRUCT)RC.ulParam2;
   } else {
      printf("error -SimLibLogon failed with %ld.¥n",ulRC);
      exit(1);
   }

   /*********************************************************/
   /* Call other Content Manager for iSeries APIs by using the */
   /* session handle obtained by calling SimLibLogon          */
   /*********************************************************/

   /*******************************************************/
   /* Logoff from system, and end a normal session       */
   /*******************************************************/
   ulRC = SimLibLogoff(
            hSession,                // Session handle
            NULL,                    // NULL indicates synchronous call
            &RC                      // pointer to return data struct
            );
   if (ulRC == SIM_RC_OK) {
      /******************/
      /* Logoff success */
      /******************/
   } else {
      printf("error - SimLibLogoff failed with %ld¥n.",ulRC);
      exit(1);
   }
   return (0);
}
```

関連関数

- **SimLibFree**
- **SimLibLogoff**

SimLibOpenItemAttr (項目属性のオープン)

形式

SimLibOpenItemAttr(*hSession, pszItemID, usClassId, ulAccessLevel, pAsyncCtl, pRC* **)**

目的

SimLibOpenItemAttr 関数を使用して、ユーザーが指定した文書またはフォルダーの属性にアクセスできるようにします。この関数は、項目と関連する属性の仮想コピーを作成して、読み取りアクセスあるいは書き込みアクセスのためにその項目をオープンします。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszItemID PITEMID - 入力

オープンして属性にアクセスしようとしている項目の ID。この ID は、項目 ID です。

usClassId USHORT - 入力

索引クラスの ID。

ulAccessLevel ULONG - 入力

項目アクセス・モード。このパラメーターの値は、項目をロックするためのアクセス・モードを示します。有効な値は、次のとおりです。

SIM_ACCESS_READ_WRITE
項目をロックします。この値を使用すると、他の処理でロックされている項目がある場合、この関数は失敗します。

SIM_ACCESS_SHARED_READ
読み取りアクセスのみのために項目をオープンします。この値を使用すると、他のユーザーが項目をロックしたかどうかに関係なく、項目がオープンされます。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。戻りコードが SIM_RC_ITEM_CHECKEDOUT の場合には、このフィールドには値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。Content Manager for iSeries システムが別のエラーを戻す場合、このフィールドには値 NULL が入ります。

ulParam1 オープンした項目について、データ型 HITEM を持つ項目ハンドルが入ります。戻りコードが SIM_RC_ITEM_CHECKEDOUT の場合、このフィールドには USERACCESSSTRUCT データ構造を指すポインターが入ります。このデータ構造には、項目をロックしたユーザーのユーザー ID が入ります。

ulParam2 項目の索引クラスを戻します。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_ASYNC_STARTED
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INUSE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_INDEX_CLASS
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USACCESSLEVEL_VALUE
- SIM_RC_INVALID_USATTRIBUTEID_VALUE
- SIM_RC_INVALID_USCLASSID_VALUE
- SIM_RC_ITEM_CHECKEDOUT
- SIM_RC_ITEM_NOT_FOUND
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PITEM_NOT_FOLDER_OR_DOCUMENT
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果:

- アプリケーションが読み取りアクセスでこの関数を使用する場合には、Content Manager for iSeries システムは、データベース内に現行属性値のコピーを作成します。他のユーザーによる同時並行アクセスあるいは後続アクセスでは、それらの値を変更できます。
- また別のアプリケーションの書き込みアクセスのために項目がオープンしているときに、このアプリケーションで読み取りアクセスのために項目をオープンする場合、項目属性の値はデータベース内の項目属性の現行値と同一です。
- 既に書き込みアクセスのために項目をオープンしている場合は、この関数は SIM_RC_INUSE を戻します。
- この関数は、仮想項目のハンドルを戻します。このハンドル *hItem* は、現行セッション内においてのみ有効です。他のセッションへの転送はできません。項目の属性を操作するためには、 **SimLibReadAttr** および **SimLibWriteAttr** 関数とともに項目ハンドルを使用します。永続的に新規の値をコピーするには、**SimLibSaveAttr** または **SimLibCloseAttr** を使用します。
- **SimLibOpenItemAtt** では、ユーザーが SIM_ACCESS_READ_WRITE 権限を持っているかどうかについての妥当性検査は行われません。この妥当性検査は、SIM_OPT_SAVE パラメーターによって **SimLibCloseAtt** を呼び出したときに行われます。

例外:

- 項目がロックされている場合には、そのロックされた項目を持つユーザーだけがその項目を処理することができます。他のユーザーは、読み取りアクセスのみを取得することができます。

- 項目がロックされていない場合、すべてのユーザーが読み取りアクセスを取得でき、書き込みアクセスを要求する適切な権限を持つ最初のユーザーが排他的更新アクセスを取得します。
- 別のユーザーが **SimLibSaveAttr** 関数を使用して、項目の属性値を保管しないで修正する場合、ユーザーが見る属性値は、その別のユーザーが見る属性値とは異なる可能性があります。

後続作業:

- 戻りコード SIM_RC_ITEM_CHECKEDOUT が戻され、アプリケーションでユーザー・アクセス情報が不要になった場合、 **SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用してバッファーを解放してください。
- 戻りコード SIM_RC_OK が戻された場合には、 **SimLibCloseAttr** を使用して項目をクローズし、項目ハンドル用のストレージを解放してください。**SimLibFree** と **SimLibCloseAttr** の両方は使用しないでください。

関連関数

- **SimLibCloseAttr**
- **SimLibReadAttr**
- **SimLibSaveAttr**
- **SimLibWriteAttr**

SimLibOpenObject (オブジェクトのオープン)

形式

SimLibOpenObject(*hSession, hObj, ulAccessLevel, ulPriority, fConflict, fOpenControl, pAsyncCtl, pRC* **)**

目的

SimLibOpenObject 関数を使用して、アプリケーションによるアクセス用に既存のオブジェクトを準備します。この関数が正常に完了すると、オブジェクトへのアクセスに使用できるオブジェクト・アクセス・ハンドルが戻されます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。 **SimLibLogon** 関数は、セッション情報を作成します。

hObj HOBJ - 入力

HOBJ データ構造内のオブジェクト・ハンドル・ブロックへのポインター。HOBJ 構造の詳細については、 161 ページの『HOBJ (記憶オブジェクトの照会のためのハンドル)』を参照してください。

ulAccessLevel ULONG - 入力

オブジェクト・アクセス・モード。このパラメーターの値は、オープンするオブジェクトのアクセス・モードを示します。

Content Manager for iSeries システムは、このアクセス状態を使用して、オープン・オブジェクトにアクセスするための同時要求を受け入れるかまたは拒否します。有効な値は、次のとおりです。

SIM_ACCESS_READ_WRITE
オブジェクトを、読み取りアクセスおよび書き込みアクセスのために、オブジェクトの最初のバイト位置でオープンします。

SIM_ACCESS_SHARED_READ
読み取りアクセスのみのために、オブジェクトの最初のバイト位置でオブジェクトをオープンします。

ulPriority ULONG - 入力

サポートされていません。

fConflict BOOL - 入力

サポートされていません。

fOpenControl BITS - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。

ulParam1 HOBJACC オブジェクト・アクセス・ハンドルの *hObjAcc* が入ります。このフィールド内の値は、アクセスしたオブジェクトの現行インスタンスを識別します。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INUSE
- SIM_RC_INVALID_ACCESS_CODE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_OBJECT_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OBJECT_CHECKEDOUT

- SIM_RC_OPEN_FAILED
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR
- SIM_RC_OBJECT_BEINGPROMOTED

使用の手引き

結果:

- この関数がオブジェクト・アクセス・ハンドルを戻す場合、このハンドルはオープン・オブジェクトへのアクセスの現行インスタンスを識別します。このハンドルは、格納されているオブジェクトを参照するのに通常使用するハンドルとは異なります。以下の関数では、オブジェクト処理 (*hObj*) ではなく、オブジェクト・アクセス・ハンドル (*hObjAcc*) を使用してください。
  - **SimLibCloseObject**
  - **SimLibReadObject**
  - **SimLibResizeObject**
  - **SimLibSeekObject**
  - **SimLibWriteObject**
- 書き込みアクセスでオブジェクトをオープンしたいときに、別のユーザーがその項目をロックしていた場合、この関数は SIM_RC_OBJECT_CHECKEDOUT を戻しますが、項目をロックしたユーザーの ID は戻しません。**SimLibGetItemInfo** 関数を使用して、そのユーザー ID を取得することができます。

例

```
SimLibLogon...

   #include <stdio.h>                   /* Standard I/O header files         */
   #include <string.h>                  /* Standard string header file       */
   #include "ekdviapi.h"                /* Content Manager for iSeries       */

 main()
{
   HSESSION hSession ;                 // from logon
   HOBJ hObj;
   UCHAR ulAccessLevel = SIM_ACCESS_SHARED_READ;
   UCHAR ulPriority = 0;               // not supported
   BOOL fConflict =   0;               // not supported
   BOOL fOpenControl = 0;              // Not supported
   RCSTRUCT RC;
   PRCSTRUCT pRC = &RC;
   POBJ          pObj;                 // Created object  handle
   USHORT        sResult;              // get rc back
   HOBJACC       hObjAcc;              // object access handle

   // create hobj
   if(0==(  pObj=(POBJ) malloc(sizeof(OBJ)))) {
      return(1);
   }
   (  pObj)->ulStruct = sizeof(OBJ);
   strcpy((  pObj)->szItemID,"DA97220AA.AAA");
   strcpy((  pObj)->chRepType,"");    // take default
   (  pObj)->ulPart = 1;
   hObj = pObj;
   /*Call the function*/

   sResult = SimLibOpenObject(
           hSession,
```

```
            hObj,
            ulAccessLevel,
            ulPriority,
            fConflict,
            fOpenControl,
            0,  // synch
            pRC);

        if (pRC->ulRC == SUCCESS) {
            // ulParam1 is HOBACC when call is successful.
            hObjAcc = pRC->ulParam1;
            // Mem containing the HOBJACC struct is freed by SimLibCloseObject.
        }
}
```

関連関数

- **SimLibCloseObject**
- **SimLibReadObject**
- **SimLibResizeObject**
- **SimLibSeekObject**
- **SimLibWriteObject**

SimLibOpenObjectByUniqueName (固有名によるオブジェクトのオープン)

形式

SimLibOpenObjectByUniqueName(*hSession, pszUniqueName, ulAccessLevel, ulPriority, fConflict, fOpenControl, pAsyncCtl, pRC* **)**

目的

SimLibOpenObjectByUniqueName は、IBM ImagePlus Workfolder Application Facility for AS/400 を使って作成された書式オーバーレイを表示するために使用します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszUniqueName PSZ - 入力

アクセスしたいオブジェクトが含まれている項目の固有の名前。

ulAccessLevel ULONG - 入力

オブジェクト・アクセス・モード。このパラメーターの値は、オープンするオブジェクトのアクセス・モードを示します。

Content Manager for iSeries システムは、このアクセス状態を使用して、オープン・オブジェクトにアクセスするための同時要求を受け入れるかまたは拒否します。有効な値は、次のとおりです。

SIM_ACCESS_READ_WRITE
オブジェクトを、読み取りアクセスおよび書き込みアクセスのために、オブジェクトの最初のバイト位置でオープンします。

SIM_ACCESS_SHARED_READ
読み取りアクセスのみのために、オブジェクトの最初のバイト位置でオブジェクトをオープンします。

ulPriority ULONG - 入力

サポートされていません。

fConflict BOOL - 入力

サポートされていません。

fOpenControl BITS - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。

ulParam1 HOBJACC オブジェクト・アクセス・ハンドルの *hObjAcc* が入ります。このフィールド内の値は、アクセスしたオブジェクトの現行インスタンスを識別します。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INUSE
- SIM_RC_INVALID_ACCESS_CODE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_OBJECT_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OBJECT_CHECKEDOUT
- SIM_RC_OPEN_FAILED
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果:

- この関数がオブジェクト・アクセス・ハンドルを戻す場合、このハンドルはオープン・オブジェクトへのアクセスの現行インスタンスを識別します。このハンドルは、格納されているオブジェクトを参照するのに通常使用するハンドルとは異なります。以下の関数では、オブジェクト・アクセス・ハンドル (*hObjAcc*) を使用してください。
  - **SimLibCloseObject**
  - **SimLibReadObject**
  - **SimLibResizeObject**
  - **SimLibSeekObject**
  - **SimLibWriteObject**
- 書き込みアクセスでオブジェクトをオープンしたいときに、別のユーザーがその項目をロックしていた場合、この関数は SIM_RC_OBJECT_CHECKEDOUT を戻しますが、項目をロックしたユーザーの ID は戻しません。**SimLibGetItemInfo** 関数を使用して、そのユーザー ID を取得することができます。

関連関数

- **SimLibCloseObject**
- **SimLibReadObject**
- **SimLibResizeObject**
- **SimLibSeekObject**
- **SimLibWriteObject**

SimLibQueryObject (オブジェクトの照会)

形式

SimLibQueryObject(*hSession, hObj, pAsyncCtl, pRC* **)**

目的

SimLibQueryObject 関数を使用して、ユーザーが指定したオブジェクトに関連する情報 (たとえば、オブジェクトのサイズ、コンテンツ・クラス、コレクション名など) を取得します。この関数は、オブジェクト情報構造のバッファーを割り振り、オブジェクトに関連するすべての情報でこの構造を満たします。オブジェクトを照会する際、オープンする必要はありません。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObj HOBJ - 入力

HOBJ データ構造内のオブジェクト・ハンドル・ブロックへのポインター。このハンドルは、照会したいオブジェクトを指定します。

HOBJ 構造の詳細については、 161 ページの『HOBJ (記憶オブジェクトの照会のためのハンドル)』を参照してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。

ulParam1 OBJINFOSTRUCT データ構造に、オブジェクトに関連する情報がすべて入っているバッファーへのポインターが入ります。このデータ構造の詳細については、 169 ページの『OBJINFOSTRUCT (オブジェクト情報構造)』を参照してください。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_ASYNC_STARTED
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_OBJECT_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PART_NOT_FOUND
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果: この関数は、OBJINFOSTRUCT 内のデータを戻します。

後続作業: この関数がオブジェクト情報を得たあとは、 **SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、バッファーを解放してください。

SimLibReadAttr (属性の読み取り)

形式

SimLibReadAttr(*hSession, hItem, usAttributeId, pAsyncCtl, pRC* **)**

目的

SimLibReadAttr 関数を使用して、ユーザーが指定したオープンされているフォルダーまたは文書の特定の属性の値を戻します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hItem HITEM - 入力

属性を読み取りたい仮想項目、オープンしているフォルダーあるいは文書のハンドル。**SimLibOpenItemAttr** 関数がこのハンドルを戻します。この項目は、読み取りあるいは書き込みアクセス・モードで現在オープンされている可能性があります。

usAttributeId USHORT - 入力

属性に割り当てられた固有 ID。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。エラーが発生すると、このフィールドには値 0 が入ります。

ulParam1 NULL 文字で終了するストリングが属性値の文字表現であるバッファーへの、ポインターが入ります。属性値が未定義である場合、この値は NULL です。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HITEM_VALUE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USATTRIBUTEID_VALUE
- SIM_RC_OUT_OF_MEMORY

使用の手引き

例外:

- 属性は常に、NULL 文字で終了するストリングとして戻されます。
- アプリケーションは、ASCII - 整数間ルーチンなどの変換ルーチンを使用して、属性値の文字表現をアプリケーションに適した形式に変更しなければならない場合があります。
- **SimLibGetAttrInfo** 関数を使用して、属性のデータ型およびデータ長を取得します。**SimLibGetItemInfo** 関数と **SimLibGetClassInfo** 関数を使用して、クラス属性を取得します。

後続作業: 属性ストリングが不要になった場合、 **SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、バッファーを解放してください。

関連関数

- **SimLibGetClassInfo**
- **SimLibGetAttrInfo**
- **SimLibGetItemInfo**
- **SimLibOpenItemAttr**

SimLibReadObject (オブジェクトの読み取り)

形式

SimLibReadObject(*hSession, hObjAcc, pBuffer, ulBytesToRead, pAsyncCtl, pRC* **)**

目的

SimLibReadObject 関数を使用して、オブジェクトからアプリケーションのデータ・バッファーに、指定されたバイト数を転送します。この関数は、オブジェクトをファイルのように操作することを可能にします。また、オブジェクト・ポインターが現在参照しているバイトからオブジェクトの読み取りを始めます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObjAcc HOBJACC - 入力

アプリケーションのデータ・バッファーに読み込みたいオープン・オブジェクトへのオブジェクト・アクセス・ハンドル。このパラメーターの値は、アクセスされたオブジェクトの現行インスタンスを識別します。

pBuffer PHBUF - 入力

データ・バッファー・ポインター。このパラメーターの値は、読み取りオブジェクト・データを戻そうとしているバッファーの、最初のバイトへのポインターを示します。

ulBytesToRead ULONG - 入力

読み取るバイト数。このパラメーターの値は、転送操作の間にオブジェクトから読み取るバイトの最大数を指定します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。

ulParam1 バッファーに書き込まれた最後のバイトの直後のバイトを指すポインターが入ります。これは、通常、読み取ったバイト数を加えたバッファーのアドレスです。

ulParam2 読み取られたバイトの実数が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_BUFFER_PTR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_OBJECT_ACCESS_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_READ_PAST_EOF

使用の手引き

準備: オブジェクトをオープンしてオブジェクト・アクセス・ハンドルを入手してから、読み取りを行ってください。

結果: 関数が正常に終了したあと、オブジェクト・ポインターは読み取られたデータの直後のバイトを参照します。

例外: 指定して読み取るバイト数がオブジェクト内のバイト数より大きい場合、この関数は、指定より少ないバイト数を転送します。

関連関数

- **SimLibCloseObject**
- **SimLibOpenObject**
- **SimLibSeekObject**

SimLibRemoveFolderItem (フォルダーからの項目の除去)

形式

SimLibRemoveFolderItem(*hSession, pszFolderID, pszItemID, pAsyncCtl, pRC* **)**

目的

SimLibRemoveFolderItem 関数を使用して、フォルダーから文書またはフォルダー項目を除去します。この関数は、指定したフォルダーの目次から項目の参照を除去します。この関数を使用するためにフォルダーをオープンする必要はありませんが、フォルダーが別のユーザーによってロックされていてはいけません。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszFolderID PITEMID - 入力

項目を除去したいフォルダーの ID。この ID は、フォルダーの項目 ID です。

pszItemID PITEMID - 入力

フォルダーから除去する項目の ID。この ID は、文書項目あるいはフォルダー項目の項目 ID です。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 戻りコードが SIM_RC_PARENT_CHECKEDOUT の場合、このフィールドには値 1 が入り、 *ulParam1* にポインターが入ることを示します。

ulParam1 値 NULL が入ります。戻りコードが SIM_RC_PARENT_CHECKEDOUT である場合、このフィールドに

は USERACCESSSTRUCT データ構造へのポインターが入ります。この構造には、フォルダーをロックしたユーザーのユーザー ID が入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_PITEMIDFOLDER_PTR
- SIM_RC_INVALID_PITEMIDFOLDER_VALUE
- SIM_RC_INVALID_PITEMIDITEM_PTR
- SIM_RC_INVALID_PITEMIDITEM_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PARENT_CHECKEDOUT
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果:

- フォルダーが別のユーザーによってロックされている場合には、そのフォルダーから項目を除去することはできません。その代わりとして、この関数は、フォルダーをロックしたユーザーのユーザー ID を戻します。

  ユーザー自身がフォルダーをロックした場合には、そのフォルダーから項目を除去することができます。

例外:

- この関数が、フォルダーの目次の一時コピーを自動的に更新することはありません。アプリケーションは、**Ip2GetTOCUpdates** 関数または **SimLibGetTOC** 関数を使用して、このフォルダーの目次を更新する必要があります。
- 自らロックした項目、または別のユーザーがロックした項目を除去することができます。親フォルダーの状況のみがテストされます。

後続作業: アプリケーションでユーザー・アクセス情報が不要になった場合は、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、USERACCESSSTRUCT データ構造が入っているバッファーを解放してください。

関連関数

- **Ip2GetTOCUpdates**
- **SimLibAddFolderItem**
- **SimLibDeleteItem**
- **SimLibFree**
- **SimLibGetTOC**

SimLibResizeObject (オブジェクトのサイズ変更)

形式

SimLibResizeObject(*hSession, hObjAcc, ulSize, pAsyncCtl, pRC* **)**

目的

SimLibResizeObject 関数を使用して、オブジェクトのサイズ (バイト単位) を、指定された新しいサイズに変更します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObjAcc HOBJACC - 入力

サイズを変更したいオブジェクトのオブジェクト・アクセス・ハンドル。このパラメーターの値は、アクセスされたオブジェクトの現行インスタンスを識別します。

ulSize ULONG - 入力

新規オブジェクト・サイズ。オブジェクト・ポインターの現在位置から始まるオブジェクト・ファイル (開始バイトを含む) を切り捨てるには、値 0 を指定します。特定のバイト・サイズに合わせてオブジェクト・ファイルを切り捨てるには、そのバイト・サイズを指定してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_OBJECT_ACCESS_HANDLE

- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_NO_WRITE_ACCESS
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_RESIZE_FAILED
- SIM_RC_SEEK_ERROR

使用の手引き

準備: SIM_ACCESS_READ_WRITE アクセスのためにオブジェクトをオープンしてから、この関数を使用してオブジェクトのサイズを変更してください。

結果:

- オブジェクト・ファイル・ポインターは、この関数の終了時にオブジェクトの末尾に設定されます。
- オブジェクトを元のものよりも小さいものと置換したい場合は、この関数を使用します。**SimLibWriteObject** 関数のあとに **SimLibResizeObject** 関数を使用して、新規データの末尾で切り捨てます。

例外: オブジェクトのサイズを大きくするためには、 **SimLibWriteObject** 関数を使用して、データをオブジェクトに追加して同時にそのサイズを大きくする必要があります。

関連関数

- **SimLibWriteObject**

SimLibSaveAttr (属性の保管)

形式

SimLibSaveAttr(*hSession, hItem, pAsyncCtl, pRC* **)**

目的

SimLibSaveAttr 関数を使用して、仮想項目の属性を永続的に保管します。この関数は仮想項目上の処理中の作業を、項目をクローズせずに、あるいはアクセス権を解放しないで保管します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hItem HITEM - 入力

仮想項目のハンドル。**SimLibOpenItemAttr** 関数がこのハンドルを戻します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_ATTRIBUTES_NOT_MODIFIED
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HITEM_VALUE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_PASSED_ATTR_DATA
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USCLASSID_VALUE
- SIM_RC_NO_WRITE_ACCESS
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR
- SIM_RC_REQUIRED_ATTRIBUTE_MISSING

使用の手引き

結果:

- 書き込みアクセスのために仮想項目をオープンして修正した場合、この関数は、データベースの属性の上に仮想項目の属性をコピーします。
- 索引クラスが変更された場合には、この関数は、新規索引クラスにユーザー定義属性の新規セットを保管し、古い属性を削除します。

関連関数

- **SimLibOpenItemAttr**

SimLibSearch (検索)

形式

SimLibSearch(*hSession, pszItemFilter, pLinkCriteria, usStatDyn, usTypeFilter, fWipFilter, usSuspendFilter, usIndexClass, usNumCriteria, pCriteria, ulMemListRequest, pAsyncCtl, pRC* **)**

目的

SimLibSearch 関数を使用して、指定されたユーザー定義属性値と一致する項目をデータベース内で見つけます。

この関数は、検索基準と一致する項目をユーザーに戻します。索引クラスを指定する場合、索引クラス内のユーザー定義属性の値を検索することができます。索引クラスを指定しない場合、この関数は、指定されたユーザー定義属性すべてを含む索引クラスのみを検索します。たとえば、「account number」が 12345 と等しい索引クラスをすべて検索する要求の場合、検索は、ユーザー定義の属性として「account number」を含むこれらの索引クラスに限定されます。索引クラスと属性の組み合わせを複数指定することができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszItemFilter PITEMID - 入力

サポートされていません。

pLinkCriteria PVOID - 入力

サポートされていません。

usStatDyn USHORT - 入力

サポートされていません。

usTypeFilter USHORT - 入力

検索する項目のタイプ。有効な値は、次のとおりです。

SIM_DOCUMENT
文書を検索します。

SIM_FOLDER
フォルダーを検索します。

SIM_FOLDER_DOC
フォルダーと文書の両方を検索します。

fWipFilter BITS - 入力

サポートされていません。

usSuspendFilter USHORT - 入力

サポートされていません。

usIndexClass USHORT - 入力

サポートされていません。

usNumCriteria USHORT - 入力

pCriteria 配列のフィールド数。

pCriteria PLIBSEARCHCRITERIASTRUCT - 入力

検索したい各ビューの検索基準を指定する配列へのポインター。*pCriteria* は、少なくとも 1 つのフィールドの配列を指す必要があります。LIBSEARCHCRITERIASTRUCT 構造の詳細については、166 ページの『LIBSEARCHCRITERIASTRUCT (検索基準情報構造)』を参照してください。

ulMemListRequest

BOOL - 入力

このパラメーターは、検索結果を戻す方法、または戻される属性値を制御します。有効な値は、次のとおりです。

SIM_SEARCH_MEMLIST
メモリー・バッファーに検索結果を戻します。

SIM_SEARCH_MEMLIST_ONE

サポートされていません。

SIM_SEARCH_USER_ATTR

メモリー・バッファーに項目の項目 ID とユーザー属性を戻します。

SIM_SEARCH_USER_SYSTEM_ATTR

メモリー・バッファーに項目 ID、ユーザー属性、およびシステム属性を戻します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはバッファーへのポインターが入ることを示します。入力検索基準と一致するものが 1 つもない場合は、このフィールドには値 0 が入ります。

ulParam1 *ulMemListRequest* パラメーターに SIM_SEARCH_MEMLIST を設定すると、このフィールドにはバッファーへの PITEMID ポインター

が入ります。バッファー内の配列は、検索基準に一致する文書およびフォルダーの項目 ID を提供します。

ulMemListRequest パラメーターに SIM_SEARCH_USER_ATTR または SIM_SEARCH_USER_SYSTEM_ATTR を設定すると、このフィールドには、検索基準に一致する項目の属性データを含む SNAPSHOTSTRUCT の配列へのポインターが入ります。

ulParam2 基準 (*ulParam1* が参照する配列内のフィールド数) と一致する項目数が入ります。LIBSEARCHCRITERIASTRUCT 構造の *ulReturnLimit* フィールドの値は、この数を制限します。

検索基準と一致するものが 1 つもない場合は、このフィールドには値 0 が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_ATTR_NOT_IN_VIEW
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_FSEARCH
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_INDEX_CLASS
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_PATTRIBUTELIST_VALUE
- SIM_RC_INVALID_PITEMIDFOLDER_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_SEARCH_STRING
- SIM_RC_INVALID_USATTRIBUTEID_VALUE
- SIM_RC_INVALID_USITEMTYPE_VALUE
- SIM_RC_INVALID_VIEWID
- SIM_RC_NO_SEARCH_CRITERIA
- SIM_RC_NO_SEARCH_VIEWS
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

325 ページの『付録 A. 検索式の指針』を参照してください。

結果:

- 入力検索基準と一致するものがない場合、この関数は正常な戻りコードを戻し、*usParam*、*ulParam1*、*ulParam2* の各フィールドにはすべて値 NULL が入ります。
- 非常に明示的に検索基準を指定すると、検索で戻される項目数を少なくできます。また、非常に一般的な検索基準を指定すると、検索のパフォーマンスが低下する可能性があります。
- 全索引クラスの検索を指定すると、この関数は自動的に、式に指定された属性を持つ索引クラスのみを検索します。

後続作業: *ulMemListRequest* パラメーターに SIM_SEARCH_MEMLIST を設定した場合、この関数で検索結果情報を得たあと、 **SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** を使用してバッファーを解放してください。

SimLibSeekObject (オブジェクトのシーク)

形式

SimLibSeekObject(*hSession, hObjAcc, ulOrigin, lOffset, pAsyncCtl, pRC* **)**

目的

SimLibSeekObject 関数を使用して、ユーザーが定義した新規位置を参照するためのオブジェクト・ポインターを調整します。そのオブジェクトの次のデータ転送操作は、この新規位置から始まります。この関数を使用して、オブジェクトを変更する前にポインターを位置づけてください。この関数は、オブジェクトをファイルのように操作することを可能にします。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。 **SimLibLogon** 関数は、セッション情報を作成します。

hObjAcc HOBJACC - 入力

調整したいオブジェクト・ポインターのあるオブジェクトのオブジェクト・アクセス・ハンドル。このパラメーターの値は、アクセスされたオブジェクトの現行インスタンスを識別します。 **SimLibOpenObject** 関数がこのハンドルを戻します。

ulOrigin ULONG - 入力

ポインター起点索引。このパラメーターの値は、オブジェクト・ポインターの初期位置を示します。有効な値は、次のとおりです。

SIM_POS_BEGIN
オブジェクトの始まりを示します。

SIM_POS_CURRENT
現行のポインター位置を示します。

SIM_POS_END
オブジェクトの終わりの次のバイト位置を示します。

lOffset LONG - 入力

起点からのバイト・オフセット。このパラメーターの値は、調整されたオブジェクト・ポインターが参照するオブジェクト内の位置を指定します。この値は、*ulOrigin* パラメーターの値として指定した位置と関連づけて指定します。この値は、負のバイト・カウントも正のバイト・カウントも表します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。

ulParam1 データ型が ULONG である現行オフセット *ulOffset* が入ります。この値は、オブジェクトの始めからのバイト単位のオフセットを示します。現在位置がオブジェクトの最初である場合、この値は 0 になります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_OBJECT_ACCESS_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_SEEK_OFFSET
- SIM_RC_INVALID_SEEK_ORIGIN
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_RESIZE_FAILED
- SIM_RC_SEEK_ERROR

使用の手引き

準備: **SimLibSeekObject** 関数を呼び出すためには、まず、 **SimLibOpenObject** 関数を呼び出すことにより、オブジェクトをオープンして *hObjAcc* を取得していなければなりません。

結果: オブジェクト・ポインターを調整して、オブジェクトの最後を超えた位置も参照できます。しかし、オブジェクトの最初より前の位置を参照しようとすると、エラー・コード SIM_RC_INVALID_SEEK_OFFSET が戻されます。

関連関数

- **SimLibOpenObject**

SimLibStageObject (オブジェクトのステージング)

形式

SimLibStageObject(*hSession, hObj, ulPriority, fStageControl, pAsyncCtl, pRC* **)**

目的

SimLibStageObject 関数は、オブジェクトを 2 次ストレージからリトリーブし (取り出し) て iSeries DASD に移すために使用します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObj HOBJ - 入力

HOBJ データ構造内のオブジェクト・ハンドル・ブロックへのポインター。HOBJ 構造の詳細については、 161 ページの『HOBJ (記憶オブジェクトの照会のためのハンドル)』を参照してください。

ulPriority ULONG - 入力

優先順位の値。オブジェクトに対するサービスの優先順位を指定します。有効な値は、次のとおりです。

SIM_PRI_IMMEDIATE
オブジェクトの対話式リトリーブを試みます。

SIM_PRI_BACKGROUND
オブジェクトのリトリーブ要求を生成します。

fStageContro BITS - 入力

オブジェクトのステージング用の制御オプション・ビット。有効な値は次のとおりです。

SIM_PREFETCH
オブジェクト・サーバーへの事前取り出しを指定します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_FOPTIONS
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_HSYNC
- SIM_RC_INVALID_OBJECT_HANDLE
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY

使用の手引き

準備: この API を使ってリトリーブ要求を生成する場合、該当のオブジェクトを実際にリトリーブするには、光学式リトリーブ・プロセッサーが起動し実行されている必要があります。

結果: この関数が正常終了すると、該当のオブジェクトに対するリトリーブ要求が生成されるか、そのオブジェクトが対話式にリトリーブされます。

関連関数
- **SimLibLogon**

SimLibStoreNewObject (既存項目への新規オブジェクトの格納)

形式

SimLibStoreNewObject(*hSession, hObj, ulConCls, pSMS, pObjBuffer, ulObjSize, lSeqAfterPart, ulAffiliatedType, pAffiliatedData, pAsyncCtl, pRC* **)**

目的

SimLibStoreNewObject 関数を使用して、新規オブジェクトを既存の項目に追加します。これは、より少ないオプションとデータ・チェックですむ **SimLibCatalogObject** 関数の能率のよいバージョンです。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。 **SimLibLogon** 関数は、セッション情報を作成します。

hObj HOBJ - 入力

オブジェクト・ハンドル・ブロックへのポインター。HOBJ 構造の詳細については、 161 ページの『HOBJ (記憶オブジェクトの照会のためのハンドル)』を参照してください。

ulConCls ULONG - 入力

オブジェクトのコンテンツ・クラス ID (329 ページの『付録 B. 事前定義済みコンテンツ・クラス』を参照)。このパラメーターの値は、新規オブジェクトにあるデータの種類を知らせます。

未定義のコンテンツ・クラスを指示するには、このパラメーターに値 SIM_CC_UNKNOWN を指定します。ただし、未定義のコンテンツ・クラスを作成した場合には、他のアプリケーションは、Content Manager for iSeries コンテンツ・クラス・サービスを使用して、ユーザーが格納するオブジェクトの内容を操作する方法を判別することはできません。

pSMS PSMS - 入力

オブジェクトのシステム管理ストレージ (SMS) 構造へのポインター。この構造では *szCollectionName* のみを使用します。

pObjBuffer PVOID - 入力

オブジェクト・データが入っているメモリー・バッファーへのポインター。

ulObjSize ULONG - 入力

バイト単位のオブジェクトの合計サイズ。

lSeqAfterPart LONG - 入力

サポートされていません。

ulAffiliatedType LONG - 入力

格納する系列下オブジェクトのタイプ。定義済みの値は次のとおりです。

SIM_ANNOTATION
フォルダーまたは文書に関連した注釈を格納します。

SIM_BASE
MO:DCA や TIFF ファイルなどの基本オブジェクトを格納します。これらは、フォルダーまたは文書に関連する注釈、注、イベントではありません。

SIM_EVENT
フォルダーまたは文書に関連したイベントを格納します。

SIM_MGDS
フォルダーまたは文書に関連した MGDS (マシン生成データ・ストリーム) を格納します。

SIM_NOTE
フォルダーまたは文書に関連した注を格納します。

pAffiliatedData PVOID - 入力

ANNOTATIONSTRUCT タイプのデータ構造へのポインター。*ulAffiliatedType* パラメーターに値 SIM_ANNOTATION が含まれる場合、*pAffiliatedData* は、オブジェクト系列下に追加データが含まれるこの構造を指します。それ以外の場合は、Content Manager for

iSeries システムはこのパラメーターを無視します。
ANNOTATIONSTRUCT 構造の詳細については、 153 ページの『ANNOTATIONSTRUCT (注釈情報構造)』を参照してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_ANNOTATIONSTRUCT_PTR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_SMS_PTR
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

準備:

- *ulConCls* パラメーターでサポートされている値を取得するには、**lp2ListContentClasses** 関数を使用してください。
- この部分番号に 0 が指定されている場合には、次の順次部分番号が作成されます。部分番号がゼロ以外の場合には、その部分番号が既存のものでなければ、その部分番号が使用されます。その部分番号が既に存在する場合には、最初の使用可能な番号が戻されます。部分番号 1 は通常、基本部分です。この API により、部分番号 1 を作成する前に、部分番号 2 (たとえば、注) を作成することができます。

例外: Content Manager for iSeries システムでは、コンテンツ・クラス・パラメーターは、定義済みで、既知のコンテンツ・クラスと見なされ、妥当性検査はされません。

関連関数
- **Ip2ListContentClasses**
- **SimLibCatalogObject**

SimLibWriteAttr (属性への書き込み)

形式

SimLibWriteAttr(*hSession, hItem, usAttributeId, pszAttributeValue, pAsyncCtl, pRC* **)**

目的
SimLibWriteAttr 関数を使用して、オープン項目に関連する属性に値を割り当てます。ユーザー定義属性のみを変更することができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hItem HITEM - 入力

仮想項目のハンドル。**SimLibOpenItemAttr** 関数がこのハンドルを戻します。

SimLibWriteAttr 関数を使用するには、その時点で、項目が書き込みアクセス・モードでオープンされていなければなりません。

usAttributeId USHORT - 入力

属性に割り当てられた固有 ID。

pszAttributeValue

PSZ - 入力

属性の値が入る、NULL 文字で終了するストリング。このストリングには、 *usAttributeId* パラメーターに指定する属性に割り当てた値が入ります。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値
この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_ATTRIBUTE_READ_ONLY
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HITEM_VALUE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PASSED_ATTRIBUTE_DATA
- SIM_RC_INVALID_PATTRIBUTE_PTR
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USATTRIBUTEID_VALUE
- SIM_RC_NO_WRITE_ACCESS
- SIM_RC_OUT_OF_MEMORY

使用の手引き

準備: 整数 - ASCII 間ルーチンなどの変換ルーチンを使用して、数値データをこの関数に適した文字ストリングに変更します。

結果:

- この関数は、*pszAttributeValue* パラメーターの値を仮想項目にコピーします。
- 項目は、書き込みアクセスでオープンしなければなりません。そうしなければ、この関数は、エラー SIM_RC_NO_WRITE_ACCESS を戻します。
- この関数の使用が失敗した場合には、Content Manager for iSeries システムは、現行の属性値を維持します。

例外:

- **SimLibWriteAttr** 関数は、SIM_ATTR_FSTRING データ型のみの妥当性検査を行います。この関数は、属性データの最大長を Content Manager for iSeries 定義のストリングと比較して、これらのデータ型を検査します。**SimLibCloseAttr** および **SimLibSaveAttr** 関数は、データを **SimLibWriteAttr** 関数を通して構成したデータ型と比較して属性内容を検査します。
- **SimLibWriteAttr** 関数は、メモリーの仮想コピーのみを変更します。属性の永続データベース・コピーは更新しません。**SimLibSaveAttr** または **SimLibCloseAttr** 関数を使用して、変更内容を永続的なものにします。

関連関数

- **SimLibCloseAttr**
- **SimLibGetAttrInfo**
- **SimLibGetClassInfo**
- **SimLibOpenItemAttr**
- **SimLibSaveAttr**

SimLibWriteObject (オブジェクトへの書き込み)

形式

SimLibWriteObject(*hSession, hObjAcc, pBuffer, ulBytesToWrite, pAsyncCtl, pRC* **)**

目的

SimLibWriteObject 関数を使用して、アプリケーションのデータ・バッファーからオープン・オブジェクトに、指定されたバイト数を転送します。書き込み操作は、現行のオブジェクト・ポインターが参照しているバイトから始まります。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hObjAcc HOBJACC - 入力

書き込みを行いたいオブジェクトのオブジェクト・アクセス・ハンドル。このパラメーターの値は、アクセスされたオブジェクトの現行インスタンスを識別します。

pBuffer PHBUF - 入力

データ・バッファー・ポインター。このパラメーターの値は、オブジェクトに書き込まれるデータの最初のバイトを指すポインターを表します。

ulBytesToWrite ULONG - 入力

オブジェクトに書き込むバイト数。このパラメーターの値は、転送操作の間にオブジェクトに書き込む最大バイト数を指定します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。

ulParam1 実際に書き込まれたバイト数が入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_BUFFER_PTR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_OBJECT_ACCESS_HANDLE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_NO_WRITE_ACCESS
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_RESIZE_FAILED

使用の手引き

準備:

- 以下の関数のいずれかを使用して、 SIM_ACCESS_READ_WRITE アクセスでオブジェクトをオープンしてから、この関数を使用してください。
  - **SimLibOpenObject**
  - **SimLibCreateObject**
  - **SimLibCatalogObject**
- オブジェクトを元のものよりも小さいものと置換しようとしている場合、まず、**SimLibResizeObject** 関数を使用して、置換オブジェクトのサイズまで元のオブジェクトを切り捨てます。その後で、 **SimLibWriteObject** 関数を使用して、オブジェクトを置換することができます。置換オブジェクトが元のものよりも大きい場合、最初のサイズ変更は不要です。

結果: この関数が正常に完了すると、オブジェクト・ポインターは書き込まれたデータの直後のバイトを参照します。

例

```
   #include <stdio.h>                    /* Standard I/O header files         */
   #include <string.h>                   /* Standard string header file       */
   #include "ekdviapi.h"                 /* Content Manager for iSeries       */

 main()
{
   HSESSION hSession;                   // get from logon
   HOBJACC hObjAcc;                     // get from catalog, open, or create
   RCSTRUCT RC;
   PRCSTRUCT pRC = &RC;
   USHORT        sResult;               // return codes
   CHAR  pBuffer[4096];                // buffer
   ULONG ulBytesToWrite = 2048;

   /* fill buffer */

   /*Call the function*/

   sResult = SimLibWriteObject(
           hSession,
           hObjAcc,
           pBuffer,
           ulBytesToWrite,
           pAsyncCtl,
           pRC);
```

```
   if ((pRC->ulRC == SIM_RC_OK) &&; (ulBytesToWrite != pRC->ulParam1))
     printf("not all the bytes got written");

}
```

関連関数

- **SimLibCatalogObject**
- **SimLibCreateObject**
- **SimLibOpenObject**
- **SimLibResizeObject**
- **SimLibWriteObject**

SimWmActivateWorkPackage (作業パッケージの活動化)

形式

SimWmActivateWorkPackage(*hSession, ulWorkPackageID, ulInstanceID, pAsyncCtl, pRC* **)**

目的

SimWmActivateWorkPackage 関数は、中断した作業パッケージをリリースするために使用します。

パラメーター

hSession HSESSION — 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

ulWorkPackageID

ULONG — 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG — 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC

関連関数

- **SimWmSuspendWorkPackage**

SimWmBeginProcess (事前定義プロセスによる作業パッケージ処理の開始)

形式

SimWmBeginProcess(*hSession, pszProcessID, pszRouteName, pszWorkPackageDesc, ulNumVariables, pVariableList, usPriority, pAsyncCtl, pRC* **)**

目的

SimWmBeginProcess は、適切な項目を含む作業パッケージを作成し、事前定義プロセスに基づいてそのパッケージの処理を開始するために使用します。

パラメーター

hSession HSESSION — 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

pszProcessID PSZ - 入力

プロセスの ID。

pszRouteName PSZ - 入力

プロセス内の初期経路の名前を指すポインター。ポインターが NULL の場合は、指定したプロセス内のデフォルトの経路が使用されます。

pszWorkPackageDesc

PSZ - 入力

作業パッケージ記述を指定する NULL 文字で終了する文字ストリング。これは、タスクに関する注釈として、または作業に関するアプリケーション・データベースの詳細のための手掛かりとしてアプリケーションが使用する情報として、使用することができます。

ulNumVariables ULONG — 入力

変数配列内の項目数。最大で 2 つの項目を指定できます。配列 *pVariableList* ポインターが NULL である場合には、このフィールドは無視されます。

pVariableList PWMVARSTRUCT — 入力

ワーク・マネージメント変数の変数 ID および値を含んでいる WMVARSTRUCT 構造の配列へのポインター。

有効な変数名は以下のとおりです。

SIMWM_ITEMID
SIMWM_ITEMID の有効値は、文書またはフォルダーの項目 ID です。

SIMWM_INDEX_CLASS
SIMWM_INDEX_CLASS の有効値は、索引クラスの ID です。

usPriority USHORT - 入力

実行する作業の優先順位。この優先順位により、作業パッケージの処理順序が決定します。数値が大きいほど、優先順位が高くなります。デフォルトの優先順位を要求するには、ゼロの優先順位を使用してください。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 常にゼロです。

ulParam1 作業パッケージ ID が入ります。

ulParam2 作業パッケージ・インスタンスが入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- OIM_WB_FULL
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_INDEX_CLASS
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC

- SIM_RC_INVALID_PROCESS_NAME
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

準備: 作業パッケージを索引クラスの項目に関連づけるために、変数 SIMWM_INDEX_CLASS および SIMWM_ITEMID を指定してください。直接データベースを参照しない作業パッケージを反映するために、*pVariableList* パラメーターを NULL にすることができます。*pVariableList* が指定されていない場合には、呼び出し側のアプリケーションは、処理されるオブジェクトに作業パッケージ ID を関連づけておかなくてはなりません。

経路名を指定しない場合、作業パッケージは所定の事前定義プロセス内の最初の経路に転送されます。

例外: **SimWmBeginProcess** を使って、あるプロセスによる作業パッケージ処理を開始した場合、ワーク・バスケットのオーバーロード制限は無視されます。したがって、作業パッケージは必ずワーク・バスケットに追加されます。ただし、作業パッケージがオーバーロード制限値に達しているワーク・バスケットに入れられた場合には、そのことを示す戻りコード OIM_WB_FULL が戻ります。

SimWmChangeVariables (作業パッケージの可変値の変更)

形式

SimWmChangeVariables(*hSession, ulWorkPackageID,ulInstanceID, ulNumVariables, pVariableList, pAsyncCtl, pRC* **)**

目的

SimWmChangeVariables 関数を使用すると、作業パッケージに関連付ける新規の変数を作成したり、既存の変数を更新したりすることができます。

パラメーター

hSession HSESSION — 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

ulWorkPackageID

ULONG — 入力

作業パッケージの ID。

ulInstanceID ULONG — 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

ulNumVariables ULONG — 入力

変数配列内の項目数。

pVariableList PWMVARSTRUCT — 入力

ワーク・マネージメント変数の変数 ID および値を含んでいる WMVARSTRUCT 構造の配列へのポインター。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_WM_VARIABLE

使用の手引き

準備: 事前定義の変数 SIMWM_ACTION (*ACTION) は、あるユーザーが最後に選択したアクションを識別するために、IBM Content Manager for iSeries クライアントが使用します。この変数に割り当てる値は、アクション・リストの定義に基づいて決定します。

例外: 変数 SIMWM_ITEMID (*ITEMID) および SIMWM_INDEX_CLASS (*INDEXCLASS) は、内部用として予約済みであるため、**SimWmChangeVariables** 関数を使って新規に作成したり変更したりすることはできません。

関連関数

- **SimWmQueryVariables**

SimWmCreateWorkPackage (作業パッケージの作成)

形式

SimWmCreateWorkPackage(*hSession, pszWorkPackageDesc, ulNumVariables, pVariableList, usWorkPriority, pAsyncCtl, pRC* **)**

目的

SimWmCreateWorkPackage 関数を使用すると、特別な作業を制御するためにアプリケーションが随時使用できる作業パッケージを作成できます。これにより、アプリケーションは事前定義プロセスを必要とせずに、1 つ以上のワーク・バスケットを使って、フォルダーや文書を含んだ作業パッケージを経路指定することができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszWorkPackageDesc

PSZ - 入力

作業パッケージの記述へのポインター。これは、タスクに関する注釈として、または作業に関するアプリケーション・データベースの詳細のための手掛かりとしてアプリケーションが使用する情報として、使用することができます。

ulNumVariables ULONG - 入力

変数配列内の項目数。配列 *pVariableList* ポインターが NULL である場合には、このフィールドは無視されます。

pVariableList PWMVARSTRUCT - 入力

ワーク・マネージメント変数の変数 ID および値を含んでいる WMVARSTRUCT 構造の配列へのポインター。直接データベースを参照しない作業パッケージ、またはアプリケーションがオブジェクトに関連づける作業パッケージを反映するために、このパラメーターを NULL にすることができます。作業パッケージを索引クラスの項目に関連づけるために、変数 SIMWM_INDEX_CLASS および SIMWM_ITEMID を組み込んでください。

usWorkPriority USHORT - 入力

実行する作業の優先順位。この優先順位により、ワーク・バスケットでの作業パッケージの処理順序が決定します。数値が大きいほど、優先順位が高くなります。デフォルトの優先順位を要求するには、ゼロの優先順位を使用してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 常にゼロです。

ulParam1 作業パッケージ ID が入ります。

ulParam2 作業パッケージ・インスタンスが入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_INDEX_CLASS
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

準備: 作業パッケージを特定のライブラリー項目と関連づけるために変数を指定することができます。*pVariableList* が指定されていない場合には、呼び出し側のアプリケーションは、処理されるオブジェクトに作業パッケージ ID を関連づけておかなくてはなりません。pVariableList が指定されている場合には、作業パッケージ ID が API で参照されるたびに、ワーク・マネージメント・インターフェースは常に、データをアプリケーションに戻します。たとえば、呼び出し側のアプリケーションがワーク・バスケットから次の作業パッケージを取得すると、項目 ID も戻されることになります。

結果: 新しい作業パッケージが作成されます。

後続作業: 作業パッケージをワーク・バスケットに経路指定するには、**SimWmRouteWorkPackage** を呼び出す必要があります。

関連関数

- **SimWmRouteWorkPackage**

SimWmEndCollectionPoint (コレクション・ポイントからの作業パッケージの強制移動)

形式

SimWmEndCollectionPoint(*hSession, ulWorkPackageID, ulInstanceID, pAsyncCtl, pRC* **)**

目的

SimWmEndCollectionPoint 関数は、コレクション・ポイントにある作業パッケージを強制的に移動させるために使用します。

パラメーター

hSession HSESSION — 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

ulWorkPackageID

ULONG — 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG — 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- OIM_ITEM_NOT_SUSPENDED
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC
- SIM_RC_PRIVILEGE_ERROR

SimWmEndProcess (処理中の作業パッケージの終了)

形式

SimWmEndProcess(*hSession, ulWorkPackageID, ulInstanceID, pAsyncCtl, pRC* **)**

目的

SimWmEndProcess 関数は、アクティブ作業パッケージを強制終了します。この関数は、ワーク・バスケットから作業パッケージを除去します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

ulWorkPackageID

ULONG - 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG - 入力

インスタンスが 1 つしか存在しない場合には、このパラメーターは無視されます。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果:

- 作業パッケージのインスタンス・フィールドの値がゼロの場合は、プロセスが終了中であると見なされます。ゼロ以外の値の場合は、経路が終了します。プロセスの途中で作業パッケージを終了すると、その作業パッケージのすべてのインスタンスが終了します。
- 特別な経路上の作業パッケージを終了するには、その作業パッケージの ID のみを指定します。

関連関数

- **SimWmCreateWorkPackage**
- **SimWmGetWorkPackage**

SimWmGetActionListInfo (アクション・リスト情報の取得)

形式

SimWmGetActionListInfo(*hSession, pszActionListName, pAsyncCtl, pRC* **)**

目的

SimWmGetActionListInfo 関数は、アクション・リストに関連付けられている詳細情報を取得するために使用します。

hSession HSESSION — 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

pszActionListName

PSZ — 入力

アクション・リストの名前へのポインター。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。

ulParam1 アクション・リスト情報を提供する WMACTIONLISTINFOSTRUCT

データ構造へのポインターを含みます。詳細については、 182 ページの『WMACTIONLISTINFOSTRUCT (アクション・リストのデータ構造)』を参照してください。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC

使用の手引き

後続作業: アプリケーションで WMACTIONLISTINFOSTRUCT データが不要になった場合は、**SimLibFree** 関数を使用して、その構造を含むバッファーを解放してください。

SimWmGetProcessInfo (プロセスに関する情報の取得)

形式

SimWmGetProcessInfo(*hSession, pszProcessID, fGetProcessInfo, pAsyncCtl, pRC* **)**

目的

SimWmGetProcessInfo 関数を使用して、システム内で定義されている特定のクラスに関する詳細情報を戻します。この関数は、特定のプロセスに関連付けられているワーク・バスケットとコレクション・ポイントの一方または両方を戻します。

パラメーター

hSession HFSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

pszProcessID PSZ - 入力

プロセス ID へのポインター。

fGetProcessInfo BITS - 入力

指定したプロセスに関して、どのような情報を戻すかを決定するフラグ・ビット。ビット単位の包含 OR 演算子 (|) を使用して、これらの値を組み合わせることができます。

SIMWM_PROCESS_WORKBASKETS
指定したプロセスに関連するすべてのワーク・バスケットの情報が戻ります。

SIMWM_PROCESS_COLLECTION_POINTS
指定したプロセスに関連するすべてのコレクション・ポイントの情報が戻ります。

SIMWM_PROCESS_ALL_LOCATIONS
指定したプロセスに関連するワーク・バスケットとコレクション・ポイントの情報が戻ります。

SIMWM_PROCESS_COUNT
指定したプロセスに関連するワーク・バスケットとコレクション・ポイントの数が戻ります。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。

ulParam1 WMPROCESSINFOSTUCT データ構造によってプロセス定義情報が提供されるバッファーへのポインターが入ります。このデータ構造の詳細については、184 ページの『WMPROCESSINFOSTRUCT (プロセス情報データ構造)』を参照してください。

ulParam2 ロケーション数が入ります。この値は、*fGetProcessInfo* の設定値に応じて変わります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_ERROR_READING_FROM_FILE
- SIM_RC_FILE_NOT_FOUND
- SIM_RC_INVALID_GETPROCESSOPTIONS
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_ITEM_NOT_FOUND
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

後続作業: アプリケーションで処理情報が不要になった場合、**SimLibFree**(*hSession*, (PVOID)*ulParam1, pRC)* 関数を使用してバッファーを解放してください。

関連関数:

- **SimWmListProcesses**

SimWmGetWorkBasketInfo (ワーク・バスケット情報の取得)

形式

SimWmGetWorkBasketInfo(*hSession, pszWorkBasketID, pAsyncCtl, pRC* **)**

目的

SimWmGetWorkBasketInfo 関数を使用して、指定されたワーク・バスケットに関する情報を戻します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszWorkBasketID

PSZ - 入力

ワーク・バスケット ID へのポインター。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。

ulParam1 WORKBASKETINFOSTRUCT データ構造が指定のワーク・バスケットに関する詳しい情報を提供するバッファーへのポインターが入ります。このデータ構造の詳細については、190 ページの『WORKBASKETINFOSTRUCT (ワーク・バスケット情報データ構造)』を参照してください。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- OIM_INVALID_PITEMIDWB_PTR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION

- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_PRC
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

後続作業: アプリケーションで WORKBASKETINFOSTRUCT データが不要になった場合は、**SimLibFree** 関数を使用してバッファーを解放してください。

関連関数

- **SimWmListWorkBaskets**

SimWmGetWorkPackage (ワーク・バスケットからの次の作業パッケージの取得)

形式

SimWmGetWorkPackage(*hSession, pszWorkBasketID, ulWorkOrder, ulWorkPackageID, ulInstanceID, pAsyncCtl, pRC* **)**

目的

SimWmGetWorkPackage 関数は、ワーク・バスケットの中に現在ある作業パッケージを取得 (オープン) します。指定のワーク・バスケットでキューに入っている作業パッケージは、他のアプリケーションでは使用できません。この関数は、特定の作業パッケージ、または指定のワーク・バスケットで現在使用可能な作業パッケージで、作業順位が次の作業パッケージを取得することができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pszWorkBasketID

PSZ - 入力

ワーク・バスケット ID へのポインター。

ulWorkOrder ULONG - 入力

ワーク・バスケットからの項目の選択に使用される順序。有効な値は、次のとおりです。

NULL サーバーは、作業の処理順序を判別し、最初に使用可能な作業パッケージまたは要求された作業パッケージを戻します。

SIMWM_ORDER_FIFO
先入れ先出し (FIFO) 順序に基づいた選択を行い、最初に使用可能な作業パッケージを戻します。

SIMWM_ORDER_LIFO
先入れ先出し (LIFO) 順序に基づいた選択を行い、最初に使用可能な作業パッケージを戻します。

SIMWM_ORDER_PRIORITY
作業パッケージの優先順位に基づいた選択を行い、最初に使用可能な作業パッケージを戻します。

SIMWM_ORDER_SYSTEM_NEXT
サーバーは、作業の処理順序を判別し、次に使用可能な作業パッケージを戻します。

SIMWM_ORDER_FIFO_NEXT
先入れ先出し (FIFO) 順序に基づき、次に使用可能な作業パッケージの選択を行います。

SIMWM_ORDER_LIFO_NEXT
先入れ先出し (LIFO) 順序に基づき、次に使用可能な作業パッケージの選択を行います。

SIMWM_ORDER_PRIORITY_NEXT
作業パッケージの優先順位に基づき、次に使用可能な作業パッケージの選択を行います。

ulWorkPackageID
ULONG - 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。最初の作業パッケージを検索する場合は、ゼロを指定してください。作業パッケージ ID を指定した場合は、 *ulWorkOrder* に指定した値に応じて、その ID を持つ作業パッケージまたは次に使用可能な作業パッケージが検索されます。

ulInstanceID ULONG - 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、 171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはデータ域へのポインターが入ることを示します。

ulParam1 戻された項目および関連ワーク・マネージメント情報を提供する SNAPSHOTSTRUCT データ構造へのポインターが入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- OIM_INVALID_PITEMIDWB_PTR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_EMPTY_WORKBASKET
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果:

- 作業パッケージ ID が指定されない場合には、この関数は、ワーク・バスケット内の最初に使用できる作業パッケージを検索します。
- 作業パッケージ ID を指定して *ulWorkOrder* を NULL にした場合は、該当する作業パッケージが検索されます。
- 作業パッケージ ID を指定して *ulWorkOrder* に SIMWM_ORDER_SYSTEM_NEXT、SIMWM_ORDER_FIFO_NEXT、SIMWM_ORDER_LIFO_NEXT 、または SIMWM_ORDER_PRIORITY_NEXT を設定した場合は、該当する作業パッケージの次に使用可能な作業パッケージが検索されます。
- 指定された作業パッケージまたはワーク・バスケット内の次の作業パッケージが検索されると、その作業パッケージには他のユーザーがアクセスできなくなります。

後続作業:

- **SimWmReturnWorkPackage** 関数を呼び出して、作業パッケージをワーク・バスケットに戻します。これにより、他のユーザーがその作業パッケージを使用できるようになります。
- **SimWmRouteWorkPackage** 関数を呼び出して、作業パッケージを別のワーク・バスケットに経路指定します。これにより、宛先ワーク・バスケットで他のユーザーがその作業パッケージを使用できるようになります。
- アプリケーションで SNAPSHOTSTRUCT データが不要になった場合は、**SimLibFree** 関数を使用してバッファーを解放してください。

関連関数

- **SimWmReturnWorkPackage**
- **SimWmRouteWorkPackage**

SimWmGetWorkPackagePriority (作業パッケージの優先順位の取得)

形式

SimWmGetWorkPackagePriority(*hSession, ulWorkPackageID, ulInstanceID, pAsyncCtl, pRC* **)**

目的

SimWmGetWorkPackagePriority 関数を使用して、作業パッケージに割り当てられている優先順位を判別します。この優先順位は、そのワーク・バスケット内に入っている項目の処理順序を示しています。項目がロックされている場合でも、その項目の現行の優先順位を判別することができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

ulWorkPackageID

ULONG - 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG - 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。

ulParam1 作業パッケージがワーク・バスケットに入った日時を提供する TIMESTAMP バッファーへのポインターが入ります。

ulParam2 指定された作業パッケージの現行の優先順位が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK

- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC

使用の手引き

後続作業: アプリケーションで TIMESTAMP データが不要になった場合は、**SimLibFree** 関数を使用してバッファーを解放してください。

関連関数

- **SimWmGetWorkPackage**
- **SimWmSetWorkPackagePriority**
- **SimWmRouteWorkPackage**

SimWmListHistory (作業パッケージのヒストリーのリスト)

形式

SimWmListHistory(*hSession, ulWorkPackageID, ulInstanceID, fHistoryRequest, pAsyncCtl, pRC* **)**

目的

SimWmListHistory 関数は、指定した作業パッケージに関するアクティビティーのログを取得するために使用します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

ulWorkPackageID

ULONG - 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG - 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

fHistoryReques BITS - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。

ulParam1 特定の作業パッケージの変数 ID および変数値を含んでいる WMHISTLOGENTRYSTRUCT 構造の配列へのポインターを含みます。

ulParam2 ulParam1 が指し示す配列内の変数の数が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY

使用の手引き

結果: 関数が正常終了すると、作業パッケージに関連するすべてのヒストリー・イベントが戻ります。

後続作業: アプリケーションで、指定の作業パッケージに関するワーク・マネージメント・ヒストリー情報が不要になった場合は、SimLibFree(hSession, (PVOID)ulParam1, pRC) 関数を使用してバッファーを解放してください。

関連関数

- **SimLibLogon**

SimWmListProcesses (プロセスのリスト)

形式

SimWmListProcesses(*hSession, pAsyncCtl, pRC* **)**

目的

SimWmListProcesses 関数は、Content Manager for iSeries システム内のすべての既存プロセスのリストを取得するために使用します。

パラメーター

hSession HSESSION — 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。

ulParam1 ITEMNAMESTRUCT 配列へのポインターが入ります。

ulParam2 *ulParam1* が指す配列内の要素の数が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_ERROR_READING_FROM_FILE
- SIM_RC_FILE_NOT_FOUND
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

例外: この関数によって、システムに定義されたプロセスがすべて提供されます。**SimWmListProcesses** 関数が戻すプロセスのいずれか 1 つについての情報を取得するには、**SimWmGetProcessInfo** 関数を使用してください。

後続作業: アプリケーションで処理リストが不要になった場合、**SimLibFree**(*hSession*, (PVOID)*ulParam1, pRC)* 関数を使用してバッファーを解放してください。

SimWmListWorkBaskets (ワーク・バスケットのリスト)

形式

SimWmListWorkBaskets(*hSession, pAsyncCtl, pRC* **)**

目的

SimWmListWorkBaskets 関数を使用して、システムのすべてのワーク・バスケット定義のリストを取得します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。

ulParam1 ITEMNAMESTRUCT 配列へのポインターが入ります。

ulParam2 *ulParam1* が指す配列内の要素の数が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_OR_FOLDER_VALUE
- SIM_RC_INVALID_PRC
- SIM_RC_LIB_CLIENT_ERROR
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

例外: この関数は、ワーク・バスケットの定義に関する詳細情報は提供しません。この詳細情報を取得するには、**SimWmListWorkBaskets** が戻す ID の 1 つを含んだ **SimWmGetWorkBasketInfo** を使用してください。

後続作業: アプリケーションで ITEMNAMESTRUCT 配列が不要になった場合は、**SimLibFree** 関数を使用してバッファーを解放してください。

関連関数

- **SimWmGetWorkBasketInfo**

SimWmMatchEvent (作業パッケージに関するイベントの充足)

形式

SimWmMatchEvent(*hSession, ulActivate, pszProcessID, pszCollectionPointName, ulWorkPackageID, ulInstanceID, ulEventType, pszEventCriteria, pAsyncCtl, pRC* **)**

目的

SimWmMatchEvent 関数は、コレクション・ポイントにある作業パッケージに関するイベントを満たすために使用します。

パラメーター

hSession HSESSION — 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

ulActivate ULONG — 入力

コレクション・ポイントを起動可能の状態にするかどうかを示すインディケーター。有効な値は、次のとおりです。

SIMWM_ACTIVATE_COLLECTION_POINT
作業パッケージが現在コレクション・ポイントにない場合に、そのコレクション・ポイントを活動化します。

SIMWM_NO_ACTIVATE_COLLECTION_POINT
作業パッケージが現在コレクション・ポイントにない場合には、そのコレクション・ポイントを活動化しません。

pszProcessID PSZ - 入力

プロセス ID へのポインター。

pszCollectionPointName
PSZ - 入力

コレクション・ポイントの名前を指すポインター。

ulWorkPackageID
ULONG — 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG — 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

ulEventType ULONG — 入力

コレクション・ポイントで満たされなければならないイベントのタイプ。有効な値は、次のとおりです。

SIMWM_EVENT_INDEX_CLASS
イベントは、指定した索引クラスの項目の到着。

SIMWM_EVENT_TIME
イベントは、ある時間枠の満了。

SIMWM_EVENT_USERDEF_MIN - SIMWM_EVENT_USERDEF_MAX
イベントは、ユーザー定義のイベント。

pszEventCriteria
PSZ - 入力

突き合わせ基準を指すポインター。*ulEventType* に SIMWM_EVENT_INDEX_CLASS を指定した場合、突き合わせ基準はいずれかの索引クラス ID でなければならなりません。*ulEventType* に SIMWM_EVENT_TIME を指定した場合、このフィールドは無視され、サーバーの現在のシステム日付が突き合わせ基準として使用されます。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- OIM_INVALID_RELEASE_CRITERIA
- OIM_INVALID_WF_ITEM
- OIM_ITEM_NOT_IN_WORKFLOW
- OIM_ITEM_NOT_SUSPENDED
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USCLASSID_VALUE

使用の手引き

この関数は、コレクション・ポイントでイベントを満たすか、作業パッケージを活動化します。指定した作業パッケージに関するイベントが一致すると、その作業パッケージ・イベントが満たされます。イベントが一致せず、活動化フラグが SIMWM_ACTIVATE_COLLECTION_POINT に設定されている場合には、その作業パッケージがコレクション・ポイントで活動化されます。

コレクション・ポイントのイベント・リスト内の最後のイベントが満たされると、作業パッケージはコレクション・ポイントからリリースされ、そのコレクション・ポイント定義のイベント・リストに対して指定されている経路を開始するために送信されます。

イベント・タイプ SIMWM_EVENT_TIME を指定してこの関数を呼び出すと、すべてのコレクション・ポイントについて、日付満了基準が満たされているかどうかが検査されます。この関数は、保留作業項目リリース関数に相当します。

SimWmQueryVariables (特定の作業パッケージ変数の照会)

形式

SimWmQueryVariables(*hSession, ulWorkPackageID, ulInstanceID, pAsyncCtl, pRC* **)**

目的

SimWmQueryVariables 関数は、特定の作業パッケージに関連付けられているすべての変数と値を戻すために使用します。

パラメーター

hSession HSESSION — 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

ulWorkPackageID

ULONG — 入力

作業パッケージの ID。

ulInstanceID ULONG — 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。その他の場合、このフィールドには値 0 が入ります。

ulParam1 特定の作業パッケージの変数 ID および変数値を含んでいる WMVARSTRUCT 構造の配列へのポインター。

ulParam2 *ulParam1* が指し示す配列内の変数の数が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK

- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC

使用の手引き

後続作業: アプリケーションで作業パッケージ変数情報が不要になった場合、**SimLibFree**(*hSession*, (PVOID)*ulParam1, pRC)* 関数を使用してバッファーを解放してください。

SimWmQueryWorkPackage (作業パッケージの照会)

形式

SimWmQueryWorkPackage(*hSession, ulWorkPackageID, ulInstanceID, pAsyncCtl, pRC* **)**

目的

SimWmQueryWorkPackage 関数を使用して、作業パッケージの内容および属性を検索します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

ulWorkPackageID

ULONG - 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG - 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはデータ域へのポインターが入ることを示します。

ulParam1 戻された項目および関連ワークフロー情報を提供する SNAPSHOTSTRUCT データ構造へのポインターが入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_INDEX_CLASS
- SIM_RC_INVALID_PRC
- SIM_RC_LIB_CLIENT_ERROR
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

後続作業: アプリケーションで SNAPSHOTSTRUCT データが不要になった場合は、**SimLibFree** 関数を使用してバッファーを解放してください。

関連関数

- **SimWmRouteWorkPackage**

SimWmReturnWorkPackage (ワーク・バスケットへの作業パッケージの返却)

形式

SimWmReturnWorkPackage(*hSession, ulWorkPackageID, ulInstanceID, usWorkPriority, pAsyncCtl, pRC* **)**

目的

SimWmReturnWorkPackage 関数を使用して、ワーク・バスケット内で現在オープンされている作業パッケージ・インスタンスをそのワーク・バスケットに戻します。この関数は **SimWmGetWorkPackage** 関数とは逆の働きをします。この関数の使用後に、その作業パッケージ・インスタンスは再び使用可能になります。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

ulWorkPackageID

ULONG - 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG - 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

usWorkPriority USHORT - 入力

実行する作業の優先順位。作業パッケージがプロセスを移動するときに、この優先順位により処理順序が決定します。数値が大きいほど、優先順位が高くなります。現行の優先順位を維持するには、ゼロを使用してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果: ユーザーが作業を完了できず、あとで再開する必要がある場合に、アプリケーションはこの機能を使用することができます。**SimWmGetWorkPackage** は作業パッケージをオープンし、**SimWmReturnWorkPackage** 関数は作業パッケージをクローズして、再びワーク・バスケット内で使用できるようにします。

関連関数

- **SimWmGetWorkPackage**
- **SimWmRouteWorkPackage**

SimWmRouteWorkPackage (作業パッケージの経路指定)

形式

SimWmRouteWorkPackage(*hSession, pszWorkBasketID, ulWorkPackageID, ulInstanceID, usWorkPriority, fRoute, pszOverrideAction, pAsyncCtl, pRC* **)**

目的

SimWmRouteWorkPackage 関数を使用して、作業パッケージをワーク・バスケットに割り当てたり、あるワーク・バスケットから別のワーク・バスケットに割り当てなおしたり、また既に定義されているプロセス内の次のステップに移動して作業を続けることができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。 **SimLibLogon** 関数は、セッション情報を作成します。

pszWorkBasketID

PSZ - 入力

ワーク・バスケットの名前を指すポインター。このパラメーターが NULL で、かつ該当の作業パッケージがいずれかのプロセスで処理中であった場合、その作業パッケージはプロセスの次のステップに進んで引き続き処理されます。

ulWorkPackageID

ULONG - 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG - 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

usWorkPriority USHORT - 入力

実行する作業の優先順位。この優先順位により、ワーク・バスケットでの作業パッケージの処理順序が決定します。数値が大きいほど、優先順位が高くなります。デフォルトの優先順位を要求するには、ゼロの優先順位を使用してください。

fRoute BITS - 入力

作業パッケージの経路指定制御。有効な値は以下のとおりです。

SIMWM_IGNORE_OVERLOAD

このパラメーターを NULL にした場合は、ワーク・バスケットのオーバーロード制限値が検査されます。

pszOverrideAction

PSZ - 入力

作業パッケージが次のワーク・バスケットに経路指定されるときに必要となるアクション・リストの名前を指すポインター。このアクション・リストは、次のワーク・バスケットに関連付けられているデフォルトのアクション・リストを指定変更します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 常にゼロです。

ulParam1 作業パッケージ ID が入ります。

ulParam2 作業パッケージ・インスタンスが入ります。

ulRC 以下のいずれかの値を含みます。

- SIM_RC_OK
- OIM_INVALID_FOVERLOAD_VALUE
- OIM_WB_FULL
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

この関数を使用すると、あるプロセスに従って項目の処理を続行したり、項目をワーク・バスケットに割り当てたり、別のワーク・バスケットに再割り当てすることができます。SIMWM_IGNORE_OVERLOAD を設定せず、かつ *pszWorkBasketID* を NULL にした場合、該当の項目は、ワーク・バスケットのオーバーロード条件の有無にかかわらず、そのワーク・バスケットに追加されます。ただし、オーバーロード条件はアプリケーションに通知されます。この関数を **SimWmQueryWorkPackage** と組み合わせることにより、作業パッケージを経路指定する前に、その作業パッケージのロケーションを判別できます。

例外: 作業パッケージがコレクション・ポイントにある場合は、そのコレクション・ポイントに定義されているイベントが満たされるまで、作業パッケージの経路指定は実行できません。

関連関数

- **SimWmCreateWorkPackage**
- **SimWmQueryWorkPackage**

SimWmSetWorkPackagePriority (作業パッケージの優先順位の設定)

形式

SimWmSetWorkPackagePriority(*hSession, ulWorkPackageID, ulInstanceID, usPriority, pAsyncCtl, pRC* **)**

目的

SimWmSetWorkPackagePriority 関数を使用して、作業パッケージの優先順位を設定します。この優先順位により、ワーク・バスケット内の作業パッケージの作業順序を制御することができます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

ulWorkPackageID

ULONG - 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG - 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

usPriority USHORT - 入力

実行する作業の優先順位。この優先順位により、作業パッケージの処理順序が決定します。数値が大きいほど、優先順位が高くなります。デフォルトの優先順位を要求するには、ゼロの優先順位を使用してください。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

例外: 優先順位の値には、1 ～ 65,535 の数字を指定できます。ただし、 Content Manager for iSeries クライアント・アプリケーションでは、1 ～ 31,999 の値のみがサポートされます。

関連関数
- **SimWmGetWorkPackage**
- **SimWmGetWorkPackagePriority**
- **SimWmRouteWorkPackage**

SimWmSuspendWorkPackage (作業パッケージの中断)

形式

SimWmSuspendWorkPackage(*hSession, ulWorkPackageID, ulInstanceID, pSuspendCriteria, pAsyncCtl, pRC* **)**

目的

SimWmSuspendWorkPackage 関数は、現在ワーク・バスケットに入っている作業パッケージ・インスタンスを中断し、その作業パッケージを選択不能な状態で保持するために使用します。この状態は、中断基準が満たされるか、作業パッケージの明示的な再活動化が行われるまで保持されます。

パラメーター

hSession HSESSION — 入力

Content Manager for iSeries セッション情報へのハンドル。 **SimLibLogon** 関数がセッション情報を作成します。

ulWorkPackageID

ULONG — 入力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID ULONG — 入力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

pSuspendCriteria

PWMSUSPENDSTRUCT — 入力

作業パッケージの中断とリリースに関する基準が含まれた単一の WMSUSPENDSTRUCT 構造を指すポインター。

pAsyncCtl PASYNCCTLSTRUCT — 入力

サポートされていません。

pRC PRCSTRUCT — 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 以下のいずれかの値を含みます。

- SIM_RC_OK
- OIM_INVALID_READY_WB
- OIM_INVALID_RELEASE_CRITERIA
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

準備:

- 中断基準には、最高で 8 つの索引クラスを指定できます。
- フォルダーは、指定した索引クラスに属しているほかの項目の到着を保留することによって中断することも、所定の期間が満了するまで中断することもできます。
- 指定した索引クラス (複数も可) によって中断を行う場合は、中断期間を併せて指定する必要があります。
- リリース基準の索引クラスとして SIM_INDEX_ANY を指定した場合は、システムに定義されたいずれかの索引クラスに属している項目が到着するまでの間、該当項目が中断されます。

結果:

- リリース基準が満たされると、WMSUSPENDSTRUCT データ構造内の基準に関連付けられているワーク・バスケットに、先に中断されたいずれかの項目が割り当てられます。

例外:

- 中断する項目は、ワーク・バスケットに入っているものでなければなりません。
- 項目が特別なプロセスで処理される場合、SIMWM_NEXT は有効なワーク・バスケットとして認識されません。
- 項目の中断状態が変化しても、その項目のチェックアウトまたはアクセス状況は変化しません。ある項目をアプリケーションがチェックアウトして中断した場合、その項目を確実にチェックインする処理は、同一のアプリケーションが行う必要があります。その項目は、所定のリリース基準が満たされると活動化されますが、該当アプリケーションによってチェックインされない限り、チェックアウト状態に置かれたままとなります。
- SIM_INDEX_ANY を単一の索引クラスとして入力した場合は、他の索引クラスを中断基準に定義することはできません。
- 現在中断状態にある項目に対して **SimWmSuspendWorkPackage** を発行しても、その項目が再度中断されることはありません。この場合、新規の中断要求は無視されますが、アプリケーションには要求の正常終了コードが戻ります。

Sim400ConvertCodepage (コード・ページの変換)

形式

Sim400ConvertCodepage(*hSession, iConvertDirection, chInputBuffer, chOutputBuffer, ulInputSize, ulOutputSize, pAsyncCtl, pRC* **)**

目的

Sim400ConvertCodepage は、ワークステーションと iSeries 間でのコード・ページ変換を処理するために使用します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

iConvertDirection

INT - 入力

以下のいずれかの値を指定します。

SIM_400_CONVERT_TO400

SIM_400_CONVERT_FROM400

chInputBuffer CHAR - 入力

サーバーに送信するバッファー。

chOutputBuffer CHAR - 入力

戻されるデータ用のスペース。

ulInputSize ULONG - 入力

サーバーへの送信時のバッファーの長さ。最大サイズは、32,700 です。

ulOutputSize ULONG - 入力

戻されるデータ用のスペースのサイズ。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

usParam この関数は、このフィールドを使用しません。

ulParam1 出力バッファーの長さが入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_LIB_CLIENT_ERROR

関連関数

- **Sim400SendReceive**

Sim400SendReceive (AS/400 へのデータの送信)

形式

Sim400SendReceive(*hSession, chInputBuffer, chOutputBuffer, ulInputSize, ulOutputSize, pAsyncCtl, pRC* **)**

目的

Sim400SendReceive 関数は、最大 32,700 バイトのデータを iSeries に送信するために使用します。サーバーに送信されるデータは、ユーザーが作成したアプリケーションで処理して、その結果をワークステーションに戻すようにすることもできます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

chInputBuffer CHAR - 入力

サーバーに送信するバッファー。

chOutputBuffer CHAR - 入力

戻されるデータ用のスペース。

ulInputSize ULONG - 入力

サーバーへの送信時のバッファーの長さ。最大サイズは、32,700 です。

ulOutputSize ULONG - 入力

戻されるデータ用のスペースのサイズ。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

usParam この関数は、このフィールドを使用しません。

ulParam1 受信したバイト数が入ります。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR

例

QVI ライブラリーのソース・ファイル QLBLSRC に含まれているサンプル・プログラム QVIRCVSND を参照してください。このサンプル・プログラムは、**Sim400SendReceive** 関数によってデータを受信する COBOL プログラムと、この関数に対して戻されるデータの例を示します。

関連関数

- **Sim400ConvertCodepage**

lp2CloseTOC (目次のクローズ)

形式

lp2CloseTOC(*hSession, hTOC, pAsyncCtl, pRC* **)**

目的

lp2CloseTOC 関数を使用して、指定された目次をクローズしてから、目次ハンドルを解放します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hTOC HTOC - 入力

クローズしたい目次のハンドル。**SimLibGetTOC** 関数を使用して、このハンドルを取得します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam この関数は、このフィールドを使用しません。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_LIB_CLIENT_ERROR
- SIM_RC_OUT_OF_MEMORY

使用の手引き

結果:

- この関数を使用して目次をクローズしたあとは、その目次ハンドル (*hTOC*) を再度使用することはできません。
- **SimLibGetTOC** 関数を使用して、新しい目次ハンドルを取得してください。

関連関数

- **Ip2CloseToc**
- **Ip2GetTOCUpdates**
- **Ip2TOCCount**
- **Ip2TOCStatus**
- **SimLibGetItemAffiliatedTOC**
- **SimLibGetTOC**

lp2GetLibSessionInfo (ライブラリー・セッションの情報の取得)

形式

lp2GetLibSessionInfo(*hSession, pAsyncCtl, pRC***)**

目的

lp2GetLibSessionInfo 関数は、現行ライブラリー・セッションに関する情報を戻すために使用します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数がセッション情報を作成します。

pAsyncCtl PASYNCCTLSTRUT - 入力

サポートされていません。

pRC PRSTRUCT - 入力/出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。

ulParam1 LIBSESSIONINIFOSTRUCT データ構造を持つバッファーへのポインターを含みます。このデータ構造の詳細については、168 ページの『LIBSESSIONINFOSTRUCT (ライブラリー・セッション情報構造)』を参照してください。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_PRC

使用の手引き

後続作業: アプリケーションで LIBSESSIONINFOSTRUCT データが不要になった場合、**SimLibFree (***hSession*, (PVOID)*ulParam1, pRC***)** 関数を使用してバッファーを解放してください。

lp2GetTOCUpdates (目次の更新の取得)

形式

lp2GetTOCUpdates(*hSession, hTOC, usUpdate, pAsyncCtl, pRC* **)**

目的

lp2GetTOCUpdates 関数を使用して、前の **SimLibGetTOC** 関数から戻された目次を最新表示します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hTOC HTOC - 入力

最新表示したい目次のハンドル。**SimLibGetTOC** 関数を使用して、このハンドルを取得します。

usUpdate USHORT - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 目次内の項目の総数が入ります。

ulParam1 更新、削除、または追加された項目数を示す、TOCENTRYSTRUCT データ構造の配列を持つバッファーへのポインターが入ります。TOCENTRYSTRUCT データ構造の詳細については、178 ページの『TOCENTRYSTRUCT (目次項目データ構造)』を参照してください。

ulParam2 目次のハンドルが入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- OIM_INVALID_FUPDATE_VALUE
- OIM_INVALID_HTOC_VALUE
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR

- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_LIB_CLIENT_ERROR
- SIM_RC_OUT_OF_MEMORY

使用の手引き

後続作業: アプリケーションで目次が不要になった場合は、**lp2CloseTOC** 関数を使用して、目次をクローズし、目次ハンドルを解放してください。

関連関数

- **SimLibGetTOC**
- **lp2CloseTOC**
- **lp2TOCStatus**
- **lp2GetTOCUpdates**

lp2ListAttrs (ユーザー定義属性のリスト)

形式

lp2ListAttrs(*hSession, pAsyncCtl, pRC* **)**

目的

lp2ListAttrs 関数を使用して、システムの属性のリストを取得します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入り、*ulParam1* にはポインターが入ることを示します。

ulParam1 *ulParam2* フィールドに 0 より大きな値が入る場合、このフィールドには NAMESTRUCT 配列を持つバッファーへのポインターが入ります。この配列内の各要素は、特定の属性名に関連づけられてい

る索引属性 ID を提供します。このデータ構造の詳細については、168 ページの『NAMESTRUCT (名前データ構造)』を参照してください。

ulParam2 *ulParam1* が指す配列内の要素の数が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_LIB_CLIENT_ERROR
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果:

- **SimLibGetAttrInfo** 関数を使用して、特定の索引属性に関する追加情報を取得します。
- 負の ID または 32767 より大きい ID を持つ属性は、システム属性です。このシステム属性を変更することはできません。
- どの索引クラスにも定義されていない属性については、 **DlpListAttrs** を使用しても情報は戻りません。

後続作業: このアプリケーションで索引属性 ID の配列が不要になった場合は、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC* **)** 関数を使用して、バッファーを解放してください。

関連関数

- **SimLibGetAttrInfo**

DlpListContentClasses (コンテンツ・クラスのリスト)

形式

DlpListContentClasses(*hSession, usContentClassType, pAsyncCtl, pRC* **)**

目的

DlpListContentClasses 関数を使用して、ライブラリー・サーバー・データベースに入っているコンテンツ・クラス・レコードを表示します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

usContentClassType

USHORT - 入力

リストするコンテンツ・クラスのタイプ。有効な値は、次のとおりです。

OIM_SA_ALL_CC
IBM 定義のコンテンツ・クラスとユーザー定義のコンテンツ・クラスの両方をリストします。

OIM_SA_IBM_CC
IBM 定義のコンテンツ・クラスのみをリストします。

OIM_SA_USR_CC
ユーザー定義コンテンツ・クラスのみをリストします。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常に完了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。この値は、*ulParam1* にポインターが入っていることを示しています。指定したコンテンツ・クラス・タイプのレコードがない場合は、このフィールドには値 0 が入ります。

ulParam1 コンテンツ・クラスのリストが入っている CONTENTCLASSINFO データ構造の配列を指すポインターが入ります。このデータ構造の詳細については、161 ページの『CONTENTCLASSINFO (コンテンツ・クラス情報構造)』を参照してください。指定したコンテンツ・クラス・タイプのレコードがない場合は、このフィールドには値 NULL が入ります。

ulParam2 ライブラリー・サーバー・データベース内にあるコンテンツ・クラスの数が入ります。*ulRC* にエラー・コードが入ると、*ulParam2* には値 NULL が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_CC_TYPE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR
- SIM_RC_QUERY_FAILED

使用の手引き

後続作業: コンテンツ・クラス情報の処理が終了した時点で、**SimLibFree(** *hSession,* (PVOID)*ulParam1, pRC***)** 関数を使用して割り振り済みストレージを解放してください。

lp2ListServers (アクセス可能なサーバーのリスト)

形式

lp2ListServers(*pServrInfo, ulServrInfoSize, fSrchfilter, pRC* **)**

目的

lp2ListServers 関数を使用して、システムでアクセス可能なすべてのサーバーに関する情報を検索します。この関数を使用すると、ログオン対話の一部として表示する適格なライブラリーを判別することができます。

パラメーター

pServrInfo PSERVERINFOSTRUCT - 入力/出力

サーバー名とタイプの配列を含むバッファーへのポインター。呼び出し側のアプリケーションが、この構造にメモリーを割り振ります。

ulServrInfoSize ULONG - 入力

SERVERINFOSTRUCT 配列に割り振られるバッファーのサイズ (バイト)。

fSrchfilter ULONG - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 1 が入ります。

ulParam1 *usParam* に 0 より大きな値が入る場合、このフィールドには SERVERINFOSTRUCT データ構造の配列へのポインターが入ります。「使用の手引き」では、 *ulServrInfoSize* パラメーターの値が *ulParam1* の内容に与える影響について説明しています。SERVERINFOSTRUCT データ構造の詳細については、173 ページの『SERVERINFOSTRUCT (サーバー情報構造)』を参照してください。

ulParam2 この呼び出しで戻されるサーバーの数が入ります。必ずしもシステムのサーバーの数ではありません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- OIM_INVALID_PSERVERINFO_PTR
- OIM_RC_INPUTBUF_TOO_SMALL
- OIM_RC_ISO_CONNECT_FAILED
- OIM_RC_ISO_LISTSVR_FAILED

使用の手引き

例外:

- 使用しているアプリケーションは、すべてのサーバーに接続することができますが、すべてにログオンできるとは限りません。サーバー上のデータベースにアクセスするには、有効なユーザー ID とパスワードが必要です。
- *ulServrInfoSize* の入力値が小さすぎてデータを受け取ることができない場合には、エラー・コード OIM_RC_INPUTBUF_TOO_SMALL が戻され、 RCSTRUCT データ構造の *ulParam2* フィールドには、検出されたサーバーの数が入ります。

関連関数

なし

Ip2QueryClassPriv (索引クラスまたはビューの特権ストリングの照会)

形式

Ip2QueryClassPriv(*hSession, usClassType, usID, pAsyncCtl, pRC* **)**

目的

Ip2QueryClassPriv 関数を使用して、ユーザーが指定した索引クラスに関する評価された特権ストリングを戻します。評価された特権ストリングとは、システム内の情報へのアクセス権を示します。アクセス権を判別するには、その特権ストリングと **Ip2QueryPrivBuffer** 関数を併用する必要があります。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。 **SimLibLogon** 関数は、セッション情報を作成します。

usClassType USHORT - 入力

サポートされていません。

usID USHORT - 入力

索引クラスの ID。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常に完了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam このパラメーターには、*ulParam1* にポインターが入っていることを示す値 1 が入っています。

ulParam1 PSZ ポインターが入っています。このポインターは、評価された特権ストリングがデータ構造に含まれる CHAR *szPrivilege*[401] バッファーの位置を指します。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_CLASS_TYPE
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_INVALID_USCLASSID_VALUE
- SIM_RC_LIB_CLIENT_ERROR
- SIM_RC_OUT_OF_MEMORY

使用の手引き

結果:

- 特権ストリングは、ログオンによって *hSession* を取得したユーザーに関するクラスに対して評価されます。評価された特権ストリングは、指定された索引クラスに関するそのユーザーの特権を指定します。この特権は、アクセス制御アルゴリズムによって計算されたものです。

後続作業: *ulParam1* が指すデータ構造がアプリケーションで不要になった場合は、 **SimLibFree(** *hSession,* (PVOID) *ulParam1, pRC***)** 関数を使用して、データ構造を解放してください。

Ip2QueryPrivBuffer (特権バッファーの照会)

形式

Ip2QueryPrivBuffer(*pszPrivilege, ulAuthority, pRC* **)**

目的

Ip2QueryPrivBuffer 関数を使用して、指定された特権バッファーに特定の権限が付与されているかどうかを判別します。

パラメーター

pszPrivilege PSZ - 入力

ユーザーに設定されている現行の特権。

ulAuthority ULONG - 入力

検索のための一般的な特権。有効な値は、次のとおりです。

OIM_ACL
アクセス・リストを作成、更新、および削除する権限を決定します。

OIM_ADD_ITEMS_TO_WB
ワーク・バスケットに項目を追加する権限を決定します。

OIM_ADD_ITEMS_TO_WF
ワークフローに項目を追加する権限を決定します。

OIM_ADD_NEW_BASE_PART
新しい文書を追加する権限を決定します。

OIM_ADD_NOTE_TO_NOTELOG
注ログに注オブジェクトを追加する権限を決定します。

OIM_ATTRS
属性を作成、更新、および削除する権限を決定します。

OIM_CC
コンテンツ・クラスを作成、更新、リスト、および削除する権限を決定します。

OIM_CHANGE_INDEX_CLASS
任意の項目の索引クラスを変更する権限を決定します。

OIM_CHANGE_ITEMS_TO_WB
ワーク・バスケット内の項目の優先順位を変更する権限を決定します。

OIM_CHANGE_ITEMS_TO_WF
現行のワークフローから新しいワークフローに項目を変更する権限を決定します。

OIM_CHECK_IN_OUT_ITEMS
フォルダーまたは文書をチェックインおよびチェックアウトする権限を決定します。

OIM_CLASS
索引クラスで索引を追加、削除して、その DLL を照会する権限を決定します。

OIM_CREATE_ITEMS
フォルダーまたは文書を作成する権限を決定します。

OIM_DB_UTILITY
UTILITY がデータベースをアクセスすることを認める権限を決定します。

OIM_DELETE_BASE_PART
文書を削除する権限を決定します。

OIM_DELETE_ITEMS
フォルダーまたは文書を削除する権限を決定します。

OIM_EXPORT
オブジェクトを含むメールをエクスポートおよび送信する権限を決定します。

OIM_FAXIN
ファクシミリを受信する権限を決定します。

OIM_FAXOUT
ファクシミリを送信する権限を決定します。

OIM_FAXSERVER
ファックス・サーバーでファクシミリの送受信を行う権限を決定します。

OIM_FILEROOM
アプリケーションで定義されているファイル・ルームにアクセスする権限を決定します。

OIM_IMPORT
メールをインポートおよび受け取る権限を決定します。

OIM_LBOS_BACKUP
LAN ベースのオブジェクト・サーバーのバックアップをとる権限を決定します。

OIM_LIB_SERV_BACKUP
ライブラリー・サーバーのバックアップをとる権限を決定します。

OIM_LIB_SERV_CONFIG
ライブラリー・サーバー構成を制御する権限を決定します。

OIM_LICENSE
データベース内のライセンス情報を更新する権限を決定します。

OIM_LINK_ITEMS
項目とフォルダーの間にリンクを追加する権限を決定します。

OIM_OCR
光学式文字認識装置を使用する権限を決定します。

OIM_PRINT
印刷する権限を決定します。

OIM_PRIV_SET
特権セットを作成、更新、および削除する権限を決定します。

OIM_READ_BASE_PART
文書部分を読み取る権限を決定します。

OIM_READ_HISTORY
ヒストリー・イベントを読み取る権限を決定します。

OIM_READ_NOTELOG
注ログを読み取る権限を決定します。

OIM_READ_TOC
フォルダー目次を読み取る権限を決定します。

OIM_READ_WORKBASKET
ワーク・バスケット情報を入手する権限を決定します。

OIM_REMOVE_ITEMS_TO_WB
ワーク・バスケットから項目を除去する権限を決定します。

OIM_REMOVE_ITEMS_TO_WF
ワークフローから項目を除去する権限を決定します。

OIM_REMOVE_LINKS
項目とフォルダーとのリンクを削除する権限を決定します。

OIM_SA_NLS
データベースでサポートされている言語を更新する権限を決定します。

OIM_SA_OBJSERV
データベース内のオブジェクト・サーバー情報を更新する権限を決定します。

OIM_SA_USER
ユーザーの一般的なログオン特権を決定します。

OIM_SA_WORKBASKET
ワーク・バスケットを作成、更新、削除する権限を決定します。

OIM_SA_WORKFLOW
ワークフローを作成、更新、削除する権限を決定します。

OIM_SCAN
イメージをスキャンする権限を決定します。

OIM_SEARCH_INDEX_INFO
ユーザーが、すべての索引クラスと各索引クラスの全項目に定義した属性を読み取る権限を決定します。

OIM_SERVER
他のクライアントの代理として操作する権限を決定します。

OIM_SMS
LAN ベースのオブジェクト・サーバーのシステム管理ストレージの管理を行う権限を決定します。

OIM_SNAPSHOT_ALL
各項目に関して **SimLibGetItemSnapshot** または **SimLibGetTOCData** 関数を使用する権限を決定します。

OIM_SUPER_ADMIN
アクセス・リストをバイパスする権限を決定します。

OIM_SUSP_AND_ACTIVATE_ITEMS
フォルダーまたは文書を中断および活動化する権限を決定します。

OIM_UPDATE_AVT_INFO
ユーザーがすべての索引クラスと各索引クラスの全項目に定義した属性を更新する権限を決定します。

OIM_UPDATE_BASE_PART
文書を更新する権限を決定します。

OIM_UPDATE_NOTELOG
注ログの中の注を更新または削除する権限を決定します。

OIM_USER_GROUPS
ユーザー・グループを作成、更新、および削除する権限を決定します。

OIM_USER_ID
ユーザー ID を作成、更新、および削除する権限を決定します。

OIM_VIEW
ビューを作成、更新、および削除する権限を決定します。

OIM_WORKFLOW_CONTINUE
項目をプロセスの後続ステップに進めて処理させる権限を判別します。

OIM_WORKFLOW_FORCE_CONTINUE
未処理の中断イベントを持つ項目を、プロセスの後続ステップで強制的に処理させる権限を判別します。

OIM_WORKFLOW_SEARCH
項目のプロセスを検索する権限を決定します。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常に完了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 指定した権限が、*pszPrivilege* で表される特権セットに入っていれば、1 が入ります。それ以外の場合、このフィールドの値は 0 になります。

ulParam1 この関数は、このフィールドを使用しません。

ulParam2 この関数は、このフィールドを使用しません。

ulRC 次の戻りコードのいずれかが入ります。
- SIM_RC_OK
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- OIM_INVALID_PSZPRIVLEGE_STRING
- SIM_INVALID_ULAUTHORITY

Ip2TOCCount (目次の項目のカウント)

形式

Ip2TOCCount(*hSession, pitemidItem, usItemType, usWipFilter, usSuspendFilter, usNbrOfClasses, pusClassIdList, pAsyncCtl, pRC* **)**

目的

Ip2TOCCount 関数を使用して、指定されたフィルター基準を満たすフォルダーまたはワーク・バスケット内の項目のカウントを取得します。この関数は、**SimLibGetTOC** に類似していますが、目次ではなく、項目のカウントのみを戻すという部分が異なります。このカウントには、権限に関係なく、すべての項目が組み込まれます。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

pitemidItem PITEMID - 入力

フォルダーまたはワーク・バスケットの項目 ID へのポインター。

usItemType USHORT - 入力

カウントする項目のタイプ。有効な値は次のとおりです。

SIM_DOCUMENT
文書をカウントします。

SIM_FOLDER
フォルダーをカウントします。

SIM_ALL
すべてのタイプの項目をカウントします。

usWipFilter USHORT - 入力

サポートされていません。

usSuspendFilter USHORT - 入力

サポートされていません。

usNbrOfClasses USHORT - 入力

pusClassIdList パラメーターの値として指定したリスト内の索引クラス ID の数。*usNbrOfClasses* パラメーターに 0 を指定すると、カウントする項目の選択基準がクラスではないことを示します。

pusClassIdList PUSHORT - 入力

カウントする項目を指示する索引クラス ID のリストのポインター。このパラメーターに NULL を指定することができるのは、*usNbrOfClasses* パラメーターにも 0 を指定した場合です。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。

ulParam1 目次の項目のカウントが入ります。フィルター基準を満たす項目がなければ、このフィールドの値は 0 になります。

ulParam2 値 0 が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC
- SIM_RC_LIB_CLIENT_ERROR
- SIM_RC_OUT_OF_MEMORY
- SIM_RC_PRIVILEGE_ERROR

使用の手引き

結果: 項目がフォルダーまたはワーク・バスケットでなければ、この関数は SIM_RC_INVALID_ITEM_TYPE を戻します。

関連関数

- **Ip2GetTOCUpdates**
- **SimLibGetTOC**

lp2TOCStatus (目次の状況の取得)

形式

lp2TOCStatus(*hSession, hTOC, usCheck, pAsyncCtl, pRC* **)**

目的

lp2TOCStatus 関数を使用して、目次が変更されたか否かを示す値を戻します。

パラメーター

hSession HSESSION - 入力

Content Manager for iSeries セッション情報へのハンドル。**SimLibLogon** 関数は、セッション情報を作成します。

hTOC HTOC - 入力

状況を調べたい目次を指すハンドル。**SimLibGetTOC** 関数がこのハンドルを戻します。

usCheck USHORT - 入力

サポートされていません。

pAsyncCtl PASYNCCTLSTRUCT - 入力

サポートされていません。

pRC PRCSTRUCT - 入出力

戻りデータ構造を指すポインター。RCSTRUCT 構造の詳細については、171 ページの『RCSTRUCT (戻りコード情報構造)』を参照してください。

戻り値

この関数が正常終了すると、RCSTRUCT データ構造内の以下のフィールドに値が戻されます。

usParam 値 0 が入ります。

ulParam1 目次が変更された場合、このフィールドの値は TRUE になります。変更がなければ、このフィールドの値は FALSE になります。

ulParam2 値 0 が入ります。

ulRC 次の戻りコードのいずれかが入ります。

- SIM_RC_OK
- OIM_EMPTY_WORKBASKET
- OIM_INVALID_HTOC_VALUE
- SIM_RC_COMMUNICATIONS_ERROR
- SIM_RC_COMPLETION_ERROR
- SIM_RC_INVALID_HSESSION
- SIM_RC_INVALID_ITEM_ID
- SIM_RC_INVALID_POINTER
- SIM_RC_INVALID_PRC

- SIM_RC_LIB_CLIENT_ERROR
- SIM_RC_OUT_OF_MEMORY

使用の手引き

例外: この関数は、目次に変更があったかどうかを知らせますが、更新は戻しません。この関数の使用後に、別の関数を使用してアプリケーションに変更を反映させることができます。この関数に要する時間が **SimLibGetTOC** 関数または **SimLibGetTOCUpdates** 関数に要する時間とほぼ同じであるため、代わりにこれらの関数を使用してください (使用可能な場合)。

- **Ip2GetTOCUpdates** 関数を使用して目次を最新表示します。
- **Ip2CloseTOC** 関数を使用してオープンされている目次をクローズし、**SimLibGetTOC** 関数でデータベース内の値を反映させるために目次を最新表示します。

関連関数

- **Ip2CloseTOC**
- **Ip2GetTOCUpdates**
- **SimLibGetTOC**

第 4 章 共通データ構造

この章では、Content Manager for iSeries で使用される共通データ構造およびデータベース・テーブルについて記述する詳細な参照情報について説明します。このデータ構造は、アルファベット順にリストされ、Content Manager for iSeries コードでは常に大文字で表記されます。それぞれのデータ構造ごとに以下の項目について説明します。

- 目的
- 有効なフィールド
- 有効なフィールド値
- 使用の指針

データ構造

AFFTOCENTRYSTRUCT (関連目次項目構造)

このデータ構造は、どのオブジェクトが項目に関連しているかを示す情報を提供します。内容は以下のとおりです。

```
typedef struct _AFFTOCENTRYSTRUCT

{
ULONG                   ulStruct;
ANNOTATIONSTRUCT        AnnotationData;
ULONG                   ulObjType;
OBJ                     Obj;
ULONG                   ulObjConCls;
ULONG                   ulObjLength;
LONG                    lObjSeqAfter;
ULONG                   ulObjFlags;
TIMESTAMP               tsCreate;
TIMESTAMP               tsChanged;

} AFFTOCENTRYSHOTSTRUCT, *PAFFTOCENTRYSTRUCT;
```

フィールド

ulStruct — ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

AnnotationData — ANNOTATIONSTRUCT — 出力

注釈オブジェクトに関連した情報。詳細については、ANNOTATIONSTRUCT (注釈情報構造)を参照してください。

ulObjType — ULONG — 出力

オブジェクトのタイプ。有効な値は、次のとおりです。

SIM_ANNOTATION
その項目が、フォルダーまたは文書に関連した注釈であることを示します。

SIM_BASE
そのオブジェクトが、混合オブジェクト文書コンテンツ・アーキテクチャー (MO:DCA) またはタグ・イメージ・ファイル・フォーマット (TIFF) ファイルのような基本オブジェクトであり、フォルダーや文書に関連づけられた注釈、注、またはイベントでないことを示します。

SIM_NOTE
その項目が、フォルダーまたは文書に関連した注であることを示します。

Obj

OBJ — 出力

オブジェクトを識別するオブジェクト処理データ構造。詳細については、HOBJ (記憶オブジェクトの照会のためのハンドル) を参照してください。

ulObjConCls

ULONG — 出力

照会するオブジェクトのオブジェクト・コンテンツ・クラス。値 SIM_CC_UNKNOWN は、未定義のコンテンツ・クラスを表します。

ulObjLength

ULONG — 出力

オブジェクトの長さをバイト数で表します。

lObjSeqAfter

LONG — 出力

項目内にある他のオブジェクトに対する、オブジェクトの順位。

制約事項: これは、 **SimLibCreateObject** 関数のサポートされない *lSeqAfterPart* パラメーターの値です。

ulObjFlags

ULONG — 出力

サポートされていません。

tsCreate

TIMESTAMP — 出力

項目またはオブジェクトが作成された日時。

tsChanged

TIMESTAMP — 出力

項目またはオブジェクトが変更された日時。

ANNOTATIONSTRUCT (注釈情報構造)

このデータ構造は、オブジェクトに関連づけられている注釈についての情報を提供します。内容は以下のとおりです。

```
typedef struct _ANNOTATIONSTRUCT

{
ULONG       ulStruct;
ULONG       ulPart;
ULONG       ulPageNumber;
USHORT      usX;
USHORT      usY;
USHORT      usT;
USHORT      usAnnotUnused;

} ANNOTATIONSTRUCT, *PANNOTATIONSTRUCT;
```

フィールド

ulStruct — ULONG — 入力/出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

ulPart — ULONG — 入力/出力

オブジェクトの部分番号。正の値だけが有効です。

ulPageNumber — ULONG — 入力/出力

注釈オブジェクトが参照するページ番号。

usX — USHORT — 入力/出力

ulPageNumber フィールドの値が参照するページ上の注釈オブジェクトが使う X 座標。

usY — USHORT — 入力/出力

ulPageNumber フィールドの値が参照するページ上の注釈オブジェクトが使う Y 座標。

usT — USHORT — 入力/出力

サポートされていません。

usAnnotUnused — USHORT — 入力/出力

予約フィールド。

ATTRINFOSTRUCT (属性情報構造)

この構造は、ユーザー定義属性を作成、修正、およびリストするために必要なデータを提供します。内容は以下のとおりです。

```
typedef struct _ATTRINFOSTRUCT

{
ULONG       ulStruct;
```

```
BOOL        fUseBidirectional;
BOOL        fSymmetricSwapping;
BOOL        fShaping;
LONG        lMin;
LONG        lMax;
BITS        fTypeFlags;
USHORT      usAttrType;
USHORT      usHorizontalOrientation;
USHORT      usVerticalOrientation;
USHORT      usMode;
USHORT      usNumericSelectionDefault;
CHAR        szAttributeName;
CHAR        achLanguageCode;

} ATTRINFOSTRUCT, *PATTRINFOSTRUCT;
```

フィールド

ulStruct
ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

fUseBidirectional
BOOL — 出力

常に FALSE に設定されます。

fSymmetricSwapping
BOOL — 入力

常に FALSE に設定されます。

fShaping
BOOL — 入力

常に FALSE に設定されます。

lMin
LONG — 入力

lMin の意味は、次に示すように、*usAttrType* パラメーターの値によって異なります。

- *usAttrType* に SIM_ATTR_FSTRING が入っている場合には、このフィールドはストリングの最小長であり、これには値 0 またはそれより大きい値が入っている必要があります。
- データを 2 バイトの文字ストリング (DBCS) にすることができる場合には、 2 バイト文字と 1 バイト文字の混在する状況でシフトイン (SI) 文字とシフトアウト (SO) 文字が使用される場合を考慮して、スペースを用意しておく必要があります。
- *usAttrType* に SIM_ATTR_LONG が入っている場合は、これは最小許容値です。

lMax
LONG — 出力

lMax の意味は、次に示すように、*usAttrType* パラメーターの値によって異なります。

- *usAttrType* に SIM_ATTR_FSTRING が入っている場合には、このフィールドはストリングの最大長であり、これには 0 より大きくて *lMin* より大きい値が入っている必要があります。
- *usAttrType* に SIM_ATTR_LONG が入っている場合は、これは最大許容値です。

fTypeFlags — BITS — 出力

サポートされていません。

usAttrType — USHORT — 出力

Content Manager for iSeries では、これは常に SIM_ATTR_VSTRING に設定されます。

usHorizontalOrientation — USHORT — 出力

サポートされていません。

usVerticalOrientation — USHORT — 出力

サポートされていません。

usMode — USHORT — 出力

サポートされていません。

usNumericSelectionDefault — USHORT — 出力

サポートされていません。

szAttributeName — CHAR[*SIM_ATTR_NAME_LENGTH*+1] — 入力/出力

アプリケーション定義の属性名が入っている NULL 文字で終了する文字ストリング。

achLanguageCode — CHAR[*SIM_LANGUAGE_CODE_LENGTH*+1] — 出力

この属性名に対応する 3 文字からなる各国語コード。言語コードの値に関しては、「*IBM National Language Design Guide: National Language Support Reference Manual Volume 2*」に説明があります。

ATTRLISTSTRUCT (属性リスト・データ構造)

このデータ構造は、ある項目に関連づける、システム定義またはユーザー定義の、単一の属性値を定義します。このデータ構造は、項目の作成時にも使用されます。内容は以下のとおりです。

```
typedef struct _ATTRLISTSTRUCT

{
ULONG       ulStruct;
PSZ         pszAttributeValue;
BITS        fAttrFlags;
USHORT      usAttrId;
USHORT      usAttrType;
```

} ATTRLISTSTRUCT, *PATTRLISTSTRUCT;

フィールド

ulStruct

ULONG — 入力/出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

pszAttributeValue

PSZ — 入力/出力

属性の値が入った NULL 文字で終了する文字ストリングへのポインター。

fAttrFlags

BITS — 出力

属性の特性を表すフラグ。これらのフラグは、属性値が、読み取り、書き込み、またはその両方を行うためにアクセス可能であるかどうかを示したり、索引クラスで必要であるかどうかを示します。有効な値が後ろにきます。これらの値を組み合わせるために、ビット単位の包含 OR 演算子 (|) を使用することができます。

SIM_ATTR_READABLE
この索引クラスに関して属性が読み取り用にアクセス可能であることを示します。

SIM_ATTR_READWRITE
この索引クラスに関して属性が読み取りおよび書き込み用にアクセス可能であることを示します。

SIM_ATTR_WRITEABLE
この索引クラスに関して属性が書き込み用にアクセス可能であることを示します。

SIM_ATTR_ALLOW_NULL
この索引クラスに属性値が必要でないことを示します。

usAttrId

USHORT — 入力/出力

属性の固有 ID。Content Manager for iSeries システム定義属性については、このリストのあとに続く表の説明を参照してください。

usAttrType

USHORT — 入力/出力

Content Manager for iSeries では、これは常に SIM_ATTR_VSTRING に設定されます。

Content Manager for iSeries は、157 ページの表 2 に示されるシステム定義属性をサポートしています。

表 2. システム定義属性値の記述

属性の名前	説明	割り当て方法
OIM_ID_ITEM_CREATE_TIMESTAMP	項目が作成されたときのタイム・スタンプ。	システムによって自動的に割り当てられ維持されます。
OIM_ID_ITEM_NAME	項目名。	項目の作成時にユーザーが割り当て、読み取りアクセスと書き込みアクセスをするために項目をオープンしたときに更新できます。
OIM_ID_SYS_MOD_TIMESTAMP	システム割り当てまたはユーザー定義の項目属性に変更が行われたときのタイム・スタンプ。	システムによって自動的に割り当てられ維持されます。
OIM_ID_ITEM_ID	項目の項目 ID。	システムによって自動的に割り当てられ維持されます。

CLASSATTRSTRUCT (クラス属性構造)

このデータ構造には、索引クラスに関して定義された属性に関する特定の情報が入っています。内容は以下のとおりです。

```
typedef  struct  _CLASSATTRSTRUCT

{
ULONG                   ulStruct;
BOOL                    fAttrRequiredField;
BITS                    fAttrAccess;
USHORT                  usAttrId;

}  CLASSATTRSTRUCT,  *PCLASSATTRSTRUCT;
```

フィールド

ulStruct

ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

fAttrRequiredField

BOOL — 出力

この属性にとって値が必須であるかどうかを示すフラグ。有効な値は、次のとおりです。

TRUE 値が必要であることを示します。

FALSE
値が必須でないことを示します。

制約事項: このフィールドは、索引クラスの場合のみ意味をもちます。ビューの場合は、意味をもちません。

fAttrAccess BITS — 出力

属性へのアクセス・タイプを示すフラグ。このフィールドは、ビューでのみ意味を持ちます。索引クラスでは意味を持ちません。有効な値は、次のとおりです。

SIM_ATTR_READABLE
読み取りアクセスであることを示します。

SIM_ATTR_READWRITE
読み取りアクセスと書き込みアクセスであることを示します。この値は、SIM_ATTR_READABLE と SIM_ATTR_WRITEABLE の組み合わせです。

SIM_ATTR_WRITEABLE
書き込みアクセスであることを示します。

usAttrId USHORT — 出力

属性の固有 ID。

CLASSINDEXATTRSTRUCT (クラス索引属性構造)

このデータ構造には、索引クラス属性テーブルにある索引の持つ属性に関する情報が入っています。内容は以下のとおりです。

```
typedef struct _CLASSINDEXATTRSTRUCT

{
ULONG       ulStruct;
USHORT      usAttrId;
USHORT      usIndexSortOrder;

} CLASSINDEXATTRSTRUCT, *PCLASSINDEXATTRSTRUCT;
```

フィールド

ulStruct ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

usAttrId USHORT — 出力

属性の固有 ID。属性はユーザー定義のものであり、システム定義のものではありません。また、この属性は、この索引が要求される索引クラス内になければなりません。

usIndexSortOrder USHORT — 出力

Content Manager for iSeries では、これは常に SIM_INDEX_ASCENDING に設定されます。

CLASSINDEXSTRUCT (クラス索引構造)

このデータ構造には、索引クラスにデータベース索引を作成するために使用される索引クラス属性が入っています。内容は以下のとおりです。

```
typedef struct _CLASSINDEXSTRUCT

{
ULONG                       ulStruct;
BITS                        fIndexFlags;
PCLASSINDEXATTRSTRUCT       pClassIndexAttr;
USHORT                      usNbrAttrIds;
CHAR                        szIndexName;

} CLASSINDEXSTRUCT, *PCLASSINDEXSTRUCT;
```

フィールド

ulStruct — ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

fIndexFlags — BITS — 出力

サポートされていません。

pClassIndexAttr — PCLASSINDEXATTRSTRUCT — 出力

クラス索引属性情報が含まれる ClassIndexAttrStruct データ構造へのポインター。詳細については、CLASSINDEXATTRSTRUCT (クラス索引属性構造) を参照してください。

usNbrAttrIds — USHORT — 出力

ClassIndexAttrStruct 構造の中にある属性 ID の数。

szIndexName — CHAR[*SIM_INDEX_NAME_LENGTH*+1] — 出力

索引クラスのデータベース索引の固有名。

CLASSINFOSTRUCT (索引クラス情報構造)

このデータ構造は、索引クラスに関する情報を提供します。内容は以下のとおりです。

```
typedef struct _CLASSINFOSTRUCT

{
ULONG                       ulStruct;
PCLASSATTRSTRUCT            pClassAttrStruct;
USHORT                      usNbrAttrIds;
USHORT                      usMaxVersions;
USHORT                      usIndexClass;
USHORT                      usViewID;
CHAR                        szACLName;
CHAR                        achLanguageCode;
```

```
CHAR                    szClassName;
CHAR                    szDescription;
CHAR                    szCollectionName;
CHAR                    szStoreSite;

} CLASSINFOSTRUCT,  *PCLASSINFOSTRUCT;
```

フィールド

ulStruct
ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

pClassAttrStruct
PCLASSATTRSTRUCT — 出力

クラス属性構造の配列へのポインター。

usNbrAttrIds
USHORT — 出力

CLASSATTRSTRUCT 配列の中にある属性 ID の数。属性をもたないクラスの場合、この値はゼロとなり、*pClassAttrStruct* フィールドには値 NULL が入ります。

usMaxVersions
USHORT — 出力

サポートされていません。

usIndexClass
USHORT — 出力

索引クラス ID。

usViewID
USHORT — 出力

既存の索引クラス・ビューの ID。

Content Manager for iSeries は、索引クラスと同じ ID を持つ 1 つのビューだけをサポートしています。

szACLName
CHAR[*SIM_ACCESS_LIST_NAME_LENGTH*+1] — 出力

索引クラスのアクセス・リスト (ACL) の名前。

achLanguageCode
CHAR[*SIM_LANGUAGE_CODE_LENGTH*+1] — 出力

この索引クラス名またはビュー名の 3 文字からなる各国語コード。言語コードの値に関しては、「*IBM National Language Design Guide: National Language Support Reference Manual Volume 2*」に説明があります。

szClassName
CHAR[*SIM_CLASS_NAME_LENGTH*+1] — 出力

指定の言語で表記された索引クラス名またはビュー名。

szDescription
CHAR[*SIM_DESCRIPTION_LENGTH*+1] — 出力

サポートされていません。

szCollectionName CHAR[*SIM_COLLECTION_NAME_LENGTH*+1] — 出力

指定の索引クラス内の新規オブジェクトに関するデフォルト・コレクション。ビューの場合、これは、そのビューと関連づけられている索引クラスのものと同じ値です。これは、**SimLibGetClassInfo** 関数のビューでのみ意味を持ちます。

szStoreSite CHAR[*SIM_SERVER_NAME_LENGTH*+1] — 出力

サポートされていません。

CONTENTCLASSINFO (コンテンツ・クラス情報構造)

この情報構造は、コンテンツ・クラスの作成および修正に必要なデータを提供します。内容は以下のとおりです。

```
typedef struct _CONTENTCLASSINFO

{
ULONG       ulStruct;
USHORT      usContentClsID;
CHAR        szContentClsName;
CHAR        szContentClsDesc;

} CONTENTCLASSINFO, *PCONTENTCLASSINFO;
```

フィールド

ulStruct ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

usContentClsID USHORT — 出力

Content Manager for iSeries が生成する固有のコンテンツ・クラス ID。

szContentClsName CHAR[9] — 出力

コンテンツ・クラスの名前。

szContentClsDesc CHAR[41] — 出力

コンテンツ・クラスの記述。

HOBJ (記憶オブジェクトの照会のためのハンドル)

このハンドルは、照会対象となる記憶オブジェクトを識別します。これは、実際には次のような構成のデータ構造へのポインターです。

```
typedef struct _OBJSTRUCT

{
```

```
ULONG       ulStruct;
ULONG       ulPart;
SHORT       sVersion;
ITEMID      szItemID;
UCHAR       chRepType;
UCHAR       chReserved;

} OBJ, *HOBJ;
```

フィールド

ulStruct
ULONG — 入力/出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

ulPart
ULONG — 入力/出力

オブジェクトの部分番号。正の値だけが有効です。

sVersion
SHORT — 入力

サポートされていません。

szItemID
ITEMID — 入力/出力

オブジェクトの項目 ID。

chRepType
UCHAR[*SIM_REP_TYPE*] — 入力/出力

サポートされていません。

chReserved
UCHAR[*SIM_OBJ_RESERVED_LENGTH*] — 入力

予約されています。

ICVIEWSTRUCT (索引クラスのビュー情報構造)

このデータ構造は、索引クラスまたは索引クラスのビュー情報構造に関する情報を提供します。内容は以下のとおりです。

```
typedef struct _ICVIEWSTRUCT

{
ULONG                   ulStruct;
struct _ICVIEWSTRUCT    *pNextView;
PATTRLISTSTRUCT         pAttr;
USHORT                  usIndexClass;
USHORT                  usViewId;
USHORT                  usNumAttributes;

} ICVIEWSTRUCT, *PICVIEWSTRUCT;
```

フィールド

ulStruct
ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

pNextView struct _ICVIEWSTRUCT * — 出力

項目に関するビュー情報のリンク済みリストで、次に位置するフィールドへのポインター。このリスト内の各フィールドは、ICVIEWSTRUCT データ構造です。Content Manager for iSeries の場合、このポインターには常に値 NULL が入ります。

pAttr PATTRLISTSTRUCT — 出力

ATTRLISTSTRUCT データ構造の配列へのポインター。各データ構造には、この項目に関する現行ビューの、システム定義またはユーザー定義の属性 ID が入っています。配列内にある 1 つのデータ構造は、1 つの属性を指定します。

usIndexClass USHORT — 出力

この項目の索引クラス ID。

usViewId USHORT — 出力

既存の索引クラス・ビューの ID。

Content Manager for iSeries は、索引クラスと同じ ID を持つ 1 つのビューだけをサポートしています。

usNumAttributes USHORT — 出力

この項目に関して存在する属性値の数。このフィールドの値は、*pAttr* フィールドが指し示す ATTRLISTSTRUCT データ構造の数に一致します。

ITEMINFOSTRUCT (項目情報構造)

このデータ構造は、要求された項目に関する情報を提供します。内容は以下のとおりです。

```
typedef struct _ITEMINFOSTRUCT

{
ULONG       ulStruct;
BOOL        fSuspended;
USHORT      usItemType;
USHORT      usIndexClass;
ULONG       ulOpenStatus;
USHORT      usWipStatus;
USERID      useridCheckout;
CHAR        szLabel;

} ITEMINFOSTRUCT, *PITEMINFOSTRUCT;
```

フィールド

ulStruct ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

fSuspended

BOOL — 出力

サポートされていません。

usItemType

USHORT — 出力

SimLibGetItemInfo 関数を使用して検索された項目のタイプ。有効な値は、次のとおりです。

SIM_DOCUMENT
項目が文書であることを示します。

SIM_FOLDER
項目がフォルダーであることを示します。

SIM_WORKBASKET
項目がワーク・バスケットであることを示します。

SIM_WORKFLOW
項目がプロセス中であることを示します。

usIndexClass

USHORT — 出力

索引クラス ID。

SimLibGetItemInfo 関数の場合、この値は照会対象の項目の索引クラス ID を指定します。

ulOpenStatus

ULONG — 出力

項目が更新可能状態になっているかどうかを示すインディケーター。このパラメーターと *useridCheckout* パラメーターは、一体となって機能し、だれが何の目的でこの項目を持っているかに関する情報を提供します。有効な値は、次のとおりです。

SIM_ACCESS_READ_WRITE
ユーザーが項目を更新可能状態にしていることを示します。

SIM_ACCESS_UNKNOWN
ユーザーが項目をまだ更新可能状態にしていないことを示します。

usWipStatus

USHORT — 出力

項目の現行の WIP 状況。このフィールドの値は、項目が中断されているかどうか、およびその項目のワークフロー状況を示します。OR 演算子は、1 つの中断状況値を、以下のグループのいずれかのワークフロー状況値と結合するために使用されます。

中断状況に関する値
サポートされていません。

ワークフロー状況に関する値

OIM-CURRENT_WORKFLOW_ITEMS
項目がプロセス中であることを示します。

OIM_ITEMS_NOT_IN_WORKFLOW
項目がプロセス中でないことを示します。

useridCheckout

USERIDENT — 出力

項目をチェックアウトした人のユーザー ID。このパラメーターと *ulOpenStatus* パラメーターは一体となって機能して、だれが何の目的でこの項目を持っているのかに関する情報を提供します。有効な値は、次のとおりです。

ユーザー自身の ID
ulOpenStatus に値 SIM_ACCESS_READ_WRITE が入っている場合には、ユーザーが項目を永続的にチェックアウトして、それを更新可能状態にしたことを示します。それ以外の場合は、ユーザーは、項目を永続的にチェックアウトしたが、まだ更新可能状態にはなっていません。

他のユーザー ID
ulOpenStatus に値 SIM_ACCESS_UNKNOWN が入っている場合には、項目をチェックアウトした別のユーザーを識別します。

ヌルのストリング
ulOpenStatus に値 SIM_ACCESS_READ_WRITE が入っている場合は、ユーザーが項目を更新可能状態にしていることを示します。それ以外の場合、項目はチェックアウトされていません。

szLabel

CHAR[*SIM_LABEL_LENGTH*+1] — 出力

項目の名前またはラベルが入っている NULL 文字で終了するストリング。

ITEMNAMESTRUCT (項目名データ構造)

このデータ構造には、ワーク・バスケットまたはプロセス項目に関連付けられた名前があります。

```
typedef  struct_ITEMNAMESTRUCT

{
```

```
ULONG                   ulStruct;
ITEMID                  WItemID;
CHAR                    szIDName;
ULONG                   ulActive;

} ITEMNAMESTRUCT, *PITEMNAMESTRUCT;
```

フィールド

ulStruct

ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

WItemID

ITEMID — 出力

ワーク・バスケットまたはプロセスの項目 ID。

szIDName

CHAR[*OIM_ITEMNAME_LENGTH*+1] — 出力

項目の記述。

ulActive

ULONG — 出力

Content Manager for iSeries の場合、ワーク・バスケットかプロセスの状況。有効な値は、次のとおりです。

SIMWM_ACTIVE
ワーク・バスケットかプロセスがアクティブであることを示します。

SIMWM_INACTIVE
ワーク・バスケットかプロセスに削除のマークが付けられていることを示します。

LIBSEARCHCRITERIASTRUCT (検索基準情報構造)

このデータ構造は、検索する索引クラスおよび検索式自体に関する情報を提供します。内容は以下のとおりです。

```
typedef struct _LIBSEARCHCRITERIASTRUCT

{
ULONG                   ulStruct;
ULONG                   ulReturnLimit;
BITS                    fSearch;
PSZ                     pszSearchString;
USHORT                  usViewID;
USHORT                  usSearchUnused;

} LIBSEARCHCRITERIASTRUCT, *PLIBSEARCHCRITERIASTRUCT;
```

フィールド

ulStruct

ULONG — 入力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

ulReturnLimit

ULONG — 入力

指定した索引クラスに関して、検索が戻す項目の最大数。*fSearch* フィールドの値として SIM_SEARCH_ALLVIEWS を指定すると、このフィールドの値は、検索する索引クラスごとにその検索の戻す項目の最大数になります。このフィールドの値として 0 を指定すると、指定した索引クラスに関して、検索基準に一致するすべての項目が戻されます。

fSearch

BITS - 入力

検索修正インディケーター。このフィールドの値は、検索に対する修正を決定します。有効な値は、次のとおりです。

SIM_SEARCH_VIEW

usViewID フィールドで指定したビューのみを検索します。この値を指定する場合は、*usViewID* フィールドで有効なビューの ID を指定する必要があります。

SIM_SEARCH_ALLVIEWS

1 つのビューだけでなく、該当する現行のビューすべてを検索します。この値を指定する場合には、*usViewID* フィールドの値としてゼロを指定する必要があります。検索基準の配列内において、データ構造のただ 1 つについてのみこの値を指定できます。

この値を指定すれば、**SimLibSearch** 関数は、 *pszSearchString* フィールド内で式に指定した属性を持つビューのみを自動的に検索します。

pszSearchString

PSZ - 入力

NULL 文字で終了するストリングへのポインター。このフィールドには、 1 つ以上の式が入ります。それぞれの式は、1 つの属性についての検索条件を記述します。論理演算子を使用して、検索の式を組み合わせてください。レベルと括弧の数には制限がありません。このリストのあとにある『検索式の指針』をお読みください。

usViewID

USHORT - 入力

既存の索引クラスの ID。

usSearchUnused
USHORT - 入力

予約フィールド。

制約事項: SimLibSearch 関数は、この値を使用しません。

検索式の指針

325 ページの『付録 A. 検索式の指針』を参照してください。

LIBSESSIONINFOSTRUCT (ライブラリー・セッション情報構造)

このデータ構造は、現行ライブラリー・セッションに関する情報を提供します。このデータ構造が適用されるのは、**SimLibLogon** 関数を使って現行セッションを開始した場合に、HSESSION パラメーターの値として指定した現行ライブラリー・セッションです。内容は以下のとおりです。

```
typedef struct _LIBSESSIONINFOSTRUCT

{
ULONG                   ulStruct;
SESSION_P               pSession;
CHAR                    szDBName;
CHAR                    szApplicationName;
PATRON_ID               szUserIDSession;

} LIBSESSIONINFOSTRUCT, *PLIBSESSIONINFOSTRUCT;
```

フィールド

ulStruct
ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

pSession
SESSION_P — 出力

ライブラリー・クライアント・セッションのハンドル。

szDBName
CHAR[*RS_STORE_ID_LENGTH+1*] — 出力

サポートされていません。

szApplicationName
CHAR[*RS_STORE_ID_LENGTH+1*] — 出力

サポートされていません。

szUserIDSession
PATRON_ID — 出力

セッション用の現行のユーザー ID。

NAMESTRUCT (名前データ構造)

このデータ構造は、属性または索引クラス・ビューのコードに関連づけられた名前を提供します。内容は以下のとおりです。

```
typedef struct _NAMESTRUCT

{
ULONG                   ulStruct;
USHORT                  usID;
CHAR                    szName;
CHAR                    szDescription;

} NAMESTRUCT, *PNAMESTRUCT;
```

フィールド

ulStruct ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

usID USHORT — 出力

有効な属性、索引クラス、または索引クラス・ビューの ID。

szName CHAR[*SIM_CLASS_NAME_LENGTH*+1] — 出力

現行の言語で表された索引クラスまたはビューの名前。

szDescription CHAR[*SIM_DESCRIPTION_LENGTH*+1] — 出力

サポートされていません。

OBJINFOSTRUCT (オブジェクト情報構造)

このデータ構造は、オブジェクトについての記憶情報を提供します。内容は以下のとおりです。

```
typedef struct _OBJINFOSTRUCT

{
ULONG                   ulStruct;
ULONG                   ulObjSize;
LONG                    lSMSRetention;
LONG                    lEstimateRetrieveTime;
ULONG                   ulAvail;
ULONG                   ulObjConCls;
USHORT                  usPageNum;
TIMESTAMP               tsCreate;
TIMESTAMP               tsExpiration;
TIMESTAMP               tsLastRef;
TIMESTAMP               tsModify;
TIMESTAMP               tsEnterSG;
TIMESTAMP               tsEnterSC;
CHAR                    szCollectionName;
CHAR                    szObjectName;
CHAR                    szMgtCls;
CHAR                    szStgCls;
CHAR                    szDataCls;
```

```
CHAR                    szStoreSite;

} OBJINFOSTRUCT, *POBJINFOSTRUCT;
```

フィールド

ulStruct
ULONG — 出力
このフィールドの長さも含め、構造の長さをバイト単位で表します。

ulObjSize
ULONG — 出力
バイト数で表されたオブジェクトの合計サイズ。

lSMSRetention
LONG — 出力
サポートされていません。

lEstimateRetrieveTime
LONG — 出力
サポートされていません。

ulAvail
ULONG — 出力
サポートされていません。

ulObjConCls
ULONG — 出力
照会するオブジェクトのオブジェクト・コンテンツ・クラス。値 SIM_CC_UNKNOWN は、未定義のコンテンツ・クラスを表します。

usPageNum
USHORT — 出力
サポートされていません。

tsCreate
TIMESTAMP — 出力
項目またはオブジェクトが作成された日時。

tsExpiration
TIMESTAMP — 出力
サポートされていません。

tsLastRef
TIMESTAMP — 出力
サポートされていません。

tsModify
TIMESTAMP — 出力
項目またはオブジェクトが最後に修正された日時。

tsEnterSG
TIMESTAMP — 出力
サポートされていません。

tsEnterSC
TIMESTAMP — 出力
サポートされていません。

szCollectionName
CHAR[*MAXCOLNMSZ*] — 入力
サポートされていません。

szObjectName
CHAR[*MAXOBJNMSZ*] — 入力

サポートされていません。

szMgtCls CHAR[*MAXMGTCLSNMSZ*] — 出力

サポートされていません。

szStgCls CHAR[*MAXSTGCLSNMSZ*] — 出力

サポートされていません。

szDataCls CHAR[*MAXDATACLSNMSZ*] — 出力

サポートされていません。

szStoreSite CHAR[*MAXSTRSITENMSZ*] — 出力

サポートされていません。

RCSTRUCT (戻りコード情報構造)

このデータ構造は、プログラミング・インターフェース関数の戻りコードとデータ情報を提供します。内容は以下のとおりです。

```
typedef  struct  _RCSTRUCT

{
ULONG                   ulStruct;
ULONG                   ulRC;
USHORT                  usReserved;
USHORT                  usParam;
ULONG                   ulParam1;
ULONG                   ulParam2;
#ifdef                  _OS400_
PVOID                   pParam1;
PVOID                   pParam2;
#endif
ULONG                   ulExtRC;
ULONG                   ulExtReason;
PVOID                   pApplData;
ULONG                   ulApplData;
ULONG                   ulReserved;
HERR                    hErrLog;

}  RCSTRUCT,  *PRCSTRUCT;
```

フィールド

ulStruct ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

ulRC ULONG — 出力

関数の戻りコード。

usReserved USHORT — 出力

サポートされていません。

usParam

USHORT — 出力

ulParam1 フィールドがデータ域へのポインターをもっているかどうかを示すフィールド。値 1 は、ポインターを持っている場合です。その他の場合、このフィールドには値 0 が入ります。

ulParam1

ULONG — 出力

データ構造かまたはデータ構造の配列のどちらかへの値またはポインター。

ulParam2

ULONG — 出力

ulParam1 フィールドにデータ構造の配列へのポインターが入っている場合に、その配列内にあるデータ構造の数を示すフィールド。

pParam1

PVOID — 出力

該当の関数がサーバー上で実行されたときに、*ulParam1* の代わりに使用されるポインター。

pParam2

PVOID — 出力

該当の関数がサーバー上で実行されたときに、*ulParam2* の代わりに使用されるポインター。

ulExtRC

ULONG — 出力

Content Manager for iSeries が直接的または間接的に呼び出した他の構成要素からの戻りコード。

ulExtReason

ULONG — 出力

サポートされていません。

pApplData

PVOID — 出力

アプリケーションがアプリケーション・データを入れるために使用できる PVOID データ・フィールド。Content Manager for iSeries は、このデータ・フィールドを使用しません。値は、プログラミング・インターフェース関数によって保持され、あとで戻されます。たとえば、アプリケーションはこのフィールドを使用してデータ構造、すなわち、データを必要とする関数の使用に先立ってアプリケーションが作成するデータ構造を指し示すことができます。関数は、その構造内のデータを使用して、ユーザー出口を処理することもできます。

ulApplData

ULONG — 出力

アプリケーションがアプリケーション・データを入れるために使用できる ULONG データ・フィールド。Content Manager for iSeries は、このデータ・フィールドを使用しません。値は、プログラミング・インターフェース関数によって保持され、あとで戻されます。たとえば、アプリケーションはこの

フィールドを使用してデータ構造、すなわち、データを必要とする関数の使用に先立ってアプリケーションが作成するデータ構造を指し示すことができます。関数は、その構造内のデータを使用して、ユーザー出口を処理することもできます。

ulReserved
ULONG — 出力

サポートされていません。

hErrLog
HERR — 出力

サポートされていません。

SERVERINFOSTRUCT (サーバー情報構造)

この構造には、システムに対して定義されたサーバーに関する情報が入っています。このデータ構造は、呼び出し側のアプリケーションに戻されます。内容は以下のとおりです。

```
typedef struct _SERVERINFOSTRUCT

{
ULONG                   ulStruct;
CHAR                    szServerName;
CHAR                    szServerType;

} SERVERINFOSTRUCT, *PSERVERINFOSTRUCT;
```

フィールド

ulStruct
ULONG — 入力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

szServerName
CHAR[*SERVERNAME_LENG*+1]— 出力

Content Manager for iSeries サーバーの名前。

szServerType
CHAR[*SERVERTYPE_LENG*+1]— 出力

サーバー・タイプ。現行のサーバー・タイプには、以下のものがあります。

サーバー・タイプ	記述
'FRNCACHE'	リスト・マネージャー・キャッシュ。
'FRNREXE'	リモート・ユーティリティー・サーバー。
'FRNCS'	構成サーバー。
'FRNOSADM'	システム管理ストレージ・サーバー。
'FRNOLM'	リスト・マネージャー・サーバー。

SMS (システム管理ストレージ・ポインター)

オブジェクトのシステム管理ストレージ (SMS) データ構造へのポインター。このデータ構造は、各種のオブジェクト・サーバー上にあるオブジェクトの SMS をサポートするために必要な情報を提供します。これは、実際には次のような構成のデータ構造へのポインターです。

```
typedef struct _SMS

{
ULONG       ulStruct;
LONG        lSMSRetention;
CHAR        szCollectionName;
CHAR        szObjectName;
CHAR        szMgtCls;
CHAR        szStgCls;
CHAR        szDataCls;
CHAR        szStoreSite;
CHAR        szStoreHint;

} SMS, *PSMS;
```

フィールド

ulStruct

ULONG — 入力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

lSMSRetention

LONG — 入力

Content Manager for iSeries がオブジェクトをシステム管理ストレージに保存している日数。有効な値の範囲は、1 から 999 999 999 までです。

szCollectionName

CHAR[*MAXCOLNMSZ*] — 入力

ASCIIZ からなるユーザー定義のコレクション名。このフィールドの値は、クライアント・データ・スペース内のゼロで終了するストリングを参照し、ユーザー定義のいくつかの有効な文字を含んでいます。この文字ストリングは、作成されるコレクションについて有用な名前を提供します。コレクション名が必要でなければ、値として NULL を指定してください。オブジェクト・サーバー上でコレクションに対してオブジェクトが割り当てられると、その後で、コレクションの割り当てを変更することはできません。

szObjectName

CHAR[*MAXOBJNMSZ*] — 入力

サポートされていません。

szMgtCls

CHAR[*MAXMGTCLSNMSZ*] — 入力

サポートされていません。

szStgCls CHAR[*MAXSTGCLSNMSZ*] — 入力

サポートされていません。

szDataCls CHAR[*MAXDATACLSNMSZ*] — 入力

サポートされていません。

szStoreSite CHAR[*MAXSTRSITENMSZ*] — 入力

オブジェクトが保管される場所であるオブジェクト・サーバーの名前。

szStoreHint CHAR[*MAXSTGHINTNMSZ*] — 入力

サポートされていません。

SNAPSHOTSTRUCT (スナップショット情報構造)

このデータ構造は、ある特定の時点における項目のビュー情報、属性情報、およびワーク・マネージメント情報を提供します。内容は以下のとおりです。

```
typedef struct _SNAPSHOTSTRUCT

{
ULONG                   ulStruct;
PWMSNAPSHOTSTRUCT       pWmSnapshot;
USHORT                  usNumWmSnapshots;
PICVIEWSTRUCT           pICView;
USHORT                  usNumViews;
USHORT                  usItemType;
ULONG                   ulOpenStatus;
ITEMID                  szItemID;
USERID                  useridCheckout;
TIMESTAMP               tsCreate;
TIMESTAMP               tsModify;

} SNAPSHOTSTRUCT, *PSNAPSHOTSTRUCT;
```

フィールド

ulStruct ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

pWmSnapshot PWMSNAPSHOTSTRUCT — 出力

タイプが WMSNAPSHOTSTRUCT であるワークフロー情報のデータ構造へのポインター。*fReadAttrInd* 入力パラメーターに SIM_WORK_ATTR を指定した場合に、**SimLibGetItemSnapshot** 関数はこの構造を戻します。それ以外の場合には、このフィールドには NULL 値が入ります。

Content Manager for iSeries は、1 つの項目が複数のワーク・バスケットにあることをサポートするの

で、このフィールドは、ある項目に関するワークフロー情報の配列になる場合があります。

usNumWmSnapshots

USHORT - 入力

pWmSnapshot が指す WMSNAPSHOTSTRUCT の配列内の要素の数。

pICView

PICVIEWSTRUCT — 出力

項目のビュー情報のリンク・リストへのポインター。このリストの各要素は、データ型として ICVIEWSTRUCT を持っています。項目がどの索引クラスにも関連づけられていない場合、またはシステム属性を取り出していない場合には、このポインターには NULL 値が入っています。

Content Manager for iSeries では現在、項目が索引クラスに関連づけられている場合には、その項目の現行の索引クラス・ビューに関する情報を含んでいるリンク・リスト内には要素が 1 つだけあります。項目がどの索引クラスにも関連づけられていない場合は、このポインターには NULL 値が入っています。

usNumViews

USHORT — 出力

SNAPSHOTSTRUCT データ構造の *pICView* フィールドが指し示すリンク・リスト内の要素の数。

Content Manager for iSeries では現在、項目が索引クラスに関連づけられている場合には、このフィールドには値 1 が入ります。この値は、データ型 ICVIEWSTRUCT の要素のリンク・リストにはその項目の現行の索引クラス・ビューに関する情報を含んだ要素が 1 つ入っていることを示します。項目が索引クラスに関連づけられていない場合には、このフィールドには値 0 が入ります。ただしこの場合でも、システム属性を検索する際には、 *pICView* ポインターはなお有効です。

usItemType

USHORT — 出力

SimLibGetItemSnapshot 関数を使用して検索される項目のタイプ。有効な値は、次のとおりです。

SIM_DOCUMENT
項目が文書であることを示します。

SIM_FOLDER
項目がフォルダーであることを示します。

ulOpenStatus

ULONG — 出力

項目が更新可能状態になっているかどうかを示すインディケーター。このパラメーターと *useridCheckout* パラメーターは、一体となって機能

し、だれが何の目的でこの項目を持っているかに関する情報を提供します。有効な値は、次のとおりです。

SIM_ACCESS_READ_WRITE
ユーザーが項目を更新可能状態にしていることを示します。

SIM_ACCESS_UNKNOWN
ユーザーが項目をまだ更新可能状態にしていないことを示します。

szItemID

ITEMID — 出力

項目 ID。

useridCheckout

USERIDENT — 出力

項目をチェックアウトした人のユーザー ID。このパラメーターと *ulOpenStatus* パラメーターは一体となって機能して、だれが何の目的でこの項目を持っているのかに関する情報を提供します。有効な値は、次のとおりです。

ユーザー自身の ID
ulOpenStatus に値 SIM_ACCESS_READ_WRITE が入っている場合には、ユーザーが項目を永続的にチェックアウトして、それを更新可能状態にしたことを示します。それ以外の場合は、ユーザーは、項目を永続的にチェックアウトしたが、まだ更新可能状態にはなっていません。

他のユーザー ID
ulOpenStatus に値 SIM_ACCESS_UNKNOWN が入っている場合には、項目をチェックアウトした別のユーザーを識別します。

ヌルのストリング
ulOpenStatus に値 SIM_ACCESS_READ_WRITE が入っている場合は、ユーザーが項目を更新可能状態にしていることを示します。それ以外の場合、項目はチェックアウトされていません。

tsCreate

TIMESTAMP — 出力

項目またはオブジェクトが作成された日時。

tsModify

TIMESTAMP — 出力

項目またはオブジェクトが最後に修正された日時。

TOCENTRYSTRUCT (目次項目データ構造)

このデータ構造には、特定のフォルダーやワーク・バスケットに入っている文書およびフォルダーのリストの中にある項目について記述している情報が入っています。内容は以下のとおりです。

```
typedef  struct  _TOCENTRYSTRUCT

{
ULONG                   ulStruct;
USHORT                  usItemStatus;
USHORT                  usIndexClass;
USHORT                  usItemType;
ITEMID                  szItemID;
TIMESTAMP               tsItemChanged;

}  TOCENTRYSTRUCT,  *PTOCENTRYSTRUCT;
```

フィールド

ulStruct
ULONG — 入力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

usItemStatus
USHORT - 入力

更新後の項目の状況。有効な値は、次のとおりです。

- **0** (修正なし)
- **SIM_TOC_ADD**
- **SIM_TOC_MODIFIED**
- **SIM_TOC_DELETE**

usIndexClass
USHORT - 入力

索引クラス ID。

usItemType
USHORT - 入力

SimLibGetTOC 関数を使用して検索された項目のタイプ。有効な値は、次のとおりです。

SIM_DOCUMENT
項目が文書であることを示します。

SIM_FOLDER
項目がフォルダーであることを示します。

szItemID
ITEMID — 入力

項目 ID。

tsItemChanged
TIMESTAMP — 入力

ライブラリー・サーバーの中に保管されている項目のタイム・スタンプ。

USERACCESSSTRUCT (ユーザー・アクセス・データ構造)

このデータ構造には、参照した項目をチェックアウトしたユーザーを記述している情報が入っています。内容は以下のとおりです。

```
typedef  struct  _USERACCESSSTRUCT

{
ULONG                   ulStruct;
ULONG                   ulAccessLevel;
USERIDENT               useridCheckout;
ITEMID                  szItemID;

}  USERACCESSSTRUCT,  *PUSERACCESSSTRUCT;
```

フィールド

ulStruct
ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

ulAccessLevel
ULONG — 出力

サポートされていません。

useridCheckout
USERIDENT — 出力

この項目をチェックアウトした人のユーザー ID。現在この項目がチェックアウトされていなければ、このフィールドには NULL 値が入っています。

szItemID
ITEMID — 出力

項目 ID。

USERLOGONINFOSTRUCT (ユーザー・ログオン情報構造)

このデータ構造は、ユーザーのセッションに関する情報を提供します。内容は以下のとおりです。

```
typedef  struct  _USERLOGONINFOSTRUCT

{
ULONG                   ulStruct;
ULONG                   ulUserType;
ULONG                   ulUserCCSID;
PSZ                     pszUserDescription;
CHAR                    szUserLanguage;
CHAR                    szSessionType;
TIMESTAMP               tsPasswordExpire;
CHAR                    szPrivString;

}  USERLOGONINFOSTRUCT,  *PUSERLOGONINFOSTRUCT;
```

フィールド

ulStruct
ULONG — 入力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

ulUserType
ULONG — 入力

サポートされていません。

ulUserCCSID
ULONG — 入力

サポートされていません。

pszUserDescription
PSZ - 入力

サポートされていません。

szUserLanguage
CHAR[*SIM_LANGUAGE_CODE_LENGTH*+1] — 入力

固定長の文字配列で、このユーザーがダイアログおよびメッセージで優先使用する言語を表します。有効な値は、IBM 標準の 3 文字からなる言語コードです。言語コードの値に関しては、「*IBM National Language Design Guide: National Language Support Reference Manual Volume 2*」に説明があります。

szSessionType
CHAR[*SIM_SESSION_TYPE_LENGTH*+1] — 入力

ログオン・セッションのタイプ。このフィールドの唯一の有効な値は、Ip2 です。

tsPasswordExpire
TIMESTAMP — 入力

現在のパスワードが失効する日付。

szPrivString
CHAR[*SIM_PRIVSTRING_LENGTH*+1] — 入力

NULL 文字で終了する文字ストリングで、ユーザーの特権ベクトルを表します。このストリングは、ASCII ゼロ、そのゼロに対応するもの、およびユーザーの対応する特権ベクトル内のものから構成されます。

WMACTIONLISTFUNCSTRUCT (アクション・リストの機能構造)

このデータ構造は、アクション・リストで定義されているアクションについての情報を提供します。

```
typedef struct _WMACTIONLISTFUNCSTRUCT

{
ULONG       ulFuncNumber;
ULONG       ulActionType;
ULONG       ulFuncCode;
CHAR        szFuncPrompt;
CHAR        szAction;
CHAR        szIcon;
```

```
CHAR                    szShortcut;
CHAR                    szExitFuncName;
CHAR                    szExitDLLName;

} WMACTIONLISTFUNCSTRUCT, *PWMACTIONLISTFUNCSTRUCT;
```

フィールド

ulFuncNumber

ULONG — 出力

アクション・リスト内のアクションのシーケンス番号。

ulActionType

ULONG — 出力

アクションが、文書項目タイプとフォルダー項目タイプのどちらに適用されるか、あるいは両方の項目タイプに適用されるかを示します。有効な値は、次のとおりです。

SIMWM_ACTION_DOCUMENT
アクションは文書項目と関連付けられています。

SIMWM_ACTION_FOLDER
アクションはフォルダー項目と関連付けられています。

SIMWM_ACTION_BOTH
アクションはフォルダー項目と文書項目の両方に関連付けられています。

ulFuncCode

ULONG — 出力

アクションが固有に識別される値。

szFuncPrompt

CHAR[*SIMWM_AL_PROMPT*+1] — 出力

このアクションに関連付けられたテキスト・プロンプト。

szAction

CHAR[*SIMWM_AL_ACTION*+1] — 出力

このアクションが選択されたときに、 SIMWM_ACTION 変数に割り当てられる値。

szIcon CHAR[*SIMWM_AL_ICON*+1] — 出力

このアクションに関連付けられたアイコン。

szShortcut

CHAR[*SIMWM_AL_SHORTCUT*+1] — 出力

このアクションに関連付けられたキーボードのショートカット。

szExitFuncName

CHAR[*OIM_WB_FUNCTION_LENGTH*+1] — 出力

ユーザー定義のアクションの場合、このフィールドには実行するユーザー出口関数の名前が入ります。

szExitDLLName

CHAR[*OIM_WB_DLL_LENGTH*+1] — 出力

ユーザー定義のアクションの場合、このフィールドには関数 *szExitFuncName* を含むダイナミック・リンク・ライブラリーの名前が入ります。

WMACTIONLISTINFOSTRUCT (アクション・リストのデータ構造)

このデータ構造は、1 つのアクション・リスト定義に関連するすべての情報を提供します。

```
typedef  struct  _WMACTIONLISTINFOSTRUCT

{
ULONG                          ulStruct;
CHAR                           szActionListName;
TIMESTAMP                      tsALCreate;
TIMESTAMP                      tsALModify;
CHAR                           szDescription;
ULONG                          ulALNumFunctions;
PWMACTIONLISTFUNCSTRUCT        pALFunctions;

}WMACTIONLISTINFOSTRUCT,  *PWMACTIONLISTINFOSTRUCT;
```

フィールド

ulStruct

ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

szActionListName

CHAR[*SIMWM_ACTION_LENGTH*+1] — 出力

アクション・リストの名前。

tsALCreate

TIMESTAMP — 出力

アクション・リストが作成された日時。

tsALModify

TIMESTAMP — 出力

アクション・リストが最後に変更された日時。

szDescription

CHAR[*SIMWM_AL_DESCRIPTION*+1] — 出力

アクション・リストの記述。

ulALNumFunctions

ULONG — 出力

アクション・リストに関連付けられた機能の名前。

pALFunctions
PWMACTIONLISTFUNCSTRUCT — 出力

アクション・リストに関連付けられた機能リストのポインター。

WMHISTLOGENTRYSTRUCT (WMEvent ヒストリー構造)

このデータ構造は、ある作業パッケージのヒストリー・ログ・エントリーのいずれかの配列に含まれる、その作業パッケージのヒストリーを提供します。

```
typedef struct _HISTLOGENTRY

{
CHAR                    szEventID;
TIMESTAMP               tsCreated;
CHAR                    szProcess;
CHAR                    szLocation;
USERIDENT               szUser;
CHAR                    szEventData;

}WMHISTLOGENTRYSTRUCT, *PWMHISTLOGENTRY;
```

フィールド

szEventID
CHAR[7] — 出力

8 文字のメッセージ ID。

tsCreated
TIMESTAMP — 出力

イベントの日時。

szProcessID
CHAR[*SIMWM_PROCESS_NAME_LENGTH*+1] — 出力

ワークフロー・プロセス名。

szLocation
CHAR[*SIMWM_LOC_NAME_LENGTH*+1] — 出力

ワークフローのロケーション名。

szUser USERIDENT — 出力

ユーザー ID。

szEventData
CHAR[256] — 出力

イベントに関連付けられたテキスト記述。

WMLOCATIONINFOSTRUCT (ワーク・プロセスのロケーション情報構造)

このデータ構造は、あるプロセスの各ロケーションに関連付けられている情報を提供します。

```
typedef struct _WMLOCATIONINFOSTRUCT

{
ULONG                   ulType;
CHAR                    szLocation;
CHAR                    szDescription;
ULONG                   ulActive;

} WMLOCATIONINFOSTRUCT, *PWMLOCATIONINFOSTRUCT;
```

フィールド

ulType ULONG — 出力

戻される情報がワーク・バスケットまたはコレクション・ポイントであるかを示します。有効な値は、次のとおりです。

SIM_WORKBASKET
ロケーションがワーク・バスケットであることを示します。

SIM_COLLECTION_POINT
ロケーションがコレクション・ポイントであることを示します。

szLocation

CHAR[*SIMWM_LOC_NAME_LENGTH*+1] — 出力

ワーク・バスケットまたはコレクション・ポイントの ID。

szDescription

CHAR[*SIMWM_LOC_DESC_LENGTH*+1] — 出力

ロケーションに関連付けられたテキスト記述。

ulActive

ULONG — 出力

サポートされていません。

WMPROCESSINFOSTRUCT (プロセス情報データ構造)

このデータ構造には、特定のプロセスに関する情報があります。

```
typedef struct _WMPROCESSINFOSTRUCT

{
ULONG                   ulStruct;
CHAR                    szProcessID;
CHAR                    szProcessDescription;
CHAR                    chAccessListName;
USHORT                  usHistoryLogDisposition;
ULONG                   ulNbrItemsInProcess;
ULONG                   ulNbrLocations;
UCHAR                   szPrivString;
PWMLOCATIONINFOSTRUCT   pLocations;

} WMPROCESSINFOSTRUCT, *PWMPROCESSINFOSTRUCT;
```

フィールド

ulStruct

ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

szProcessID

CHAR[*SIM_PROCESS_NAME_LENGTH*+1] — 出力

プロセス ID。

szProcessDescription;

CHAR[*SIM_DESCRIPTION_LENGTH*+1] — 出力

プロセスに関連付けられたテキスト記述。

chAccessListName

CHAR[*ACCESS_LIST_NAME_SIZE*+1] — 出力

プロセスのアクセス・リスト名。

usHistoryLogDisposition

USHORT — 出力

サポートされていません。

ulNbrItemsInProcess

ULONG — 出力

プロセスの作業パッケージ数。

ulNbrLocations

ULONG — 出力

プロセス内に定義されている固有のロケーションの数。

szPrivString

UCHAR[*SIM_PRIVSTRING_LENGTH*+1] — 出力

プロセスに関するユーザーの評価済み特権ストリング。

pLocations

PWMLOCATIONINFOSTRUCT — 出力

タイプ WMLOCATIONINFOSTRUCT のロケーション情報データ構造の配列を示すポインター。

WMSNAPSHOTSTRUCT (ワーク・マネージメント情報構造)

このデータ構造には、項目に関するワークフロー情報があります。内容は以下のとおりです。

```
typedef struct _WMSNAPSHOTSTRUCT

{
ULONG                   ulStruct;
USHORT                  usWIPStatus;
USHORT                  usReleaseType;
USHORT                  usPriority;
ITEMID                  szWorkFlowID;
```

```
TIMESTAMP           tsWFEntry;
TIMESTAMP           tsEnteredWB;
ITEMID              szWorkBasketID;
ULONG               ulWorkPackageID;
ULONG               ulInstanceID;
ULONG               ulLocationType;
CHAR                szLocation;
TIMESTAMP           tsEnteredLocation;
CHAR                szOverrideAction;

} WMSNAPSHOTSTRUCT, *PWMSNAPSHOTSTRUCT;
```

フィールド

ulStruct
ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

usWipStatus
USHORT — 出力

項目の現行の WIP 状況。このフィールドの値は、項目が中断されているかどうか、およびその項目のワークフロー状況を示します。OR 演算子は、1 つの中断状況値を、以下のグループのいずれかのワークフロー状況値と結合するために使用されます。

中断状況に関する値

OIM_ITEMS_SUSPENDED
項目が中断されていることを示します。

OIM_ITEMS_NOT_SUSPENDED
項目が中断されていないことを示します。

ワークフロー状況に関する値

OIM-CURRENT_WORKFLOW_ITEMS
項目がプロセス中であることを示します。

OIM_ITEMS_NOT_IN_WORKFLOW
項目がプロセス中でないことを示します。

usReleaseType
サポートされていません。

usPriority
USHORT — 出力

項目の現行の優先順位。

szWorkFlowID
ITEMID — 出力

この項目が割り当てられる対象のプロセスがあれば、そのプロセス。

tsWFEntry
TIMESTAMP — 出力

この項目が、リストされたプロセスに入力された日時。

tsEnteredWB

TIMESTAMP — 出力

この項目が、リストされたワーク・バスケットに入力された日時。

szWorkBasketID

ITEMID — 出力

この項目が割り当てられているワーク・バスケット ID。

ulWorkPackageID

ULONG — 出力

実行中の作業を表す作業パッケージ (たとえば、経路指定される文書) の ID。

ulInstanceID

ULONG — 出力

プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。

ulLocationType

ULONG — 出力

ロケーションがワーク・バスケットであるかコレクション・ポイントであるかを示すインディケーター。有効な値は、次のとおりです。

SIMWM_WORKBASKET
ロケーションがワーク・バスケットであることを示します。

SIMWM_COLLECTION_POINT
ロケーションがコレクション・ポイントであることを示します。

szLocation

CHAR [*SIMWM_LOC_NAME_LENGTH*+1] — 出力

作業パッケージが置かれているロケーションのワーク・バスケット ID またはコレクション・ポイント ID。

tsEnteredLocation

TIMESTAMP — 出力

項目がそのロケーションに入った日時。

szOverrideAction

CHAR[*SIMWM_ACTION_LENGTH*+1] — 出力

作業パッケージに割り当てられたアクション・リスト。このアクション・リストは、ワーク・バスケット定義によって定義されたデフォルトのアクション・リストを指定変更します。

WMSUSPENDSTRUCT (中断作業パッケージのデータ構造)

このデータ構造は、中断された項目のリリース基準に関するデータを提供します。内容は以下のとおりです。

```
typedef struct _WMSUSPENDSTRUCT

{
ULONG                   ulStruct;
USHORT                  usReleaseType;
TIMESTAMP               tsExpDateTime;
CHAR                    szExpWB;
CHAR                    szReadyWB;
USHORT                  usNumAwaitedClasses;
USHORT                  usAwaitedClasses;

} WMSUSPENDSTRUCT, *PWMSUSPENDSTRUCT;
```

フィールド

ulStruct

ULONG — 入力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

usReleaseType

ULONG — 入力

項目を中断状態からリリースするときに適用される基準のタイプ。有効な値は、次のとおりです。

SIMWM_SUSPEND_TIME

tsExpDateTime に指定された時間が満了するまで中断。

SIMWM_SUSPEND_ANY_CLASS

ausAwaitedClasses にリストされているいずれかの索引クラスの項目をフォルダーが受け取るまで中断。この値を指定する場合も、*tsExpDateTime* の時間を事前に設定する必要があります。

SIMWM_SUSPEND_ALL_CLASS

ausAwaitedClasses にリストされている各クラスの項目をフォルダーが受け取るまで中断。この値を指定する場合も、 *tsExpDateTime* の時間を事前に設定する必要があります。

tsExpDateTime

TIMESTAMP — 入力

作業パッケージを中断状態からリリースする日時。

usNumAwaitedClasses

USHORT - 入力

ausAwaitedClasses 配列内の索引クラス項目の数

ausAwaitedClasses に SIM_INDEX_ANY を入力する場合は、この数を 1 にする必要があります。

usAwaitedClasses

CHAR[*SIMWM_MAX_AWAIT_CLASSES*] — 入力

特定のフォルダーの作業パッケージに適用する中断基準として指定可能な、1 ～ 8 つまでの索引クラスの配列。索引クラスとして SIM_INDEX_ANY

を指定することにより、任意の索引クラスのいずれかの項目が到着するまでの間、フォルダーの作業パッケージを中断することもできます。

szExpWB

CHAR[*SIM_ITEM_ID_LENGTH*+1] — 入力

中断された作業パッケージが、中断有効期間の満了後に送信される宛先ワーク・バスケットの ID。SIMWM_NEXT を指定した場合、その作業パッケージはプロセスの次のステップに進んで引き続き処理されます。

szReadyWB

CHAR[*SIM_ITEM_ID_LENGTH*+1] — 入力

所定の索引クラスに属している項目のいずれか 1 つが、該当フォルダー項目に追加されることによって中断基準が満たされたときに、そのフォルダー内で中断されていた作業パッケージが送信される宛先ワーク・バスケットの ID。SIMWM_NEXT を指定した場合、その作業パッケージはプロセスの次のステップに進んで引き続き処理されます。

WMVARSTRUCT (作業パッケージ変数データ構造)

このデータ構造には、システム定義またはユーザー定義の作業パッケージ変数の ID および関連値が入っています。内容は以下のとおりです。

```
typedef struct _WMVARSTRUCT

{
ULONG                   ulStruct;
CHAR                    szVarName;
CHAR                    szVarValue;

} WMVARSTRUCT, *PWMVARSTRUCT;
```

フィールド

ulStruct — ULONG — 入力/出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

szVarName — CHAR [*SIMWM_VAR_NAME_LENGTH*+1] — 入力/出力

変数の名前。以下の定数は、システム変数の名前を表します。

SIMWM_ITEMID
経路指定される項目。

SIMWM_INDEX_CLASS
項目の索引クラス。

SIMWM_PRIORITY
作業パッケージの優先順位。

SIMWM_ACTION
ユーザーが選択するアクション。

szVarValue CHAR[*SIMWM_VAR_NAME_LENGTH*+1] — 入力/出力

変数の値を含むストリングへのポインター。

WORKBASKETINFOSTRUCT (ワーク・バスケット情報データ構造)

このデータ構造は、ワーク・バスケットの作成および修正に使用する情報を提供します。内容は以下のとおりです。

```
typedef  struct  _WORKBASKETINFOSTRUCT

{
ULONG           ulStruct;
CHAR            szWorkBasketName;
CHAR            chAccessListName;
USHORT          usWBLoadLimit;
BOOL            bRemoveAfterIndex;
BOOL            bSystemCntl;
CHAR            szUserFunName;
CHAR            szUserDLLName;
UCHAR           szWorkBasketPrivString;
ULONG           ulItemStatusFlag;
CHAR            szDefaultAction;
USHORT          usWorkbasketType;
CHAR            szEntryFunName;
CHAR            szEntryDLLName;
CHAR            szExitFunName;
CHAR            szExitDLLName;
CHAR            szUserDefWBExitFunName;
CHAR            szUserDefWBExitDLLName;

}  WORKBASKETINFOSTRUCT,  *PWORKBASKETINFOSTRUCT;
```

フィールド

ulStruct ULONG — 出力

このフィールドの長さも含め、構造の長さをバイト単位で表します。

szWorkBasketName CHAR[*OIM_WB_NAME_LENGTH*+1] — 出力

ワーク・バスケットの名前。

chAccessListName CHAR[*ACCESS_LIST_NAME_SIZE*+1] — 出力

ワーク・バスケットのアクセス・リスト名。

usWBLoadLimit USHORT — 出力

ワーク・バスケットのオーバーロードの制限値。

既にこの制限値に達しているワーク・バスケットに対して項目の追加を試みた場合、その項目は追加さ

れません。ただし、必要な場合には、項目の追加時にこの制限値を指定変更することにより、その項目を追加できます。

bRemoveAfterIndex

BOOL — 出力

索引付けの完了後に、該当の項目をワーク・バスケットから自動的に除去するかどうかを指示するフラグ。有効な値は、次のとおりです。

TRUE 索引付けが完了した後、項目をこのワーク・バスケットから除去します。

FALSE
索引付けが完了した後も、項目をこのワーク・バスケットから除去しません。

bSystemCntl

BOOL — 出力

ワーク・バスケット内の項目の優先度をシステムが制御するかどうかを示すフラグ。有効な値は、次のとおりです。

TRUE これがシステム割り当てワーク・バスケットであることを示します。システムは、ワーク・バスケット内の次の項目を、要求されたときにユーザーに提供します。作業パッケージの優先順位か日付、およびワーク・バスケットに関して定義されている順序 (LIFO、FIFO、または優先順位) が、その順序を決定します。

FALSE
これがシステム割り当てワーク・バスケットではないことを示します。ユーザーはワーク・バスケットの中から任意の項目を選択することができます。

szUserFunName

CHAR[*OIM_WB_FUNCTION_LENGTH*+1] — 出力

ワーク・バスケットの持つオーバーロード・トリガーが *usWBLoadLimit* フィールドの値として指定された限界を超えたときに呼び出すユーザー出口関数の名前。DLL と関数名は、アプリケーションによって使用されます。

szUserDLLName

CHAR[*OIM_WB_DLL_LENGTH*+1] — 出力

ユーザー出口関数が入っている DLL の名前。DLL と関数名は、アプリケーションによって使用されます。

szWorkBasketPrivString

UCHAR[*SIM_PRIVSTRING_LENGTH*+1] — 出力

ワーク・バスケットに関するユーザーの評価済み特権ストリング。

ulItemStatusFlag

ULONG - 出力

ワーク・バスケット状況フラグ。有効な値は、次のとおりです。

SIMWM_ACTIVE
ワーク・バスケットがアクティブであることを示します。

SIMWM_INACTIVE
ワーク・バスケットに削除のマークが付けられていることを示します。

szDefaultAction
CHAR(*SIMWM_ACTION_LENGTH*+1) — 出力

このワーク・バスケットに関連付けられているデフォルトのアクション・リスト。

usWorkbasketType
USHORT — 出力

ワーク・バスケット・タイプ。ユーザー定義のワーク・バスケットを示す、50 ～ 99 までの値。

szEntryFunName
CHAR[*OIM_WB_FUNCTION_LENGTH*+1] — 出力

ワーク・バスケット内のいずれかの項目が選択されオープンされた時に、アプリケーションが呼び出すユーザー出口関数。

szEntryDLLName
CHAR[*OIM_WB_DLL_LENGTH*+1] — 出力

エントリー・ユーザー出口関数が入っている DLL の名前。

szExitFunName
CHAR[*OIM_WB_FUNCTION_LENGTH*+1] — 出力

ユーザーがワーク・バスケット内のいずれかの項目の作業を完了した時に、アプリケーションが呼び出すユーザー出口関数。

szExitDLLName
CHAR[*OIM_WB_DLL_LENGTH*+1] — 出力

完了ユーザー出口関数が入っている DLL の名前。

szUserDefWBExitFunName
CHAR[*OIM_WB_FUNCTION_LENGTH*+1] — 出力

ワーク・バスケットがユーザー定義のワーク・バスケットである場合に、アプリケーションが呼び出すユーザー出口関数。

szUserDefWBExitDLLName
CHAR[*OIM_WB_DLL_LENGTH*+1] — 出力

ユーザー定義ワーク・バスケット関数が入っている DLL の名前。

第 5 章 OLE オートメーション・インターフェースの使用

Content Manager for iSeries クライアントとともに提供される API を使用することにより、Windows ベースの別のアプリケーションの Content Manager for iSeries へのログオン、文書およびフォルダーの検索の実行、検索結果、フォルダー、またはワーク・バスケットの目次 (TOC) リストの表示が可能になり、また、文書を表示して注釈を付けることもできます。これには、OLE 2.0 Automation に基づく API を使用します。

OLE オートメーションを使用したプログラミング

OLE オートメーションを使用すると、アプリケーションのコマンド操作をアプリケーション外から操作できるようになります。Windows クライアントは、Visual Basic (バージョン 3.0 以降)、Visual C++、PowerBuilder などのプログラミング環境で作成したプログラムから操作することのできる OLE オートメーション・オブジェクトを提供します。Windows クライアントのオブジェクトを操作するには、個々のオブジェクトのプロパティー とメソッド を知っている必要があります。

プロパティー

プロパティー は、Visual Basic の変数と似ていますが、Windows クライアントのオブジェクトの中にあるところが違います。変数を読み取ったり書き込んだりできるように、プロパティーも設定 (つまり書き込み) または取得 (つまり読み取り) することができます。すべてのプロパティーが読み取り/書き込みプロパティーなのではなく、読み取り専用のプロパティーもあれば、書き込み専用のプロパティーもあります。たとえば、*Application* オブジェクトの *Visible* プロパティーは読み取り/書き込みプロパティーで、プログラムが現在画面上で可視かどうかを確認するために使用されます。プロパティーの値が True に設定されているとき、プログラムは現在可視です。*Visible* プロパティーの値を False に設定すると、そのプログラムは隠れてしまいます。一方、*Item* オブジェクトの *Name* プロパティーは読み取り専用で、Content Manager for iSeries が項目を参照するときに使用する名前が含まれています。書き込み専用プロパティーの例は *Application* オブジェクトの *Password* プロパティーです。

メソッド

メソッド は、Visual Basic プロシージャーや関数プロシージャーに類似しています。メソッドを呼び出すことによって、Windows クライアントの内部で操作を実行する (つまり、コマンド命令を呼び出す) ことができます。たとえば、*Application* クラスの *OpenWorkbasket* メソッドは「ワーク・バスケットのオープン (Open Workbasket)」ダイアログをオープンします。

Windows クライアントのオブジェクト

Windows クライアント OLE オートメーション・オブジェクトは Microsoft® のガイドラインに沿って設計されています。したがって、Microsoft の指針に従っているすべてのアプリケーションと同様に、Windows クライアントには *Application* オブジェクト、 *Documents* コレクション・オブジェクト、および *Document* オブジェクトがあります。

Windows クライアントにはさらに、複数の *Item* オブジェクトを管理するための *Items* コレクション・オブジェクトと、文書、フォルダー、ワーク・バスケットといった Content Manager for iSeries の項目に対して情報を提供し、インターフェースをとる *Item* オブジェクトがあります。さらに、イメージ・ビューアーで現在オープンされている文書を保持する *Image* オブジェクトも提供されています。

Error という名前の情報専用のクラスは、どんなエラーが起きているかをアプリケーションが判別できるようにするために提供されています。

最後に、Windows クライアントは 2 つのヘルパー・オブジェクト (*EnumDocument* および *EnumItem*) もサポートしています。これらは、Visual Basic がオブジェクト反復を提供するために必要とするものですが、Visual Basic を使用したプログラミング時には作成されません。

コレクション・オブジェクトは、他のオブジェクトを保持するという意味では配列に似ています。*Documents* コレクションは *Document* オブジェクトを保持しますが、 *Items* コレクションは *Item* オブジェクトを保持します。すべての OLE オートメーション・コレクションは同じメソッドとプロパティーを共用します。

OLE オートメーションを使用したプログラミングと、Windows クライアントとともに提供されるオブジェクトの概要については、196 ページの『プログラミングのヒント』を参照してください。

Windows クライアント OLE オートメーション API は、Visual Basic だけでなく、OLE オートメーションをサポートするすべてのプログラム言語または第 4 世代言語 (4GL) で使用できます。

Application オブジェクト

Windows クライアントのメイン・オブジェクトは、*Application* オブジェクトです。プログラムは、*Application* オブジェクトへのアクセスを取得すると、Windows クライアントの他のすべてのオブジェクトを取得したり、作成したりできるようになります。

Application オブジェクトのメソッドとプロパティーは、 Windows クライアント全体に適用されます。たとえば、*Logon* メソッドは Content Manager for iSeries へのログオンを呼び出し、 *Quit* メソッドはプログラムの終了を呼び出します。したがって、Windows クライアントとのインターフェースをとるように設計されたプログラムは、まず、*Application* オブジェクトを作成する必要があります。

稼働中の Windows クライアントは、Content Manager for iSeries との対話に使用できます。TOC (OLE オートメーション内の *Document* オブジェクトと等価) をオープンして、項目 (*Item* オブジェクト) を検索または作成し、文書 (*Image* オブジェクト) を表示できます。

Documents コレクション

Documents コレクションは、TOC (フォルダー、検索結果、またはワーク・バスケット) を保持しているキューにたとえることができます。これらの TOC は *Document* オブジェクトによって表されます。

ほとんどの *Document* オブジェクトは、*Documents* メソッドの *OpenTOC* にパラメーターとして *Item* オブジェクトを指定して呼び出せばオープンできます。

Document オブジェクト

Documents コレクションの *OpenTOC* メソッドによって *Document* オブジェクトが作成されると、そのオブジェクトを表示し、いくつものメソッドを実行できるようになります。たとえば、ユーザーが *Document* TOC に現在選択している任意の項目を照会できます。

Error オブジェクト

エラーが発生すると、Content Manager for iSeries の戻りコードを含む、エラーに関する情報がすべてこのオブジェクトに格納されます。

Image オブジェクト

Image は特殊な文書を表します。これは、その時点で見ることのできる Content Manager for iSeries 文書です。*Image* オブジェクトは、その *OpenDocument* メソッドに *Item* オブジェクトをパラメーターとして指定して呼び出すことによりオープンされます。

Items コレクション

Items コレクション・オブジェクトは、関連する *Items* のリストです。たとえば、*Document* メソッド *Selections* は、現在選択されている項目すべてを含む *Items* コレクションを戻します。これには、コレクションから特定の *Item* オブジェクトを戻すメソッドがあり、 *Item* オブジェクトと *Items* コレクション・インスタンスを削除するハウスキーピング・メソッドもあります。

一度に複数の *Items* コレクションを定義することも可能です。ただし、*Items* コレクションを取得する方法にはメソッドから戻されたときにそれを取得するしかないため、*Items* コレクションを常に最新の状態に保つことが別途必要になります。

Item オブジェクト

Item オブジェクトは、文書、フォルダー、ワーク・バスケットなどの Content Manager for iSeries の項目を表します。*Item* オブジェクトを使用すると、項目を表示し (項目を他のオブジェクトのパラメーターとして渡すことにより)、項目の索引クラスとキー・フィールドを照会し、項目を再索引付けし、多数の他の処理を実行することができます。

Item オブジェクトには、それ自体を記述するプロパティーも含まれています。

プログラミングのヒント

OLE オートメーション API を使用して、Windows クライアントをユーザーのアプリケーションに組み込むことができます。この API を使用して Windows クライアントを組み込むには、ユーザーのアプリケーションの開発環境が OLE オートメーションのオブジェクトにアクセス可能でなければなりません。たとえば、Microsoft Visual Basic、Microsoft Visual C++、および PowerBuilder のような開発環境があります。Microsoft Excel および Microsoft Access のようなアプリケーションでの統合と似ています。

以下に、オブジェクトの解放とエラーの処理に関する情報を含む、 OLE オートメーションを使用したプログラミングのヒントを紹介します。

オブジェクトの解放

OLE オートメーションを使用したプログラミングではオブジェクトの解放に留意することが求められます。それは、オブジェクトを割り振るプログラムは使用後にそのオブジェクトを解放しなければならないからです。たとえば、Windows クライアントのオブジェクトを次のように Visual Basic で作成するとします。

```
Dim MyItem As Object
Set MyItem = MyApp.GetWorkbasket("To be indexed")
```

この操作で、Windows クライアントは *Item* オブジェクトを保持するためのストレージを割り振り、そのオブジェクトを指すポインターを戻します。ポインターは *MyItem* 変数に保管されます。

この *Item* オブジェクトを解放するには、次のステートメントを使用します。

```
Set MyItem = Nothing
```

この操作で、Windows クライアントは、*Item* オブジェクトに割り振ったストレージを解放します。オブジェクトを解放しないと、最終的に Windows クライアントのストレージが不足することになります。また、Windows クライアントは、オープンされたままのオブジェクトがある場合、実際には終了しません。

エラーの処理

Windows クライアントは、エラーを検出すると、例外を投げます。Visual Basic では、例外は *OnError* ステートメントでキャッチすることができます。例外が起きることを予期してエラーをキャッチするプログラムでは、メソッドを呼び出した後に戻りコードを検査する必要はありません。

Windows クライアントのエラーを処理するための実行可能な方法は、プログラム開始処理で *On Error Resume Next* ステートメントを実行し、メソッドからの戻り時に組み込み Visual Basic の *Err* 変数をテストすることです。*Err* が非ゼロの場合はエラーが発生しており、*Error* オブジェクトを調べて詳細情報を入手することができます (*Error* オブジェクトは、*Application* オブジェクトのプロパティーとして探すことができます)。*Error* オブジェクトには、実際のエラー・コードとエラー・メッセージ・ストリングが含まれます。

ほとんどのメソッドはエラー状況を戻します。この状況のタイプは VT_I4 であり、これは Visual Basic では Long データ型に変換されます。エラー状況は、ゼロ (正常) か非ゼロ (エラー検出) です。エラーが検出されると、*Error* オブジェクトが調べられて、問題に関する詳細を取得することができます。

プロパティーと引き数タイプ

引き数とプロパティーは、第 7 章にリストされています。これらのタイプを Visual Basic のタイプや Visual C++ のタイプへ変換するには、次の表を参照してください。

OLE タイプ	VisualBasic	C++	説明
VT_BSTR	String	Char 配列、ゼロ終了	ASCII ストリング。文字データのタイプを入れることができるが、通常はユーザー可読文字が入る。
VT_DISPATCH	Object	IDispatch*	OLE オブジェクトへの参照。メソッドまたはプロパティーを読み取って、戻すオブジェクト・タイプを判別する。
VT_VARIENT (安全配列)	Array (VB 4.0 以上のみ)	IVarient*	オブジェクトの安全配列。安全配列が使用されている領域では、オブジェクト・タイプは VT_BSTR です。
VT_I4	Number	Long	長整数。正と負の両方が可能。許容範囲は -2 147 483 648 ～ +2 147 483 647 です。
VT_EMPTY	(なし)	void	値はなし。
VT_UNKNOWN	(なし)	IVarient*	OLE オートメーションで内部的に使用されている構造。
VT_BOOL	Boolean	Int	2 つの可能な値 (TRUE または FALSE) を持つ論理値。

Visual Basic プログラムのサンプル

この節では、Windows クライアントを開始し、“To be indexed” ワーク・バスケットを表示する Visual Basic プログラムのコードを示します。その後、このプログラムはワーク・バスケット内の最初の項目が文書またはフォルダーであれば、それを表示します。プログラムを読みやすくするために、エラー処理は考慮に入れていません。プログラムを Visual Basic に入力し、繰り返し F8 キーを押してプログラムをトレースすると、このプログラムから最もよく学習できます。

```
' This example invokes the Windows クライアント and causes it to display the
' To be indexed workbasket, then displays the first item in the workbasket,
' whether it is a document or a folder.
' Data declarations
    Dim VicApp As Object
    Dim Workbasket As Object
    Dim Docs As Object
    Dim Doc As Object
    Dim Item As Object
' Get the application objects
    Set VicApp = CreateObject("Vic.Application")
```

```
' Set login information
    VicApp.User = "GLEND"
    VicApp.Password = "PASSWORD"
' Log into Content Manager for iSeries
    VicApp.Logon
' Get the workbasket item
    Set Workbasket = VicApp.GetWorkbasket("To be indexed")
' Display the workbasket
    Set Docs = VicApp.Documents
    Set Doc = Docs.OpenTOC(Workbasket)
' Get next item from workbasket
    Set Item = Workbasket.NextWorkbasketItem
' Find out if the item is a folder or a document
    If (Item.Type = 1) Then
        ' Document! Display it.
        VicApp.Image.OpenDocument Item
  Else
        ' Must be a folder. Display it.
        Docs.OpenTOC Item
     End If
' Clean up
    Set Workbasket = Nothing
    Set Docs = Nothing
    Set Doc = Nothing
    Set Item = Nothing
    VicApp.Quit
    Set VicApp = Nothing
```

この例では、Windows クライアントがロードされ、デフォルトの Windows クライアントのライブラリー・サーバーへのログオン時に使用されるユーザー名とパスワードが構成されます。次に、Windows クライアントのログオンが実行されます。

“To be indexed” ワーク・バスケット項目を入手した後に、ワーク・バスケットは *Documents* オブジェクトを使用してオープンされます。

次のステップは、ワーク・バスケット内の次の項目を入手して、それが文書かフォルダーかを判別します。項目がフォルダーである場合、それは *Documents* オブジェクトに渡されますが、文書であれば *Image* オブジェクトに渡されます。

最後に、Windows クライアントが終了します。

Windows 用 OLE オブジェクトのプロパティーおよびメソッド

この章では、すべての Windows クライアント・アプリケーション・オブジェクトに関連するプロパティーおよびメソッドについて説明します。

Application オブジェクト

Application オブジェクトは、アプリケーション・レベルの状態 (たとえば、ログオンや終了など) を取得し設定します。

プロパティー

Application オブジェクトには、以下のプロパティーがあります。

Application

Application プロパティーは、Application オブジェクトを戻します。

データ型: VT_DISPATCH (Application)

Document

Document プロパティーは、Document オブジェクトのコレクションを保持します。Windows クライアントの「文書」は、「目次 (Table of Contents)」ビューと同じ意味です。

データ型: VT_DISPATCH (Documents)

Error 最新のメソッド・エラーに関するエラー情報。

データ型: VT_DISPATCH (Error)

HWnd

このプロパティーは、クライアントのメイン・ウィンドウ・ハンドルを戻します。これは読み取り専用プロパティーです。

データ型: VT_14

Image Image プロパティーは、イメージ・ビューアーで現在見ることのできる IBM Content Manager for iSeries 文書を保持します。見ることのできる文書が存在しない場合には、Image プロパティーは NULL を戻します。

データ型: VT_DISPATCH (Image)

KeyFieldTranslation

KeyFieldTranslation プロパティーは、KeyFieldTranslation プロパティーの値に応じて、検索または設定された値を変換するように、または変換しないように Item.KeyFields プロパティーを設定します。

データ型: VT_BOOL

NewPassword

NewPassword プロパティーは、ユーザーのパスワードを変更するために使用します。このプロパティーは、Logon メソッドを呼び出す前に設定する必要があります。ユーザーがログオンに成功すると、ユーザーのパスワードが変更されます。デフォルト値は NULL です。

データ型: VT_BSTR

Password

Password プロパティーは、IBM Content Manager for iSeries ライブラリー・サーバーにログオンするために Logon メソッドを呼び出すときに使用されるパスワードです。使用可能な値および結果については、 Application オブジェクトの Logon メソッドの説明を参照してください。

データ型: VT_BSTR

Server

Server プロパティーには、Logon メソッドを呼び出すときにログオンの対象となるライブラリー・サーバーの名前が入っています。使用可能な値および結果については、 Application オブジェクトの Logon メソッドの説明を参照してください。

データ型: VT_BSTR

User User プロパティーは、Application オブジェクトの Logon メソッドを呼び出して、使用可能な値および結果を入手します。

データ型: VT_BSTR

Visible

Visible プロパティーには、Windows Client フレーム・ウィンドウの可視状況が入っています。デフォルト値は False (0) です。

データ型: VT_BOOL

メソッド

Application オブジェクトは、以下のメソッドをサポートしています。

Activate

このメソッドは、クライアントを強制的にフォアグラウンド (前景) に表示します。

パラメーター: なし

戻り値: なし

ClassArray

ClassArray メソッドは、Logon メソッドの実行時に定義されたすべての索引クラスの名前が入っている VT_BSTR の安全配列を戻します。

パラメーター: なし

戻り値: VT_VARIANT (VT_BSTR の安全配列)

ClassKeyFieldArray

ClassKeyFieldArray メソッドは、Logon メソッドの実行時に指定の索引クラスと関連づけられたすべてのキー・フィールドの名前が入っている VT_BSTR の安全配列を戻します。

パラメーター: VT_BSTR としての IndexClass

戻り値: VT_VARIANT (VT_BSTR の安全配列)

ClassKeyFieldList

ClassKeyFieldList メソッドは、Logon メソッドの実行時に指定の索引クラスと関連づけられたすべてのキー・フィールドを持つストリングを戻します。これらのキー・フィールドは、ストリング区切り引き数で区切られます。

パラメーター: VT_BSTR としての IndexClass、VT_BSTR としての Separator

戻り値: VT_BSTR

ClassList

ClassList メソッドは、Logon メソッドの実行時に定義されたすべての索引クラスのリストを持つストリングを戻します。これらの索引クラスは、ストリング区切り引き数で区切られます。

パラメーター: VT_BSTR としての Separator

戻り値: VT_BSTR

ContentClassArray

ContentClassArray メソッドは、Logon メソッドの実行時に定義されたすべてのコンテンツ・クラスの名前が入っている VT_BSTR の安全配列を戻します。

パラメーター: なし

戻り値: VT_VARIANT (VT_BSTR の安全配列)

ContentClassList

ContentClassList メソッドは、Logon メソッドの実行時に定義されたすべてのコンテンツ・クラスを持つストリングを戻します。これらのコンテンツ・クラスは、区切り引き数で区切られます。

パラメーター: VT_BSTR としての Separator

戻り値: VT_BSTR

CreateDocument

CreateDocument メソッドは、新たに作成された文書を表す *Item* オブジェクトを戻します。これはオブジェクト (ページ) を含んでおらず、NOINDEX 索引クラスを用いて索引付けされます。ソース・キー・フィールドには Source 引き数の値が入り、名前キー・フィールドには User プロパティーの内容が入り、さらにタイム・スタンプ・キー・フィールドには文書が作成された正確な日時が入ります。

パラメーター: VT_BSTR としての Source

戻り値: VT_DISPATCH (Item)

CreateFolder

CreateFolder メソッドは、新たに作成されたフォルダーを表す Item オブジェクトを戻します。これはその TOC 内に項目を含んでおらず、*NOINDEX* 索引クラスを用いて索引付けされます。ソース・キー・フィールドには Source 引き数の値が入り、ユーザー *ID* キー・フィールドには User プロパティーの内容が入り、さらにタイム・スタンプ・キー・フィールドには文書が作成された正確な日時が入ります。

パラメーター: VT_BSTR としての Source

戻り値: VT_DISPATCH (Item)

DisableMenus

この DisableMenus メソッドでは、メニュー・クラスを使用不可にすることができます。使用不可にするメニューを指定するには、*Flags* 引き数を使用します。このメソッドの有効な値を以下に示します。

- IP2_DISABLE_CHECKINOUT (0x001)
  項目のチェックインまたはチェックアウトをできないようにします。
- IP2_DISABLE_DELETE (0x002)
  項目の削除をできないようにします。
- IP2_DISABLE_EXPORT (0x004)
  項目のエクスポートをできないようにします。
- IP2_DISABLE_FAXOUT (0x008)
  項目のファックス送信をできないようにします。
- IP2_DISABLE_FOLDER_FUNCTIONS (0x0010)
  既存のフォルダーへの項目の追加、新規フォルダーへの項目の追加、またはフォルダーからの項目の削除をできないようにします。
- IP2_DISABLE_INDEX_CLASS_CHANGE (0x0020)

異なる索引クラスへの変更をできないようにします。ただし、索引クラスのキー・フィールドは編集できます。

- IP2_DISABLE_INDEX_VALUE_CHANGE (0x0040)

  異なる索引クラスへの変更や、索引クラスからのキー・フィールドの編集をできないようにします。ユーザーは、メニューをブラウズして、ウィンドウにリストされた値をコピーすることができます。この値を指定すると、*IP2_DISABLE_INDEX_CLASS_CHANGE* フラグは無視されます。

- IP2_DISABLE_NOTE_APPEND (0x0100)

  前に保管した注を編集したり、新しい注を追加したりできないようにします。ブラウズ・モードでは、既存の注をオープンしてコピーすることができます。注が存在しない場合、「注釈ログ (Note Log)」ウィンドウは表示されません。この値を指定すると、*IP2_DISABLE_NOTE_EDIT* フラグは無視されます。

- IP2_DISABLE_NOTE_EDIT (0x0080)

  前に保管した注を編集できないようにします。ただし、新しい注を追加することはできます。

- IP2_DISABLE_OPTIONS (0x8000)

  「オプション (Options)」→「設定 (Preferences)」または**「オプション (Options)」→「レイアウト (Layout)」**メニュー・オプションを使用できないようにします。

- IP2_DISABLE_PRINT (0x0200)

  項目の印刷をできないようにします。

- IP2_DISABLE_SEARCH (0x4000)

  検索をできないようにします。

- IP2_DISABLE_WORKBASKET_ACTIVATE (0x0400)

  中断状態から項目の除去をできないようにします。

- IP2_DISABLE_WORKBASKET_REMOVAL (0x0800)

  ワーク・バスケットからの項目の除去や、ワーク・バスケット間の項目の経路指定をできないようにします。

- IP2_DISABLE_WORKBASKET_SUSPEND (0x1000)

  項目の中断をできないようにします。

- IP2_DISABLE_WORKFLOW (0x2000)

  ワークフロー内の項目の開始、項目のワークフローの変更、項目のワークフローの完了、または、ワークフローからの項目の除去をできないようにします。

これらの値を組み合わせると、一度に複数のクラスを使用不可にすることができます。 *Flags* 引き数に 0 を指定して DisableMenus メソッドを呼び出すと、メニューがすべて使用可能になります。

オプショナル引き数の *Hide* を使用すれば、メニュー・オプションを使用不可にするのではなく、削除することができます。ただし、メニュー項目を削除すると、低い制限値を設定しても、元に戻すことはできません。

パラメーター: VT_I4 としての Flags、VT_VARIANT としての Hide (オプショナル、通常は VT_BOOL)

戻り値: VT_NONE

ExtendedPrintSetup

ExtendedPrintSetup メソッドを使用すると、外部アプリケーションにより、クライアントのデフォルト印刷動作を変更することができます。記述されるオプションは、標準のユーザー・インターフェースからは設定できない拡張印刷機能です。

パラメーター: VT_BOOL としての Comments、VT_BOOL としての Borders、VT_BOOL としての SinglePage、 VT_BOOL としての HorizPages、VT_BOOL としての PageNumbers、VT_I2 としての NumRows、 VT_I2 としての NumColumns

戻り値: VT_I4

- *Comments* は、注釈を印刷しないための、もう一つの方法です。このオプションは、Item.PrintItem メソッド内の *PrintMarkup* 引き数を複製します。
- *Borders* は、イメージ周辺の単一ピクセル線を使用可能または使用不可にします。この機能は、*SinglePage* を false に設定している場合に最も役立ちます (次の項目を参照)。
- *SinglePage* に *NumRows*、*NumColumns*、および *HorizPages* の各引き数を指定すると、ページ上でのイメージの配置を決めることができます。*SinglePage* が true の場合は、各ページに 1 つのイメージしか印刷されません。*SinglePage* が false の場合は、他の 3 つの引き数によって、各ページにイメージをいくつ印刷するのかを決めることができます。
- *HorizPages* は、*SinglePage* 引き数が false の場合に使用されます。*HorizPages* は、印刷ページ上でのイメージの向きを指定します。水平の場合は true、垂直の場合は false を指定します。
- *PageNumbers* は、各イメージ上にページ番号を印刷します。PageNumbers を true に設定すると、ページ番号が各イメージの左上隅に印刷されます (1 ページに複数のイメージが表示されることもあります)。
- *NumRows* および *NumColumns* は、 *SinglePage* 引き数が false の場合に使用されます。 *NumRows* および *NumColumns* により、単一の印刷ページ上で水平方向と垂直方向にいくつのイメージを表示するのかを決めることができます。

GetWorkbasket

GetWorkbasket メソッドは、Name 引き数に指定されているワーク・バスケットに関連づけられている Item オブジェクトを戻します。このワーク・バスケット名には大文字小文字の区別がないことに注意してください。

パラメーター: VT_BSTR としての Name

戻り値: VT_DISPATCH (Item)

ItemID

ItemID メソッドは、項目 ID が指定された Item オブジェクトを戻します。ItemID プロパティーについては、 Item オブジェクトのプロパティーの説明を参照してください。

パラメーター: VT_BSTR としての *Item*

戻り値: VT_DISPATCH (Item)

KeyFieldArray

KeyFieldArray メソッドは、Logon メソッドの実行時に定義されたすべての索引クラスの名前が入っている VT_BSTR の安全配列を戻します。

パラメーター: なし

戻り値: VT_VARIANT (VT_BSTR の安全配列)

KeyFieldList

KeyFieldList メソッドは、Logon メソッドの実行時に定義されたすべてのキー・フィールドを持つストリングを戻します。これらのキー・フィールドは、ストリング区切り引き数で区切られます。

パラメーター: VT_BSTR としての Separator

戻り値: VT_BSTR

Logon

Logon メソッドは IBM Content Manager for iSeries にログオンします。User、Password および Server の各プロパティーがすべて設定されている場合には、ログオンがその情報を用いて試みられます。上記のプロパティーのいずれかが設定されなかった場合、または 1 回目のログオン試行が正常に行われなかった場合には、残りの情報をオペレーターが入力するためのログオン画面が表示されます。Logon メソッドを呼び出す前に Password プロパティーは設定されているが、*User* プロパティーが設定されなかった場合には、パスワード情報は無視されます。

Server プロパティーは、最後にログオンされたライブラリー・サーバーを用いて、または正常なログオンが行われなかった場合は「LIBSRVR2」を用いて事前初期設定されます。

戻り値は、正常なログオンが行われた場合は 0 であり、エラーが発生した場合はゼロ以外の値です。

パラメーター: なし

戻り値: VT_I4

OpenBasicSearch

OpenBasicSearch メソッドは、基本検索ダイアログ・ボックスを表示し、オペレーターが検索入力できるようにします。結果として生ずる Document オブジェクトは戻されないことに注意してください。

パラメーター: なし

戻り値: なし

OpenScan

OpenScan メソッドは、「スキャン (scan)」ダイアログ・ボックスを表示し、オペレーターがスキャン・セッションをオープンできるようにします。結果として生ずる Document オブジェクトは戻されないことに注意してください。

パラメーター: なし

戻り値: なし

OpenWorkbasket

OpenWorkbasket メソッドは、ワーク・バスケット選択ダイアログ・ボックスを表示し、ユーザーがワーク・バスケットを選択しオープンできるようにします。結果として生ずる Document オブジェクトは戻されないことに注意してください。

パラメーター: なし

戻り値: なし

PrintSetup

PrintSetup メソッドを使用すると、外部アプリケーションにより、クライアントのデフォルト印刷動作を変更することができます。このメソッドで指定した値は、 OLE オートメーション印刷やユーザーによって開始される印刷のデフォルト印刷設定として保管されます。

パラメーター: VT_BSTR としての Printer、VT_I2 としての PaperSize、VT_BOOL としての Portrait、 VT_I2 としての Copies、VT_VARIANT としての Scaling (オプショナル、通常は VT_BOOL)

戻り値: VT_I4

- *Printer* は、出力先のプリンター名を指定します。
- *PaperSize* は、用紙の種類を定義します。希望する用紙の種類を指定するには、以下のリストの対応する番号 (1 ～ 41) を割り当てます。
  1. レター 8 1/2 x 11 in
  2. レター・スモール 8 1/2 x 11 in
  3. タブロイド 11 x 17 in
  4. レジャー 17 x 11 in
  5. リーガル 8 1/2 x 14 in
  6. ステートメント 5 1/2 x 8 1/2 in
  7. エグゼクティブ 7 1/4 x 10 1/2 in
  8. A3 297 x 420 mm
  9. A4 210 x 297 mm
  10. A4 スモール 210 x 297 mm
  11. A5 148 x 210 mm
  12. B4 (JIS) 250 x 354
  13. B5 (JIS) 182 x 257 mm
  14. フォリオ 8 1/2 x 13 in
  15. クォート 215 x 275 mm
  16. 10x14 in
  17. 11x17 in
  18. ノート 8 1/2 x 11 in
  19. エンベロープ #9 3 7/8 x 8 7/8
  20. エンベロープ #10 4 1/8 x 9 1/2
  21. エンベロープ #11 4 1/2 x 10 3/8

22. エンベロープ #12 4 \276 x 11
23. エンベロープ #14 5 x 11 1/2
24. C サイズ・シート
25. D サイズ・シート
26. E サイズ・シート
27. エンベロープ DL 110 x 220mm
28. エンベロープ C5 162 x 229 mm
29. エンベロープ C3 324 x 458 mm
30. エンベロープ C4 229 x 324 mm
31. エンベロープ C6 114 x 162 mm
32. エンベロープ C65 114 x 229 mm
33. エンベロープ B4 250 x 353 mm
34. エンベロープ B5 176 x 250 mm
35. エンベロープ B6 176 x 125 mm
36. エンベロープ 110 x 230 mm
37. エンベロープ・モナーク 3.875 x 7.5 in
38. 6 3/4 エンベロープ 3 5/8 x 6 1/2 in
39. 米国標準ファンフォールド 14 7/8 x 11 in
40. ドイツ標準ファンフォールド 8 1/2 x 12 in
41. ドイツ・リーガル・ファンフォールド 8 1/2 x 13 in

- *Portrait* は、印刷の方向 (true = 縦、false = 横) を定義します。
- *Copies* は、印刷部数を指定します。
- *Scaling* は、「ページ合わせ」サイズ、または「標準」サイズのどちらで印刷を行なうかを指定します。 Scaling を `True` (0 以外) に設定するか、または省略すると、印刷は「ページ合わせ」サイズで行なわれます。 Scaling を `False` に設定すると、印刷は「標準」サイズで行なわれます。

QueryPrivilege

QueryPrivilege メソッドにより、外部アプリケーションは、現在ログオンしているユーザーの実際の特権を判別することができます。外部アプリケーションは、一般特権または特定の特権 (索引クラスまたはワーク・バスケット用の特権など) を確認することができます。

パラメーター: VT_I4 としての Authority、VT_VARIANT としての Context (オプショナル、VT_BOOL(Item) または VT_BSTR)

戻り値: VT_BOOL

- *Authority* は、検査する特権を定義します。この値には、フォルダー管理機能 **Ip2QueryPrivBuffer** によってサポートされる OIM_ 値を設定することができます。
- *Context* は、各種コンテキストの評価済みの特権を判別します。 *Context* の値を入力しないと、ユーザーの一般特権が戻されます。 *Context* には、以下の値を設定することもできます。
  - Item オブジェクトへのディスパッチ (フォルダー、ドキュメント、またはワーク・バスケットなど)

– 索引クラス名 (VT_BSTR)
– ワークフロー名 (VT_BSTR)

Quit Quit メソッドは、Windows クライアントのアプリケーションを終了させます。すべてのオープン文書 (TOC)、すべてのイメージ・ビューアー・セッション、およびすべての未処理の Item および Items オブジェクトがクローズされます。

パラメーター: なし

戻り値: なし

Search

Search メソッドは、オプショナルの索引クラスおよびキー・フィールドのワイルドカード検索ストリングを用いてシステム・ファイル・ルーム上で行われた検索の結果を表す Item を戻します。検索結果フォルダーは、索引クラスが変更されない限り、クローズ時に自動的に削除されます。検索ストリングの形式は、 166 ページの『LIBSEARCHCRITERIASTRUCT (検索基準情報構造)』で定義されています。

TypeFilter=1 の場合は、フォルダーのみが戻されます。

TypeFilter=2 の場合は、文書のみが戻されます。

TypeFilter の値が上記以外の値の場合は、文書とフォルダーの両方が戻されます。

WipFilter=1 の場合は、ワークフローの中にない項目が戻されます。

WipFilter=2 の場合は、現在ワークフローの中にある項目が戻されます。

WipFilter=4 の場合は、ワークフローから取り消された項目が戻されます。

WipFilter=8 の場合は、ワークフローを完了した項目が戻されます。

WipFilter 値は、バイナリー OR オペレーターと結合されることがあります。

SuspendFilter=1 の場合は、活動状態の項目が戻されます。活動状態または中断中の項目。

SuspendFilter=2 の場合は、中断中の項目が戻されます。

SuspendFilter がその他の値の場合は、中断中の項目または活動状態の項目が戻されます。

パラメーター:

VT_BSTR としての IndexClass (オプショナル)
VT_BSTR としての SearchString (オプショナル)
VT_VARIANT としての TypeFilter (オプショナル、通常は VT_I2)
VT_VARIANT としての WipFilter (オプショナル、通常は VT_I2)
VT_VARIANT としての SuspendFilter (オプショナル、通常は VT_I2)

戻り値: VT_DISPATCH (Item)

SetPrintRect

SetPrintRect メソッドでは、ページに印刷されるイメージが入る長方形を定

義することができます。このメソッドで指定した値は、 OLE オートメーション印刷やユーザーによって開始される印刷のデフォルト印刷設定として保管されます。

パラメーター: VT_I2 としての RectLeft、VT_I2 としての RectTop、VT_I2 としての RectRight、VT_I2 としての RectBottom

戻り値: なし

この 4 つの引き数により、用紙の左上隅からボックスの各辺までの距離 (mm) が定義されます。印刷する長方形を「なし」にリセットするには、すべての引き数を 0 に設定して SetPrintRect メソッドを再度呼び出します。

重要 : 用紙に収まらない長方形を指定すると、イメージの一部またはすべてが用紙に印刷されません。

WorkbasketArray

WorkbasketArray メソッドは、Logon メソッドの実行時に定義されたすべてのワーク・バスケットの名前が入っている VT_BSTR の安全配列を戻します。

パラメーター: なし

戻り値: VT_VARIANT

WorkbasketList

WorkbasketList メソッドは、Logon メソッドの実行時に定義されたすべてのワーク・バスケットのリストを持つストリングを戻します。これらのワーク・バスケットは、ストリング区切り引き数で区切られます。

パラメーター: VT_BSTR としての Separator

戻り値: VT_BSTR

Document オブジェクト

Document オブジェクトは、目次 (TOC) に関する情報を保持しています。

プロパティー

Application

Application プロパティーは、Application オブジェクトを戻します。

データ型: VT_DISPATCH (Application)

Count Count プロパティーは、TOC にリストされている項目の数を戻します。

データ型: VT_14

Item Item プロパティーは、この Document (TOC) に関連づけられている Item オブジェクトを戻します。

データ型: VT_DISPATCH (Item)

Page Page プロパティーには、選択されたページ番号が入っています。このプロパティーは、文書についてのみ有効であり、ワーク・バスケットまたはフォルダーについては無効です。デフォルト値は 0 です。

データ型: VT_I4

PageCount

PageCount プロパティーには、文書内のページの数が入っています。このプロパティーは、文書についてのみ有効であり、ワーク・バスケットまたはフォルダーについては無効です。デフォルト値は 0 です。これは読み取り専用プロパティーです。

データ型: VT_I4

Parent

Parent プロパティーは、Document オブジェクトの親 (これは Documents コレクション・オブジェクトです) を戻します。

データ型: VT_DISPATCH (Documents)

SelectedCount

SelectedCount プロパティーは、TOC 内で選択される項目の数を戻します。

データ型: VT_14

Type Type プロパティーは、文書内でオープンされている項目のタイプ (フォルダー、ワーク・バスケット、または文書) を戻します。実際の値は、以下のとおりです。

1 - 文書

2 - フォルダー

3 - ワーク・バスケット

1024 - スキャン (基本スキャン・ビューアー。これ以外のプロパティーまたはメソッドはこのタイプでは作動しません)

デフォルトは 0 (エラー) です。これは読み取り専用プロパティーです。

データ型: VT_I4

メソッド

Document オブジェクトは、以下のメソッドをサポートしています。

Activate

Activate メソッドは、この文書と関連づけられている TOC ウィンドウをフォアグラウンド (前景) に引き出します。

パラメーター: なし

戻り値: VT_I4

CaretIndex

CaretIndex メソッドは、脱字記号 (^) 項目 (グリッド内に点線による長方形を含む項目) の索引をフォルダーまたはワーク・バスケット内に戻します。

パラメーター: なし

戻り値: VT_I4

ClearSelect

ClearSelect メソッドは、TOC 内の現行の選択をすべて消去します。

パラメーター: なし

戻り値: VT_I4

Close Close メソッドは、関連文書 (TOC) に関連づけられているウィンドウをクローズし、*Documents* コレクションからその文書を除去します。コレクションの残りの Document オブジェクトは、コレクション内のギャップを防ぐためにシフトダウンされます。

パラメーター: VT_VARIANT (オプショナル、通常は VT_BOOL)

戻り値: VT_I4

CloseIt

CloseIt メソッドは Close メソッドと同じ働きをします。このメソッドは、VisualBasic をサポートするためだけにインプリメントされています。これは、 VisualBasic が Close を予約語として使用しているためです。CloseIt メソッドは、関連文書 (TOC) に関連づけられているウィンドウをクローズし、Documents コレクションからその文書を除去します。コレクションの残りの Document オブジェクトは、コレクション内のギャップを防ぐためにシフトダウンされます。

パラメーター: VT_VARIANT (オプショナル、通常は VT_BOOL)

戻り値: VT_I4

DisplayPage

DisplayPage メソッドは、文書内の指定されたページを強制的に表示します。このメソッドは、文書についてのみ有効であり、ワーク・バスケットまたはフォルダーについては無効です。

パラメーター: VT_I4 としての Page

戻り値: VT_I4

FirstPage

FirstPage メソッドは、文書の先頭ページを表示します。このメソッドは、文書についてのみ有効であり、ワーク・バスケットまたはフォルダーについては無効です。

パラメーター: なし

戻り値: VT_I4

IndexedItem

IndexedItem メソッドは、フォルダーまたはワーク・バスケットからその索引 (Index 引き数を用いて指定されたもの) に基づいて Document オブジェクトからの 1 つの項目を戻します。

パラメーター: VT_I4 としての Index

戻り値: VT_DISPATCH (Item)

LastPage

文書の最後のページを表示します。このメソッドは、文書についてのみ有効であり、ワーク・バスケットまたはフォルダーについては無効です。

パラメーター: なし

戻り値: VT_I4

Maximize

Maximize メソッドは、メイン・クライアント・ウィンドウの Document オブジェクトを最大化し、他のすべての Document オブジェクトを隠します。

パラメーター: なし

戻り値: VT_I4

Minimize

Minimize メソッドは、メイン・クライアント・ウィンドウの Document オブジェクトを最小化します。

パラメーター: なし

戻り値: VT_I4

NextPage

NextPage メソッドは、文書内の次のページ (現行ページの 1 つ後のページ) を表示します。このメソッドは、文書についてのみ有効であり、ワーク・バスケットまたはフォルダーについては無効です。

パラメーター: なし

戻り値: VT_I4

PreviousPage

PreviousPage メソッドは、文書内の前のページ (現行ページの 1 つ前のページ) を表示します。このメソッドは、文書についてのみ有効であり、ワーク・バスケットまたはフォルダーについては無効です。

パラメーター: なし

戻り値: VT_I4

Restore

Restore メソッドは、メイン・クライアント・ウィンドウの Document オブジェクトをその元の状態 (最小化されたものでも最大化されたものでもないもの) に復元します。

パラメーター: なし

戻り値: VT_I4

Selections

Selections メソッドは、Document (TOC) で選択されたすべての *Item* オブジェクトが入っている Items コレクションを戻します。

パラメーター: なし

戻り値: VT_DISPATCH (Items)

SelectRange

SelectRange メソッドは、TOC 内の項目の範囲を選択します。引き数は、選択される最初の項目と最後の項目のゼロ・ベースの索引です。

パラメーター:

VT_I4 としての First

VT_I4 としての Last

戻り値: VT_I4

Zoom Zoom メソッドは、Document オブジェクトのズーム比率を変更します。たとえば、ズーム比率を 100 に設定した場合には、イメージはピクセルごとにフルサイズで表示されます。ズーム比率を 50 に設定した場合には、イメ

ージはハーフサイズで表示されます。Zoom メソッドは、文書に関してのみ有効であり、フォルダーまたはワーク・バスケットに関しては無効です。

パラメーター: VT_I4 としての Percent

戻り値: VT_I4

ZoomFit

ZoomFit メソッドを使用して、文書イメージを表示用長方形に適合させることができます。Type 引き数は、適合方法を指定するためのものです。1 は適合の高さを意味し、0 は適合の幅を意味します。ZoomFit メソッドは、文書に関してのみ有効であり、フォルダーまたはワーク・バスケットに関しては無効です。

パラメーター: VT_14 としての Fit

戻り値: VT_I4

ZoomRect

ZoomRect メソッドを使用して、Document オブジェクトでズームの対象となる長方形を指定することができます。Left (左)、Top (上)、Right (右)、Bottom (下) の各引き数を使用して、表示用長方形 (ビューアー・ウィンドウ) でできるだけ大きく表示するための長方形の境界を指定します。これらの引き数はピクセルで指定されます。ZoomRect メソッドは、文書に関してのみ有効であり、フォルダーまたはワーク・バスケットに関しては無効です。

パラメーター:

VT_I4 としての Left

VT_I4 としての Top

VT_I4 としての Right

VT_I4 としての Bottom

戻り値: VT_I4

Documents オブジェクト

Documents コレクション・オブジェクトは、すべてのオープン Document オブジェクト (TOC) の集合体です。

プロパティー

Documents オブジェクトには、以下のプロパティーがあります。

Active Active プロパティーは、現在処理対象になっている Document オブジェクトの索引を保持します。これは読み取り専用プロパティーです。

データ型: VT_I4

Application

Application プロパティーは、Application オブジェクトを戻します。

データ型: VT_DIPSATCH (Application)

Count Count プロパティーは、現在コレクションに入っている Document オブジェクトの数を保持します。

データ型: VT_14

Parent

Parent プロパティーは、Documents コレクション・オブジェクトの親 (これは Application オブジェクトです) を戻します。

データ型: VT_DISPATCH (Application)

メソッド

Document オブジェクトは、以下のメソッドをサポートしています。

Cascade

Cascade メソッドは、最小化されていないすべてのオープン Document オブジェクトをカスケード表示で整列させます。

パラメーター: なし

戻り値: VT_I4

Close Close メソッドは、Document オブジェクトに関連づけられているすべてのウィンドウをクローズし、 Document コレクションから Document オブジェクトを除去します。

パラメーター: なし

戻り値: なし

CloseIt

重要 : CloseIt メソッドは Close メソッドと同じ働きをします。このメソッドは、VisualBasic をサポートするためだけにインプリメントされています。これは、 VisualBasic が Close を予約語として使用しているためです。Close メソッドは、Document オブジェクトに関連づけられているすべてのウィンドウをクローズし、 Document コレクションから Document オブジェクトを除去します。

パラメーター: なし

戻り値: VT_I4

Item Item メソッドは、コレクションの中に入っている複数の Document オブジェクトのうちの 1 つを戻します。

パラメーター: VT_I4 としての Index

戻り値: VT_DISPATCH (Document)

OpenDocument

OpenDocument メソッドは、文書用に新しい Document オブジェクトを作成し、それを Documents コレクションに追加します。Browse 引き数が TRUE に設定されている場合には、文書はロックされずにオープンされ、他のユーザーがその文書をオープンできます。

パラメーター:

VT_DISPATCH (Item) としての Index

VT_VARIANT としての Browse (オプショナル、通常は VT_BOOL)

戻り値: VT_DISPATCH (Document)

OpenTOC

OpenTOC メソッドは、指定されたワーク・バスケットまたはフォルダー用

に、新しい Document オブジェクトを作成し、それを Documents コレクションに追加します。Browse 引き数が TRUE に設定されている場合には、フォルダーはロックされずにオープンされ、他のユーザーがその文書をオープンできます。Browse がワーク・バスケットに影響を与えることはありません。

パラメーター:

VT_DISPATCH (Item) としての Index

VT_VARIANT としての Browse (オプショナル、通常は VT_BOOL)

戻り値: VT_DISPATCH (Document)

Tile Tile メソッドは、最小化されていないすべてのオープン Document オブジェクトをタイル表示で整列させます。Vertical 引き数は、各オブジェクトを主として縦方向 (非ゼロ) か横方向 (ゼロ) のいずれに設定する必要があるかを指定するためのものです。

パラメーター: VT_I4 としての Vertical

戻り値: VT_I4

Error オブジェクト

Error オブジェクトは、Windows クライアントでのメソッドの実行中に発生した可能性のあるエラーに関する詳細を記述します。

プロパティー

Error オブジェクトには、以下のプロパティーがあります。

ErrorMessage

ErrorMessage プロパティーには、どの処理が失敗したかおよびその時点で Windows クライアントが何を行っていたかを記述する記述エラー・コードが入っています。

データ型: VT_BSTR

ExtReturnCode

ExtReturnCode プロパティーには、エラーが検出されたときに戻された拡張戻りコードが入っています。

データ型: VT_14

ReturnCode

ReturnCode プロパティーには、エラーが検出されたときに戻されたエラー・コードが入っています。OLE Automotation メソッドが戻すエラー・コードは、「メッセージとコード」に記載されている一貫した 4 桁のコード、または、表 3 に示すような、`frnwole.h` ヘッダー・ファイルに記述されている値で、いずれも標準化されたエラー・コードです。

データ型: VT_14

表 3. 標準化された OLE API 戻りコード

OLEAPI_RC_NOT_LOGGED_ON	12000
OLEAPI_RC_INVALID_INDEXCLASS	12001
OLEAPI_RC_INSUFFICIENT_MEMORY	12002

表 3. 標準化された OLE API 戻りコード (続き)

OLEAPI_RC_NO_ITEMS_FOUND	12003
OLEAPI_RC_INVALID_WORKBASKET	12004
OLEAPI_RC_ALREADY_LOGGED_ON	12005
OLEAPI_RC_INVALID_ARGUMENT	12006
OLEAPI_RC_NO_DOC_OPEN	12007
OLEAPI_RC_INVALID_ITEM	12008
OLEAPI_RC_INDEX_OUT_OF_RANGE	12009
OLEAPI_RC_INVALID_KEYFIELD	12010
OLEAPI_RC_ERROR_PRINTING	12011
OLEAPI_RC_INVALID_CONTENT_CLASS	12012
OLEAPI_RC_ITEM_NOT_FOLDER	12013
OLEAPI_RC_ITEM_NOT_WORKBASKET	12014
OLEAPI_RC_ITEM_NOT_WORKFLOW	12015
OLEAPI_RC_ERROR_GETTING_PART	12016
OLEAPI_RC_ERROR_UNLOCKING	12017
OLEAPI_RC_INVALID_DOCUMENT	12018
OLEAPI_RC_NOT_TOC_DOCUMENT	12019
OLEAPI_RC_INSUFFICIENT_PRIVS	12020
OLEAPI_RC_NO_SELECTIONS	12021
OLEAPI_RC_NOT_DOC_DOCUMENT	12022
OLEAPI_RC_ITEM_NOT_TOC	12023
OLEAPI_RC_ITEM_NOT_DOCUMENT	12024
OLEAPI_RC_TEMP_FOLDER	12030
OLEAPI_RC_VALIDATION_ERROR	12040
OLEAPI_RC_UNABLE_TO_QUIT	12100
OLEAPI_RC_FAX_NOT_INSTALLED	12110
OLEAPI_RC_FAX_GEN_ERROR	12111
OLEAPI_RC_FAX_EMPTY_TOC	12112
OLEAPI_RC_FAX_NODOCSIN_TOC	12113

メソッド

Error オブジェクトにはメソッドがありません。

Image オブジェクト

重要 : Image オブジェクトでは、一度に複数の文書をオープンできるようにするために Document および Documents オブジェクトを使用することをお勧めします。

Image オブジェクトは、現在見ることのできる文書を保持しています。

プロパティー

Image オブジェクトは、以下のプロパティーをサポートしています。

Application

Application プロパティーは、Application オブジェクトを戻します。

データ型: VT_DISPATCH (Application)

Item Item プロパティーは、Image オブジェクトに関連づけられている *Item* オブジェクトを戻します。

データ型: VT_DISPATCH (Item)

Page Page プロパティーには、選択されたページ番号が入っています。

データ型: VT_14

Parent

Parent プロパティーは、Image オブジェクトの親 (これは Application オブジェクトです) を戻します。

データ型: VT_DISPATCH (Application)

メソッド

Image オブジェクトは、以下のメソッドをサポートしています。

Close Close メソッドは、Image オブジェクトに関連づけられているすべてのウィンドウをクローズします。Save 引き数が True の場合には、そのオブジェクトに加えられた変更が保管されます。Save 引き数が False の場合には、変更は廃棄されます。Save 引き数が指定されない場合には、メッセージ・ボックスが、変更を保管したいか否かをユーザーに尋ねます。

パラメーター: VT_VARIANT としての Save (オプショナル、通常は VT_BOOL)

戻り値: なし

CloseIt

重要 : CloseIt メソッドは Close メソッドと同じ働きをします。このメソッドは、VisualBasic をサポートするためだけにインプリメントされています。これは、 VisualBasic が Close を予約語として使用しているためです。CloseIt メソッドは、 Image オブジェクトに関連づけられているすべてのウィンドウをクローズします。Save 引き数が True の場合には、そのオブジェクトに加えられた変更が保管されます。Save 引き数が False の場合には、変更は廃棄されます。Save 引き数が指定されない場合には、メッセージ・ボックスが、変更を保管したいか否かをユーザーに尋ねます。

パラメーター: VT_VARIANT としての Save (オプショナル、通常は VT_BOOL)

戻り値: なし

DisplayPage

DisplayPage メソッドは、イメージ・ビューアーの中の指定されたページを強制的に表示します。

パラメーター: VT_I4 としての Page

戻り値: VT_I4

FirstPage

FirstPage メソッドは、ビューアーの先頭ページを表示します。

パラメーター: なし

戻り値: VT_I4

LastPage

LastPage メソッドは、ビューアーの最後のページを表示します。

パラメーター: なし

戻り値: VT_I4

NextPage

NextPage メソッドは、ビューアー内の次のページ (現行ページの 1 つ後のページ) を表示します。

パラメーター: なし

戻り値: VT_I4

OpenDocument

OpenDocument メソッドは、イメージ・ビューアー内の新しい IBM Content Manager for iSeries 文書をオープンします。Index 引き数は、オープンされる項目です。項目がワーク・バスケットまたはフォルダーでない場合には、Item エラーが発生します。

パラメーター: VT_DISPATCH (Item) としての Index

戻り値: VT_I4

PreviousPage

PreviousPage メソッドは、ビューアー内の前のページ (現行ページの 1 つ前のページ) を表示します。

パラメーター: なし

戻り値: VT_I4

Item オブジェクト

Item オブジェクトは、フォルダー、ワーク・バスケット、文書などの項目を表します。

プロパティー

Item オブジェクトは、以下のプロパティーをサポートしています。

Application

Application プロパティーは、Application オブジェクトを戻します。

データ型: VT_DISPATCH (Application)

CheckedStatus

CheckedStatus プロパティーは、チェックアウトされた項目 (存在する場合) を持つユーザーのユーザー ID を戻します。

データ型: VT_BSTR

Class Class プロパティーは、項目の索引クラスです。キー・フィールド値に加えられた変更内容は、UpdateIndex メソッドが呼び出されるまで更新されません。これは読み取り/書き込みプロパティーです。

データ型: VT_BSTR

ItemID

ItemID プロパティーは、IBM Content Manager for iSeries ファイル・ルームの各項目を固有に定義するストリングです。

データ型: VT_BSTR

Name Name プロパティーは、項目の IBM Content Manager for iSeries の名前を戻します。このプロパティーは、索引クラスの作成時に ID (存在する場合) として選択されたキー・フィールドに基づいています。項目がワーク・バスケットの場合は、ワーク・バスケット名が戻されます。

データ型: VT_BSTR

PartCount

PartCount プロパティーは、文書に格納されている部分の数を戻します。

データ型: VT_14

Parent

Parent プロパティーは、Image オブジェクトの親 (これは Application オブジェクトまたは Items オブジェクトのどちらでもかまいません) を戻します。

データ型: VT_DISPATCH (Application または Items)

Priority

Priority プロパティーは、項目のワーク・バスケット優先順位を戻します。有効な値は 1 〜 31,999 です。ここで、1 は最も低い優先順位です。項目がワーク・バスケットの中にない場合には、Priority プロパティーは、クラスのデフォルト優先順位を戻します。なお、このプロパティーは読み取り専用プロパティーです。

データ型: VT_14

SystemAssigned

ワーク・バスケットがシステムが割り当てたワーク・バスケットの場合は、TRUE を戻します。

データ型: VT_BOOL

TOCCount

TOCCount プロパティーは、この目次の中で索引付けされている項目の数を戻します。

データ型: VT_14

Type Type プロパティーは、項目の項目タイプを戻します。値 1 は文書、値 2 はフォルダー、値 3 はワーク・バスケットをそれぞれ意味します。

データ型: VT_14

メソッド

Item オブジェクトは、以下のメソッドをサポートしています。

Activate

Activate メソッドは、中断された項目から中断状況を取り除きます。

パラメーター: なし

戻り値: VT_I4

AddAnnotationPart

AddAnnotationPart メソッドでは、既存の文書に注釈部分を追加することができます。 Path 引き数には、文書で使用する新しい注釈部分への絶対パスを指定する必要があります。注釈部分が既に存在している場合は、新しい注釈ファイルに置き換えられます。拡張子には「.T_L」が使用されます。別の拡張子を指定しても「.T_L」が使用されることに注意してください。

パラメーター: VT_BSTR としての Path

戻り値: VT_I4

AddPart

AddPart メソッドは、オブジェクトとしてのファイルを項目に追加します。絶対パスおよびコンテンツ・クラスを指定する必要があります。オプショナルでユーザーは、ライブラリー・サーバーが規則に従ってオブジェクト・サーバーとコレクションを新しい部分に選択するように指定することができます (通常は、ユーザーのデフォルト・オブジェクト・サーバーおよびコレクション)。

パラメーター:

VT_BSTR としての Path

VT_BSTR としての ContentClass

VT_VARIANT としての SMSOption (オプショナル、通常は VT_BOOL)

SMSOption が TRUE (0 以外) に設定されるか、または省略されると、索引クラスのデフォルト・オブジェクト・サーバーおよびコレクションが使用されます (最初の動作)。 SMSOption が FALSE (0) に設定されると、ライブラリー・サーバーは、構成 (通常はユーザーのデフォルト設定) に基づいて、オブジェクト・サーバーおよびコレクションを選択します。

戻り値: VT_14

AddToFolder

AddToFolder メソッドは、Item オブジェクトを別の Item オブジェクトとして、指定されたフォルダーに追加します。

パラメーター: VT_DISPATCH (Item) としての Folder

戻り値: VT_I4

ChangeNotes

ChangeNotes メソッドは、引き数値を注釈ログとして保管します。

パラメーター: VT_BSTR としての Value

戻り値: VT_I4

ChangeWorkflow

ChangeWorkflow メソッドでは、項目を送るワークフローを新たに指定します。新しいワークフローは、名前に WorkFlow 引き数を付けて指定します。

パラメーター: VT_BSTR としての WorkFlow

戻り値: VT_I4

CheckIn

CheckIn メソッドは、項目をチェックインし、他のユーザーがその項目を変更できるようにします。

パラメーター: なし

戻り値: VT_I4

CheckOut

CheckOut メソッドは、現行ユーザーに対して項目をチェックアウトし、他のユーザーがその項目を変更できないようにします。

パラメーター: なし

戻り値: VT_I4

Close Close メソッドは、Open メソッドまたは NextWorkbasketItem メソッドを用いて以前にロックされた項目 (結果として生ずる項目であり、ワーク・バスケットではない) をアンロックします。

パラメーター: なし

戻り値: VT_I4

CloseIt

重要 : CloseIt メソッドは Close メソッドと同じ働きをします。このメソッドは、VisualBasic をサポートするためだけにインプリメントされています。これは、 VisualBasic が Close を予約語として使用しているためです。CloseIt メソッドは、Open メソッドまたは NextWorkbasketItem メソッドを用いて以前にロックされた項目 (結果として生ずる項目であり、ワーク・バスケットではない) をアンロックします。

パラメーター: なし

戻り値: VT_I4

CloseNotes

CloseNotes メソッドは、変更内容を保管せずに、オープン中の注釈ログをクローズします。

パラメーター: なし

戻り値: VT_I4

CloseParts

CloseParts メソッドは、変更内容を保管せずに、すべてのオープン部分ファイル (ページ) をクローズします。

パラメーター: なし

戻り値: VT_I4

CompleteWorkflow

CompleteWorkflow メソッドは、ワークフローを正常に完了した項目にマークを付けます。

パラメーター: なし

戻り値: VT_I4

Delete Delete メソッドは、ファイル・ルームから項目を除去します。これはリカバリー不能の操作であるため、注意してこのメソッドを使用してください。

パラメーター: なし

戻り値: VT_I4

DeletePart

DeletePart メソッドは、指定されたオブジェクト (部分) を項目から削除します。

パラメーター: VT_I4 としての Index

戻り値: VT_I4

FaxItem

FaxItem メソッドは、項目 (ロードされている場合) をファクシミリ・サブシステムに送ります。引き数 withSubFolderContents (指定され、かつ True (非ゼロ) に設定されている場合) を使用して、フォルダーに入っている文書をファックスで送ることができます。

パラメーター: VT_VARIANT としての withSubFolderContents (オプショナル、通常は VT_BOOL)

戻り値: VT_I4

GetAnnotationFile

GetAnnotationFile メソッドは、項目の注釈ファイルを検索します。

パラメーター: なし

戻り値: VT_BSTR

GetHistoryLog

GetHistoryLog メソッドは、項目の作業ヒストリーを検索します。

パラメーター: なし

戻り値: VT_BSTR

GetKeyFields

GetKeyFields メソッドは、項目の所定のキー・フィールドの値を戻します。

パラメーター: VT_BSTR としての Name

戻り値: VT_BSTR

GetNotes

GetNotes メソッドは、IBM Content Manager for iSeries から注釈ログのテキストを検索し、それを呼び出し側アプリケーションに戻します。最初にこのメソッドが呼び出されると、Item オブジェクトは IBM Content Manager for iSeries でチェックアウトされます。

パラメーター: なし

戻り値: VT_BSTR

GetPartContentClass

GetPartContentClass メソッドは、Index 引き数で指定された部分ファイルのコンテンツ・クラスを戻します。

パラメーター: VT_I4 としての Index

戻り値: VT_BSTR

GetPartFile

GetPartFile メソッドは、IBM Content Manager for iSeries からオブジェクト・ファイルを取り出し、それをローカル・ワークステーション上に保管し、さらに絶対パスをその一時ファイルに戻します。最初にこのメソッドが呼び出されると、Item オブジェクトは IBM Content Manager for iSeries でチェックアウトされます。

パラメーター: VT_I4 としての Index

戻り値: VT_BSTR

GetParentFolders

GetParentFolders メソッドは、フォルダーの Items コレクションを戻します。これらの各フォルダーには、メソッドを呼び出したフォルダーの文書が含まれます。

パラメーター: なし

戻り値: VT_DISPATCH (Items)

GetTOCItem

GetTOCItem メソッドは、TOC から指定された Item オブジェクトを戻します。

パラメーター: VT_I4 としての Index

戻り値: VT_DISPATCH (Item)

NextWorkbasketItem

NextWorkbasketItem メソッドは、ワーク・バスケット内の優先順位の順序で使用可能な次の項目を戻します。

パラメーター: なし

戻り値: VT_DISPATCH (Item)

Open Open メソッドは、項目をロックします。項目がロックされると、他のどのユーザーも、索引情報または部分を変更することができません。項目をアンロックするには、Close または CloseIt メソッドを使用する必要があります。

パラメーター: なし

戻り値: なし

PreStage

PreStage メソッドは、将来の検索のためにオフライン部分を設定します。このメソッドは、Item.GetPartFile が 6265 (SIM_RC_OBJECT_BEINGPROMOTED) 例外を戻したときに呼び出します。この例外は、部分オブジェクトがオフラインの記憶装置上にあることを示すものです。

パラメーター: VT_14 としての Index

戻り値: なし

PrintItem

PrintItem メソッドは、現行の印刷オプションを使用して、現在選択されているプリンターへ項目を出力します。 ShowDialog が true の場合は「印刷

(print)」ダイアログが表示され、ここで別のプリンターを選択したり、オプションを変更したり、印刷を取り消したりすることができます。

パラメーター: VT_BOOL としての ShowDialog、VT_VARIANT としての PrintImage、VT_VARIANT としての StartPage、 VT_VARIANT としての EndPage、VT_VARIANT としての PrintMarkup、VT_VARIANT としての PrintIndex、 VT_VARIANT としての PrintNoteLog、VT_VARIANT としての PrintTOC、VT_VARIANT としての PrintContents

戻り値: VT_I4

- *PrintImage* (オプショナル) は、文書の基本部分 (イメージとも呼ばれる) を印刷するかどうかを指定します。 *PrintMarkup* (オプショナル) も選択すると、定義済みの注釈がすべてイメージ上に印刷されます。
- *StartPage* (オプショナル) および *EndPage* (オプショナル) は、印刷したい基本部分ページの範囲を指定します。ページは、1 から通しで番号が付けられています。

  例 :

  – 5 ページの文書のうち、中央の 3 ページを印刷するには、*StartPage* に 2、*EndPage* に 4 を設定します。
  – 文書全体を印刷するには、*StartPage* に 1、*EndPage* に 10000 (または十分に大きな数) を設定します。
- *PrintIndex* および *PrintNotelog* では、索引情報および注釈ログ情報を、文書およびフォルダーに印刷するかどうかを指定することができます。ワーク・バスケットは、これらの引き数を無視します。ただし、ワーク・バスケットで *PrintIndex* および *PrintNotelog* を true に設定すれば、注釈ログ情報および索引情報を、ワーク・バスケットに含まれている文書およびフォルダーに印刷することができます。
- *PrintTOC* と *PrintContents* は、ワーク・バスケットおよびフォルダーの印刷方法を指定します。*PrintTOC* が true の場合は、フォルダーまたはワーク・バスケットに含まれている項目のリストが印刷されます。
  PrintContents が true の場合は、目次に含まれているフォルダーおよび文書も印刷されます。

RefreshTOC

RefreshTOC メソッドは、ワーク・バスケットまたはフォルダーの TOC を再抽出します。このメソッドを呼び出さなかった場合には、ワーク・バスケットまたはフォルダーの TOC に加えられた変更は、Item クラスのメソッドでは認識されません。

パラメーター: なし

戻り値: VT_I4

RemoveFromFolder

RemoveFromFolder メソッドは、引き数として指定されたフォルダーから項目を除去します。

パラメーター: VT_DISPATCH (Item) としての Folder

戻り値: VT_I4

RemoveFromWorkbasket

RemoveFromWorkbasket メソッドは、引き数として指定されたワーク・バスケットから項目を除去します。

パラメーター: VT_DISPATCH (Item) としての Workbasket

戻り値: VT_I4

RemoveFromWorkflow

RemoveFromWorkflow メソッドは、ワークフローから取り消されている項目をマークします。

パラメーター: なし

戻り値: VT_I4

RouteToWorkbasket

RouteToWorkbasket メソッドは、この項目をあるワーク・バスケットに追加し、その項目が現在入っているワーク・バスケットからその項目を除去します。ワーク・バスケットは、その Item オブジェクトによって指定されます。Force 引き数が TRUE として指定されている場合、ワーク・バスケットが既にいっぱいであっても、項目はそのワーク・バスケットに追加されます。

パラメーター:

VT_DISPATCH (Item) としての Workbasket

VT_VARIANT としての Priority (オプショナル、通常は VT_I4)

VT_VARIANT としての Force (オプショナル、通常は VT_BOOL)

戻り値: VT_I4

SavePart

SavePart メソッドは、指定された部分ファイルおよびその注釈ファイルに対して加えられた変更内容を保管します。

戻り値: VT_I4

SetKeyFields

SetKeyFields メソッドは、項目の所定のフィールドの値を設定します。更新したキー・フィールドをサーバーに保管する場合は、UpdateIndex メソッドを呼び出す必要があります。

パラメーター: VT_BSTR としての Name、VT_BSTR としての NewValue

戻り値: なし

StartWorkflow

StartWorkflow メソッドは、項目を指定されたワークフローに追加します。

パラメーター:

VT_BSTR としての Workflow

VT_VARIANT としての Workbasket (オプショナル、通常は VT_DISPATCH)

VT_VARIANT としての Priority (オプショナル、通常は VT_14)

戻り値: VT_I4

Suspend

Suspend メソッドを使用すると、項目が中断され、それ以後のイベントが保留にされます。このイベントは日時ですが、フォルダー項目に組み込まれる項目になる場合もあります。

パラメーター:

VT_VARIANT としての Timestamp (オプショナル、通常は VT_BSTR)

VT_VARIANT としての TimeoutWorkbasket (オプショナル、通常は VT_DISPATCH)

VT_VARIANT としての Classes (オプショナル、通常は VT_BSTR)

VT_VARIANT としての Criteria (オプショナル、通常は VT_I4)

VT_VARIANT としての ReadyWorkbasket (オプショナル、通常は VT_DISPATCH)

Timestamp 引き数が指定されると、項目は中断され、時間イベントが保留にされます。時間イベントが生成されると、項目が活動化され、TimeOutWorkbasket ワーク・バスケットに入れられます。Timestamp 引き数は、次のような形式でなければなりません。

```
1997-09-30-08.05.23.000000
```

Classes 引き数 (フォルダー項目の場合にのみ有効) が指定される場合、項目は中断され、時間イベントまたはフォルダー・イベントが保留にされます。時間イベントが生成されると、項目が活動化され、*TimeOutWorkbasket* ワーク・バスケットに入れられます。フォルダー・イベントがタイムアウトの前に生成されると、項目が活動化され、*ReadyWorkbasket* ワーク・バスケットに入れられます。

オプショナルの *Classes* 引き数は、セミコロン (;) で区切られた索引クラスのリストが入っているストリングです。このリストは、どの索引クラスが活動化を生成するのかを示すために使用されます。

オプショナルの *Criteria* 引き数 (これは、フォルダー項目の場合にのみ有効です) は、OR 条件を指示するための 0、または AND 条件を指示するための 1 にする必要があります。この条件は、*Classes* 引き数に指定されている索引クラスのうちの 1 つまたはすべてを、フォルダーが活動化される前に索引付けする必要があるかどうかを決めるときに使用されます。

戻り値: VT_I4

UpdateIndex

UpdateIndex メソッドは、Class プロパティーおよび/または SetkeyFields メソッドを使用して索引クラスまたはキー・フィールド (あるいはその両方) に加えられた変更内容を保管します。このメソッドが呼び出されるまでは、変更内容は保管されません。

パラメーター: なし

戻り値: VT_I4

Items コレクション

Items コレクションは、Item オブジェクトのリストを保持しており、これにより、ユーザーはリストに入っているオブジェクトにアクセスすることができます。Items コレクションは通常、Document のメソッド SelectionList の結果として生じたものです。

プロパティー

Application

Application プロパティーは、Application オブジェクトを戻します。

データ型: Application

Count Count プロパティーは、Items コレクションで参照される Item オブジェクトの数を戻します。

データ型: VT_14

Parent

Parent プロパティーは、Items コレクションの親 (これは通常、Document オブジェクトです) を戻します。

データ型: VT_DISPATCH (Document)

メソッド

_NewEnum

_NewEnum メソッドは、IID_IEnumVARIANT をサポートする未知のものを戻します。_NewEnum は、他のメソッドのようには呼び出すことができない制限付きメソッドです。このメソッドは、Visual Basic のようなマクロ言語でのループ構成体を実現するために使用されます。

パラメーター: なし

戻り値: VT_UNKNOWN

Close Close メソッドは、*Items* コレクションをクローズします。

パラメーター: なし

戻り値: VT_I4

CloseIt

重要 : CloseIt メソッドは Close メソッドと同じ働きをします。このメソッドは、VisualBasic をサポートするためだけにインプリメントされています。これは、 VisualBasic が Close を予約語として使用しているためです。CloseIt メソッドは、 Items コレクションをクローズします。

パラメーター: なし

戻り値: VT_I4

Item Item メソッドは、 Items コレクションから Item オブジェクトを戻します。

パラメーター: VT_I4 としての Index

戻り値: VT_DISPATCH (Item)

第 6 章 高水準プログラミング・インターフェースのサンプル

Visual Basic 用高水準プログラミング・インターフェースのサンプル

Content Manager for iSeries クライアント高水準プログラミング・インターフェースは、頻繁に使用されるフォルダーおよび文書管理関数のセットです。これらの高水準関数は、ユーザーが Content Manager for iSeries 内の文書およびフォルダーにどのようにアクセスするかを反映する簡単な呼び出しインターフェースを持っています。Visual Basic を使用する Content Manager for iSeries クライアント高水準プログラミング・インターフェースの特長には、以下のような点があります。

- 頻繁に使用されるフォルダーおよび文書の管理機能のセットとして、約 30 の関数があります。
- Windows クライアントのアプリケーションを介した Content Manager for iSeries への単一ワークステーション・ログオン
- Visual Basic OLE オートメーション・ソース・コードの提供

さらに、Windows クライアントを使用すると、複数のアプリケーションが同時に Content Manager for iSeries にアクセスすることができます。

一般使用

Content Manager for iSeries クライアント高水準プログラミング・インターフェースは、Content Manager for iSeries データ・モデルの基本構成要素 (文書、フォルダー、およびワーク・バスケット) との対話を行います。Content Manager for iSeries の文書は、密接に関連したオブジェクトまたは部分のセットから構成されます。

Content Manager for iSeries クライアント高水準プログラミング・インターフェースは、*単一の基本部分* (たとえば、スキャンされた文書やワード・プロセッシング・ファイル) と *単一の注部分* から構成される典型的な Content Manager for iSeries 文書の作成、表示、更新、および削除を行う機能を提供します。複数の基本部分を含む文書で Content Manager for iSeries 高水準プログラミング・インターフェースを使用すると、予期しない結果や望ましくない結果が生じる可能性があります。Content Manager for iSeries データ・モデルに関する追加情報については、5 ページの『論理データ・モデルの理解』を参照してください。

Windows クライアントの OLE オートメーション・インターフェースは、複数の基本部分から成る文書を処理する機能を備えています。Visual Basic のソース・コードが提供されるので、VHLPI をカスタマイズして他の文書構成を処理することができます。

VHLPI 関数によって戻されるデータのリストは、ログオンしているユーザー ID の特権設定に基づいてフィルタリングすることができます。また、VHLPI 関数に対してパラメーターとして指定される索引クラスおよび属性の名前は、通常は大文字小文字が区別されるので注意してください。

Visual Basic のパラメーターと変数

VHLPI 関数にパラメーターとして渡される Visual Basic 変数はすべて、 Variant または Variant Array タイプでなければなりません。Variant Array が渡された場合は、エレメント索引 0 を除いて、配列のサイズが配列のエレメント 0 に含まれていなければなりません。

NULL 値は、変数を空ストリング "" に割り当てることによって設定できます。

VHLPI コード・モジュール FRNWWFVB.BAS とともに組み込まれているグローバル変数がいくつかあります。FRNWWFVB.BAS を組み込んだ Visual Basic プログラムはすべて、これらのグローバル変数にアクセスできます。これらのグローバル変数は、次のとおりです。

- **VhlApplObj** - Windows クライアント' のアプリケーション・オブジェクト
- **VhlDocsObj** - Windows クライアント' の文書コレクション・オブジェクト
- **VhlErrorObj** - Windows クライアント' のエラー・オブジェクト

これらのグローバル変数は、**VbVhlLoadFuncs** 関数によって作成され、**VbVhlDropFuncs** 関数によって解放されます。Visual Basic プログラムは、VHLPI 関数を使用する前に **VbVhlLoadFuncs** を呼び出し、終了してこれらのオブジェクトを解放する前に **VbVhlDropFuncs** を呼び出す必要があります。

これらの変数が作成されると、Visual Basic プログラムはメソッドを呼び出すことができ、それらに関連するプロパティーの入手や設定も行うことができます。たとえば、Windows クライアントがどのサーバーにログオンしているかを調べるには、以下を実行します。

```
' Create Objects
ulRC = VbVhlLoadFuncs

' Get what server is logged on
Server$ = VhlApplObj.Server

' Display the server name
MsgBox "The server is " & Server$
```

Windows クライアントへのアクセス

Windows クライアントを使用して、Content Manager for iSeries との一定のログオン・セッションを維持することができます。このプログラムは、開始されると Content Manager for iSeries にログオンし、オペレーター・コマンドを待ちます。ログオン後は、確立された Content Manager for iSeries ログオン・セッションは、他のアプリケーションにより OLE オートメーション・インターフェースを介して使用できます。

Windows クライアントのログオン・セッションを使用することにより、他のアプリケーションは Content Manager for iSeries にログオンする必要がなくなり、むしろ Windows クライアントから OLE オートメーションの Application オブジェクトを作成する必要があります。このことは次を実行することによって行えます。

```
Set VhlApplObj = CreateObject("Vic.Application")
```

ここで、VhlApplObj は VHLPI コード・モジュール FRNWWFVB.BAS に組み込まれているグローバル変数オブジェクトです。

VbVhlLoadFuncs 関数はこの処理を行うほかに、他のグローバル・データ・オブジェクトの初期設定も行います。Visual Basic プログラムでは、**VbVhlLoadFuncs** および **VbVhlDropFuncs** を使用して Windows クライアントへのアクセスの取得と終了を行うことをお勧めします。

上記の説明は、Windows クライアントが開始され、ログオンしたあとで、後続の Visual Basic アプリケーションが実行される状況について述べたものです。それ以外の場合には、次の節で説明するように、Visual Basic アプリケーションはログオン・コマンドとログオフ・コマンドを発行する必要があります。

Windows クライアントでのログオン/ログオフの使用

Visual Basic アプリケーションが実行される前に Windows クライアントが開始されてログオンしていない場合には、アプリケーションは **VbVhlLoadFuncs** ではなく **VbVhlLogon** を呼び出す必要があります。**VbVhlLogon** は Windows クライアントを開始し、ログオン・メソッドを発行して Content Manager for iSeries にログオンします。

Windows クライアントが Content Manager for iSeries にログオンすると、ユーザー ID またはサーバー情報が異なっていても、その後のログオンの試みによって別のログオン試行が引き起こされることはありません。それ以降のログオンでは、オリジナルのログオン・セッションが使用され、エラー表示は出されません。

VbVhlLogoff は、他のアプリケーションがログオン・セッションを使用している場合でも、ログオフ・メソッドを発行して Windows クライアントをクローズします。Windows クライアントを終了させたくない場合には、**VbVhlDropFuncs** を使用して、現行のアプリケーションに対するアクセスだけを終了してください。

Windows 用高水準プログラミング・インターフェース API のサンプル

VbVhlAddFolderItem (フォルダーへの項目の追加)

形式

VbVhlAddFolderItem (*ItemId, FolderId*)

目的

この機能を使用して、文書またはフォルダー (その項目 ID によって指定) を既存のフォルダー (そのフォルダーの項目 ID によって指定) に追加します。

パラメーター

ItemId - 入力

フォルダーに追加される文書またはフォルダーの項目 ID。

FolderId

- 入力

フォルダーの項目 ID。

使用の手引き

追加する項目の項目 ID とフォルダーの項目 ID は、両方とも有効でなければなりません。

Visual Basic ソース・コード

```
Function VbVhlAddFolderItem (ItemID, FolderId)

    ' Declarations
    Dim ItemObj As Object
    Dim FolderObj As Object

    ' Setup Error handler
    On Error GoTo VhlAddFolderError
    ulRC = 0

    ' Get the Folder Object
    Set FolderObj = VhlApplObj.ItemID(FolderId)

    ' Get the ItemID Object
    Set ItemObj = VhlApplObj.ItemID(ItemID)

    ' Put ItemId into Folder
    ulRC = ItemObj.AddToFolder(FolderObj)

VhlAddFolderEnd:

    ' Free the objects
    Set ItemObj = Nothing
    Set FolderObj = Nothing

    ' Set return value to error code
    VbVhlAddFolderItem = ulRC

    Exit Function

VhlAddFolderError:

    ' Set return value to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlAddFolderEnd

End Function
```

VbVhlAdminItemNoteLog (管理者文書注釈ログの管理)

形式

VbVhlAdminItemNoteLog (*ItemID, FuncInd, NoteText*)

目的

この機能を使用して、項目の注釈ログを置換、削除、取得、または追加します。

パラメーター

ItemID - 入力

項目の ID。

FuncInd

- 入力

関数標識。次のどれかでなければなりません。

"APPEND"
項目の注釈ログを *NoteText* に追加します。

"DELETE"
項目の注釈ログを削除します。

"REPLACE"
項目の注釈ログを *NoteText* で置換します。

"GET" 項目の注釈ログを *NoteText* にコピーします。

NoteText

- 入出力

注釈のテキスト値が入っている Visual Basic 変数名。

- *FuncInd = GET* の場合は、この関数は項目の注釈ログをこの Visual Basic 変数にコピーします。
- *FuncInd = REPLACE* の場合は、この関数は要求された項目の注釈ログをこの Visual Basic 変数の内容で置換します。
- *FuncInd = APPEND* の場合は、この関数はこの Visual Basic 変数に入っているテキストを要求された項目の注釈ログに追加します。

使用の手引き

文書の項目 ID は有効でなければなりません。

Visual Basic ソース・コード

```
Function VbVhlAdminItemNoteLog (ItemId, FuncInd, NoteText)

    ' Declarations
    Dim ItemObj As Object

    ' Setup Error handler
    On Error GoTo VhlAdminNoteError
    ulRC = 0

    ' Get the Item object
    Set ItemObj = VhlApplObj.ItemID(ItemId)

    ' Determine what to do
    Select Case FuncInd
    Case "APPEND"
       OldNoteText = ItemObj.GetNotes
       ulRC = ItemObj.ChangeNotes(OldNoteText & NoteText)
    Case "DELETE"
       ulRC = ItemObj.ChangeNotes("")
    Case "REPLACE"
       ulRC = ItemObj.ChangeNotes(NoteText)
    Case "GET"
       NoteText = ItemObj.GetNotes
    End Select

VhlAdminNoteEnd:

    ' Free the object
    Set ItemObj = Nothing
```

```
    ' Set return value to error code
    VbVhlAdminItemNoteLog = ulRC

    Exit Function

VhlAdminNoteError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlAdminNoteEnd

End Function
```

VbVhlChangeItemIndex (項目の索引クラスの変更)

形式

VbVhlChangeItemIndex (*ItemId, ClassName, AttrName(), AttrValue()*)

目的

この機能を使用して、既存の文書またはフォルダー (項目 ID によって指定) に、別の索引クラス名および索引クラス属性 (名前/値) を関連付けます。

パラメーター

ItemId - 入力

変更される文書またはフォルダーの項目 ID。

ClassName

- 入力

項目の新しい索引クラスの名前。

AttrName()

- 入力

AttrValue() の属性値の配列に対応する属性名の配列。これらの属性名は、指定された *ClassName* 用に定義されている必要があります。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

AttrValue()

- 入力

AttrName() の属性名の配列に対応する属性値の配列。これらの属性値は、この属性用の索引クラス *ClassName* に定義されているデータ型にとって有効でなければなりません。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

使用の手引き

指定する ItemId と索引クラス名は、この関数を使用する前に存在している必要があります。さらに、入力配列リスト内の属性がこの索引クラス用に定義されている必要があり、この索引クラスのすべての必要な 属性が配列リストで*指定*されている必要があります。

属性名と属性値の配列を指定する際、*属性名* 配列の各要素には同じ配列指数を持つ *属性値* 配列要素が対応している必要があります。

Visual Basic ソース・コード

```
Function VbVhlChangeItemIndex (ItemID, ClassName, AttrName(), AttrValue())

    ' Declarations
    Dim ItemObj As Object

    ' Setup Error handler
    On Error GoTo VhlChgIndexError
    ulRC = 0

    ' Get the search results folder
    Set ItemObj = VhlApplObj.ItemID(ItemID)

    ' Update Item index class
    ItemObj.Class = ClassName

    ' Update the Item attributes
    For i = 1 To AttrName(0)
       ItemObj.KeyFields(AttrName(i)) = AttrValue(i)
    Next
    ' Update the Items Index Class and attribute information
    ulRC = ItemObj.UpdateIndex

VhlChgIndexEnd:

    ' Free the objects
    Set ItemObj = Nothing

    ' Set return value to error code
    VbVhlChangeItemIndex = ulRC

    Exit Function

VhlChgIndexError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlChgIndexEnd

End Function
```

VbVhlCloseDocViews (文書イメージ・ビュー・ウィンドウのクローズ)

形式

VbVhlCloseDocViews (*fUpdate*)

目的
この関数は、現在イメージ・ビューアーに表示されている文書をクローズします。

パラメーター

fUpdate

- 入力

表示中の文書への変更を保管するかどうかを指定するフラグ (*True* または *False*)。

使用の手引き
現在イメージ・ビューアーに表示されている文書表示ウィンドウは、この関数の実行後にクローズされます。*fUpdate* パラメーターは、文書への変更 (注釈、強調表示、など) を保管するかどうかを指定します。

Visual Basic ソース・コード

```
Function VbVhlCloseDocViews (fUpdate)

    ' Declarations
    Dim ImageObj As Object

    ' Setup Error handler
    On Error GoTo VhlCloseDocError
    ulRC = 0

    ' Close Document being displayed
    Set ImageObj = VhlApplObj.Image
    If Not (ImageObj Is Nothing) Then
       ImageObj.CloseIt (fUpdate)
    End If

VhlCloseDocEnd:
    ' Set return value to error code
    VbVhlCloseDocViews = ulRC

    Exit Function

VhlCloseDocError:
    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlCloseDocEnd

End Function
```

VbVhlCopyDoc (文書のコピーの作成)

形式

VbVhlCopyDoc (*NewDocID, DocID, ClassName, AttrName(), AttrValue()*)

目的
この機能を使用して、新規文書を作成し、既存の文書から新規文書へすべてのオブジェクトをコピーします。新規文書は、新しい索引クラスに設定することも、デフォルトの索引クラスに設定することもできます。

パラメーター

NewDocID
- 出力

作成された文書の項目 ID はこの Visual Basic 変数に戻されます。

DocID - 入力

元の文書の項目 ID。

ClassName
- 入力

新しい文書用の新しい索引クラスの名前。*NULL* に設定すると、索引クラスは *NOINDEX* に設定されます。

AttrName()
- 入力

AttrValue() の属性値の配列に対応する属性名の配列。これらの属性名は、指定された *ClassName* 用に定義されている必要があります。*ClassName* が *NULL* の場合は使用されません。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

AttrValue()
- 入力

AttrName() の属性名の配列に対応する属性値の配列。これらの属性値は、この属性用の索引クラス *ClassName* に定義されているデータ型にとって有効でなければなりません。 *ClassName* が *NULL* の場合は使用されません。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

使用の手引き

文書項目 ID は有効でなければなりません。*ClassName* が NULL でない場合、この項目 ID はこの関数を使用する前に存在している必要があります。さらに、入力配列リスト内の属性がこの索引クラス用に定義されている必要があり、この索引クラスのこの索引クラスを一意的に索引付けするために使用されるすべての必要な属性が新しい属性配列リストで指定されている必要があります。

ClassName が NULL の場合は、新しい文書の索引クラスは *NOINDEX* に設定され、属性 *Source* は ″COPY″ に、属性 Name と Timestamp はユーザー ID と現在時刻に設定されます。

新しく作成された文書の項目 ID は、指定された Visual Basic 変数 *NewDocID* に保管されます。

Visual Basic ソース・コード

```
Function VbVhlCopyDoc (NewDocId, DocId, ClassName, AttrName(), AttrValue())

    ' Declarations
    Dim ItemObj As Object
    Dim NewItemObj As Object

    ' Setup Error handler
    On Error GoTo VhlCopyDocError
```

```
    ulRC = 0

    ' Get the Document object
    Set ItemObj = VhlApplObj.ItemID(DocId)
    ' Make sure the object is a document
    If ItemObj.Type <> 1 Then
       ' Return with error - SBVI_BAD_DOCUMENT
       ulRC = 909
       GoTo VhlCopyDocEnd
    End If

    ' Create a new document
    Set NewItemObj = VhlApplObj.CreateDocument("COPY")
    NewDocId = NewItemObj.ItemID

    ' Update the new document with Index Class information if provided
    If (ulRC = 0) And (ClassName <> "") Then
       ' Change the Items Index Class
       ulRC = VbVhlChangeItemIndex(NewDocId, ClassName, AttrName(), AttrValue())
    End If

    ' Copy document base parts into new document
    i = 0
    While (ulRC = 0) And (i < ItemObj.PartCount)
       ContentClass = ItemObj.GetPartContentClass(i)
       TempFile = ItemObj.GetPartFile(i)
       ulRC = NewItemObj.AddPart(TempFile, ContentClass)
       i = i + 1
    Wend
    ' Close the original document
    RC = ItemObj.CloseParts

VhlCopyDocEnd:

    ' Free the objects
    Set ItemObj = Nothing
    Set NewItemObj = Nothing

    ' Set return value to error code
    VbVhlCopyDoc = ulRC

    Exit Function

VhlCopyDocError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlCopyDocEnd

End Function
```

VbVhlCreateFolder (新規フォルダーの作成)

形式

VbVhlCreateFolder (*FolderId, ClassName, AttrName(), AttrValue()*)

目的

この機能を使用して、指定された索引クラス名および索引属性 (名前/値) を使用してフォルダーを作成します。

パラメーター

FolderId

- 出力

作成されたフォルダーの項目 ID が保管されている Visual Basic 変数の名前。

ClassName

- 入力

フォルダーの索引クラスの名前。NULL の場合は、「*NOINDEX*」 という名前が使用されます。

AttrName()

- 入力

AttrValue() の属性値の配列に対応する属性名の配列。これらの属性名は、指定された *ClassName* 用に定義されている必要があります。*ClassName* が *NULL* の場合は使用されません。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

AttrValue()

- 入力

AttrName() の属性名の配列に対応する属性値の配列。これらの属性値は、この属性用の索引クラス *ClassName* に定義されているデータ型にとって有効でなければなりません。 *ClassName* が *NULL* の場合は使用されません。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

使用の手引き

指定する索引クラス名は、この関数を使用する前に定義されている必要があります。さらに、入力配列リスト内の属性名がこの索引クラス用に定義されている必要があり、この索引クラスのすべての必要な 属性が配列リストで指定されている必要があります。

ClassName が NULL の場合は、新しいフォルダーの索引クラスは *NOINDEX* に設定され、属性 *Source* は ″CREATE″ に、属性 Name と Timestamp はユーザー ID と現在時刻に設定されます。

新しく作成されたフォルダーの項目 ID は、指定された Visual Basic 変数 *FolderId* に保管されます。

Visual Basic ソース・コード

```
Function VbVhlCreateFolder (FolderId, ClassName, AttrName(), AttrValue())

    ' Declarations
    Dim FolderObj As Object

    ' Setup Error handler
    On Error GoTo VhlCreFoldError
    ulRC = 0

    ' Create the folder
    Set FolderObj = VhlApplObj.CreateFolder("CREATE")
```

```
    FolderId = FolderObj.ItemID
    If (ulRC = 0) And (ClassName <> "") Then
       ' Change the Items Index Class
       ulRC = VbVhlChangeItemIndex(FolderId, ClassName, AttrName(), AttrValue())
    End If

VhlCreFoldEnd:

    ' Free the object
    Set FolderObj = Nothing

    ' Set return value to error code
    VbVhlCreateFolder = ulRC

    Exit Function

VhlCreFoldError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlCreFoldEnd

End Function
```

VbVhlCreateFolderAddItem (フォルダーの作成と項目の追加)

形式

VbVhlCreateFolderAddItem (*FolderId, ItemId, ClassName, AttrName(), AttrValue()*)

目的

この機能を使用して、指定された索引クラス名および索引属性 (名前/値) を使用してフォルダーを作成します。この関数を使用して、新しく作成されたフォルダーに文書またはフォルダー (項目 ID によって指定) を追加することもできます。

パラメーター

FolderId
- 出力

作成されたフォルダーの項目 ID が保管されている Visual Basic 変数の名前。

ItemId - 入力

新しく作成されたフォルダーに追加される文書またはフォルダーの項目 ID。

ClassName
- 入力

フォルダーの索引クラスの名前。NULL の場合は、「*NOINDEX*」 という名前が使用されます。

AttrName()
- 入力

AttrValue() の属性値の配列に対応する属性名の配列。これらの属性名は、指定された *ClassName* 用に定義されている必要があります。*ClassName* が *NULL* の場合は使用されません。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

AttrValue()

- 入力

AttrName() の属性名の配列に対応する属性値の配列。これらの属性値は、この属性用の索引クラス *ClassName* に定義されているデータ型にとって有効でなければなりません。 *ClassName* が *NULL* の場合は使用されません。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

使用の手引き

指定する項目 ID と索引クラスは、この関数を使用する前に定義されている必要があります。さらに、入力配列リスト内の属性名がこの索引クラス用に定義されている必要があり、この索引クラスのすべての必要な 属性が配列リストで指定されている必要があります。

ClassName が NULL の場合は、新しいフォルダーの索引クラスは *NOINDEX* に設定され、属性 *Source* は ″CREATE″ に、属性 Name と Timestamp はユーザー ID と現在時刻に設定されます。

作成されたフォルダーの項目 ID は、指定された Visual Basic 変数 *FolderID* に保管されます。

Visual Basic ソース・コード

```
Function VbVhlCreateFolderAddItem (FolderId, ItemID, ClassName,
                                   AttrName(), AttrValue())

    ' Declarations
    Dim FolderObj As Object
    Dim ItemObj As Object

    ' Setup Error handler
    On Error GoTo VhlCreFoldAddError
    ulRC = 0

    ' Create the folder
    Set FolderObj = VhlApplObj.CreateFolder("CREATE")
    FolderId = FolderObj.ItemID

    ' Get the ItemID Object
    Set ItemObj = VhlApplObj.ItemID(ItemID)

    ' Put ItemId into Folder
    ulRC = ItemObj.AddToFolder(FolderObj)

    If (ulRC = 0) And (ClassName <> "") Then
       ' Change the Items Index Class
       ulRC = VbVhlChangeItemIndex(FolderId, ClassName, AttrName(), AttrValue())
    End If

VhlCreFoldAddEnd:
    ' Free the objects
    Set FolderObj = Nothing
    Set ItemObj = Nothing
```

```
    ' Set return value to error code
    VbVhlCreateFolderAddItem = ulRC

    Exit Function

VhlCreFoldAddError:
    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlCreFoldAddEnd

End Function
```

VbVhlDeleteItem (項目の削除)

形式

VbVhlDeleteItem (*ItemID*)

目的

この機能を使用して、項目 ID Content Manager for iSeries によって指定される文書またはフォルダーを削除します。

パラメーター

ItemID - 入力

Content Manager for iSeries から削除される文書またはフォルダーの項目 ID。

使用の手引き

指定された文書またはフォルダーは物理的には削除されません。

Visual Basic ソース・コード

```
Function VbVhlDeleteItem (ItemID)
    ' Declarations
    Dim ItemObj As Object

    ' Setup Error handler
    On Error GoTo VhlDeleteError
    ulRC = 0

    ' Get the ItemID Object
    Set ItemObj = VhlApplObj.ItemID(ItemID)

    ' Delete the Item
    ulRC = ItemObj.DeleteIt

VhlDeleteEnd:
    ' Free the objects
    Set ItemObj = Nothing

    ' Set return value to error code
    VbVhlDeleteItem = ulRC

    Exit Function

VhlDeleteError:
```

```
    ' Set return value to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlDeleteEnd

End Function
```

VbVhlDisplayDocView (文書イメージの表示)

形式

VbVhlDisplayDocView (*DocId, fUpdate*)

目的

この関数は、文書イメージ (項目 ID で指定) をイメージ・ビューアーに表示します。

パラメーター

DocId - 入力

表示される文書イメージ項目 ID。

fUpdate

- 入力

現在表示中の文書への変更を保管するかどうかを指定するフラグ (*True* または *False*)。

使用の手引き

現在イメージ・ビューアーに表示されている文書表示ウィンドウは、指定された新しい文書が表示される前にクローズされます。*fUpdate* パラメーターは、前の文書への変更 (注釈、強調表示、など) を保管するかどうかを決定します。

Visual Basic ソース・コード

```
Function VbVhlDisplayDocView (ItemID, fUpdate)

    ' Declarations
    Dim ItemObj As Object
    Dim ImageObj As Object

    ' Setup Error handler
    On Error GoTo VhlDispDocError
    ulRC = 0

    ' Get the Item object
    Set ItemObj = VhlApplObj.ItemID(ItemID)

    ' Close Document being displayed
    Set ImageObj = VhlApplObj.Image
    If Not (ImageObj Is Nothing) Then
       ImageObj.CloseIt (fUpdate)
    End If

    ' Display Document
    ulRC = ImageObj.OpenDocument(ItemObj)
```

```
VhlDispDocEnd:

    ' Free the object
    Set ItemObj = Nothing

    ' Set return value to error code
    VbVhlDisplayDocView = ulRC

    Exit Function

VhlDispDocError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlDispDocEnd

End Function
```

VbVhlDisplayVIItem (Windows クライアントを使用した項目の表示)

形式

VbVhlDisplayVIItem (*ItemID, fUpdate*)

目的

Windows クライアントのアプリケーションを使用して、文書、フォルダー、またはワーク・バスケットの内容を表示します。文書はイメージ・ビューアーで表示されます。フォルダーとワーク・バスケットは Windows クライアントのメイン・ウィンドウに別個のウィンドウとして表示されます。

パラメーター

ItemID - 入力

表示される文書またはフォルダーの項目 ID。

fUpdate

- 入力

現在表示中の文書への変更を保管するかどうかを指定するフラグ (*True* または *False*)。これは指定された項目 ID が文書の場合にのみ使用されます。

使用の手引き

文書、フォルダー、またはワーク・バスケットの情報は、Windows クライアントのアプリケーションを使用して表示されます。文書はイメージ・ビューアーで表示されます。フォルダーとワーク・バスケットは Windows クライアントのメイン・ウィンドウに別個のウィンドウとして表示されます。*fUpdate* パラメーターは文書が指定されている場合にのみ使用されます。このフラグは、現在表示中の文書への変更を保管するかどうかを決定します。

Visual Basic ソース・コード

```
Function VbVhlDisplayVIItem (ItemID, fUpdate)

    ' Declarations
    Dim ItemObj As Object
    Dim ImageObj As Object
    Dim FolderObj As Object

    ' Setup Error handler
    On Error GoTo VhlDispItemError
    ulRC = 0

    ' Get the Item object
    Set ItemObj = VhlApplObj.ItemID(ItemID)

    ' Find out if the item is a folder or a document
    If (ItemObj.Type = 1) Then
       ' Close Document being displayed
       Set ImageObj = VhlApplObj.Image
       If Not (ImageObj Is Nothing) Then
          ImageObj.CloseIt (fUpdate)
       End If
       ' Display Document
       ulRC = ImageObj.OpenDocument(ItemObj)
    Else
        ' Must be a folder. Display it.
        Set FolderObj = VhlDocsObj.OpenTOC(ItemObj)
    End If

VhlDispItemEnd:
    ' Free the object
    Set ItemObj = Nothing
    Set FolderObj = Nothing
    Set ImageObj = Nothing

    ' Set return value to error code
    VbVhlDisplayVIItem = ulRC

    Exit Function

VhlDispItemError:
    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlDispItemEnd

End Function
```

VbVhlDropFuncs (VHLPI 関数へのアクセスの終了)

形式

VbVhlDropFuncs()

目的

この API は、Windows クライアントの OLE 自動化インターフェースへのアクセスを終了するために使用します。VHLPI 関数のその後の使用はすべて失敗します。

使用の手引き

この関数を実行した後は、Visual Basic プログラムは VHLPI 関数を呼び出すことができません。これらの関数へのアクセスを確立するには VbVhlLoadFuncs API を使用します。

Visual Basic ソース・コード

```
Function VbVhlDropFuncs ()

    ' Setup Error handler
    On Error GoTo VhlDropError

    ' End access with OLE interface
    ulRC = 0
    Set VhlDocsObj = Nothing
    Set VhlErrorObj = Nothing
       Set VhlApplObj = Nothing

VhlDropEnd:

    ' Set return value to error code
    VbVhlDropFuncs = ulRC

    Exit Function

VhlDropError:

    ' Set return code to error code
    ulRC = Err
    Resume VhlDropEnd

End Function
```

VbVhlExportDocObj (文書の基本オブジェクトのエクスポート)

形式

VbVhlExportDocObj (*DocId, FileName, PartNum*)

目的

この関数は、文書 (*DocId* により指定) の基本オブジェクトが入ったディスク・ファイルを作成します。

パラメーター

DocId - 入力

基本部分をエクスポートされる文書の項目 ID。

FileName

- 入力

作成されるファイルの名前 (パスを含む)。

PartNum

- 入力

エクスポートする基本オブジェクトの部分番号。「0」は最初の基本部分を表します。

使用の手引き

文書項目 ID は有効でなければなりません。また、文書の基本オブジェクトはファイル内に表現できるものでなければなりません。

Visual Basic ソース・コード

```
Function VbVhlExportDocObj (DocId, Filename, PartNum)

    ' Declarations
    Dim DocObj As Object

    ' Setup Error handler
    On Error GoTo VhlExportDocError
    ulRC = 0

    ' Get the document object
    Set DocObj = VhlApplObj.ItemID(DocId)

    ' Copy document base part into file
    TempFile = DocObj.GetPartFile(PartNum)
    Name TempFile As Filename
    ' Close the document
    RC = DocObj.CloseParts

VhlExportDocEnd:

    ' Free the object
    Set DocObj = Nothing

    ' Set return value to error code
    VbVhlExportDocObj = ulRC

    Exit Function

VhlExportDocError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlExportDocEnd

End Function
```

VbVhlGetVIUserID (ログオン・ユーザー ID の取得)

形式

VbVhlGetVIUserID()

目的

この関数は、ログオン・ユーザー ID を戻すために使用します。

使用の手引き

エラーの場合 (たとえば、ログオン・セッションが存在しない場合など) は、NULL ユーザー ID が戻されます。

Visual Basic ソース・コード

```
Function VbVhlGetVIUserID ()

    ' Setup Error handler
    On Error GoTo VhlGetUserError
    ulRC = 0

    ' Set return value to UserId
    VbVhlGetVIUserID = VhlApplObj.User

VhlGetUserEnd:

    Exit Function

VhlGetUserError:

    ' Set return code to error code
    VbVhlGetVIUserID = VhlErrorObj.ReturnCode

    Resume VhlGetUserEnd

End Function
```

VbVhlImportDocObj (文書の基本オブジェクトのインポート)

形式

VbVhlImportDocObj (*DocId, FileName, ContentClass, ClassName, AttrName(), AttrValue()*)

目的

この関数は、文書の基本部分のディスク・ファイル形式から文書の基本オブジェクトを作成します。

パラメーター

DocId - 出力

文書の項目 ID が保管されている Visual Basic 変数の名前。

FileName

- 入力

文書の基本部分が入っているファイル (ファイル拡張子を含む) の名前 (パスを含む)。

ContentClass

- 入力

ファイルのコンテンツ・クラス名です。

ClassName

- 入力

文書の索引クラスの名前。NULL または指定されていない場合には、「*NOINDEX*」という名前が使用されます。

AttrName()
- 入力

AttrValue() の属性値の配列に対応する属性名の配列。これらの属性名は、指定された *ClassName* 用に定義されている必要があります。*ClassName* が *NULL* の場合は使用されません。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

AttrValue()
- 入力

AttrName() の属性名の配列に対応する属性値の配列。これらの属性値は、この属性用の索引クラス *ClassName* に定義されているデータ型にとって有効でなければなりません。 *ClassName* が *NULL* の場合は使用されません。

注: 配列指数 0 は配列要素の数を含んでいなければなりません。

使用の手引き

指定する索引クラス名は、この関数を使用する前に存在している必要があります。さらに、入力配列リスト内の属性名がこの索引クラス用に定義されている必要があり、この索引クラスのすべての必要な 属性が配列リストで指定されている必要があります。

作成された文書の項目 ID は、指定された Visual Basic 変数 *DocId* に保管されます。

Visual Basic ソース・コード

```
Function VbVhlImportDocObj (DocId, Filename, ContentClass,
                            ClassName, AttrName(), AttrValue())

    ' Declarations
    Dim DocObj As Object

    ' Setup Error handler
    On Error GoTo VhlImportDocError
    ulRC = 0

    ' Create the document and add the file
    Set DocObj = VhlApplObj.CreateDocument("IMPORT")
    DocId = DocObj.ItemID
    ulRC = DocObj.AddPart(Filename, ContentClass)
    If (ulRC = 0) And (ClassName <> "") Then
       ' Change the Items Index Class
       ulRC = VbVhlChangeItemIndex(DocId, ClassName, AttrName(), AttrValue())
    End If

VhlImportDocEnd:

    ' Free the object
    Set DocObj = Nothing

    ' Set return value to error code
    VbVhlImportDocObj = ulRC

    Exit Function

VhlImportDocError:

    ' Set return code to error code
```

```
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlImportDocEnd

End Function
```

VbVhlListContClasses (すべてのコンテンツ・クラスのリスト)

形式

VbVhlListContClasses (*CCList()*)

目的

この機能を使用して、すべてのコンテンツ・クラスをリストします。

パラメーター

CCList()

- 出力

すべてのコンテンツ・クラスのリストが保管されている Visual Basic 変数の名前。この Visual Basic 変数名は、*CCList(0)* に索引カウント (戻されたコンテンツ・クラスの数) を持つ配列変数になります。Visual Basic 配列の形式は次のとおりです。

CCList(0)- コンテンツ・クラスの数

CCList(n)- コンテンツ・クラス名 *n*

使用の手引き

この機能を使用して、IBM 定義のコンテンツ・クラスとユーザー定義のコンテンツ・クラスの両方をリストします。

Visual Basic ソース・コード

```
Function VbVhlListContClasses (CCList())

    ' Declarations
    Dim i, ulStart, ulEnd, ulLen, ulTotLen As Long

    ' Setup Error handler
    On Error GoTo VhlContListError
    ulRC = 0

    ' Get the list of Cont Classes
    strRet = VhlApplObj.ContentClassList(";")
       ulTotLen = Len(strRet)

    ' Add Cont classes to List array
    i = 0
    ReDim CCList(1)
    CCList(0) = 0
    ulStart = 1
    Do
       ' Each name separated by a ";"
       ulEnd = InStr(ulStart, strRet, ";")
       If (ulEnd = 0) Then
          ulEnd = ulTotLen + 1
       End If
```

```
        ulLen = ulEnd - ulStart

        ' Set next array variable to Cont Class name
        i = i + 1
        ReDim Preserve CCList(i + 1)
        CCList(i) = Mid$(strRet, ulStart, ulLen)

        ' Setup for next loop
        ulStart = ulEnd + 1
    Loop Until (ulStart >= ulTotLen)

    ' Set total number of Cont Classes in array
    CCList(0) = i

VhlContListEnd:

    ' Set return value to error code
    VbVhlListContClasses = ulRC

    Exit Function

VhlContListError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlContListEnd

End Function
```

VbVhlListFolderItems (フォルダー・コンテンツのリスト)

形式

VbVhlListFolderItems (*ItemList(), FolderID, IndexClass()*)

目的

この機能を使用して、フォルダー (フォルダーの項目 ID によって指定) に含まれ、オプションの索引クラス配列指定に一致する、文書およびフォルダーの項目 ID をすべてリストします。

パラメーター

ItemList()

- 出力

指定されたフォルダーの目次に含まれていて、オプションの索引クラスに一致する文書およびフォルダーのリストが保管されている Visual Basic 変数の名前。この Visual Basic 変数名は *ItemList(0,0)* に保管されている索引カウント (戻された項目 ID の数) を持つ配列変数になり、戻された項目ごとに次のような 3 つの Visual Basic 配列要素を持つ構造になります。

ItemList(n,1)- 項目 ID

ItemList(n,2)- 項目タイプ

--(1) 文書

--(2) フォルダー

--(?)不明

ItemList(n,3)- 索引クラス

FolderID

- 入力

リストするフォルダーの項目 ID。

IndexClass()

- 入力

戻された項目をフィルターに掛けるオプションの索引クラス。要素を指定しなかった場合は、フォルダーの目次にあるすべての項目が、索引クラスにかかわりなく戻されます。

注: 配列指数 0 はリストの中の配列要素の数を含んでいなければなりません。

使用の手引き

フォルダーの項目 ID はこの呼び出しの前に存在していなければなりません。この関数は、ワーク・バスケットのコンテンツをリストするためにも使用できます。

Visual Basic ソース・コード

```
Function VbVhlListFolderItems (ItemList(), FolderId, IndexClass())

    ' Declarations
    Dim FolderObj As Object
    Dim ContentObj As Object
    Dim ulTOCCnt, ulStart, ulEnd, ulLen, ulTotLen As Long

    ' Setup Error handler
    On Error GoTo VhlLstFldError
    ulRC = 0

    ' Get the Folder Object
    Set FolderObj = VhlApplObj.ItemID(FolderId)

    ' Setup return array based on size of folder
    ulTOCCnt = FolderObj.TOCCount
    ReDim ItemList(ulTOCCnt + 1, 4)
    ItemList(0, 0) = 0

    ' Get the list of Item Objects in the Folder
    j = 1
    For i = 1 To ulTOCCnt
       Set ContentObj = FolderObj.GetTOCItem(i - 1)
       ItemList(j, 0) = 3
       ItemList(j, 1) = ContentObj.ItemID
       ItemList(j, 2) = ContentObj.Type
       ItemList(j, 3) = ContentObj.Class
       Set ContentObj = Nothing
       ' Check if Index Class filter was provided
       Found = False
       If IndexClass(0) <> 0 Then
          For k = 1 To IndexClass(0)
             If IndexClass(k) = ItemList(j, 3) Then
                Found = True
                Exit For
             End If
          Next k
       Else
          Found = True
```

```
       End If
       ' Only send back Items found in Index Class list
       If Found Then
          ItemList(0, 0) = j
          j = j + 1
       End If
    Next i

VhlLstFldEnd:

    ' Free the objects
    Set ContentObj = Nothing
    Set FolderObj = Nothing

    ' Set return value to error code
    VbVhlListFolderItems = ulRC

    Exit Function

VhlLstFldError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlLstFldEnd

End Function
```

VbVhlListFolderItemsAttr (フォルダー内容とその属性のリスト)

形式

VbVhlListFolderItemsAttr(*ItemList(), FolderId*)

目的

この機能を使用して、フォルダー (フォルダーの項目 ID によって指定) に含まれ、文書およびフォルダーの項目 ID をすべてリストします。

パラメーター

ItemList()

- 出力

指定されたフォルダーの目次に含まれている文書およびフォルダーのリストが保管されている Visual Basic 変数の名前。この Visual Basic 変数名は *ItemList(0,0)* に保管されている索引カウント (戻された項目 ID の数) を持つ配列変数になり、戻された項目ごとに次のような Visual Basic 配列要素を持つ構造になります。

ItemList(n,0)- 配列のサイズ

ItemList(n,1)- 項目 ID

ItemList(n,2)- 項目タイプ

--(1) 文書

--(2) フォルダー

--(?)不明

ItemList(n,3)- 索引クラス

ItemList(n,3+m) - 属性名 *m*

ItemList(n,3+m) - 属性値 *m*

FolderId

- 入力

リストするフォルダーの項目 ID。

使用の手引き

フォルダーの項目 ID はこの呼び出しの前に存在していなければなりません。この関数は、ワーク・バスケットのコンテンツをリストするためにも使用できます。

Visual Basic ソース・コード

```
Function VbVhlListFolderItemsAttr (ItemList(), FolderId)

    ' Declarations
    Dim FolderObj As Object
    Dim ContentObj As Object
    Dim ulTOCCnt, ulStart, ulEnd, ulLen, ulTotLen As Long

    ' Setup Error handler
    On Error GoTo VhlLstFldAttrError
    ulRC = 0

    ' Get the Folder Object
    Set FolderObj = VhlApplObj.ItemID(FolderId)

    ' Setup return array based on size of folder
    ulTOCCnt = FolderObj.TOCCount
    ReDim ItemList(ulTOCCnt + 1, 4)
    ItemList(0, 0) = 0

    ' Get the list of Item Objects in the Folder
    For i = 1 To ulTOCCnt
       Set ContentObj = FolderObj.GetTOCItem(i - 1)
       ItemList(i, 1) = ContentObj.ItemID
       ItemList(i, 2) = ContentObj.Type
       ItemList(i, 3) = ContentObj.Class
       ItemList(0, 0) = i
       ItemList(i, 0) = 3

       ' Get the list of Index Class attributes
       strRet = VhlApplObj.ClassKeyFieldList(ContentObj.Class, ";")
       ulTotLen = Len(strRet)
       j = 3
       ulStart = 1
       ' Add attributes to List array
       Do
          ' Each name separated by a ";"
          ulEnd = InStr(ulStart, strRet, ";")
          If (ulEnd = 0) Then
             ulEnd = ulTotLen + 1
          End If
          ulLen = ulEnd - ulStart
          AttrName = Mid$(strRet, ulStart, ulLen)

          ' Set next array variables to attribute name and value
          j = j + 1
          ReDim Preserve ItemList(i, j + 2)
          ItemList(i) = AttrName
          j = j + 1
          ItemList(i, j) = ContentObj.KeyFields(AttrName)
```

```
          ' Setup for next loop
          ulStart = ulEnd + 1
       Loop Until (ulStart >= ulTotLen)

       ' Reset total number of variables in array
       ItemList(i, 0) = j
       ' Free the current Item object
       Set ContentObj = Nothing

    Next i

VhlLstFldAttrEnd:

    ' Free the objects
    Set ContentObj = Nothing
    Set FolderObj = Nothing

    ' Set return value to error code
    VbVhlListFolderItemsAttr = ulRC

    Exit Function

VhlLstFldAttrError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlLstFldAttrEnd

End Function
```

VbVhlListIndexClassAttr (索引クラスの全属性のリスト)

形式

VbVhlListIndexClassAttr(*AttrList(), ClassName*)

目的

この関数は、指定された索引クラスのすべての属性をリストします。

パラメーター

AttrList()

- 出力

指定された索引クラス名のすべての属性名のリストが保管されている Visual Basic 変数の名前。この Visual Basic 変数名は、*AttrList(0)* に保管されている索引カウント (戻された属性の数) を持つ配列変数になります。Visual Basic 配列の形式は次のとおりです。

AttrList(0)- 属性の数

AttrList(n)- 属性名 *n*

ClassName

- 入力

すべての属性名をリストする必要がある索引クラス名。

使用の手引き

この関数は、ユーザーがアクセス権を持つ索引クラス名の属性のみをリストします。

Visual Basic ソース・コード

```
Function VbVhlListIndexClassAttr (AttrList(), ClassName)

    ' Declarations
    Dim i, ulStart, ulEnd, ulLen, ulTotLen As Long

    ' Setup Error handler
    On Error GoTo VhlClassAttrError
    ulRC = 0

    ' Get the list of Index Class attributes
    strRet = VhlApplObj.ClassKeyFieldList(ClassName, ";")
    ulTotLen = Len(strRet)

    ' Add attributes to List array
    i = 0
    ReDim AttrList(1)
    AttrList(0) = 0
    ulStart = 1
    Do
       ' Each name separated by a ";"
       ulEnd = InStr(ulStart, strRet, ";")
       If (ulEnd = 0) Then
          ulEnd = ulTotLen + 1
       End If
       ulLen = ulEnd - ulStart

       ' Set next array variable to attribute name
       i = i + 1
       ReDim Preserve AttrList(i + 1)
       AttrList(i) = Mid$(strRet, ulStart, ulLen)

       ' Setup for next loop
       ulStart = ulEnd + 1
    Loop Until (ulStart >= ulTotLen)

    ' Set total number of attributes in array
    AttrList(0) = i

VhlClassAttrEnd:

    ' Set return value to error code
    VbVhlListIndexClassAttr = ulRC

    Exit Function

VhlClassAttrError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlClassAttrEnd

End Function
```

VbVhlListIndexClasses (すべての索引クラスのリスト)

形式

VbVhlListIndexClasses(*IxClassList()*)

目的

この関数は、ユーザーがアクセスできるすべての索引クラスをリストするために使用します。

パラメーター

IxClassList()

- 出力

戻された索引クラスが保管されている Visual Basic 変数の名前。この Visual Basic 変数名は、*IxClassList(0)* に保管されている索引カウント (戻された索引クラスの数) を持つ配列変数になります。Visual Basic 配列の形式は次のとおりです。

IxClassList(0)- 索引クラスの数

IxClassList(n)- 索引クラス名 *n*

Visual Basic ソース・コード

```
Function VbVhlListIndexClasses (IxClassList())

    ' Declarations
    Dim i, ulStart, ulEnd, ulLen, ulTotLen As Long

    ' Setup Error handler
    On Error GoTo VhlClassListError
    ulRC = 0

    ' Get the list of Index Classes
    strRet = VhlApplObj.ClassList(";")
    ulTotLen = Len(strRet)

    ' Add Index classes to List array
    i = 0
    ReDim IxClassList(1)
    IxClassList(0) = 0
    ulStart = 1
    Do
       ' Each name separated by a ";"
       ulEnd = InStr(ulStart, strRet, ";")
       If (ulEnd = 0) Then
          ulEnd = ulTotLen + 1
       End If
       ulLen = ulEnd - ulStart

       ' Set next array variable to Index Class name
       i = i + 1
       ReDim Preserve IxClassList(i + 1)
       IxClassList(i) = Mid$(strRet, ulStart, ulLen)

       ' Setup for next loop
       ulStart = ulEnd + 1
    Loop Until (ulStart >= ulTotLen)
```

```
    ' Set total number of Index Classes in array
    IxClassList(0) = i

VhlClassListEnd:

    ' Set return value to error code
    VbVhlListIndexClasses = ulRC

    Exit Function

VhlClassListError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlClassListEnd

End Function
```

VbVhlListItemCC (基本オブジェクトのコンテンツ・クラスのリスト)

形式

VbVhlListItemCC(*ItemCC, ItemId, PartNum*)

目的

この関数は、指定された項目 ID の基本オブジェクトに関連したコンテンツ・クラスをリストします。

パラメーター

ItemCC

- 出力

戻されたコンテンツ・クラス名が保管されている Visual Basic 変数の名前。

ItemId - 入力

項目 ID。

PartNum

- 入力

コンテンツ・クラス情報を戻す文書の部分番号。「0」は最初の基本部分を表します。

使用の手引き

項目 ID はこの呼び出しの前に存在していなければなりません。

Visual Basic ソース・コード

```
Function VbVhlListItemCC (ItemCC, ItemId, PartNum)

    ' Declarations
    Dim ItemObj As Object
```

```
    ' Setup Error handler
    On Error GoTo VhlItemCCError
    ulRC = 0

    ' Get the Item object
    Set ItemObj = VhlApplObj.ItemID(ItemId)

    ' Copy content class of document base part
    ItemCC = ItemObj.GetPartContentClass(PartNum)

VhlItemCCEnd:

    ' Free the object
    Set ItemObj = Nothing

    ' Set return value to error code
    VbVhlListItemCC = ulRC

    Exit Function

VhlItemCCError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlItemCCEnd

End Function
```

VbVhlListItemInfo (項目の索引クラスおよび属性情報をリストする)

形式

VbVhlListItemInfo(*ItemInfo(), ItemId*)

目的

この関数は、指定された項目 ID に関する情報 (項目タイプ、索引クラス・タイプ、属性名、属性値など) をリストします。

パラメーター

ItemInfo

- 出力

項目情報が保管されている Visual Basic 変数の名前。この Visual Basic 変数名は、 *ItemInfo(0)* に保管されている索引カウント (配列変数のサイズ) を持つ配列変数になり、次のような構造になります。

ItemInfo(0)- 配列サイズ

ItemInfo(1)- 項目 ID

ItemInfo(2)- 項目タイプ

--(1) 文書

--(2) フォルダー

--(3) ワーク・バスケット

--(?)不明

ItemInfo(3)- 索引クラス

ItemInfo(3+m) - 属性名 *m*

ItemInfo(3+m) - 属性値 *m*

ItemId - 入力

項目 ID。

使用の手引き

項目 ID はこの呼び出しの前に存在していなければなりません。索引クラスと属性は、ワーク・バスケット項目には適用されません。

Visual Basic ソース・コード

```
Function VbVhlListItemInfo (ItemList(), ItemID)

    ' Declarations
    Dim ItemObj As Object
    Dim ulTOCCnt, ulStart, ulEnd, ulLen, ulTotLen As Long

    ' Setup Error handler
    On Error GoTo VhlListInfoError
    ulRC = 0

    ' Get the Item Object
    Set ItemObj = VhlApplObj.ItemID(ItemID)

    ' Get the list of Item Objects in the Folder
    ReDim ItemList(10)
    ItemList(0) = 3
    ItemList(1) = ItemObj.ItemID
    ItemList(2) = ItemObj.Type
    ItemList(3) = ItemObj.Class

    ' Workbaskets don't have attributes
    If ItemList(2) > 2 Then
       GoTo VhlListInfoEnd
    End If

    ' Get the list of Index Class attributes
    strRet = VhlApplObj.ClassKeyFieldList(ItemObj.Class, ";")
    ulTotLen = Len(strRet)
    i = 3
    ulStart = 1
    ' Add attributes to List array
    Do
       ' Each name separated by a ";"
       ulEnd = InStr(ulStart, strRet, ";")
       If (ulEnd = 0) Then
          ulEnd = ulTotLen + 1
       End If
       ulLen = ulEnd - ulStart
       AttrName = Mid$(strRet, ulStart, ulLen)

       ' Set next array variables to attribute name and value
       i = i + 1
       ReDim Preserve ItemList(i + 2)
       ItemList(i) = AttrName
       i = i + 1
       ItemList(i) = ItemObj.KeyFields(AttrName)

       ' Setup for next loop
       ulStart = ulEnd + 1
```

```
    Loop Until (ulStart >= ulTotLen)

    ' Set total number of variables in array
    ItemList(0) = i

VhlListInfoEnd:

    ' Free the objects
    Set ItemObj = Nothing

    ' Set return value to error code
    VbVhlListItemInfo = ulRC

    Exit Function

VhlListInfoError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlListInfoEnd

End Function
```

VbVhlListWBItems (ワーク・バスケット・コンテンツのリスト)

形式

VbVhlListWBItems(*ItemList(), WorkBasket*)

目的

この関数は、ワーク・バスケット (名前によって指定) に含まれている文書およびフォルダーの項目 ID をすべてリストします。

パラメーター

ItemList()

- 出力

項目 ID が保管されている Visual Basic 変数の名前。この Visual Basic 変数名は、項目の数が *ItemList(0)* に、項目 ID が *ItemList(1)* から *ItemList(n)* までに保管されている配列変数となります。

WorkBasket

- 入力

ワーク・バスケット名。

使用の手引き

ワーク・バスケット名は有効でなければなりません。

Visual Basic ソース・コード

```
Function VbVhlListWBItems (ItemList(), WBItemID)

    ' Declarations
    Dim WBObj As Object
    Dim ContentObj As Object
    Dim ulTOCCnt As Long
```

```
    ' Setup Error handler
    On Error GoTo VhlLstWBItemError
    ulRC = 0

    ' Get the WB Object
    Set WBObj = VhlApplObj.ItemID(WBItemID)

    ' Setup return array based on size of WB
    ulTOCCnt = WBObj.TOCCount
    ReDim ItemList(ulTOCCnt + 1)
    ItemList(0) = 0

    ' Get the list of Item Objects in the WB
    j = 1
    For i = 1 To ulTOCCnt
       Set ContentObj = WBObj.GetTOCItem(i - 1)
       ItemList(j) = ContentObj.ItemID
       Set ContentObj = Nothing
       ItemList(0) = j
       j = j + 1
    Next i

VhlLstWBItemEnd:

    ' Free the objects
    Set ContentObj = Nothing
    Set WBObj = Nothing

    ' Set return value to error code
    VbVhlListWBItems = ulRC

    Exit Function

VhlLstWBItemError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlLstWBItemEnd

End Function
```

VbVhlListWorkBaskets (すべてのワーク・バスケット名のリスト)

形式

VbVhlListWorkBaskets(*WkBasketList()*)

目的

この関数は、すべてのワーク・バスケット名および説明をリストするために使用します。

パラメーター

WkBasketList()

- 出力

定義されたワーク・バスケット名のリストが保管されている Visual Basic 変数の名前。この Visual Basic 変数名は、*WkBasketList(0)* に保管されてい

る索引カウント (戻されたワーク・バスケットの数) および *WkBasketList(1)* から *WkBasketList(n)* までに保管されているワーク・バスケット名を持つ配列変数になります。

Visual Basic ソース・コード

```
Function VbVhlListWorkBaskets (WBList())

    ' Declarations
    Dim i, ulStart, ulEnd, ulLen, ulTotLen As Long

    ' Setup Error handler
    On Error GoTo VhlListWBError
    ulRC = 0

    ' Get the list of WorkBaskets
    strRet = VhlApplObj.WorkBasketList(";")
    ulTotLen = Len(strRet)

    ' Add Index classes to List array
    i = 0
    ReDim WBList(1)
    WBList(0) = 0
    ulStart = 1
    Do
       ' Each name separated by a ";"
       ulEnd = InStr(ulStart, strRet, ";")
       If (ulEnd = 0) Then
          ulEnd = ulTotLen + 1
       End If
       ulLen = ulEnd - ulStart

       ' Set next array variable to Index Class name
       i = i + 1
       ReDim Preserve WBList(i + 1)
       WBList(i) = Mid$(strRet, ulStart, ulLen)

       ' Setup for next loop
       ulStart = ulEnd + 1
    Loop Until (ulStart >= ulTotLen)

    ' Set total number of Index Classes in array
    WBList(0) = i

VhlListWBEnd:

    ' Set return value to error code
    VbVhlListWorkBaskets = ulRC

    Exit Function

VhlListWBError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlListWBEnd

End Function
```

VbVhlLoadFuncs (VHLPI 関数へのアクセスの取得)

形式

VbVhlLoadFuncs()

目的

この関数は、Visual Basic 用の VHLPI 関数へのアクセス権を取得するために使用します。これにより、Visual Basic プログラムでこれらの関数を呼び出すことが可能になります。

使用の手引き

この関数を実行した後は、Visual Basic プログラムは任意の VHLPI 関数を呼び出すことができます。これらの関数へのアクセスを終了するには VbVhlDropFuncs 関数を使用します。

Visual Basic ソース・コード

```
Function VbVhlLoadFuncs ()

    ' Setup Error handler
    On Error GoTo VhlLoadError
    ulRC = 0

    ' Get the application object
    Set VhlApplObj = CreateObject("Vic.Application")

    ' Setup Global Application Objects
    Set VhlDocsObj = VhlApplObj.Documents
    Set VhlErrorObj = VhlApplObj.Error

VhlLoadEnd:

    ' Set return value to error code
    VbVhlLoadFuncs = ulRC

    Exit Function

VhlLoadError:

    ' Set return code to error code
    ulRC = Err
    Resume VhlLoadEnd

End Function
```

VbVhlLogoff (IBM Content Manager for iSeries へのアクセスの終了)

形式

VbVhlLogoff()

目的

この API は、Windows クライアントへのアクセスを終了し、Windows クライアントをクローズするために使用します。VHLPI 関数のその後の使用はすべて失敗します。

使用の手引き

この関数を実行した後は、Windows クライアントがクローズされ、 Visual Basic プログラムは VHLPI 関数を呼び出せなくなります。これらの関数へのアクセスを確立するには VbVhlLogon API を使用します。

Visual Basic ソース・コード

```
Function VbVhlLogoff ()

    ' Setup Error handler
    On Error GoTo VhlLogoffError

    ' Logoff from the system
    ulRC = 0
    VhlApplObj.Quit
    Set VhlDocsObj = Nothing
    Set VhlErrorObj = Nothing
    Set VhlApplObj = Nothing

VhlLogoffEnd:

    ' Set return value to error code
    VbVhlLogoff = ulRC

    Exit Function

VhlLogoffError:

    ' Set return code to error code
    ulRC = Err
    Resume VhlLogoffEnd

End Function
```

VbVhlLogon (IBM Content Manager for iSeries へのアクセスの取得)

形式

VbVhlLogon()

目的

この関数は、Visual Basic にログオンし、Windows クライアントの VHLPI 関数へのアクセス権を取得するために使用します。これにより、Visual Basic プログラムでこれらの関数を呼び出すことが可能になります。

使用の手引き

この関数を実行した後は、Visual Basic プログラムは任意の VHLPI 関数を呼び出すことができます。Windows クライアントをログオフしてクローズするには、VbVhlLogoff 関数を使用します。これらの関数へのアクセスを終了するだけの場合は、VbVhlDropFuncs 関数を使用します。

Visual Basic ソース・コード

```
Function VbVhlLogon (UserId, Password, LibServer)

    ' Setup Error handler
    On Error GoTo VhlLogonError
    ulRC = 0

    ' Get the application object
    Set VhlApplObj = CreateObject("Vic.Application")

    ' Set logon information
    VhlApplObj.User = UserId
    VhlApplObj.Server = LibServer
    VhlApplObj.Password = Password

    ' Display the Logon screen and Log onto the system
    ulRC = VhlApplObj.Logon
    If (ulRC = 0) Then
       ' Setup Global Application Objects
       Set VhlDocsObj = VhlApplObj.Documents
       Set VhlErrorObj = VhlApplObj.Error
    Else
       ' Release application object
       Set VhlApplObj = Nothing
    End If

VhlLogonEnd:

    ' Set return value to error code
    VbVhlLogon = ulRC

    Exit Function

VhlLogonError:

    ' Set return code to error code
    ulRC = Err
    Resume VhlLogonEnd

End Function
```

VbVhlRemoveFolderItem (フォルダーからの項目の除去)

形式

VbVhlRemoveFolderItem(*ItemId, FolderId*)

目的

この関数は、文書またはフォルダー (項目 ID によって指定) をフォルダー (フォルダーの項目 ID によって指定) から除去します。

パラメーター

ItemId - 入力

削除する文書またはフォルダーの項目 ID。

FolderId

- 入力

フォルダーの項目 ID。

使用の手引き

指定された文書またはフォルダーは物理的には削除されません。フォルダーとの関連付けがなくなるだけです。

Visual Basic ソース・コード

```
Function VbVhlRemoveFolderItem (ItemID, FolderId)

    ' Declarations
    Dim ItemObj As Object
    Dim FolderObj As Object

    ' Setup Error handler
    On Error GoTo VhlRemFolderError
    ulRC = 0

    ' Get the Folder Object
    Set FolderObj = VhlApplObj.ItemID(FolderId)

    ' Get the ItemID Object
    Set ItemObj = VhlApplObj.ItemID(ItemID)

    ' Put ItemId into Folder
    ulRC = ItemObj.RemoveFromFolder(FolderObj)

VhlRemFolderEnd:

    ' Free the objects
    Set ItemObj = Nothing
    Set FolderObj = Nothing

    ' Set return value to error code
    VbVhlRemoveFolderItem = ulRC

    Exit Function

VhlRemFolderError:

    ' Set return value to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlRemFolderEnd

End Function
```

VbVhlScanDoc (文書のスキャン)

形式

VbVhlScanDoc()

目的

この関数はスキャン機能を呼び出します。これにより、ユーザーはイメージをスキャンし、新しい文書を作成できます。ユーザーが「スキャン (Scan)」ウィンドウをクローズしたとき、作成された文書の項目 ID は**戻されません**。

使用の手引き

ユーザーはスキャン機能と対話して作業を行います。したがって、ユーザーは、スキャン機能へのコマンドを通して、文書をいつどのように作成するかを制御します。

Visual Basic ソース・コード

```
Function VbVhlScanDoc ()

    ' Setup Error handler
    On Error GoTo VhlScanDocError
    ulRC = 0

    ' Scan some documents
    VhlApplObj.OpenScan

VhlScanDocEnd:

    Exit Function

VhlScanDocError:

    ' Set return code to error code
    VbVhlScanDoc = VhlErrorObj.ReturnCode

    Resume VhlScanDocEnd

End Function
```

VbVhlSearchAdv (項目の高機能検索)

形式

VbVhlSearchAdv(*ItemList(), ClassName, Criteria, TypeFilter, WIPFilter, SuspendFilter*)

目的

この関数は、提供された検索基準と一致する項目 ID をすべてリストします。戻された項目 ID のリストは、各種のフィルター・パラメーターに提供された値に基づいてフィルター操作できます。

パラメーター

ItemList()

- 出力

項目 ID の文書リストが保管されている Visual Basic 変数の名前。この Visual Basic 変数名は、項目の数が *ItemList(0)* に、項目 ID が *ItemList(1)* から *ItemList(n)* までに保管されている配列変数となります。

ClassName
- 入力

索引クラスの名前。

基準 - 入力

検索基準。 『使用の手引き』を参照してください。

TypeFilter
- 入力

検索する項目の値のタイプ。有効な値は以下のとおりです。

- *1(SIM_DOCUMENT)*
- *2(SIM_FOLDER)*
- *other(SIM_FOLDER_DOC)*

WIPFilter
- 入力

戻される項目の Work In Progress 状況。複数の基準が必要な場合は、WIP 状況の値の *OR* を使用することができます。有効な値は以下のとおりです。

- *1(OIM_ITEMS_NOT_IN_WORKFLOW)*
- *2(OIM_CURRENT_WORKFLOW_ITEMS)*
- *4(OIM_CANCELLED_WORKFLOW_ITEMS)*
- *8(OIM_COMPLETED_WORKFLOW_ITEMS)*

SuspendFilter
- 入力

戻される項目の中断状況。有効な値は以下のとおりです。

- *1(OIM_ITEMS_NOT_SUSPENDED)*
- *2(OIM_ITEMS_SUSPENDED)*
- *other(OIM_ITEMS_ALL)*

使用の手引き

指定する索引クラス名は、この関数を使用する前に存在している必要があります。また、検索仕様の中の属性 ID もこの索引クラス用に定義されている必要があります。

検索基準の構文は、″Attribute Operator Value″ です。ここで、

- *Attribute* は属性の ID で、IBM Content Manager for iSeries の中で定義されている必要があります。この属性 ID の形式は Annn です。この nnn は属性番号です。
- *Operator* は演算を表すテキスト・ストリングです。有効な ″Operator″ 値は、EQ、==、LEQ、<=、GEQ、>=、LT、<、GT、>、NEQ、<>、IN、NOTIN、LIKE、NOTLIKE、BETWEEN、NOTBETWEEN です。

- *Value* はテキスト、数、またはワード *NULL* です。″Value″ テキストには、文字 ’%’ または文字 ’_’ を含めることもできます。’%’ は任意の数の文字に一致し、’_’ は任意の 1 文字に一致します。有効な ″Operator Value″ 検索基準の例を次に示します。
  - ″LIKE E%″
  - ″< 123″
  - ″== NULL″

システムは、検索基準を使用し、動的 SQL 照会を介してデータベース内で一致する項目 ID を見つけます。

Visual Basic ソース・コード

```
Function VbVhlSearchAdv (ItemList(), ClassName, Criteria,
                                     TypeFilter, WIPFilter, SuspendFilter)

    ' Declarations
    Dim FolderObj As Object
    Dim ContentObj As Object
    Dim ulTOCCnt, ulStart, ulEnd, ulLen, ulTotLen As Long

    ' Setup Error handler
    On Error GoTo VhlSearchAdvError
    ulRC = 0

    ' Get the search results folder
    Set FolderObj = VhlApplObj.Search(ClassName, Criteria,
                                     TypeFilter, WIPFilter, SuspendFilter)

    ' Setup return array based on size of folder
    ulTOCCnt = FolderObj.TOCCount
    ReDim ItemList(ulTOCCnt + 1)
    ItemList(0) = 0

    ' Get the list of Item Objects in the Folder
    For i = 1 To ulTOCCnt
       Set ContentObj = FolderObj.GetTOCItem(i - 1)
       ItemList(i) = ContentObj.ItemID
       Set ContentObj = Nothing
       ItemList(0) = i
    Next

VhlSearchAdvEnd:
    ' Free the objects
    Set ContentObj = Nothing
    Set FolderObj = Nothing

    ' Set return value to error code
    VbVhlSearchAdv = ulRC

    Exit Function

VhlSearchAdvError:
    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlSearchAdvEnd

End Function
```

VbVhlSearchItem (項目の検索)

形式

VbVhlSearchItem(*ItemList(), ClassName, Criteria*)

目的

この関数は、提供された検索基準と一致する属性名/値を含んでいる、指定された索引クラス名の項目 ID をすべてリストします。

パラメーター

ItemList()

- 出力

項目 ID の文書リストが保管されている Visual Basic 変数の名前。この Visual Basic 変数名は、項目の数が *ItemList(0)* に、項目 ID が *ItemList(1)* から *ItemList(n)* までに保管されている配列変数となります。

ClassName

- 入力

索引クラスの名前。

基準 - 入力

検索基準。 267 ページの『使用の手引き』を参照してください。

使用の手引き

指定する索引クラス名は、この関数を使用する前に存在している必要があります。また、検索仕様の中の属性 ID もこの索引クラス用に定義されている必要があります。

検索基準の構文は、″Attribute Operator Value″ です。ここで、

- *Attribute* は属性の ID で、定義されている必要があります。この属性 ID の形式は Annn です。この nnn は属性番号です。
- *Operator* は演算を表すテキスト・ストリングです。有効な ″Operator″ 値は、EQ、==、LEQ、<=、GEQ、>=、LT、<、GT、>、NEQ、<>、IN、NOTIN、LIKE、NOTLIKE、BETWEEN、NOTBETWEEN です。
- *Value* はテキスト、数、またはワード *NULL* です。″Value″ テキストには、文字 '%' または文字 '_' を含めることもできます。'%' は任意の数の文字に一致し、'_' は任意の 1 文字に一致します。有効な ″Operator Value″ 検索基準の例を次に示します。
  - ″LIKE E%″
  - ″< 123″
  - ″== NULL″

システムは、検索基準を使用し、動的 SQL 照会を介してデータベース内で一致する項目 ID を見つけます。

Visual Basic ソース・コード

```
Function VbVhlSearchItem (ItemList(), ClassName, Criteria)

    ' Declarations
    Dim FolderObj As Object
    Dim ContentObj As Object
    Dim ulTOCCnt, ulStart, ulEnd, ulLen, ulTotLen As Long

    ' Setup Error handler
    On Error GoTo VhlSearchError
    ulRC = 0

    ' Get the search results folder
    Set FolderObj = VhlApplObj.Search(ClassName, Criteria)

    ' Setup return array based on size of folder
    ulTOCCnt = FolderObj.TOCCount
    ReDim ItemList(ulTOCCnt + 1)
    ItemList(0) = 0

    ' Get the list of Item Objects in the Folder
    For i = 1 To ulTOCCnt
       Set ContentObj = FolderObj.GetTOCItem(i - 1)
       ItemList(i) = ContentObj.ItemID
       Set ContentObj = Nothing
       ItemList(0) = i
    Next

VhlSearchEnd:

    ' Free the objects
    Set ContentObj = Nothing
    Set FolderObj = Nothing

    ' Set return value to error code
    VbVhlSearchItem = ulRC

    Exit Function

VhlSearchError:

    ' Set return code to error code
    ulRC = VhlErrorObj.ReturnCode

    Resume VhlSearchEnd

End Function
```

第 7 章 Content Manager for iSeries プログラミング・インターフェースのサーバー API

Content Manager for iSeries クライアント API のサーバー・バージョン

Content Manager for iSeries クライアント API には、同等の機能を持つ Content Manager for iSeries サーバー API が用意されています。これらの API の一部を使用したサンプル・プログラムも、COBOL、RPG、および C 言語で利用できます。詳細については、QVI ライブラリーの QSMPSRC ソース・ファイルに含まれている各サンプル・プログラムを参照してください。また、ソース・ファイル QVIRPGCPY、QVICBLCPY、および H には、データ構造のサンプルが含まれています。ILE C/400®、ILE COBOL/400®、ILE RPG/400®、または VisualAge/400 を使用して、カスタム・モジュールを作成してください。次に、サービス・プログラム QVIAPI を使ってそれらの新しいモジュールをバインドすることにより、1 つのプログラムを作成してください。

クライアント API に比べ、サーバー・バージョンの API には以下の相違点があります。これらの詳細については、「Content Manager for iSeries アプリケーション・プログラミングのガイドとリファレンス」(SC88-4007) を参照してください。

- Content Manager for iSeries でのポインターは 16 バイトであるため、データ構造 RCSTRUCT で戻るすべてのポインターは、ulParam1 および ulParam2 ではなく pParam2 を介してアクセスされます。
- サーバー API を Content Manager for iSeries で実行すると、そのサーバー・コードは、API の呼び出し側アプリケーションと同じジョブ・スペース内で実行されます。つまり、別個のジョブとしては開始されません。
- **SimLibOpenObject** によってオープンできるのは、Content Manager for iSeries でアクセス可能なイメージ・データだけです。
- 2 つのワークステーション API Sim400SendReceive および Sim400ConvertCodepage には、同等のサーバー・バージョンはありません。**Sim400SendReceive** と **Sim400ConvertCodepage** は、ワークステーション上でのみ使用することができます。
- 任意の API の振る舞いを検査する VI400TST プログラムは、どのワークステーションの Content Manager for iSeries 上でも実行できます。

Content Manager for iSeries のサーバー専用 API

次に示す Content Manager for iSeries API は、サーバー専用です。ワークステーションについては、同様の名前をもつ API はありません。

QVISNDRCV (バッファーの送信と受信)

目的

QVISNDRV は、ワークステーションとのデータの送受信のための汎用関数です。この関数は、Content Manager for iSeries クライアントを使用して文書を表示するため

に、Content Manager for iSeries アプリケーションが使用できます。この関数には、文書ワークステーションをクローズするためのリセット・オプションも組み込まれています。

パラメーター

Communication_Type
INT- 入力/出力

使用する通信タイプ。有効な値は以下のとおりです。

0 検出。

装置記述による判別結果に従って、該当のアプリケーション用の接続が使用されます。エラーが発生しない限り、値 1 または 2 が戻ります。印刷時などに特定のワークステーション・アドレスを使用することが定義されている場合を除き、この値が使用されます。

1 APC (CPI-C)。明示的に APPC を使用する場合。

2 TCP/IP。たとえば、TCP/IP を使用する場合。

Partner_Address
CHAR[20]- 入力/出力

少なくとも 1 つの末尾ブランクを持つ、ワークステーション・アドレス。有効値は、CPI-C 用の完全修飾 LU 名または TCP/IP アドレスです。*Cmmunication_Type* が 0 に設定されている場合には、このフィールドは無視されますが、ワークステーション・アドレスがこのフィールドに戻ります。

Partner_TPName
CHAR[20] - 入力/出力

APPC のトランザクション・プログラム名。ブランクのままにした場合は、Content Manager for iSeries によってデフォルト値 EKDVICLA が設定されます。

Partner_ModeName
CHAR[10]

少なくとも 1 つの末尾ブランクを持つ、APPC のモード名。ブランクのままにした場合、モード名は #INTER となります。

Partner_PortNumber
INT - 入力/出力

TCP/IP 接続の場合は、ワークステーションのポート番号。0 にした場合のデフォルト値は 31015 です。

communication_handle
CHAR[20]

通信ハンドルが入ります。このフィールドがブランクで、バッファー・サイズがゼロ以外の場合は、該当のワークステーションに会話が割り振られるか、ワークステーションに接続するためのソケットがオープンされます。このフィールドがブランクでなく、かつバッファー・サイズが 0 の場合は、会話の割り振りが解除されるか、ソケットがクローズされます。

dllname

CHAR - 入力/出力

ワークステーションにロードされる DLL の、ヌル文字またはブランクで終了する名前。DLL の関数は、次のとおりにしてください。

```
int vi400comm (int * buffer_size, char * buffer)
```

ゼロ以外の値が戻った場合、ワークステーション・プログラムは終了します。その場合に、別の表示要求を開始できるようにするには、ユーザーはワークステーションを再始動する必要があります。

ブランクのままにした場合は、ホストで開始される表示要求をサポートするために、Content Manager for iSeries によってデフォルトの EKDVIDSP.DLL が設定されます。

host_code_page

INT - 入力

0 を指定すると、QVISNDRCV は現行のコード・ページを抽出します。バッファー内のすべてのデータは、変換可能な文字でなければなりません。変換されていないバイナリー・データを送信するには、-1 を指定します。

buffer_size

INT - 入力/出力

送信元および受信先となるバッファーのサイズを指すポインター。最大サイズは 32760 バイトです。ホストに 0 が戻った場合は、会話またはソケットがクローズされます。ゼロ以外の値が戻った場合、バッファーにはワークステーションから送信されたデータが含まれています。送信または受信が可能なサイズは 32500 バイト以内です。残りのバイト分は、制御情報に関するデータです。

buffer CHAR - 入力/出力

送信元および受信先となるバッファーを指すポインター。これは少なくとも、指定した *buffer_size* または戻されたバッファー・サイズと同じ長さにしてください。戻されたバッファー・サイズが、戻されたデータ容量よりも小さい場合、明示的なエラーは発生しませんが、呼び出し側プログラムが失敗するおそれがあります。

戻り値

この関数は、Content Manager for iSeries コードにエラーがあった場合には、整数の戻りコードを戻します。

Content Manager for iSeries を使用してホストで開始される表示要求をサポートするサンプル・コードが用意されています。このコードは、呼び出し側のアプリケーションに返信されるバッファー内に、以下の文字の戻りコードを戻します。

1 Content Manager for iSeries が開始されていませんでした。

2 ヌル・バッファーが渡されました。

3 先頭バイトが R (リセット)、D (表示) のいずれでもありませんでした。

4 項目 ID の長さが無効です。

5 項目 ID が無効です。

6 項目へのアクセス上の問題が発生しました。

7 Content Manager for iSeries エラー

使用の手引き

すべてのパラメーターは参照によって渡されます。文字変数は、ヌルまたはブランクで終了できます。

ILE C/400、ILE COBOL/400、ILE RPG/400、または VisualAge/400 を使用して、カスタム・モジュールを作成してください。次に、サービス・プログラム QVISNDRCV を使ってそれらの新しいモジュールをバインドすることにより、1 つのプログラムを作成してください。

通信用として、2 種類のワークステーション・プログラムが提供されます。APPC 通信用の EKDVICLA と、TCP/IP 通信用の EKDVICLT です。これらがデフォルトで呼び出された場合、ワークステーションのアドレスと通信タイプは自動的に判別されます。

APPC 通信の場合、プログラム EKDVICLA は事前に開始しておくこともできますし、接続マネージャーによって開始されるトランザクション・プログラムとして定義することもできます。APPC 用のパーソナル・コミュニケーションズのサポート機能を使用して、EKDVICLA をトランザクション・プログラムとして定義する場合は、「*Receive_Allocate timeout*」を 0 に設定し、「*Dynamically loaded*」、「*Queued TP*」、および「*Background process*」にチェック・マークを付けます。プログラムは、iSeries 上のいずれかのプログラムからの要求時にまだ実行されていない場合には、自動的に開始されます。タイムアウト値を 0 に設定すると、プログラム は、会話の割り振りが解除された後であってもアクティブ状態で維持されます。

TCP/IP 通信の場合、プログラム EKDVICLT はワークステーション上で事前に開始しておく必要があります。ポート番号 (31015) が受け入れ可能でない場合は、EKDVICLT の開始時に、別の値をパラメーターとして渡すことができます。

ソース・プログラムのサンプル

ご使用の QVI ライブラリーのファイル QCSRC に含まれているソース・プログラム・サンプル QVIDSPTST を参照してください。このプログラムは、サーバー上の C プログラムから QVISNDRCV を呼び出すためのプログラミング例として提供されています。このプログラムに含まれる情報、定義、および構造は、カスタム・コードの作成時に役立つよう考慮されています。

第 8 章 Content Manager for iSeries ユーザー出口

Content Manager for iSeries が提供するユーザー出口とは、ユーザー自身の処理ルーチンを指定できる、プログラム内のある特定のポイントのことをいいます。データベースにアクセスしたり、別のアプリケーションと統合することによって、一定のレベルのカスタマイズが可能な出口プログラムを作成することができます。

クライアントのユーザー出口

ここで示すユーザー出口ポイントは、Content Manager for iSeries によって呼び出されます。Windows クライアントと一緒に、以下のユーザー出口を使用してください。

AlternateSearchUserExit (代替検索ユーザー出口)

形式

SHORT AlternateSearchUserExit(*hSession, hWnd, szUserID usTypeFilter, fWipFilter, usSuspendFilter, usIndexClass, usNumCriteria, pCriteria, pItemIdResultFolder***)**

目的

AlternateSearchUserExit を使用して、クライアント・アプリケーションの検索関数をユーザー独自の検索ルーチンと置換します。出口は、検索操作の結果を検索結果フォルダーに戻します。

パラメーター

hSession

HSESSION - 入力

SimLibLogon により戻されるセッション・ハンドル。

hWnd HWND - 入力

ウィンドウのハンドル。デバイス・マネージャーはこのハンドルを使用して、エンド・ユーザー・インターフェースの操作 (エラー・メッセージの表示など) が行われるウィンドウを識別します。

pszUserID

PSZ - 入力

検索結果を受け取るユーザーのユーザー ID が入っている、0 で終わる文字ストリング。このパラメーターは大文字小文字の区別は行いません。

usTypeFilter

USHORT - 入力

検索する項目のタイプ。有効な値は、次のとおりです。

SIM_DOCUMENT
項目が文書であることを示します。

SIM_FOLDER
項目がフォルダーであることを示します。

SIM_FOLDER_DOC
項目がフォルダーまたは文書であることを示します。

fWipFilter

BITS - 入力

検索する項目の処理中作業状況。有効な値は次のとおりです。ビット包含 OR 演算子 (|) を使用すれば、値を結合することができます。

OIM_ITEMS_NOT_IN_WORKFLOW
ワークフローにない項目を検索します。

OIM_CURRENT_WORKFLOW_ITEMS
ワークフローにある項目を検索します。

OIM_CANCELLED_WORKFLOW_ITEMS
ワークフローから除去された項目を検索します。

OIM_COMPLETED_WORKFLOW_ITEMS
ワークフローを完了した項目を検索します。

OIM_ALL
オブジェクトの処理中作業状況に関係なく検索します。この値は他の値と結合しないでください。他の値をすべて使用するのと同じことになってしまいます。

usSuspendFilter

USHORT - 入力

検索する項目の中断状況。有効な値は、次のとおりです。

OIM_ITEMS_SUSPENDED
中断されている項目を検索します。

OIM_ITEMS_NOT_SUSPENDED
中断されていない項目を検索します。

OIM_ALL
オブジェクトの中断状況に関係なく検索します。この値は他の値と結合しないでください。他の値をすべて使用するのと同じことになってしまいます。

usIndexClass

USHORT - 入力

検索結果のために作成するフォルダーの索引クラスに対する、索引クラス ID。作成されたフォルダーに割り当てる索引クラスに、必須属性がないようにしてください。もしあった場合、検索ファイルとフォルダーは作成されません。

作成するフォルダーに索引クラスを割り当てない場合は、このパラメーターに値 0 を指定してください。

fMemListRequest パラメーターの値が TRUE であるか、 *usStatDyn* パラメーターの値が SIM_SEARCH_BUILD_ONLY である場合、 IBM Content Manager for iSeries はこの値を無視します。

usNumCriteria

USHORT - 入力

pCriteria 配列内のエレメントの数。

pCriteria

PLIBSEARCHCRITERIASTRUCT - 入力

検索する各視点の検索基準を指定する配列を指すポインター。このポインターが指す配列には、少なくとも 1 つのエレメントが必要です。

pItemIdResultFolder

PITEMID - 入力

検索結果フォルダーを指すポインター。

戻り値

この出口から SIM_RC_OK が戻され、検索操作が正常終了したことが示されます。この他の戻り値は、異常終了を示し、エラーとしてログに記録されます。

正常終了すると、この関数は *ItemidResultFolder* 出力パラメーターの値で検索結果フォルダーを識別します。

注釈

代替検索ユーザー出口ルーチンは、ビュー・レベルで機能します。基本検索を実行する際、特定のビューを検索すると、クライアント・アプリケーション・プログラムはそのビューの出口をロードします。すべてのビューを検索すると、クライアント・アプリケーション・プログラムは NOINDEX クラスの基本ビューの出口をロードします。拡張検索の場合、クライアント・アプリケーション・プログラムは NOINDEX クラスの基本ビューの出口をロードします。

ChangeSMSUserExit (システム管理ストレージ変更ユーザー出口)

形式

SHORT ChangeSMSUserExit(*hwnd, pExitStruct, pfContinue***)**

目的

このユーザー出口ルーチンは、ライブラリー・オブジェクト・ウィンドウがクローズされる前に索引クラスの項目を変更すると呼び出されます。この出口は項目の ITEMID を受け取り、デフォルト処理を継続するかどうかを示すフラグを戻します。デフォルトの処理では、オブジェクト・サーバーおよび項目の新しい索引クラスで定義されたコレクション情報を使用して、各項目のパーツについて **SimLibChangeObjSMS** が呼び出されます。

システム管理プログラムを使用して、このユーザー出口ルーチンを、索引クラスの設定ノートブックに指定します。「*システム管理ガイド*」を参照してください。

パラメーター

hwnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用してメッセージを表示し、アプリケーション・ウィンドウに関連付けることができます。

pExitStruct

PUSEREXITSTRUCT - 入力

オープン文書のユーザー定義属性フィールドおよび他の関連情報が *pExitStruct* パラメーターに渡されます。

pfContinue

PBOOL - 出力

継続フラグを指すポインター。この値を TRUE に設定して、デフォルト処理を継続します。

内部表記

USEREXITSTRUCT:

```
typedef  struct
{
HSESSION
        hSession;
ITEMID
        uidItem;
USHORT
        itemidWorkflowId;
BOOL  fIsUnindexed;
USHORT
        hOrigClass;
USHORT
        hClass;
CHAR  szUserId[LST_USERID_LEN+1];
CHAR  szUserHandle[LST_USERID_LEN+1];
USHORT
        usAccessLevel;
SHORT
        sFields;
FIELDVALUE *
        pFields;
} USEREXITSTRUCT;

typedef  USEREXITSTRUCT  *  PUSEREXITSTRUCT
```

値の意味を以下に説明します。

hSession

SimLibLogon により戻されるセッション・ハンドル。

uidItem
変更される現行文書またはフォルダーの ITEMID。

itemidWorkflowId
変更されるオープン済みの文書またはフォルダーのワークフロー ID。オブジェクトがワークフローにない場合、この値は NULL です。

fIsUnindexed
オブジェクトが、システム内で索引付けされていない新規文書の場合には、この値は TRUE です。

hOrigClass
オープン済み文書またはフォルダーの、元のクラス ID。

hClass オープン済み文書またはフォルダーの、現行クラス ID。

szUserId[LST_USERID_LEN+1]
文書またはフォルダーを保管しているユーザーのユーザー ID。

szUserHandle[LST_USERID_LEN+1]
このパラメーターは予約済みです。

usAccessLevel
この文書またはフォルダーに対してユーザーが持っているアクセス権。有効な値は次のとおりです。
ユーザーが UPDATE モードでこのオブジェクトをオープンする場合は、 UX_PRIV_WRITE です。

sFields *pFields* パラメーターで出口に渡されるフィールドの数。

pFields FIELDVALUE データ構造の配列を指すポインター。この文書またはフォルダーのユーザー定義属性の構成および内容が、これらのデータ構造で出口に渡されます。

FIELDVALUE:

```
typedef struct
{
USHORT
        usFieldId;
USHORT
        usDataType;
USHORT
        usMaxLength;
BOOL    fIsReq;
PSZ     pBuffer;
} FIELDVALUE;

typedef FIELDVALUE * PFIELDVALUE
```

値の意味を以下に説明します。

usFieldId
ユーザー定義属性 ID。

usDataType
usFieldId パラメーターの属性の IBM Content Manager for iSeries データ型。これはデータ型を示すのと同等の数字です。

usMaxLength
「索引形式 (Index Form)」ウィンドウに表示される *pBuffer* パラメーターのうち、 NULL 終止符を除いた最大のバイト数。

fIsReq フィールドが必要な場合、この値は TRUE です。

pBuffer ASCIIZ 表示形式の属性の現行値。バッファー長は *usMaxLength* パラメーターの値に 1(NULL 終止符の分) を足した値です。

結果
SUCCESS の場合はゼロの SHORT が戻されます。ゼロ以外の値が戻されると、デフォルト処理が実行されます。

呼び出しが成功すると、 *pfContinue* パラメーターに戻される値が検査されます。

注釈
出口ルーチンは、渡されているバッファーを解放することはできません。出口に送信された項目はすべて読み取り専用のコピーです。この出口では、これらのデータ構造を修正する必要があります。

このユーザー出口ルーチンが呼び出されると、索引形式はクローズします。

クラスにレコード保管ユーザー出口ルーチンと SMS 変更ユーザー出口ルーチンの両方が指定されている場合、レコード保管ユーザー出口ルーチンが先に呼び出されます。

DetNextWBUserExit (次ワーク・バスケット判別ユーザー出口)

形式

SHORT DetNextWBUserExit(*hwnd, usOperation, sNumberofITEMIDs, pListofITEMIDs, pExitStruct, pNextWorkBasketITEMID, pfComplete, pfContinue***)**

目的
クライアント・アプリケーション・プログラムは、 IBM Content Manager for iSeries 内の 3 つの関数のうちの 1 つからこのユーザー出口ルーチンを呼び出します。特定の索引クラスに関連しているこの出口は、このクラスの項目を別のワーク・バスケットに経路指定し、ワークフローの項目を開始、またはそのワークフローを変更します。

項目の索引クラスがこのユーザー出口ルーチンを定義している場合には、ユーザーが「処理 (Process)」メニューの「経路指定先 (Route To)」オプションを選択すると必ず、クライアント・アプリケーション・プログラムはこの出口を呼び出します。デフォルトでは、IBM Content Manager for iSeries は項目がワークフローにあるか

どうかを判別します。あった場合、ワークフロー内の次のワーク・バスケットを判別します。システムは、結果のダイアログ・ボックスでこのワーク・バスケットを選択します。

クライアント・アプリケーション・プログラムは、項目がワークフローにあるかどうかにかかわらず、「経路指定先 (Route To)」ダイアログ・ボックスを表示する前にこのユーザー出口ルーチンを呼び出します。ユーザー出口ルーチンがワーク・バスケット ITEMID を戻した場合は、ワーク・バスケットは「経路指定先 (Route To)」ダイアログ・ボックスで選択されたものとして表示されます。項目を経路指定する、異なるワーク・バスケットを選択することもできます。ユーザー出口ルーチンは要求された処理はいかなるものも実行でき、また経路操作を継続しないよう IBM Content Manager for iSeries に通知することもできます。この場合、「経路指定先 (Route To)」ダイアログ・ボックスは表示されません。

クライアント・アプリケーションはまた、ユーザーが「処理 (Process)」メニューでワークフロー開始オプションまたはワークフロー変更オプションを選択すると、このユーザー出口ルーチンを呼び出します。ワークフロー開始オプションのデフォルト処理には、ワークフローの最初のワーク・バスケットへの項目の経路指定が組み込まれています。ワークフロー変更アクションには、最初のワーク・バスケットへの項目の経路指定を任意で行うことができます。

ワークフロー操作が自動的に行われている間は、次ワーク・バスケット判別ユーザー出口は呼び出されません。

クライアント・アプリケーション・プログラムは、項目の実際の経路指定より先にこのユーザー出口ルーチンを呼び出します。システムは、指定されたワーク・バスケットにこの項目を経路指定します。この場合、有効なワーク・バスケットが必ず戻されます。なぜならワークフローの項目は、ワーク・バスケットがワークフローの一部でない場合も、常にワーク・バスケットにあるからです。

システム管理プログラムを使用して、このユーザー出口ルーチンを、索引クラス設定ノートブックの「次のワーク・バスケット (Next workbasket)」フィールドに指定します。「*IBM Content Manager for iSeries システム管理ガイド*」を参照してください。

パラメーター

hwnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用して、メッセージを表示し、アプリケーション・ウィンドウと関連付けることができます。

usOperation

USHORT - 入力

ユーザー出口ルーチンを呼び出した操作を示す値。値は以下のうちのいずれかです。

- UX_ROUTE
- UX_START_WORKFLOW
- UX_CHANGE_WORKFLOW

sNumberofITEMIDs

USHORT - 入力

pListofITEMIDs パラメーターが指すリストにある ITEMID の数を指定する値。この数値が 1 より大きい場合は、フォルダーまたはワーク・バスケットの目録から複数のオブジェクトを選択してください。

pListofITEMIDs

PITEMID - 入力

ユーザーが経路指定しようとしている文書およびフォルダーの ITEMID のリストを指すポインター。

pExitStruct

PUSEREXITSTRUCT - 入力

経路指定された文書またはフォルダーが、出口が呼び出されたときにオープンされていると、オブジェクトのユーザー定義属性および他の関係する情報が、 *pExitStruct* 構造に渡されます。データ構造内の値には、「索引形式 (Index Form)」ウィンドウのクラスおよび属性に加えられた変更が含まれています。

経路指定されたオブジェクトがオープンされていない場合は、*pListofITEMIDs* パラメーターは、「目次 (Table of Contents)」ウィンドウからユーザーが選択した 1 つ以上の ITEMIDS のリストを指します。*pExitStruct* の値は、現行値を含む *szUserId* パラメーターを除いて NULL です。

pNextWorkBasketITEMID

PITEMID - 入出力

最初は、IBM Content Manager for iSeries が推奨する次のワーク・バスケットの ITEMID を指すポインターが入っています。経路指定されている文書またはフォルダーがワークフロー内にない場合、最初の ITEMID 値にはゼロが含まれています。ユーザー出口ルーチンがシステム内の有効なワーク・バスケットの項目 ID を戻す場合にのみ、この値を置き換えてください。

pfComplete

PBOOL - 入出力

IBM Content Manager for iSeries が、このオブジェクトを、ユーザー出口ルーチンが制御を戻すときにワークフローを完了したとマークするよう推奨している場合、このパラメーターを TRUE に設定してください。文書またはフォルダーが完了したとマークされると、システムは自動的にその文書またはフォルダーをワーク・バスケットから除去します。

推奨: 完了したとマークすることのできないワークフロー・オブジェクトをユーザーが複数選択する場合は、このパラメーターを TRUE に設定しないでください。

pfContinue

PBOOL - 出力

このパラメーターを FALSE に設定してアクションへの経路を取り消してください。この値を使用すると、ユーザーが推奨値を指定変更しなくてもすべての経路指定をユーザー出口ルーチンが実行することができます。このフラ

グを FALSE に設定すると、「経路指定先 (Route To)」ダイアログ・ボックスは表示されません。ワークフローの保管、開始、またはワークフロー操作の変更が行われている間にユーザー出口ルーチンが呼び出されると、このパラメーターはシステムに無視されます。

内部表記

USEREXITSTRUCT::

```
typedef struct
{
HSESSION
        hSession;
ITEMID
        uidItem;
ITEMID
        itemidWorkflowid;
BOOL  fIsUnindexed;
USHORT
        hOrigClass;
USHORT
        hClass;
CHAR  szUserId[LST_USERID_LEN+1];
CHAR  szUserHandle[LST_USERID_LEN+1];
USHORT
        usAccessLevel;
SHORT
        sFields;
FIELDVALUE *
        pFields;
} USEREXITSTRUCT;

typedef USEREXITSTRUCT * PUSEREXITSTRUCT
```

値の意味を以下に説明します。

uidItem

項目が 1 つだけ経路指定されている場合、ユーザーが経路指定する現行文書またはフォルダーの項目 ID。2 つ以上の項目を経路指定する場合は、値は NULL です。ユーザーがこのオブジェクトをオープンしない場合は、この値は NULL です。

itemidWorkflowId

ワークフロー開始アクションまたはワークフロー変更アクションの間に呼び出された場合、この値はユーザーが経路指定する文書またはフォルダーのワークフロー ID になります。単一の文書またはフォルダーのアクションへの経路を選択した場合も、値はワークフロー ID になります。2 つ以上の文書またはフォルダーについて「経路指定先 (Route to)」オプションを選択すると、値は NULL になります。

flsUnindexed

この値は常に NULL です。

hOrigClass
この値は常に NULL です。

hClass この値は常に NULL です。

szUserId[LST_USERID_LEN+1]
値は、文書またはフォルダーを経路指定したユーザーのユーザー ID です。

szUserHandle[LST_USERID_LEN+1]
このパラメーターは予約済みです。

usAccessLevel
この値は常に NULL です。

sFields この値は常に NULL です。

pFields この値は常に NULL です。

FIELDVALUE:

```
typedef struct
{
USHORT
        usFieldId;
USHORT
        usDataType;
USHORT
        usMaxLength;
BOOL    fIsReq;
PSZ     pBuffer;
} FIELDVALUE;

typedef FIELDVALUE * PFIELDVALUE
```

値の意味を以下に説明します。

usFieldId
ユーザー定義属性 ID。

usDataType
usFieldId パラメーターの属性の IBM Content Manager for iSeries データ型。これはデータ型を示すのと同等の数字です。

usMaxLength
「索引形式 (Index Form)」ウィンドウに表示される *pBuffer* パラメーターのうち、 NULL 終止符を除いた最大のバイト数。

fIsReq フィールドが必要な場合、この値は TRUE です。

pBuffer ASCIIZ 表示形式の属性の現行値。バッファー長は *usMaxLength* パラメーターの値に 1 (NULL 終止符の分) を足した値です。

結果

SUCCESS の場合は、ゼロの SHORT の値が戻されます。その他の値はエラーと見なされ、エラー・メッセージが表示されます。

出口が正常に終了すると、*pfComplete* パラメーターがチェックされます。このパラメーターが TRUE に設定されると、IBM Content Manager for iSeries は、選択したオブジェクトにこのワークフローが完了したことをマークするよう推奨するメッセージ・ボックスを表示します。オブジェクトがワークフローにないと、このパラメーターは無視されます。

pfComplete パラメーターが TRUE に設定されていないと、 *pNextWorkBasketITEMID* パラメーターの値は、選択されたオブジェクトへの推奨宛先として使用されます。この値は有効な IBM Content Manager for iSeries ワーク・バスケット ITEMID を指しているはずです。次のワーク・バスケットまたは完了のどちらの場合でも、これらの勧告を指定変更することができます。経路指定操作の間および *pfContinue* が FALSE である場合にユーザー出口ルーチンを呼び出すと、「経路指定先 (Route To)」ダイアログ・ボックスは表示されません。

注釈

出口ルーチンは、渡されているバッファーを解放することはできません。出口に送信された項目はすべて読み取り専用のコピーです。このようなデータ構造は、この出口で修正しないでください。*pfContinue* パラメーターが FALSE に設定されていない限り、ワークフロー状況の変更、データ構造にリストされているオブジェクトのワーク・バスケットの変更、あるいはこのユーザー出口ルーチンのパラメーターの変更を行う OIM 関数呼び出しを実行しないでください。

「処理 (Process)」メニューの「経路指定先 (Route To)」オプションの実行中にユーザー出口ルーチンを呼び出すと、 USEREXITSTRUCT データ構造のパラメーターは、 *szUserId* パラメーターを除いてすべて NULL になります。「処理 (Process)」メニューのワークフロー開始アクションまたはワークフロー変更アクションの実行中にユーザー出口ルーチンを呼び出すと、 USEREXITSTRUCT データ構造のパラメーターは、 *itemidWorkflowId* パラメーターおよび *szUserId* パラメーターを除いてすべて NULL になります。このような場合、通常はユーザー定義属性の詳細が入っている FIELDVALUE データ構造は、ユーザー出口ルーチンに渡されません。この出口を処理するのに、ユーザー定義属性に関する情報が必要な場合は、適切な OIM 関数呼び出しを使用して、必要なデータを取得してください。以下の関数呼び出しを参照してください。

- SimLibGetAttrInfo
- SimLibGetClassInfo
- SimLibGetItemInfo
- SimLibGetItemType
- SimLibItemSnapshot
- SimLibOpenItem
- SimLibReadItemAttr

目次から複数の項目を選択して経路指定すると、リストの最初の項目のクラスに DetNextWBUserExit があるかどうかがチェックされます。このユーザー出口ルーチンを最初のクラスに指定すると、クラスに関係なく、選択された項目すべてのためにこのユーザー出口ルーチンが呼び出されます。最初の項目のクラスにユーザー出口ルーチンを指定しないと、ユーザー出口ルーチンは呼び出されません。

DetermineWorkflowUserExit (ワークフロー判別ユーザー出口)

形式

SHORT DetWorkflowUserExit(*hwnd, puidItem, pExitStruct, puidWorkflow, puidWorkbasket***)**

目的

保管時にワークフロー内の項目を自動的に開始するように定義された索引クラスを持つ文書またはフォルダーをユーザーが保管すると、クライアント・アプリケーション・プログラムはこのユーザー出口ルーチンを呼び出します。クライアント・アプリケーション・プログラムがこのユーザー出口ルーチンを呼び出すのは、これらの項目が今までワークフローになかった場合のみです。このユーザー出口ルーチンは特定の索引クラスに指定されていますが、同じユーザー出口ルーチンを複数の索引クラスに使用することができます。

IBM Content Manager for iSeries は、システム管理者が指定したとおりに、索引クラスのデフォルトのワークフローを自動的にユーザー出口ルーチンに提供します。ユーザー出口ルーチンは、異なるワークフロー、またはデフォルト・ワークフローで項目が開始されるように、あるいはどのワークフローでも項目が開始されないように指定することができます。

また、このユーザー出口ルーチンでは、項目が経路指定されるワーク・バスケットを任意で指定することもできます。ワークフローに項目が入るように指定した場合、項目はワーク・バスケットの中に入っていなければなりません。そのワーク・バスケットが、項目が存在するワークフローの中にない場合であっても同様です。ユーザー出口ルーチンで明示的にワーク・バスケットを指定しないと、 IBM Content Manager for iSeries は項目をワークフローの最初のワーク・バスケットに経路指定します。

システム管理プログラムを使用して、このユーザー出口ルーチンを、索引クラス設定ノートブックの「自動ワークフロー (Automatic workflow)」フィールドに指定します。「*システム管理ガイド*」(SC88-4004) を参照してください。

パラメーター

hwnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用してメッセージを表示し、アプリケーション・ウィンドウに関連付けることができます。

puidItem

PITEMID - 入力

保管される項目の ITEMID を指すポインター。

pExitStruct

PUSEREXITSTRUCT - 入力

文書またはフォルダーのユーザー定義属性およびその他の関連情報が、*pExitStruct* データ構造に渡されます。

puidWorkflow

PITEM - 入出力

項目が開始されるワークフロー項目 ID を指すポインター。ユーザー出口ルーチンへの入力として提供されるワークフロー ID は、システム管理者が指定した、クラスのデフォルトのワークフローです。

ユーザー出口ルーチンは、以下のいずれかにワークフロー ID を設定します。

変更なし
デフォルトのワークフローを使用します。

ワークフロー項目 ID
開始される項目をここに定義します。

null ID (少なくとも UID null の最初の文字は設定する)
自動ワークフロー処理を取り消します。項目はワークフローでは開始されません。

puidWorkBasket

PITEMID - 出力

ワークフローで開始されたあと、保管される項目が経路指定されるワーク・バスケット ITEMID を指すポインター。このパラメーターは、出口が呼び出されたときに NULL ITEMID を指します。項目が、ワークフローの最初のワーク・バスケット以外のワーク・バスケットに経路指定されると、ユーザー出口ルーチンは、このパラメーターに有効なワーク・バスケットの項目 ID を設定します。ユーザー出口が戻ったときにこのパラメーターがまだ NULL ITEMID である場合、 IBM Content Manager for iSeries は項目をワークフローの最初のワーク・バスケットに経路指定します。

内部表記

USEREXITSTRUCT:

typedef struct
{
HSESSION
hSession
ITEMID
uidItem;
USHORT
itemidWorkflowId;
BOOL *fIsUnindexed;*
USHORT
hOrigClass;
USHORT
hClass;
CHAR *szUserId[LST_USERID_LEN+1];*
CHAR *szUserHandle[LST_USERID_LEN+1];*
USHORT
usAccessLevel;

SHORT
sFields;
FIELDVALUE *
pFields;
} USEREXITSTRUCT;

typedef USEREXITSTRUCT * PUSEREXITSTRUCT

値の意味を以下に説明します。

hSession
SimLibLogon により戻されるセッション・ハンドル。

uidItem
保管される現行文書またはフォルダーの ITEMID。

itemidWorkflowId
このパラメーターは常に NULL です。

fIsUnindexed
オブジェクトが、システム内で索引付けされていない新規文書の場合には、この値は TRUE です。

hOrigClass
オープン済み文書またはフォルダーの、元のクラス ID。

hClass オープン済み文書またはフォルダーの、現行クラス ID。ユーザーが新しい索引クラスを指定しない限り、この値は *hOrigClass* パラメーターと同じです。

szUserId[LST_USERID_LEN+1]
文書またはフォルダーを保管しているユーザーのユーザー ID。

szUserHandle[LST_USERID_LEN+1]
このパラメーターは予約済みです。

usAccessLevel
この文書またはフォルダーに対してユーザーが持っているアクセス権。このユーザー出口に有効な値は以下のとおりです。
ユーザーが UPDATE モードでこのオブジェクトをオープンする場合は、 UX_PRIV_WRITE です。

sFields *pFields* パラメーターで出口に渡されるフィールドの数。

pFields FIELDVALUE データ構造の配列を指すポインター。この文書またはフォルダーのユーザー定義属性の構成および内容が、これらのデータ構造で出口に渡されます。

FIELDVALUE:
typedef struct
{
USHORT
usFieldId;
USHORT
usDataType;

```
    USHORT
            usMaxLength;
    BOOL    fIsReq;
    PSZ     pBuffer;
    } FIELDVALUE;

typedef FIELDVALUE * PFIELDVALUE
```

値の意味を以下に説明します。

usFieldId
: ユーザー定義属性 ID。

usDataType
: *usFieldId* パラメーターの属性の IBM Content Manager for iSeries データ型。これはデータ型を示すのと同等の数字です。これらの数字の定義ステートメントと内容の要件については、 frnpfi.h ヘッダー・ファイルのセクション「Attribute types」を参照してください。

usMaxLength
: 「索引形式 (Index Form)」ウィンドウに表示される *pBuffer* パラメーターのうち、 NULL 終止符を除いた最大のバイト数。

fIsReq フィールドが必要な場合、この値は TRUE です。このパラメーターが TRUE に設定され、この FIELDVALUE データ構造が出口で修正されると、 *pBuffer* の値は NULL に変更できません。

pBuffer ASCIIZ 表示形式の属性の現行値。バッファー長は *usMaxLength* パラメーターの値に 1 (NULL 終止符の分) を足した値です。

結果

このユーザー出口ルーチンは、タイプ SHORT の値を戻します。ユーザー出口ルーチンが正常に終了すると、値ゼロが戻されます。その他の値が戻された場合は、項目はワークフローで開始されず、エラー・メッセージが表示されます。

ユーザー出口ルーチンが正常に終了すると、 *puidWorkflow* パラメーターで指定されたワークフローで項目が開始されます。このパラメーターが null のワークフロー ID を指定すると、項目はワークフローで開始されず、自動ワークフロー処理が取り消されます。ユーザー出口ルーチンが、ワーク・バスケットに項目を経路指定するよう指定した場合は、項目はワークフローで開始されたあと、IBM Content Manager for iSeries によって、そのワーク・バスケットに経路指定されます。ユーザー出口がワーク・バスケットを指定していないと、項目はワークフローの最初のワーク・バスケットに経路指定されます。

注釈

このユーザー出口ルーチンは、渡されるバッファーの修正または解放を行うことはできません。出口がワークフロー ID に NULL を戻してワークフロー処理を取り消さない限り、このユーザー出口ルーチンで、**lp2StartWorkFlow** 関数を使用してこの項目をワークフローで開始しないでください。

この出口は、索引値の修正または項目のクラスの変更が行われたあとに、文書またはフォルダーが保管されると呼び出されます。索引値が修正されないか、項目の保

管時に索引形式がオープンされないと、呼び出されません。この出口は、レコード保管ユーザー出口ルーチンが呼び出されたあとに呼び出されます。

自動ワークフロー処理は、保管される項目がその時点およびそれ以前にワークフローにない場合にのみ実行されます。このためこのユーザー出口ルーチンは、保管される項目がワークフローに一度もなかった場合にのみ呼び出されます。

このユーザー出口ルーチンを指定する索引クラスを持つフォルダーが自動ファイリングの間に作成されると、このユーザー出口ルーチンが呼び出されます。

ユーザー出口ルーチンに渡される索引値は表示形式です。データがデータベースに格納される、内部形式ではありません。

GetAttributeValueList (属性値リストの取得)

形式

INT_cdecl GetAttributeValueList(*hSession, nClassView, nAttrID, reserved, pControlType, pSortOption, ListValues, pNumValues, nMaxValueLen***)**

目的

このユーザー出口ルーチンを使用すると、「索引を編集 (Edit Index)」ウィンドウを拡張して、組み合わせリスト・ボックスを組み込むことができます。リストに表示する実際の値は、このユーザー出口ルーチンによって戻されます。このユーザー出口ルーチンは、 `frnwueal.dll` という名前のダイナミック・リンク・ライブラリー (DLL) に入れておく必要があります。このユーザー出口ルーチンは、索引クラスのそれぞれの属性 ID ごとに呼び出されます。戻される情報は以下のとおりです。

- 制御のタイプについての情報 (入力フィールド、入力フィールド付きの組み合わせリスト・ボックス、入力フィールドなしの組み合わせリスト・ボックス)。
- 順序付けオプション (リストをソートするか、配列にリストされているとおりに表示するか)。
- 表示するリストの値。

%FRNROOT%¥SAMPLES ディレクトリーには、このユーザー出口のサンプルがあります。

パラメーター

hSession

HSESSION - 入力

SimLibLogon により戻されるセッション・ハンドル。

nClassView

INT - 入力

ふさわしい制御タイプを判別するために以下の *nAttrID* フィールドが検査される索引クラス・ビュー。

nAttrID

INT - 入力

ふさわしい制御タイプを判別するために検査されるキー・フィールド。

reserved

PVOID

将来の利用のために予約済みのパラメーター。現在は NULL に設定されています。

pControlType

INT * - 出力

以下のいずれかの値を戻します。

- 0 - 標準の入力フィールドの場合
- 1 - テキスト入力が可能なリスト・ボックスの場合
- 2 - 静的リスト・ボックスの場合

pSortOption

INT * - 出力

以下のいずれかの値を戻します。

- 0 - 組み合わせ値の順序を戻された順序のままにする
- 1 - リストをアルファベット順にソートする

ListValues

PPSZ - 出力

それぞれがゼロで終了する 1 つのストリング (リスト・ボックスに表示される値の 1 つを表している) を指す文字ポインターの配列。フィールドが標準編集フィールド (*pControlType=0) である場合、この配列のサイズは 1 になります。これは、以下の GetValueListLength が **pNumValues* で 1 を戻すことも意味します。

制限: 指定する値の数は、以下の pNumValues で指定する数を超えてはなりません。

pNumValues

INT * - 入出力

入力では、GetValueListLength() 関数が *pNumValues* に戻す値の数です。この値をそのまま使用することもできますし、実際に使用される値がもっと少ないのであれば、値を小さくすることもできます。この値を増やす必要はありません。

nMaxValueLen

INT - 入力

GetValueListLength() で *pMaxValueLen* に戻される値。

結果/戻り値

成功の場合は 0。エラーの場合は非ゼロです。

注釈

このユーザー出口ルーチンは、「索引の編集 (Edit Index)」ウィンドウのすべてのフィールドごとに呼び出されるので、実行速度は速くなければなりません。

GetValueListLength (値リストの長さの取得)

形式

INT_cdecl GetValueListLength(*hSession, nClassView, nAttrID, reserved, pNumValues, pMaxValueLen***)**

目的

このユーザー出口ルーチンは、指定した索引クラスから、指定した属性 ID の値の数と値の最大長を戻します。クライアントは、この関数をすべての索引クラスのすべてのフィールドごとに呼び出し、属性用の値のリストがあるかどうか、またもしあればそれに割り振るスペースの量を判別します。このユーザー出口ルーチンは、`frnwueal.dll` という名前の DLL に入れておく必要があります。

%FRNROOT%¥SAMPLES ディレクトリーには、このユーザー出口ルーチンのサンプルがあります。

パラメーター

hSession

HSESSION - 入力

SimLibLogon により戻されるセッション・ハンドル。

nClassView

INT - 入力

ふさわしい制御タイプを判別するために以下の *nAttrID* フィールドが検査される索引クラス・ビュー。

nAttrID

INT - 入力

ふさわしい制御タイプを判別するために検査される属性。

pNumValues

INT * - 出力

デフォルトは 0 (*pNumValues=0) です。属性が複数の値を持つ場合は、値の数に設定します。

pMaxValueLen

INT * - 出力

デフォルトは 0 (*pMaxValueLen=0) です。属性が複数の値を持つ場合は、すべての値の最大長に設定します。

結果/戻り値

成功の場合は 0。エラーの場合は非ゼロです。

注釈

この関数は、「索引の編集 (Edit Index)」ウィンドウのすべてのフィールドごとに呼び出されるので、実行速度が速くなるようにしてください。

OverloadTriggerUserExit (過負荷トリガー・ユーザー出口)

形式

SHORT OverloadTriggerUserExit(*hwnd, usOperation, usNumberofITEMIDs, usIndex, pListofITEMIDs, pExitStruct, pWorkBasketITEMID, pNewWorkBasketITEMID***)**

目的

過負荷条件に達したワーク・バスケットに文書またはフォルダーが追加されると、中断基準を満たしたために追加される場合を除き、毎回このユーザー出口ルーチンが呼び出されます。ワーク・バスケットへの項目の追加は、「処理 (Process)」メニューから「経路指定先 (Route to)」オプションを選択し、次にシステム内の利用可能なワーク・バスケットのリストから宛先を選択することによって行います。新しいクラスが自動的に項目をワークフローに割り当てるよう指定しても、ワーク・バスケットに項目が追加されます。このユーザー出口ルーチンは、「ワークフローの開始 (Start workflow)」または「ワークフローの変更 (Change workflow)」操作の間や、スキャンまたはインポート操作の間に呼び出すこともできます。

中断基準は以下のとおりです。

- 満了時刻チェック・ユーティリティーにより検出されたタイムアウト。
- **SimLibAddFolderItem** 関数を使用したフォルダーへの項目の追加。ユーザー・インターフェースを介して項目をフォルダーへ追加すると、このユーザー出口ルーチンが起動されます。

過負荷トリガーは、システム運用管理プログラムで指定された、ワーク・バスケットに許可された項目の最大数です。過負荷条件がワーク・バスケットに起動されると、ユーザー出口ルーチンが処理されます。

デフォルトでは、オーバーフロー条件が起こったことを知らせるメッセージが IBM Content Manager for iSeries によって表示され、ユーザーは、経路を取り消すか、別のワーク・バスケットを宛先として選択するか、あるいは項目を元のワーク・バスケットに強制的に入れることができます。このユーザー出口ルーチンを使用して、デフォルトの IBM Content Manager for iSeries 処理を置換することもできます。バックアップとして使用でき、 IBM Content Manager for iSeries の代わりになるワーク・バスケットの ITEMID を戻す代替ワーク・バスケットを指定することもできます。

システム管理プログラムを使用して、このユーザー出口ルーチンを、ワーク・バスケット設定ノートブックに指定します。「*システム管理ガイド*」(SC88-4004) を参照してください。

パラメーター

hwnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用してメッセージを表示し、アプリケーション・ウィンドウに関連付けることができます。

usOperation

USHORT - 入力

ユーザー出口ルーチンを呼び出した操作を示す値。値は以下のうちのいずれかです。

- UX_SAVE_ITEM
- UX_ROUTE
- UX_START_WORKFLOW
- UX_CHANGE_WORKFLOW
- UX_SCAN_ITEM
- UX_IMPORT_ITEM

usNumberofITEMIDs

USHORT - 入力

pListofITEMIDs パラメーターにある ITEMID の番号。

usIndex

USHORT - 入力

過負荷条件が起こる原因になる項目。

pListofITEMIDs

PITEMID - 入力

ワーク・バスケットに経路指定される項目の ITEMID のリストを指すポインター。

pExitStruct

PUSEREXITSTRUCT - 入出力

経路指定される文書またはフォルダーが出口処理時にオープンされると、ユーザー定義属性フィールドおよびオブジェクトに関するその他の情報が *pExitStruct* パラメーターに渡されます。データ構造内の値には、ユーザーが「索引形式 (Index Form)」ウィンドウでクラスおよび属性に対して行った変更が含まれています。

経路指定されたオブジェクトがオープンされていない場合、 *pListofITEMIDs* パラメーターは、ユーザーが「目次 (Table of Contents)」ウィンドウから選択した 1 つ以上の ITEMID のリストを指します。現行値が入っている *szUserId* を除いて、 *pExitStruct* パラメーターはすべて NULL です。

pWorkBasketITEMID

PITEMID - 入力

過負荷トリガーの原因である、元の宛先ワーク・バスケットの ITEMID を指すポインター。

pNewWorkBasketITEMID

PITEMID - 出力

NULL ITEMID が入っているバッファーを指すポインター。これを、バックアップ宛先として使用されるシステム内で有効なワーク・バスケットの ITEMID と置換してください。

内部表記

USEREXITSTRUCT:

```
typedef  struct
{
HSESSION
        hSession
ITEMID
        uidItem;
USHORT
        itemidWorkflowId;
BOOL  fIsUnindexed;
USHORT
        hOrigClass;
USHORT
        hClass;
CHAR  szUserId[LST_USERID_LEN+1];
CHAR  szUserHandle[LST_USERID_LEN+1];
USHORT
        usAccessLevel;
SHORT
        sFields;
FIELDVALUE  *
        pFields;
}  USEREXITSTRUCT;

typedef  USEREXITSTRUCT  *  PUSEREXITSTRUCT
```

値の意味を以下に説明します。

hSession
: **SimLibLogon** により戻されるセッション・ハンドル。

uidItem
: 経路指定される現行文書またはフォルダーの ITEMID。

itemidWorkflowId
: 経路指定されるオープン済みの文書またはフォルダーのワークフロー ID。オブジェクトがこのユーザーによってオープンされない場合、またはオブジェクトがワークフローにない場合には、この値は NULL です。

fIsUnindexed
: オブジェクトが、システム内で索引付けされていない新規文書の場合には、この値は TRUE です。オブジェクトがこのユーザーによってオープンされない場合には、この値は FALSE です。

hOrigClass
: オープン済み文書またはフォルダーの、元のクラス ID。オブジェクトがこのユーザーによってオープンされない場合には、この値は NULL です。

hClass オープン済み文書またはフォルダーの、現行クラス ID。ユーザーが新しい

索引クラスを指定していない限り、この値は *hOrigClass* パラメーターと同じです。オブジェクトがこのユーザーによってオープンされない場合には、この値は NULL です。

szUserId[LST_USERID_LEN+1]
文書またはフォルダーを経路指定するユーザーのユーザー ID。

szUserHandle[LST_USERID_LEN+1]
このパラメーターは予約済みです。

usAccessLevel
この文書またはフォルダーに対してユーザーが持っているアクセス権。オブジェクトがこのユーザーによってオープンされない場合には、この値は NULL です。有効な値は、次のとおりです。
ユーザーが BROWSE モードでこのオブジェクトをオープンする場合は、 UX_PRIV_READ です。
ユーザーが UPDATE モードでこのオブジェクトをオープンする場合は、 UX_PRIV_WRITE です。

sFields *pFields* パラメーターで出口に渡されるフィールドの数。このユーザーがオブジェクトをオープンしていないか、そのオブジェクトの「索引形式 (Index Form)」ウィンドウがクローズされている間に、ユーザーがオープン済み文書またはフォルダーに対して「経路指定先 (Route to)」オプションを選択する場合、この値はゼロになります。

pFields FIELDVALUE データ構造の配列を指すポインター。この文書またはフォルダーのユーザー定義属性の構成および内容が、これらのデータ構造で出口に渡されます。このユーザーがオブジェクトをオープンしていないか、そのオブジェクトの「索引形式 (Index Form)」ウィンドウがクローズされている間に、ユーザーがオープン済みの文書またはフォルダーに対して「経路指定先 (Route to)」オプションを選択する場合、この値は NULL になります。

FIELDVALUE:

```
typedef struct
{
USHORT
        usFieldId;
USHORT
        usDataType;
USHORT
        usMaxLength;
BOOL    fIsReq;
PSZ     pBuffer;
} FIELDVALUE;

typedef FIELDVALUE * PFIELDVALUE
```

値の意味を以下に説明します。

usFieldId
ユーザー定義属性 ID。

usDataType

usFieldId パラメーターの属性の IBM Content Manager for iSeries データ型。これはデータ型を示すのと同等の数字です。これらの数字の定義ステートメントと内容の要件については、 frnpfi.h ヘッダー・ファイルのセクション「Attribute types」を参照してください。

usMaxLength

「索引形式 (Index Form)」ウィンドウに表示される *pBuffer* パラメーターのうち、 NULL 終止符を除いた最大のバイト数。

fIsReq フィールドが必要な場合、この値は TRUE です。

pBuffer ASCIIZ 表示形式の属性の現行値。バッファー長は *usMaxLength* パラメーターの値に 1 (NULL 終止符の分) を足した値です。

結果

SUCCESS の場合、ゼロの SHORT の値が戻されます。呼び出しが正常に終了すると、*pNewWorkBasketITEMID* パラメーターの値が代替宛先として使用されます。これは、有効なIBM Content Manager for iSeriesワーク・バスケット ItemID でなくてはなりません。この ItemID がオーバーフローしたワーク・バスケットと同じか、この値に NULL ITEMID が入っていると、項目は元のワーク・バスケットに強制的に入れられ、オーバーフロー条件は無視されます。

呼び出しがゼロ以外の値を戻すと、エラー・メッセージが表示され、項目はどのワーク・バスケットにも経路指定されません。

保管中にユーザー出口ルーチンが呼び出されている時にエラーが戻されると、項目は保管されますが、ワークフローへの配置またはワーク・バスケットへの経路指定は行われません。

新しいワーク・バスケットを経路指定した結果、別の過負荷が起きた場合は、再びこのユーザー出口ルーチンが呼び出されます。

注釈

出口ルーチンは、渡されているバッファーを解放することはできません。出口に送信されるパラメーターは、すべて読み取り専用コピーです。バッファーを修正または割り振り解除することはできません。

出口が完了しても、まだ *pNewWorkBasketITEMID* パラメーターが NULL である場合は、選択された項目が強制的に元のワーク・バスケット宛先に入れられます。

「処理 (Process)」メニューから「経路指定先 (Route to)」オプションをユーザーが選択する際、「索引形式 (Index Form)」ウィンドウがオープンされていないと、USEREXITSTRUCT データ構造の *pFields* ポインターおよび *sFields* パラメーターは NULL になります。この場合、通常はユーザー定義属性の詳細が入っている FIELDVALUE データ構造は、ユーザー出口ルーチンに渡されません。

2 つのワーク・バスケットを、互いにバックアップとして割り当てないでください。そのようにしてしまうと、2 つのワーク・バスケットが過負荷の場合は、循環参照の無限ループが始まる可能性があります。

QuerySortUserExit (照会ソート・ユーザー出口)

形式

SHORT QuerySortUserExit(*hSession, hwnd, pSortList, usItemCount, pszUserId, usSortObject*)

目的

このユーザー出口ルーチンは、出口が定義されるクラスの文書またはフォルダーが入っているフォルダーまたはワーク・バスケットがオープンされると呼び出されます。この出口には、ファイル・キャビネット検索の結果作成されるフォルダーが組み込まれています。この出口をプログラミングして、画面に表示される前に、フォルダーまたはワーク・バスケットの目次を分類および修正することができます。この関数を使用すると、IBM Content Manager for iSeries が提供するデフォルトの昇順または降順以外の、特定のソート順序を定義することができます。出口を使用して、選択した文書およびフォルダーをフィルターに掛け、表示とオブジェクトへのユーザー・アクセスをしないようにすることもできます。また、フォルダーの目録を印刷するより先に、この出口を呼び出すこともできます。

IBM Content Manager for iSeries を使用して、このユーザー出口ルーチンをクラスごとに割り当てることができます。フォルダーまたはワーク・バスケット目録に表される各クラスは、そのクラスに指定されたユーザー出口ルーチンに従ってソートされます。フォルダーまたはワーク・バスケットの中の複数のクラスがこのユーザー出口ルーチンを呼び出すと、まず各出口ルーチンが順番に呼び出されて完了し、そのあとで内容が表示されます。特定のクラスに割り当てられた文書およびフォルダーだけが、そのクラスのために呼び出された出口ルーチンに渡されます。

システム管理プログラムを使用して、このユーザー出口ルーチンを、索引クラス設定ノートブックの「ソート (Sort)」フィールドに指定します。「*システム管理ガイド*」(SC88-4004) を参照してください。

パラメーター

hSession

HSESSION - 入力

SimLibLogon により戻されるセッション・ハンドル。

hwnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用してメッセージを表示し、アプリケーション・ウィンドウに関連付けることができます。

pSortList

PUSERSORTSTRUCT - 入出力

分類される文書およびフォルダーの配列を指すポインター。各文書またはフォルダーは、USERSORTSTRUCT データ構造により表されます。

usItemCount

USHORT - 入力

pSortList パラメーターの文書およびフォルダーの番号。

pszUserId

PSZ - 入力

フォルダーまたはワーク・バスケットをオープンするユーザーのユーザー ID 名。これは、API を介して指定された ID です。

usSortObject

USHORT - 入力

表示されるオブジェクトのタイプ。有効な値は、次のとおりです。

フォルダー表示用に目次がソートされている場合は、SIM_FOLDER です。検索結果フォルダーが組み込まれています。

ワーク・バスケット表示用に目次がソートされている場合は、SIM_WORKBASKET です。

内部表記

USERSORTSTRUCT:

```
typedef struct
{
USHORT
        usType;
USHORT
        usClass;
ITEMID
        uid;
USHORT
        usPriority;
PCHAR *
        pszVals;
PCHAR *
        pszWbVals;
ATLIST *
        pAttrList;
BOOL  fCheckedOut;
USHORT
        usFlags;
} USERSORTSTRUCT;
```

typedef USERSORTSTRUCT * PUSERSORTSTRUCT;

値の意味を以下に説明します。

usType

オブジェクトのタイプ。有効な値は、次のとおりです。

オブジェクトが文書の場合は、SIM_DOCUMENT です。

オブジェクトがフォルダーの場合は、SIM_FOLDER です。

usClass

このオブジェクトの索引クラスの現行ビュー ID。

uid このオブジェクトの IBM Content Manager for iSeries ITEMID。

usPriority

このオブジェクトの優先順位。

pszVals

この文書またはフォルダーの ASCIIZ 形式での表示値の配列を指すポインター。これらの値には、ユーザー定義属性および以下のシステム属性が含まれます。

ワークフロー名
Priority
チェックアウト ID
中断状況

この索引クラスでユーザーの現行レイアウト内に存在しない各属性については、配列内に NULL ASCIIZ ストリングが表示されます。この配列の各値には、*pAttrList* パラメーターからの ATLIST データ構造に、対応する値があります。

pszWbVals

オブジェクトのワーク・バスケット表示値を指すポインター。値が以下の順序で組み込まれています。

Priority
ワーク・バスケットへの入力日付
ワーク・バスケットへの入力時刻
クラス名

これは、このワーク・バスケットへの入力時刻を示す日時スタンプです。*usSortObject* が SIM_FOLDER と等しい場合には、このポインターは NULL です。

pAttrList

ATLIST データ構造を指すポインター。このデータ構造には、このオブジェクトのために表示される属性値に関する詳しい情報が入っています。これらの値には、ユーザー定義属性および以下のシステム属性が含まれます。

ワークフロー名
Priority
チェックアウト ID
中断状況

fCheckedOut

オブジェクトがチェックアウトされた場合、この値は TRUE です。

usFlags

このオブジェクトがソートされた目次に表示されないようにする場合、このパラメーターを SF_HIDE に設定してください。カウントは、隠されない項目数を反映します。

ATLIST:

```
typedef struct
{
USHORT
        usClass;
USHORT
        usCount;
```

```
    ATINFO *
        patinfo;
    USHORT
        usUserCount;
    USHORT *
        patidUserList;
    } ATLIST;

typedef ATLIST * PATLIST;
```

値の意味を以下に説明します。

usClass
: このオブジェクトに対して格納される索引クラスの現行表示 ID。 USERSORTSTRUCT データ構造の *usClass* パラメーターと同じ値です。

usCount
: *patinfo* パラメーターで参照される配列にリストされる属性の数。

patinfo ATINFO データ構造の配列を指すポインター。それぞれのユーザー定義属性および以下のシステム属性には、別個のデータ構造があります。

- ワークフロー名
- Priority
- チェックアウト ID
- 中断状況

このユーザー出口ルーチンが印刷操作から呼び出されると、ユーザーの現行レイアウトの属性だけが配列に組み込まれます。

usUserCount
: *patidUserList* パラメーターで参照される配列にリストされる属性の番号。

patidUserList
: USHORT の配列を指すポインター。「目次 (Table of Contents)」ウィンドウに表示される、このオブジェクトのユーザー定義属性ごとに個別の USHORT があります。これらの属性は、この索引クラスに割り当てられた属性のリストから各ユーザーが選択します。こうして選択された属性のみが表示可能です。

ATINFO:

```
typedef struct
{
USHORT
    atid;
PATTRINFOSTRUCT
    pai;
} ATINFO;

typedef ATINFO * PATINFO;
```

値の意味を以下に説明します。

atid *pai* パラメーターで指される ATTRINFOSTRUCT データ構造で定義された属性 ID。

pai ATTRINFOSTRUCT データ構造を指すポインター。このデータ構造には、システム内の属性名、データ型、最小長、最大長が入っています。

結果

SUCCESS の場合、ゼロの SHORT の値が戻されます。フォルダーまたはワーク・バスケットの目次は、順不同で表示されます。

出口が正常に終了すると、項目は USERSORTSTRUCT 配列 (*pSortList* [0], *pSortList* [1]) にソートされた順に表示されます。*usFlags* パラメーターが SF_HIDE に設定されていると、文書またはフォルダーは、その索引クラスの他のオブジェクトと一緒に表示されることはありません。

注釈

出口ルーチンは、渡されているバッファーを解放することはできません。この出口では、「目次 (Table of Contents)」ウィンドウのユーザー・レイアウトを変更することはできません。

属性値は、出口では修正できません。これらの属性は、ATLIST データ構造の *patidUserList* パラメーターにリストされます。

オープンされているワーク・バスケットが、システム管理プログラムを介してシステム割り当て作業に対して指定されていると、この出口は呼び出されません。ユーザーがワーク・バスケットを優先モードで表示すると、 IBM Content Manager for iSeries はユーザー出口ルーチンが戻した順序を無視します。隠されるよう指定された項目は表示されません。

この関数は、リストの表示より先に処理されます。

SaveRecordUserExit (レコード保管ユーザー出口)

形式

SHORT SaveRecordUserExit(*hwnd, pPreSaveStruct, ppszErrorMsgs, ppusFieldIdsInError***)**

目的

このユーザー出口ルーチンは、文書またはフォルダーのユーザー定義属性に加えた変更を保管するようユーザーが選択すると呼び出されます。索引属性フィールドは、出口に渡されて処理されます。「索引形式 (Index Form)」ウィンドウに入力された新しい属性データは、既存のファイルの情報と突き合わせを行うことによって妥当性を検査することができます。この出口では、ユーザー定義属性フィールドを変更することもできます。

フィールドは、出口で修正されると、レコードがデータベースに書き込まれる前に、 IBM Content Manager for iSeries システムにより監査されます。監査では、以下の指針を使用して、戻されたデータが比較されます。

データ型 - データの形式および内容は、属性に対するデータ型の要件に適合している必要があります。

最小長 - データ・ストリングの最小長要件、またはあるデータ型の最小数値は、属性に指定されているものと一致している必要があります。
最大長 - データ・ストリングの最大長要件、またはあるデータ型の最大数値は、属性に指定されているものと一致している必要があります。

必須フィールドを指定します。

出口はエラー・メッセージのリストを戻して、ユーザー指定の値のエラーを示すことができます。エラー・メッセージは「索引形式 (Index Form)」ウィンドウに表示されます。エラー・メッセージに対応する属性の表示フィールドは、「索引形式 (Index Form)」ウィンドウに疑問符でフラグが付けられます。

システム管理プログラムを使用して、このユーザー出口ルーチンを、索引クラス設定ノートブックの「保管 (Save)」フィールドに指定します。「*システム管理ガイド*」(SC88-4004) を参照してください。

パラメーター

hwnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。これを使用してメッセージを表示し、アプリケーション・ウィンドウに関連付けることができます。

制限 保管時にフレームが使用不可になるため、このウィンドウ (hwnd) をダイアログの親として使用することのないようにしてください。デスクトップを親として使用し、このウィンドウ (hwnd) を所有者として使用することができます。

pPreSaveStruct

PUSEREXITSTRUCT - 入出力

文書またはフォルダーのユーザー定義属性およびその他の関連情報が、*pPreSaveStruct* パラメーターに渡されます。

ppszErrorMsgs

PSZ * - 出力

ポインターのアドレス。ポインターは、エラー・メッセージを表す ASCIIZ ストリングのデータ・ストリームを指すように、ユーザーの出口ルーチンによって設定されている必要があります。エラー・メッセージは、それぞれ *ppusFieldIdsInError* パラメーターの属性 ID に対応していなくてはなりません。エラー・データ・ストリームの必須形式は、「結果」のセクションで定義されます。このバッファーはユーザー出口ルーチンによって割り振られますが、割り振り解除は IBM Content Manager for iSeries が行います。

ppusFieldIdsInError

PUSHORT *- 出力

ppszErrorMsgs パラメーターに戻されるエラー・メッセージに関連した属性 ID の配列を指すポインターのアドレス。有効な属性 ID は、FIELDVALUE データ構造の *usFieldId* パラメーターで出口に渡されます。このバッファーはユーザー出口ルーチンによって割り振られますが、割り振り解除は IBM Content Manager for iSeries が行います。

内部表記

USEREXITSTRUCT:

```
typedef  struct
{
HSESSION
        hSession
ITEMID
        uidItem;
USHORT
        itemidWorkflowId;
BOOL  fIsUnindexed;
USHORT
        hOrigClass;
USHORT
        hClass;
CHAR  szUserId[LST_USERID_LEN+1];
CHAR  szUserHandle[LST_USERID_LEN+1];
USHORT
        usAccessLevel;
SHORT
        sFields;
FIELDVALUE *
        pFields;
} USEREXITSTRUCT;

typedef  USEREXITSTRUCT  *  PUSEREXITSTRUCT
```

値の意味を以下に説明します。

hSession
: **SimLibLogon** により戻されるセッション・ハンドル。

uidItem
: 保管される現行文書またはフォルダーの ITEMID。

itemidWorkflowId
: このパラメーターは常に NULL です。

fIsUnindexed
: オブジェクトが、システム内で索引付けされていない新規文書の場合には、この値は TRUE です。

hOrigClass
: オープン済み文書またはフォルダーの、元のクラス ID。

hClass
: オープン済み文書またはフォルダーの、現行クラス ID。ユーザーが新しい索引クラスを指定しない限り、この値は *hOrigClass* パラメーターと同じです。

szUserId[LST_USERID_LEN+1]
: 文書またはフォルダーを保管しているユーザーのユーザー ID。

szUserHandle[LST_USERID_LEN+1]
このパラメーターは予約済みです。

usAccessLevel
この文書またはフォルダーに対してユーザーが持っているアクセス権。このユーザー出口ルーチンに有効な値は以下のとおりです。
ユーザーが UPDATE モードでこのオブジェクトをオープンする場合は、 UX_PRIV_WRITE です。

sFields *pFields* パラメーターで出口に渡されるフィールドの数。

pFields FIELDVALUE データ構造の配列を指すポインター。この文書またはフォルダーのユーザー定義属性の構成および内容が、これらのデータ構造で出口に渡されます。

FIELDVALUE:

```
typedef struct
{
USHORT
        usFieldId;
USHORT
        usDataType;
USHORT
        usMaxLength;
BOOL    fIsReq;
PSZ     pBuffer;
} FIELDVALUE;

typedef FIELDVALUE * PFIELDVALUE
```

値の意味を以下に説明します。

usFieldId
ユーザー定義属性 ID。

usDataType
usFieldId パラメーターの属性の IBM Content Manager for iSeries データ型。これはデータ型を示すのと同等の数字です。これらの数字の定義ステートメントと内容の要件については、 frnpfi.h ヘッダー・ファイルのセクション「Attribute types」を参照してください。

usMaxLength
「索引形式 (Index Form)」ウィンドウに表示される *pBuffer* パラメーターのうち、 NULL 終止符を除いた最大のバイト数。

fIsReq フィールドが必要な場合、この値は TRUE です。このパラメーターが TRUE に設定され、この FIELDVALUE データ構造が出口で修正されると、 *pBuffer* の値は NULL に変更できません。

pBuffer ASCIIZ 表示形式の属性の現行値。バッファー長は *usMaxLength* パラメーターの値に 1 (NULL 終止符の分) を足した値です。

結果

SUCCESS の場合は、ゼロの SHORT の値が戻されます。他の値が戻されると、「保管 (Save)」操作は終了します。そして、エラー・メッセージが表示されます。

出口ルーチンが正常に終了すると、 *ppszErrorMsgs* パラメーターのエラー・ストリング・ポインターのアドレスが問い合わせされます。エラー・ストリング・ポインターが NULL であるか、 NULL エラー・ストリングを指していると、FIELDVALUE データ構造に戻される属性値、データ型、最小および最大長要件が、IBM Content Manager for iSeries に監査されます。監査エラーは「索引エラー (Index Errors)」ウィンドウに表示されます。監査エラーのある「索引形式 (Index Form)」ウィンドウの属性フィールドの横には、それぞれ疑問符が付いています。この場合は、エラーを訂正して再度保管オプションを選択し、 IBM Content Manager for iSeries データベースにレコードを保管してください。

ppszErrorMsgs パラメーターのエラー・ストリング・ポインターは、表示されるメッセージを指します。エラー・メッセージ・ストリングの形式は以下のとおりです。

```
string1 (zero-terminated) <one or more zero-terminated
strings >zero terminator
```

このエラー・メッセージは「索引エラー (Index Errors)」ウィンドウに表示されます。ゼロで終わるストリングは、ウィンドウの新規の行に表示されます。IBM Content Manager for iSeries が検出した監査エラーは、同じ「索引エラー (Index Errors)」ウィンドウに表示されます。

出口がエラー・メッセージ・ストリングを戻すと、 *ppusFieldIdsInError* パラメーターにアドレスされるポインターは、エラー時の属性 ID の配列を指すように設定されます。この配列には、*ppszErrorMsgs* パラメーターが参照するエラー・ストリング内のメッセージごとに、1 つの属性 ID がなくてはなりません。「索引形式 (Index Form)」ウィンドウでのユーザー定義属性名が、「索引エラー (Index Errors)」ウィンドウの対応するエラー・メッセージの左側に表示されます。「索引形式 (Index Form)」ウィンドウのフィールドの横には、疑問符が表示されます。

注釈

出口ルーチンは、渡されているバッファーを解放することはできません。出口に送信される項目は、FIELDVALUE データ構造のユーザー定義属性を除いて、すべて読み取り専用コピーです。

この出口は、オブジェクトのユーザー定義属性の修正が行われたあとに、文書またはフォルダーが保管されると呼び出されます。また、システム管理ストレージ方式変更ユーザー出口ルーチンとこの出口が現行索引クラスに指定されている場合、こちらの出口の方が先に呼び出されます。ユーザー定義属性フィールドの妥当性検査は、ユーザー出口ルーチンが完了してから実行されます。IBM Content Manager for iSeries は、ユーザーにエラー・メッセージを表示したあと、出口が割り振ったエラー・メッセージ・バッファーを解放します。

「索引形式 (Index Form)」ウィンドウがオープンされていないか、ユーザーがオブジェクトのクラスまたは属性を変更していないと、この出口は処理されません。

UserActionUserExit (ワークフロー・ユーザー・アクション・ユーザー出口)

形式

SHORT EXPENTRY UserActionUserExit(*hSession, hWnd, pWorkManagementInfo, pExitStruct, pszAction*)

ワーク・バスケットでユーザー定義のアクション (機能コード 0050) が選択されると、クライアント・アプリケーション・プログラムは、このユーザー出口を呼び出します。

このユーザー出口を指定して、アクション・リスト定義内にあるユーザー・アクション関数と関連付ける場合は、Workflow Builder 機能を使用します。

パラメーター

hSession

HSESSION - 入力

SimLibLogon により戻されるセッション・ハンドル。

hWnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用して、メッセージを表示し、アプリケーション・ウィンドウと関連付けることができます。

pWorkManagementInfo

PWMSNAPSHOTSTRUCT - 入力

WMSNAPSHOTSTRUCT データ構造が選択された項目に関する詳細な作業管理情報を提供するバッファーへのポインター。

pExitStruct

PUSEREXITSTRUCT - 入力

選択された項目に関連する索引クラスと属性情報が入ります。

pszAction

PSZ - 入力

選択されたアクションが入ったヌル終了文字ストリング。これは、*ACTION 変数の値です。

結果

このユーザー出口は、タイプ SHORT の値を戻します。ユーザー出口が正常に終了すると、値ゼロが戻されます。この他の値が戻された場合、エラー・メッセージが表示されます。

UserOptionUserExit (ユーザー・オプション・ユーザー出口)

形式

SHORT EXPENTRY UserOptionUserExit(*hSession, hWnd, pExitStruct*)

項目の「選択済み (Selected)」メニューからユーザー定義のオプションが選択されると、クライアント・アプリケーション・プログラムは、このユーザー出口を呼び出します。

このユーザー出口は、システム管理機能を使用して索引クラス・プロファイルに指定します。詳細については、「*IBM Content Manager for iSeries システム管理ガイド*」を参照してください。

パラメーター

hSession

HSESSION - 入力

SimLibLogon により戻されるセッション・ハンドル。

hWnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用して、メッセージを表示し、アプリケーション・ウィンドウと関連付けることができます。

pExitStruct

PUSEREXITSTRUCT - 入力

選択された項目に関連する索引クラスと属性情報が入ります。

このユーザー出口は、タイプ SHORT の値を戻します。ユーザー出口が正常に終了すると、値ゼロが戻されます。この他の値が戻された場合、エラー・メッセージが表示されます。

WBItemSelectedUserExit (項目選択ワーク・バスケット・ユーザー出口)

形式

SHORT EXPENTRY WBItemUserExit(*hSession, hWnd, pWorkManagementInfo, pExitStruct, pfContinue*)

クライアント・アプリケーション・プログラムは、ワーク・バスケットで項目が選択されると、このユーザー出口を呼び出します。この出口は、項目がユーザーに表示される前に呼び出されます。

システム管理機能を使用して、このユーザー出口をワーク・バスケット・プロファイルに指定します。詳細については、「*IBM Content Manager for iSeries: システム管理ガイド*」を参照してください。

パラメーター

hSession

HSESSION - 入力

SimLibLogon により戻されるセッション・ハンドル。

hWnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用して、メッセージを表示し、アプリケーション・ウィンドウと関連付けることができます。

pWorkManagementInfo

PWMSNAPSHOTSTRUCT - 入力

WMSNAPSHOTSTRUCT データ構造が選択された項目に関する詳細な作業管理情報を提供するバッファーへのポインター。

pExitStruct

PUSEREXITSTRUCT - 入力

選択された項目に関連する索引クラスと属性情報が入ります。

pfContinue

PBOOL - 出力

継続フラグを指すポインター。選択された項目の表示を継続する場合は、この値を TRUE に設定します。項目の表示をバイパスする場合は、この値を FALSE に設定します。

結果

このユーザー出口は、タイプ SHORT の値を戻します。ユーザー出口が正常に終了すると、値ゼロが戻されます。この他の値が戻された場合、エラー・メッセージが表示されます。

WBItemCompletedUserExit (項目完了ワーク・バスケット・ユーザー出口)

形式

SHORT EXPENTRY WBItemCompletedUserExit(*hSession, hWnd, pWorkManagementInfo, pExitStruct, pszAction, pfContinue*)

項目の処理を完了させるワーク・バスケットでアクションが選択されると、クライアント・アプリケーション・プログラムは、このユーザー出口を呼び出します。この出口は、そのアクションがクライアントに処理される前に呼び出されます。

システム管理機能を使用して、このユーザー出口をワーク・バスケット・プロファイルに指定します。詳細については、「*IBM Content Manager for iSeries システム管理ガイド*」を参照してください。

パラメーター

hSession

HSESSION - 入力

SimLibLogon により戻されるセッション・ハンドル。

hWnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用して、メッセージを表示し、アプリケーション・ウィンドウと関連付けることができます。

pWorkManagementInfo

PWMSNAPSHOTSTRUCT - 入力

WMSNAPSHOTSTRUCT データ構造が選択された項目に関する詳細な作業管理情報を提供するバッファーへのポインター。

pExitStruct

PUSEREXITSTRUCT - 入力

選択された項目に関連する索引クラスと属性情報が入ります。

pszAction

PSZ - 入力

選択されたアクションが入ったヌル終了文字ストリング。これは、SIMWM_ACTION 変数の値です。

pfContinue

PBOOL - 出力

継続フラグを指すポインター。選択された項目の表示を継続する場合は、この値を TRUE に設定します。項目の表示をバイパスする場合は、この値を FALSE に設定します。

結果

このユーザー出口は、タイプ SHORT の値を戻します。ユーザー出口が正常に終了すると、値ゼロが戻されます。この他の値が戻された場合、エラー・メッセージが表示されます。

UserDefinedWBUserExit (ユーザー定義ワーク・バスケット・ユーザー出口)

形式

SHORT EXPENTRY UserDefinedWBUserExit(*hSession, hWnd, pWorkBasketInfo, pszUserid*)

クライアント・アプリケーション・プログラムは、ユーザーがタイプ 50 から 99 までのワーク・バスケットのオープンを選択すると、このユーザー出口を呼び出します。ユーザー定義のワーク・バスケットのタイプとこの出口を使用することによって、Content Manager for iSeries が提供するプロセス制御機能とワーク・バスケッ

ト制御機能を利用できるようになります。ただし、ワーク・バスケットとその内容とのインターフェースは、この出口を通してユーザー独自の定義によって制御されます。

システム管理機能を使用して、このユーザー出口をワーク・バスケット・プロファイルに指定します。詳細については、「*IBM Content Manager for iSeries システム管理ガイド*」を参照してください。

パラメーター

hSession

HSESSION - 入力

SimLibLogon により戻されるセッション・ハンドル。

hWnd HWND - 入力

メッセージ・ボックスのアンカー・ウィンドウ。このパラメーターを使用して、メッセージを表示し、アプリケーション・ウィンドウと関連付けることができます。

pWorkBasketInfo

PWORKBASKETINFOSTRUCT - 入力

WORKBASKETINFOSTRUCT データ構造がユーザー定義のワーク・バスケットに関する詳細な情報を提供するバッファーへのポインター。

pszUserID

PSZ - 入力

このユーザー出口を呼び出すユーザーのユーザー ID が入っている、null で終わる文字ストリング。

結果

このユーザー出口は、タイプ SHORT の値を戻します。ユーザー出口が正常に終了すると、値ゼロが戻されます。この他の値が戻された場合、エラー・メッセージが表示されます。

サーバーのユーザー出口

ここで示すユーザー出口ポイントは、Content Manager for iSeries サーバー上で呼び出されます。

注: サーバー出口ポイントから Content Manager for iSeries API を呼び出す際には、その呼び出しで、最後の API が呼び出された後に、必ず SimLibLogoff API が呼び出されるようにする必要があります。これを行わないと、以降の呼び出し時に、予期しない結果を招く恐れがあります。

Content Manager for iSeries は、OS/400 Registration Facility 機能を使用して、呼び出す出口プログラムを決定します。出口プログラムを追加するためには、Work with Registration Information (WRKREGINF) コマンドを入力します。「Work with Registration Information」画面で、使用したい出口ポイントと形式名を探します (表 1 に出口ポイントと形式名のリストがあります)。オプション 8 (「Work with Exit

Programs」) を選択して、それぞれの出口ポイントと形式名に合った出口プログラムを選択します。「Work with Exit Programs」画面で、次のようにします。

- 現在、出口ポイント用に定義されているプログラムがない場合は、オプション 1 (「追加 (Add)」) を使用して出口プログラムの項目を追加します。プログラム番号 1 とそのプログラムのプログラム名とライブラリー名を入力します。
- 既に定義されているプログラムがある場合で、そのプログラムまたはライブラリーの名前を変更したいときは、最初にオプション 4 (「除去 (Remove)」) を使用して現在の入力項目を削除した後、オプション 1 (「追加 (Add)」) を使用して新しいプログラムを追加入力する必要があります。登録機能では、複数の出口プログラムを登録できますが、Content Manager for iSeries では出口ポイント 1 つにつき 1 つの出口プログラムしかサポートされません。

Content Manager for iSeries 出口ポイントがリストに表示されない場合は、コマンド・プロンプトで次のアクションを実行して、出口ポイントを追加してください。

```
CALL EKDCSUEREG PARM(' ' ' ')
```

この呼び出しが完了すると、出口ポイントのリストが登録されます。

表 4. Content Manager for iSeries の出口ポイント

出口ポイント名	フォーマット名	出口プログラム名
QIBM_VI-LOGON	VIF0100	ユーザー定義
QIBM_VI_LOGOFF	VIF0100	ユーザー定義
QIBM_VI_SAVE_ATTR	VIF0100	ユーザー定義
QIBM_VI_CRT_OBJECT	VIF0100	ユーザー定義
QIBM_VI_DLT_OBJECT	VIF0100	ユーザー定義
QIBM_VI_OPEN_OBJECT	VIF0100	ユーザー定義
QIBM_VI_CRT_ITEM	VIF0100	ユーザー定義
QIBM_VI_ITEM_CREATED	VIF0100	ユーザー定義
QIBM_VI_DLT_ITEM	VIF0100	ユーザー定義
QIBM_VI_IMP_CREATED	VIF0100	ユーザー定義
QIBM_VI_IMP_ITEM	VIF0100	ユーザー定義
QIBM_VI_ADD_FLR_ITEM	VIF0100	ユーザー定義
QIBM_VI_ROUTE_WP	VIF0100	ユーザー定義
QIBM_VI_GET_WP	VIF0100	ユーザー定義
QIBM_VI_RETURN_WP	VIF0100	ユーザー定義
QIBM_VI_END_PROCESS	VIF0100	ユーザー定義
QIBM_VI_SET_VARIABLE	VIF0100	ユーザー定義

Logon User Exit (ログオン・ユーザー出口)

このユーザー出口は、**SimLibLogon** を使用して、Content Manager for iSeries へのログオン要求が出されると呼び出されます。

表 5. ログオン・ユーザー出口のパラメーター

フィールド	説明	入出力	形式	サイズ
ユーザー ID	ログオンするユーザーのユーザー ID	入力	文字	10
アドレス	ワークステーション名またはアドレス	入力	文字	15

Logoff User Exit (ログオフ・ユーザー出口)

このユーザー出口は、**SimLibLogoff** を使用して、Content Manager for iSeries からのログオフ要求が出されると呼び出されます。

表 6. ログオフ・ユーザー出口のパラメーター

フィールド	説明	入出力	形式	サイズ
パラメーターなし	このユーザー出口は、**SimLibLogoff** を使用して、Content Manager for iSeries に対するログオフ要求が出されると呼び出されます。			

Save Attributes User Exit (属性保管ユーザー出口)

このユーザー出口は、 **SimLibSaveAttr** または **SimLibCloseAttr** を使用して、文書またはフォルダーの属性に対する変更の保管要求が出された場合に呼び出されます。この出口ポイントは、属性が実際に更新される前になります。このため、ユーザーは出口プログラム内で属性の妥当性検査や変更を行うことができます。変更された属性の妥当性検査は、出口からの戻り時には行われません。

この出口は、特権の検証と入力妥当性検査の前に呼び出されます。

表 7. 属性保管出口のパラメーター

フィールド	説明	入出力	形式	サイズ
項目 ID	属性が変更される項目の ID。	入力	文字	16
索引クラス	ユーザー定義の属性セットをこの項目に関連づけるための索引クラス ID。	入力	バイナリー	4
属性	項目に関連する属性の ID と値のテーブル。	入力/出力	文字	*
戻り値	後続の処理をどのように継続するかを示します。有効な値は以下のとおりです。 • 0 - 通常の処理。属性が変更されます。 • 非ゼロ - エラー処理。属性変更要求は処理されません。	出力	バイナリー	4

属性パラメーターの形式は以下のとおりです。

表 8. 属性

フィールド	型
属性の数	バイナリー (4)
属性テーブル	Char (*)

属性テーブルは、属性テーブルの項目の配列によって構成されます。属性テーブル内の項目の数は、属性の数パラメーター (上記を参照) の値に基づいて決まります。

表 9. 属性テーブルの項目

フィールド	型
属性 ID	バイナリー (4)
属性タイプ	バイナリー (4)
属性の長さ	バイナリー (4)
属性値	Char (*)

Create Object User Exit (オブジェクト作成ユーザー出口)

このユーザー出口は、**SimLibCreateObject** を使用してオブジェクトの作成要求が出された場合に呼び出されます。この出口ポイントは、オブジェクト作成要求が処理された後になります。したがって、項目 ID と部分番号は新規オブジェクトです。この出口が呼び出されるのは、作成要求が正常に処理された場合だけです。

表 10. オブジェクト作成出口のパラメーター

フィールド	説明	入出力	形式	サイズ
項目 ID	オブジェクトの項目 ID。	入力	文字	16
部分番号	オブジェクトの部分番号。	入力	バイナリー	4
バージョン	オブジェクトのバージョン番号。	入力	バイナリー	4
関連タイプ	関連値のタイプは以下のとおりです。 • SIM_ANNOTATIVE • SIM_BASE • SIM_EVENT • SIM_MGDS • SIM_NOTE	入力	バイナリー	4

Delete Object User Exit (オブジェクト削除ユーザー出口)

このユーザー出口は、**SimLibDeleteObject** を使用してオブジェクトの削除要求が出された場合に呼び出されます。この出口ポイントは、削除要求が処理された後になります。

表 11. オブジェクト削除出口のパラメーター

フィールド	説明	入出力	形式	サイズ
項目 ID	オブジェクトの項目 ID。	入力	文字	16
部分番号	オブジェクトの部分番号。	入力	バイナリー	4
削除オプション	有効な削除オプションは以下のとおりです。 • SIM_DELETE_ITEM — 残っている部分番号がない場合、項目を削除します。 • SIM_DELETE_OBJECT — 残っている部品番号がなくても、項目を削除しません。	入力	バイナリー	4
戻りコード	オブジェクト削除要求を処理した後の戻りコード。	入力	バイナリー	4

Open Object User Exit (オブジェクト・オープン・ユーザー出口)

このユーザー出口は、**SimLibOpenObject** を使用してオブジェクトのオープン要求が出された場合に呼び出されます。この出口ポイントは、要求が処理される前に呼び出されます。

表 12. オブジェクト・オープン出口のパラメーター

フィールド	説明	入出力	形式	サイズ
項目 ID	オブジェクトの項目 ID。	入力	文字	16
部分番号	オブジェクトの部分番号。	入力/出力	バイナリー	4
アクセス・レベル	オブジェクトをオープンした際に、オブジェクトに与えられるアクセスのタイプ。有効な値は以下のとおりです。 • SIM_ACCESS_READ_WRITE • SIM_ACCESS_SHARED_READ	入力/出力	バイナリー	4
戻り値	処理がどのように継続されるかを示します。 **0** 通常の処理。オブジェクトはオープンされます。 **非ゼロ** エラー処理。オープン要求は処理されず、この戻り値がユーザーに戻されます。	出力	バイナリー	4

Create Item User Exit (項目作成ユーザー出口)

このユーザー出口は、**SimLibCreateItem** を使用して項目の作成要求が出された場合に呼び出されます。この出口ポイントは、項目が作成される前になります。このため、ユーザーは出口プログラム内で索引クラスや関連する属性の妥当性検査や変更を行うことができます。

この出口は、特権の検証と入力妥当性検査の前に呼び出されます。

表 13. 項目作成出口のパラメーター

フィールド	説明	入出力	形式	サイズ
項目のタイプ	作成する項目のタイプ。有効な値は、次のとおりです。 • SIM_DOCUMENT - 項目が文書であることを示します。 • SIM_FOLDER - 項目がフォルダーであることを示します。	入力	バイナリー	4
索引クラス	ユーザー定義の属性セットをこの項目に関連づけるための索引クラス ID。	入力/出力	バイナリー	4
属性	作成された項目に関連する属性の ID と値のテーブル。	入力/出力	文字	*
戻り値	後続の処理をどのように継続するかを示します。有効な値は以下のとおりです。 • 0 - 通常の処理。項目が作成されます。 • 非ゼロ - エラー処理。項目の作成要求は処理されません。	出力	バイナリー	4

属性パラメーターの定義については、 314 ページの表 8 を参照してください。

Item Created User Exit (項目作成完了ユーザー出口)

このユーザー出口は、**SimLibCreateItem** を使用して項目の作成要求が出された場合に呼び出されます。この出口ポイントは、項目が作成された後になります。

表 14. 項目作成完了出口のパラメーター

フィールド	説明	入出力	形式	サイズ
項目のタイプ	作成された項目のタイプ。有効な値は、次のとおりです。 • SIM_DOCUMENT - 項目が文書であることを示します。 • SIM_FOLDER - 項目がフォルダーであることを示します。	入力	バイナリー	4
索引クラス	ユーザー定義の属性セットをこの項目に関連づけるための索引クラス ID。	入力	バイナリー	4
属性	作成された項目に関連する属性の ID と値のテーブル。	入力	文字	*
項目 ID	作成された項目の ID。	入力	文字	16

属性パラメーターの定義については、 314 ページの表 8 を参照してください。

Delete Item User Exit (項目削除ユーザー出口)

このユーザー出口は、**SimLibDeleteItem** を使用して項目の削除要求が出された場合に呼び出されます。この出口ポイントは、項目が削除された後になります。

表 15. 項目削除出口のパラメーター

フィールド	説明	入出力	形式	サイズ
項目 ID	削除された項目の ID。	入力	文字	16
戻りコード	項目削除要求を処理した後の戻りコード。	入力	バイナリー	4

Object Import Create Item User Exit (オブジェクト・インポートによる項目作成ユーザー出口)

このユーザー出口は、オブジェクト・インポート機能で項目が作成される際に呼び出されます。この出口ポイントは、項目が作成される前になります。このため、ユーザーは出口プログラム内で索引クラスや関連する属性の妥当性検査や変更を行うことができます。

この出口は、特権の検証と入力妥当性検査の前に呼び出されます。

表 16. オブジェクト・インポートによる項目作成出口のパラメーター

フィールド	説明	入出力	形式	サイズ
項目のタイプ	作成する項目のタイプ。有効な値は、次のとおりです。 • SIM_DOCUMENT - 項目が文書であることを示します。 • SIM_FOLDER - 項目がフォルダーであることを示します。	入力	バイナリー	4
索引クラス	ユーザー定義の属性セットをこの項目に関連づけるための索引クラス ID。	入力/出力	バイナリー	4
属性	作成された項目に関連する属性の ID と値のテーブル。	入力/出力	文字	*
戻り値	後続の処理をどのように継続するかを示します。有効な値は以下のとおりです。 • 0 - 通常の処理。項目が作成されます。 • 非ゼロ - エラー処理。項目の作成要求は処理されません。	出力	バイナリー	4

属性パラメーターの定義については、 314 ページの表 8 を参照してください。

Object Import Item Created User Exit (オブジェクト・インポートによる項目作成完了ユーザー出口)

このユーザー出口は、オブジェクト・インポート機能で項目が作成される際に呼び出されます。この出口ポイントは、項目が作成された後になります。

表 17. オブジェクト・インポートによる項目作成完了出口のパラメーター

フィールド	説明	入出力	形式	サイズ
項目のタイプ	作成された項目のタイプ。有効な値は、次のとおりです。 • SIM_DOCUMENT - 項目が文書であることを示します。 • SIM_FOLDER - 項目がフォルダーであることを示します。	入力	バイナリー	4
索引クラス	ユーザー定義の属性セットをこの項目に関連づけるための索引クラス ID。	入力	バイナリー	4
属性	作成された項目に関連する属性の ID と値のテーブル。	入力	文字	*
項目 ID	作成された項目の ID。	入力	文字	16

属性パラメーターの定義については、 314 ページの表 8 を参照してください。

Add Folder Item User Exit (フォルダーへの項目追加ユーザー出口)

このユーザー出口は、**SimLibAddFolderItem** を使用してフォルダーへの項目追加要求が出された場合に呼び出されます。この出口ポイントは、フォルダーに項目が追加される前になります。宛先フォルダーを変更するオプションがあります。

この出口は、特権の検証と入力妥当性検査の前に呼び出されます。

表 18. フォルダーへの項目追加出口のパラメーター

フィールド	説明	入出力	形式	サイズ
フォルダー ID	項目を追加するフォルダーの ID。	入力/出力	文字	16
項目 ID	フォルダーに追加する項目の ID。	入力	文字	16
戻り値	後続の処理をどのように継続するかを示します。有効な値は以下のとおりです。 • 0 - 通常の処理。項目がフォルダーに追加されます。 • 非ゼロ - エラー処理。フォルダーへの項目の追加要求は処理されません。	出力	バイナリー	4

Route Work Package User Exit (作業パッケージ経路指定ユーザー出口)

このユーザー出口は、**SimWmRouteWorkPackage** を使用して作業パッケージの経路指定要求が出された場合に呼び出されます。この出口ポイントは、作業パッケージが経路指定される前になります。宛先ワーク・バスケットを変更するオプションがあります。

この出口は、特権の検証と入力妥当性検査の前に呼び出されます。

表 19. 作業パッケージ経路指定出口のパラメーター

フィールド	説明	入出力	形式	サイズ
ワーク・バスケット	ワーク・バスケット ID	入力/出力	文字	11
作業パッケージ ID	実行中の作業を表す作業パッケージの ID。	入力	バイナリー	4
インスタンス ID	プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。	入力	バイナリー	4
優先順位	作業の優先順位作業パッケージがプロセスを移動するときに、この優先順位により処理順序が決定します。数値が大きいほど、優先順位が高くなります。	入力/出力	バイナリー	4
継続	• 0 - 通常の経路指定処理を継続します。 • 非ゼロ - 出口内で必要なすべての処理が実行されました。追加の処理はバイパスします。	出力	バイナリー	4
戻り値	継続が非ゼロの場合、戻されるエラー・コードです。	出力	バイナリー	4

Get Work Package User Exit (作業パッケージ取得ユーザー出口)

このユーザー出口は、**SimWmGetWorkPackage** を使用して作業パッケージ取得要求が出された場合に呼び出されます。この出口の作業順序と作業パッケージ ID およびインスタンスは、指定変更されることがあります。

この出口は、特権の検証と入力妥当性検査の前に呼び出されます。

表 20. 作業パッケージ取得出口のパラメーター

フィールド	説明	入出力	形式	サイズ
ワーク・バスケット	ワーク・バスケット ID	入力	文字	11

表 20. 作業パッケージ取得出口のパラメーター (続き)

作業順序	ワーク・バスケットからの項目の選択に使用される順序。有効な値は以下のとおりです。 • SIMWM_ORDER_FIFO - 先入れ先出し法 (FIFO) の順序に基づいた選択を行い、最初に使用可能な作業パッケージを戻します。 • SIMWM_ORDER_LIFO - 後入れ先出し法 (LIFO) の順序に基づいた選択を行い、最初に使用可能な作業パッケージを戻します。 • SIMWM_ORDER_PRIORITY - 作業パッケージの優先順位に基づいた選択を行い、最初に使用可能な作業パッケージを戻します。 • SIMWM_ORDER_SYSTEM_NEXT - サーバーが作業順序を決定し、次に使用可能な作業パッケージを戻します。 • SIMWM_ORDER_FIFO_NEXT - 先入れ先出し法 (FIFO) の順序に基づき、次に使用可能な作業パッケージを選択します。 • SIMWM_ORDER_LIFO_NEXT - 後入れ先出し法 (LIFO) の順序に基づき、次に使用可能な作業パッケージを選択します。 • SIMWM_ORDER_PRIORITY_NEXT - 作業パッケージの優先順位に基づき、次に使用可能な作業パッケージを選択します。 • NULL - 作業パッケージ ID が指定されている場合は、この作業パッケージを選択します。作業パッケージ ID が 0 の場合は、サーバーが作業順序を決定し、最初に使用可能な作業パッケージを戻します。	入力/出力	バイナリー	4
作業パッケージ ID	実行中の作業を表す作業パッケージの ID。	入力/出力	バイナリー	4
インスタンス ID	プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。	入力/出力	バイナリー	4

Return Work Package User Exit (作業パッケージ・リターン・ユーザー出口)

このユーザー出口は、**SimWmReturnWorkPackage** を使用して作業パッケージを戻す要求が出された場合に呼び出されます。この出口ポイントは、作業パッケージ・リターン要求が処理された後になります。優先順位は指定変更されることがあります。

この出口は、特権の検証と入力妥当性検査の前に呼び出されます。

表 21. 作業パッケージ・リターン出口のパラメーター

フィールド	説明	入出力	形式	サイズ
作業パッケージ ID	実行中の作業を表す作業パッケージの ID。	入力	バイナリー	4
インスタンス ID	プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。	入力	バイナリー	4
優先順位	実行する作業の優先順位。作業パッケージがプロセスを移動するときに、この優先順位により処理順序が決定します。数値が大きいほど、優先順位が高くなります。	入力/出力	バイナリー	4

End Process User Exit (プロセス終了ユーザー出口)

このユーザー出口は、**SimWmEndProcess** を使用して経路での作業パッケージの終了要求が出された場合に呼び出されます。この出口ポイントは、作業パッケージを終了させる前になります。

この出口は、特権の検証と入力妥当性検査の前に呼び出されます。

表 22. プロセス終了出口のパラメーター

フィールド	説明	入出力	形式	サイズ
作業パッケージ ID	実行中の作業を表す作業パッケージの ID。	入力	バイナリー	4
インスタンス ID	プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。	入力	バイナリー	4
継続	• 0 - 通常の終了処理を継続します。 • 非ゼロ - 出口内で必要なすべての処理が実行されました。追加の処理はバイパスします。	出力	バイナリー	4
戻り値	継続が非ゼロの場合、戻されるエラー・コードです。	出力	バイナリー	4

Set Variable User Exit (変数設定ユーザー出口)

このユーザー出口は、ワークフローの処理中に、変数が問い合わされる場合に呼び出されます。プロセスは、まず、変数が以下のいずれかであるかどうかを判別します。

- プロセス
- 索引クラス
- 既存の変数
- キー・フィールド

変数が上記のいずれにも該当しない場合、プロセスはその変数が外部変数であると想定し、このユーザー出口を呼び出して可変値を取得します。

表 23. 変数設定出口のパラメーター

フィールド	説明	入出力	形式	サイズ
作業パッケージ ID	実行中の作業を表す作業パッケージの ID。	入力	バイナリー	4
インスタンス ID	プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。	入力	バイナリー	4
変数名	処理される変数の名前。	入力	文字	10
可変値	変数の値。	入力/出力	文字	40
戻り値	後続の処理をどのように継続するかを示します。 **0** 通常の処理。変数を作成します。 **非ゼロ** エラー処理。変数の作成要求は処理されません。	出力	バイナリー	4

プロセス定義用のサーバーのユーザー出口

このユーザー出口は、プロセス定義にユーザー出口ノードが含まれている場合に、プロセスの 1 つのステップとして呼び出されます。

前述のサーバーのユーザー出口と異なり、ユーザー出口ノード・ユーザー出口は、OS/400 Registration Facility に入力されます。ユーザー出口ノードを Workflow Builder にプロセス定義の一部として定義する場合は、呼び出すプログラムとライブラリーを指定します。

表 24. ユーザー出口ノード出口のパラメーター

フィールド	説明	入出力	形式	サイズ
作業パッケージ ID	実行中の作業を表す作業パッケージの ID。	入力	10 進数	10
インスタンス ID	プロセス内のある並列パスと別の並列パスとを区別する作業パッケージ・インスタンスの ID。	入力	10 進数	5

表 24. ユーザー出口ノード出口のパラメーター (続き)

戻り値	戻り値が 0 の場合、経路定義内の次のコマンドから処理が継続します。戻り値にこの他の値がある場合は、エラー・ログ・ファイルにエラー・メッセージ EKD-1111 が記録され、経路定義の次のコマンドが処理されます。	出力	10 進数	4
予約済み	将来使用するために予約済みです。	入力	文字	512

付録 A. 検索式の指針

この付録では、Content Manager for iSeries クライアント・アプリケーションの検索の際に従うべき指針を示します。

検索の論理演算子

以下に、有効な論理演算子を優先順位に従って示します。

NOT または ^
後続の条件を否定します。

AND または &
前述の条件と後続の条件がともに真でなければなりません。

OR または | 前述の条件と後続の条件のどちらか一方が真です。

以下は、優先順位のルールを示す例です。

次のストリングで W、X、Y、および Z はそれぞれ式を表します。

```
W OR X AND NOT Y AND Z
```

デフォルトの優先順位規則を使用すると、上記のストリングは次のストリングと同じです。

```
W OR ( X AND (NOT Y) AND Z )
```

括弧を使用することにより、優先順位を変更したり、ストリングの意味を変えることができます。たとえば、次のようになります。

```
(W OR X) AND NOT (Y AND Z)
```

注: 論理演算子の入力は、大文字、小文字、または大文字小文字混合でもかまいません。

検索式

各検索式は、属性、演算子、値、要素、意味 の形式をとります。

属性

以下の形式の文字ストリングです。

```
Annn
```

この文字ストリングの各フィールドには、以下の意味があります。

A 属性 (attribute)。属性の入力は、大文字、小文字、または大文字小文字混合でもかまいません。

nnn 10 進数の属性 ID。この値は、Content Manager for iSeries に存在する属性がユーザー定義属性であるかシステム定義属性であるかを識別するためのものです。

演算子

関係演算子。論理演算子は大文字小文字どちらでも、また大文字小文字混合でも入力することができます。有効な演算子を以下に示します。

演算子	意味
EQ または ==	等しい
LEQ または <=	より小さいかまたは等しい
GEQ または >=	より大きいかまたは等しい
LT または <	より小
GT または >	より大きい
NEQ または <>	等しくない
IN	値のリストの中にある
NOTIN	値のリストの中にない
LIKE	類似している
NOTLIKE	類似していない
BETWEEN	2 つの値の間である
NOTBETWEEN	2 つの値の間でない

値

ストリング値、数値、または値 NULL。

ストリング値は引用符で囲む必要があります。長さ 0 のストリングを指定する場合には、引用符を 2 つ続けて指定します。2 つのブランクから成るストリングを指定する場合は、 2 つの引用符の間に 2 つのブランクを使用します。長さ 0 のストリングも、 2 つのブランクから成るストリングも、値 NULL と等価ではないことに注意してください。

数値の前には正符号 (+) または負符号 (-) を付けることができます。任意で、数値をストリングとして指定することも可能です。

値 NULL を指定するには、予約語 ″null″ を使用します。値 NULL が指定できるのは演算子 EQ と NEQ に対してのみです。次に示すのが有効な値の例です。

```
"XXXXX"
null
"123"
+123
123
```

注: 値 ″123″、+123、および 123 は等価です。

検索の関係演算子

以下の関係演算子を使用する際には、値のストリングを特定の形式で指定する必要があります。

- BETWEEN
- NOTBETWEEN
- LIKE
- NOTLIKE
- IN
- NOTIN

BETWEEN 演算子または NOTBETWEEN 演算子を使用する際には、式の中の値ストリングをすべて同じ形式で指定する必要があります。次に示すのが有効な式の例です。

```
A1 BETWEEN 100 200
A51 BETWEEN '1995-01-01' '2020-09-29'
A49 BETWEEN   '1900-01-01-00.00.00.000000'  '1920-02-02-00.00.00.000003'
A50 BETWEEN   '13.00.00'  '17.00.00'
A2 NOTBETWEEN "FIRST" "LAST"
```

LIKE 演算子または NOTLIKE 演算子のいずれかを使用する場合は、SQL 形式にパーセント記号 (%) または下線文字 (_) を使用して、部分ストリングの検索を指定します。

パーセント記号を指定すると、任意の文字を表すことができます。たとえば、次の式は文字 S で始まる任意の値を検索します。

```
A3 LIKE "S%"
```

下線文字を指定すると、特定の位置にある任意の文字を表すことができます。たとえば、次の式は文字ストリング PA で始まり、3 文字目に任意の文字を含み、4 文字目の最後の位置に文字 K を含む任意の値を検索します。

```
A8 LIKE "PA_K"
```

IN 演算子または NOTIN 演算子のいずれかを使用する場合、ストリング値をアポストロフィ (') で囲み、全体の値を小括弧で囲む必要があります。また、式の中の 2 つの値の間にコンマ (,) を入れる必要があります。次に示すのが有効な式の例です。

```
A4 IN "('Monday','Tuesday','Wednesday')"
A50 NOTIN "('15.30.03') "
A51 NOTIN "('1994-08-31') "
A49 NOTIN "('1920-02-02-00.00.00.000001') "
A5 NOTIN "(1,3,5,7,9)"
```

式に指定した属性が、このデータ構造の中で式用にユーザーが指定した索引クラスに属していない場合、検索メソッドは失敗します。このような場合には、式の中に正しい構造の部分があってもなくても、関数は失敗します。

次の例では、指定した索引クラスに含まれている属性が属性 10 と属性 12 だけであると、関数は失敗します。

```
(A12 == 3) OR (A38 < 5)
```

上の例の式は、指定した索引クラスに属性 38 が含まれていないために、メソッドが失敗してしまいます。

索引クラスの値としてヌル・ストリング ("") を指定すると、メソッドは検索ストリング内の式に指定した属性を含む索引クラスだけを自動的に検索します。その式がシステム属性 ID だけからなっている場合、関数はすべての現行索引クラスを検索します。

プロセス/ロケーションの検索

プロセスとロケーションは、検索範囲に指定できる唯一のシステム定義属性です。それぞれに関連付けられている属性 ID は、 SIM_INDEX_ATTR_PROCESS と SIM_INDEX_ATTR_LOCATION です。検索範囲にプロセスとロケーションを指定するには、最初の検索式にプロセス基準を指定する必要があります。ロケーションの指定はオプショナルですが、ロケーションを指定する場合は 2 番目の検索式にロケーション基準 (ロケーション・タイプを含む) を指定する必要があります。検索式の値要素には、有効なプロセスまたはロケーションの ID が含まれるようにしてください。また、この検索式に演算子を使用することはできません。有効なロケーション・タイプは、SIMWM_WORKBASKET と SIMWM_COLLECTION_POINT です。たとえば、次のようになります。

```
A-20 "PAPPLICANT " A-21 3 "WWORK05    "
```

付録 B. 事前定義済みコンテンツ・クラス

表 25 は、Content Manager for iSeries についての事前定義済みコンテンツ・クラスをリストしています。

表 25. 事前定義済みコンテンツ・クラス

コンテンツ・クラス	説明
SIM_CC_ADVWRITE	HP AdvanceWrite Plus 形式
SIM_CC_AIX_EXE	AIX® 実行可能プログラム
SIM_CC_AIXCMD	AIX コマンド・ファイル
SIM_CC_AMIPRO	Ami Pro 形式
SIM_CC_AOCA	オーディオ・オブジェクト・コンテンツ・アーキテクチャー (AOCA) データ専用
SIM_CC_ASCII	フラット ASCII テキスト
SIM_CC_BCOCA	タイル・バー・コード・オブジェクト・コンテンツ・アーキテクチャー (BCOCA) データ専用
SIM_CC_BKMGR_READ	BookManager® 読み取り形式
SIM_CC_BINARY	不定様式バイナリー・データ
SIM_CC_DESCRIBE	DeScribe テキスト・エディター
SIM_CC_DIGITAL	Digital DX および WPS-Plus 形式
SIM_CC_DWRITE	DisplayWrite®
SIM_CC_EBCDIC	フラット EBCDIC テキスト
SIM_CC_ENABLE	Enable 形式
SIM_CC_EXCEL	Microsoft Excel
SIM_CC_FAXGRP3	グループ 3 形式のファクシミリ・イメージ
SIM_CC_FRN_NOTE	アプリケーションの注ログ
SIM_CC_FRN_HISTORY	アプリケーションのヒストリー・ログ
SIM_CC_FWORK	Framework 形式
SIM_CC_GOCA	グラフィックス・オブジェクト・コンテンツ体系 (GOCA) データ専用
SIM_CC_IBMFFT	DCA - Final Form テキスト
SIM_CC_IBMWA	IBM Writing Assistant
SIM_CC_INTER	Interleaf Publisher 形式
SIM_CC_IOCA_FS11	イメージ・オブジェクト・コンテンツ・アーキテクチャー (IOCA) データ専用
SIM_CC_IOCA_IRM	IOCA、IRM バージョン (規格外)
SIM_CC_IOCA_TILED	タイル IOCA 専用
SIM_CC_LEGACY	Legacy 形式
SIM_CC_MacWrite	MacWrite 形式
SIM_CC_MASS	MASS 11 形式

表 25. 事前定義済みコンテンツ・クラス (続き)

コンテンツ・クラス	説明
SIM_CC_MGDS	IBM マシン生成データ・ストリーム (MGDS) 形式 (たとえば、書式用)
SIM_CC_RICHTEXT	Microsoft Rich Text 形式
SIM_CC_MODCA_FORM	混合オブジェクト文書コンテンツ・アーキテクチャー (MO:DCA) の書式オーバーレイ構造
SIM_CC_MODCA_IS2	MO:DCA-P 文書
SIM_CC_MODCA_PAGE	MO:DCA ページ構造専用
SIM_CC_MSCRIPT	Lotus® Manuscript 形式
SIM_CC_MULTIMATE	Multimate** および Multimate/Advantage** 形式
SIM_CC_MSTSOFT	Mastersoft 内部形式
SIM_CC_OFSWRITE	Office Writer
SIM_CC_OS2EXE	OS/2® バージョン 2 の実行可能プログラム
SIM_CC_OS2CMD	OS/2 バージョン 2 のコマンド・ファイル
SIM_CC_OS2DLL	OS/2 バージョン 2 のダイナミック・リンク・ライブラリー (DLL)
SIM_CC_OS2V12_BMP	OS/2 バージョン J1.2 のビットマップ
SIM_CC_OS2V13_BMP	OS/2 バージョン J1.3 のビットマップ
SIM_CC_OS2V2_BMP	OS/2 バージョン J2.0 のビットマップ
SIM_CC_PCX	PCX
SIM_CC_PEACH	PeachText 5000 形式
SIM_CC_PFS	PFS:First Choice 形式
SIM_CC_POSTSCRIPT	PostScript データ
SIM_CC_PPDS	プリンター・データ・ストリーム
SIM_CC_PRS	フリーランス表示
SIM_CC_PWRITE	Professional Write 形式
SIM_CC_QAWRITE	QA Write 形式
SIM_CC_QUATTRO	Quattro Pro 形式
SIM_CC_RFILE	Rapid File 形式
SIM_CC_RFT	IBM RFT:DCA
SIM_CC_TARGA	TARGA
SIM_CC_TEXT	テキスト (コード・ページは認識されないか、可変である)
SIM_CC_TIFF_G3_FINE	タグ・イメージ・ファイル・フォーマット (TIFF) ヘッダー、高解像度ファクシミリ
SIM_CC_TIFF_G3_STANDARD	TIFF ヘッダー、標準ファクシミリ
SIM_CC_TIFF_IRM	TIFF の IRM バージョン、単一ページ
SIM_CC_TIFF_SINGLE_STRIP	単一ストリップのラスター
SIM_CC_TIFF5	TIFF V5、許可された複数ページ
SIM_CC_TIFF5_PAGE	TIFF V5、単一ページ
SIM_CC_TIFF6	TIFF V6、複数ページ
SIM_CC_TIFF6_PAGE	TIFF V6、単一ページ

表 25. 事前定義済みコンテンツ・クラス (続き)

コンテンツ・クラス	説明
SIM_CC_TOTALWORD	Total Word 形式
SIM_CC_UNIPLEX	Uniplex onGo 形式
SIM_CC_UNKNOWN	コンテンツ・クラス不明
SIM_CC_USER	ユーザー定義のコンテンツ・クラスの開始
SIM_CC_VKS	Volkswriter 形式
SIM_CC_WANGPC	WANG PC 形式
SIM_CC_WG1	Lotus 1-2-3/G によるグラフィックス
SIM_CC_WINV3_BMP	Microsoft Windows バージョン 3 のビットマップ
SIM_CC_WINWRITE	Windows Write 形式
SIM_CC_WKS	Lotus スプレッドシート形式
SIM_CC_WORD	Microsoft Word 形式
SIM_CC_WORDSTAR	Wordstar 形式
SIM_CC_WP	WordPerfect 形式
SIM_CC_WRITENOW	WriteNow 形式
SIM_CC_XYWRITE	XyWrite 形式

付録 C. 外部参照

Content Manager for iSeries をご使用のお客様の多くは、 iSeries 上やネットワーク内に別のデータ・リポジトリーを所有しており、 Content Manager for iSeries Windows クライアントとプログラミング・インターフェースを使用してこれらのデータへアクセスできるようにしたいという希望を持っています。こうした*外部文書*は、検索、フォルダーへの追加、ワークフロー内への組み込みなどを含め、 Content Manager for iSeries 文書と全く同じように扱われる必要があります。エンド・ユーザーが文書のロケーションを知らなくても済むように、また、そのコンテンツを管理しているのが Content Manager for iSeries ではないということを意識しなくて済むようにする必要があります。

Content Manager for iSeries 文書と同様に外部文書にもアクセスできるようにするという要件に対応するため、External Reference のサポートが提供されるようになりました。 External Reference は、単に Content Manager for iSeries が既に使用している索引付け情報に、別のアプリケーションが文書のコンテンツを検索するために必要とするロケーション情報を追加したものです。最も単純なものは、iSeries ファイル・システム内、またはワークステーションからアクセス可能なネットワーク・ドライブ上に保管されているファイルのパス名などになります。

Content Manager for iSeries に対して External Reference を定義するには、まず、ロケーション情報と Content Manager for iSeries の索引付け情報 (索引クラス、キー・フィールド、およびコンテンツ・クラスなど) を含むファイルを作成する必要があります。このファイル内の各レコードは、Content Manager for iSeries に索引付けされる、1 つの文書を表します。それぞれの文書ごとに Content Manager for iSeries API を呼び出す代わりに、一度にファイル内のすべての文書を索引付けすることによって、処理時間が最小化されます。モデル 510 の場合、1000 個の文書の索引付けに要する時間は約 7 秒です。

以下の 4 つのタイプの External Reference がサポートされています。

- OS/400 ファイル
- ワークステーション・ファイル (またはネットワーク・ファイル)
- サーバー上で呼び出されたプログラムの検索したデータ
- ワークステーション上で呼び出されたプログラムの検索したデータ

External Reference は、索引付けを行うと、 Content Manager for iSeries API を介してアクセスすることが可能になります。これらの API は、現在、 Content Manager クライアント、Content Connect クライアント、およびビジネス・パートナーやお客様の作成したアプリケーションによって使用されています。これらのアプリケーションは、文書が他のロケーションに保管されていても Content Manager for iSeries によって索引付けされていれば、これに透過的にアクセスできるようになります。

このソリューションでは、Content Manager Client のコンテンツ・クラス機能が鍵となります。 Content Manager Client が文書をオープンする際、その文書に関連付けられているコンテンツ・クラスによって、文書を Content Manager Viewer で表示するか、あるいは別のアプリケーションに渡すかが制御されます。例えば、ビデオ・

クリップやオーディオ・クリップがインポートされる場合、ユーザーは、コンテンツ・クラスを AVI と指定します。この文書をオープンする際、Content Manager Client は MPLAY32 を開始してビデオを再生します。コンテンツ・クラス機能を使用することで、 Content Manager for iSeries によってあらゆるタイプの文書に索引付けを行い、検索インターフェースを介して検索して、Content Manager Viewer または代替プログラムのいずれかで表示することが可能となります。

External Reference には、数多くの用途があります。例えば、多数の文書 (イメージ、ビデオ・クリップ、テキストなど) を CD や DVD に保管し、全ユーザー用にそれを複製してから、これらの文書を Content Manager for iSeries に索引付けすることができます。各ファイルへのパス名を保管しておくことにより、ユーザーは、ダイヤルアップ接続を使用している場合でも文書をす早く検索することができます。 iSeries 上のファイルや LAN 接続されたドライブ上のファイルにも、同じ索引付けの方法を使用することができます。

さらなる柔軟性を提供するため、ワークステーション上または iSeries 上のどちらでもプログラムを呼び出してデータを検索することができます。例えば、Content Manager for iSeries に索引付けされた文書が、従業員レコードを参照しているとします。呼び出されたプログラムは複数のデータベースから情報を収集して、ワークステーションに返されるシンプルなテキスト表記やイメージ、さらには HTML 文書などを作成することができます。 Content Manager クライアントはコンテンツ・クラスを使用して、結果を直接表示するか、または、文書を別のプログラムに渡して表示させます。

External Reference のサポートを使用することで、あらゆる情報を Content Manager for iSeries で索引付けし、 Content Manager Client を介して検索、管理、および表示することができます。企業内のすべての文書に対する単一の中央索引を保守することができ、文書を動的に構成するための柔軟性が向上します。

外部参照の作成

1 つ以上の文書を外部参照として索引付けするには、ファイル EKD0314 内にレコードを作成してから、索引付けプログラム QVIXRFINX を実行します。 EKD0314 には、以下のフィールドが定義されています。

INDEX CLASS
これは、文書が索引付けされる Content Manager for iSeries の索引クラスの名前です (記述ではありません)。指定された索引クラスが存在していない場合は、後で作成することができます。 (索引クラスが存在しないと、 Content Manager for iSeries API を介して文書を検索したり、この API を使用してアプリケーションから文書を検索することができません)。

KEY1-KEY8 DATA
これらのフィールドには、文書を索引付けするための属性が含まれています。これらは、EKD0314 で指定されたのと全く同じものが、そのまま EKD0312 (索引付けファイル) に書き込まれます。

CREATE DATE、CREATE TIME、USER ID
これらのフィールドは、それぞれの値がそのまま書き込まれます。

CONTENT CLASS
Content Manager Client によって直接処理できるタイプの文書の場合には、 0

を使用します。それ以外の場合には、EKD0318 を参照して適切なコンテンツ・クラスを探してください。コンテンツ・クラスが存在しない場合には、DFU または他のユーティリティーを使用して、コンテンツ・クラスを定義してください。現在、コンテンツ・クラスを定義するための管理インターフェースは存在しません。

EXTERNAL REFERENCE TYPE

以下の 4 つのタイプがサポートされています。

1 External Reference フィールドには、iSeries 上の別のプログラム (Object Handler) がデータの検索に使用する情報が含まれます。この例では、完全修飾の iSeries ライブラリー名、ファイル名、およびメンバー名がプログラム QVIXRFSMP に渡されます。

2 External Reference フィールドには、完全修飾の iSeries パスが含まれます。これは、前出のライブラリー/ファイル/メンバーか、または IFS ファイルの名前になります。

3 External Reference フィールドには、ワークステーション上の別のプログラム (Object Handler) がデータの検索に使用する情報が含まれます。指定するプログラムは、以下の原型を持つ関数名 `vi400extref` が含まれている DLL でなければなりません。

```
int _Export _System  vi400extref (
          HSESSION    hSession,     /* CM for iSeries session handle */
          char        * extref,     /* External reference string,
                                       converted to workstation
                                       code page.                    */
          FILE        * filehandle) /* Handle of file to receive
                                       document content.  Must not
                                       be closed.                    */
```

タイプ 3 の参照の場合、オブジェクト・ハンドラーは、ワークステーション DLL の名前です。 SimLibOpenObject でエラーが発生した場合には、非ゼロの戻りコードが戻されます。

4 External Reference フィールドには、ワークステーションからアクセス可能な、完全修飾パス名が含まれます。

EXTERNAL REFERENCE

ロケーション・データ (Object Handler に渡される情報またはファイル名のいずれか) です。

OBJECT HANDLER LIBRARY

タイプ 1 の参照の場合のみ使用します。Object Handler プログラムが含まれている iSeries ライブラリーの名前です。

OBJECT HANDLER PROGRAM

タイプ 1 の参照の場合に使用します。External Reference を受け取る iSeries プログラムの名前です。これは、以下の構造を入力として受け取る、スタンドアロン・プログラムです。

RCAREA CHAR(8)

エラーを示すために、非ブランクの戻りコードが EKD0080 に書き込まれます。

FILENAME CHAR(256)

Content Manager for iSeries API を介して戻されるコンテンツは、指定された一時ファイルに書き込まれる必要があります。

EXTREF CHAR(256)

文書のコンテンツを見付けるために使用される、外部参照またはロケーション情報です。

ITEM ID

このフィールドは、初期状態においてブランクでなければなりません。このフィールドは、索引付けプロセスによって作成された項目 ID に設定されます。非ブランクの場合、レコードは既に索引付けされているものとみなされ、QVIXRFINX ではスキップされます。

EKD0314 を作成すると、索引付けプログラム QVIXRFINX を実行できます。このプログラムは、コマンド行または別のプログラムから呼び出すことができます。すべての必要ファイルがオープンされ、必要な数の文書 ID が予約されて、各文書が索引付けされます。プログラム障害が発生した場合は、QVIXRFINX を再始動して、まだ索引付けされていないレコードのみを処理することができます。

制限事項: 最良のパフォーマンスを得るために、文書はこのバッチ・アプローチを使用して索引付けされます。現在、こうした文書を別のアプリケーションから索引付けするための API は提供されていません。セキュリティー検査は行われないため、QVIXRFINX の権限を付与するユーザーを限定する必要があります。処理時に、フィールドの妥当性検査は行われません。

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あり、米国以外の国においては本書で述べる製品、サービス、またはプログラムを提供し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。

〒106-0032
東京都港区六本木 3-2-31
IBM World Trade Asia Corporation
Licensing

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
J74/G4
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

本プログラムに関する上記の情報は、適切な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、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、さまざま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。

商標

以下は、IBM Corporation の商標です。

IBM
Advanced Peer-to-Peer Networking
AIX
Application System/400
APPN
AS/400
BookManager
C/400
CICS
COBOL
DisplayWrite
iSeries
Operating System/2
Operating System/400
OS/2
OS/400
Redbooks
RPG/400

Lotus および 1-2-3 は、IBM Corporation の商標です。

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。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名などはそれぞれ各社の商標または登録商標です。

用語集

この用語集には、本書およびプロダクトの資料ライブラリーの中で使用されている用語および略語についての定義を記載しています。この用語集に記載されていない用語または略語については、「*IBM コンピューティング辞典*」(ZZ88-0118-09) を参照してください。

この用語集では、次の相互参照を使用しています。

- **～と対比**。反対の意味または実質的に異なる意味を持つ用語を示します。
- **～を参照**。この用語が含まれている複合語を参照する場合に使用します。
- **～も参照**。同義語ではないが、類似した意味を持つ用語を参照する場合に使用します。
- **～と同義**。その用語と同じ意味で、それよりも望ましい用語の項目に定義があることを示します。

［ア行］

アウトバウンド (outbound). アプリケーション・プログラムから装置への伝送に関する用語。*インバウンド (inbound)* と対比。

アクション・リスト (action list). スーパーバイザーによって定義されたアクションの承認リスト。ユーザーはワーク・バスケットの項目を処理しながら、これらのアクションを実行できる。

アクセス・リスト (access list). 1 つ以上の個別ユーザー ID またはユーザー・グループと、各ユーザー ID またはユーザー・グループに関連づけられた*権限セット*から構成されているリスト。アクセス・リストを使用して、Content Manager for iSeries における項目へのユーザー・アクセスを制御する。アクセス・リストに関連づけることができる項目は、*索引クラス*、*ワーク・バスケット*、および*プロセス* である。

後入れ先出し法 (last in first out (LIFO)). キュー技法の 1 つで、キューに最も新しく入れられた項目から先に取り出す方法。

アプリケーション・プログラマー (application programmer). ユーザーのシステムのためのプログラミング・システムおよびその他のアプリケーションを設計するプログラマー。

アプリケーション・プログラム・インターフェース (application program interface (API)). IBM システム制御プログラムまたはライセンス・プログラムと、そのプログラムのユーザーとの間の、正式に定義されたプログラミング言語インターフェース。

イメージ (image). (1) 単一のページの情報。1 枚の紙を、スキャンまたはディジタル化することによって生成される。 (2) 光、音、電子放射、あるいはピクチャーから生じたかまたはピクチャーによって反射された他の放射をセンシングすることによって生成された、ピクチャーの電子表現。イメージは、既存のピクチャーを参照せずにソフトウェアによって直接生成することもできる。ページ・イメージ *(page image)* も参照。

イメージ・オブジェクト・コンテンツ・アーキテクチャー (Image Object Content Architecture (IOCA)). イメージの交換および表示に使用する構成要素の構造化されたコレクション。

イメージ・データ (image data). イメージを定義するラスター情報の長方形の配列。多くの場合、イメージ・データはスキャン処理によって作成される。

イメージ・ホスト (image host). スキャンされてインポートされた文書が永続的に保管されるシステム。*光ディスク・ライブラリー・サブシステム (optical library subsystem)* も参照。

イメージ・ワークステーション (image workstation). イメージ機能を実行できるプログラマブル・ワークステーション。

インスタンス (instance). プロセス内での作業パッケージのオカレンス。このプロセスが複数の並列経路から構成されている場合には、1 つの作業パッケージについて複数のインスタンスが存在する。

インバウンド (inbound). 外部ソースからアプリケーション・プログラムに向けた通信に関する用語。端末からアプリケーション・プログラムへの伝送など。*アウトバウンド (outbound)* と対比。

インポート (importing). ファイルを使用して文書が iSeries に入力される処理。ただし、スキャン処理は除

く。Content Manager for iSeries 内のインポートされた文書は、スキャンされた文書と同様に、 DASD および光ディスクに保管し、また、表示しプリントすることができる。

エクスポート (export). システム・フォルダー内の文書のデータをファイルに書き込むために使用する処理。エクスポートおよびインポート処理は、システム間で文書を転送するために使用することができる。

オブジェクト (object). (1) アクションの実行対象にできる項目。

1 つの名前で参照されるデータの集まり。

システム内で最小の単位。Content Manager for iSeries システムの場合、これは通常、単一イメージの文書である。 (2) オブジェクト・サーバー上に保管される任意のバイナリー・データ・エンティティー。Content Manager for iSeries データ・モデルでは、*オブジェクト* は特に文書の内容または部分を指す。

オブジェクト権限 (object authority). オブジェクトを使用または制御する権利。

オブジェクト・サーバー (object server). クライアント・アプリケーションが保管およびアクセスするオブジェクトまたは情報を、物理的に格納する IBM Content Manager for iSeries の構成要素。

オブジェクト・ディレクトリー (object directory). イメージ文書記憶に使用される iSeries オブジェクト・ディレクトリーを識別するために、Content Manager for iSeries で使用される制御ファイル。

オペレーター (operator). システム管理用タスクを毎日処理する人。

[カ行]

カートリッジ (cartridge). (1) 磁気テープ、繰り出しリール、および巻き取りリールからなる記憶装置。保護カバーに入っている。 (2) 光ディスク記憶の場合、光ディスクを収め保護するプラスチック・ケース。光ディスク装置への挿入が可能。*光ディスク (optical disk)* および*カートリッジ保管スロット (cartridge storage slots)* も参照。

カートリッジ保管スロット (cartridge storage slots). カートリッジを保管する光ディスク・ライブラリーの保管域。

解像度 (resolution). コンピューター・グラフィックスにおいて、イメージの明瞭度の基準のことで、表示画面上の行数と列数あるいは、区域単位当たりのペル数で表される。

回転 (rotate). 文書表示ウィンドウおよびスキャン文書表示ウィンドウの機能の 1 つ。方向付けは選択されたオプションに応じて決まる。

拡張対等通信ネットワーク機能 (APPN) (Advanced Peer-to-Peer Networking (APPN)). 直接には接続されていない複数の APPC システム間のネットワークにおいてデータを経路指定するデータ通信サポート。

拡張プログラム間通信機能 (APPC) (advanced program-to-program communications (APPC)). iSeries サーバー上のプログラムが互換性のある通信サポートを持つ他のシステム上のプログラムと通信できるようにするためのデータ通信サポート。この通信サポートは、 SNA LU セッション・タイプ 6.2 プロトコルを使用する iSeries の通信手段である。

カスタマイズ (customization). データ処理インストール・システムまたはネットワークを、特定ユーザーの要件に合わせて設計変更する処理。

各国語サポート (national language support (NLS)). 米国英語版のプロダクトを、他の言語または地域の要件に合うように変更もしくは変換すること。これには、プロダクトの使用可能化または改良のほかに、ノーメンクレチャー、MRI、またはプロダクト資料の翻訳が含まれる。

管理者 (administrator). システム管理、制御、セキュリティー、および、ケース統計を担当する人。システム管理者と同義。

キーワード (keyword). パラメーターを識別する名前または記号。

キー・フィールド (key field). 項目の属性の 1 つで、その項目に関する情報のタイプを表す。たとえば、顧客データ項目には、その顧客の名前および社会保障番号に関するキー・フィールドが入っている場合がある。

グループ 3 (Group III). 国際電信電話諮問委員会 (CCITT) によって公布された規格に準拠する圧縮アルゴリズム。

経路 (route). ワーク・バスケット、コレクション・ポイント、および決定ポイントの間で処理を移動させる一連のステップ。

決定ポイント (decision point). (1) 各作業パッケージ内の特定の情報に応じて、作業パッケージが現行経路上

で処理を続けるか、代替経路に切り替えるポイントのこと。決定ポイントは、変数名、値、および経路から構成されるテーブルである。 (2) 決定ポイントは、作業プロセスの一部である。たとえば、決定ポイントは、『新規口座の開設』 という作業プロセスの一部である作業パッケージが、信用情報に基づいて承認または否認を受け取るポイント。

コレクション・ポイント (collection point) も参照。

現行の文書 (current document). 現在処理中の文書。

言語プロファイル (language profile). 時刻形式や日付形式など、地域に固有のパラメーターを定義するために Content Manager for iSeries で使用される制御ファイル。

検索基準 (search criteria). Content Manager for iSeries において、ライブラリー・サーバーで実行される論理検索を表すために使用されるテキスト・ストリング。

光ディスク (optical). 光ディスク記憶に関する用語。

項目 (item). (1) 属性およびオブジェクトのセット (イメージ・データ、注釈、注、または他の内容が入っている 1 つ以上のファイル) であり、これらが一緒になって保険金請求やフォルダーなどの物理文書を表す。

文書 (document) も参照。 (2) ライブラリー・サーバーが管理する情報の最小単位。項目とはフォルダー、文書、ワーク・バスケット、あるいはワークフローのことを指す。ライブラリー・サーバー機能の外ではオブジェクト と呼ばれる。

コレクション (collection). 一般的に同様の性能、使用可能度、バックアップ、および保存の特性を持つオブジェクト・グループに関連付けられているストレージ管理制御の定義。

コレクション・ポイント (collection point). (1) 作業パッケージが処理を続けるために特定のイベントが発生するかまたは同期をとるのを待つポイントのこと。 (2) コレクション・ポイントは、作業プロセスの一部である。たとえば、『新規口座の開設』 という作業プロセスの一部である作業パッケージの場合、コレクション・ポイントで、信用情報が検査されるまで待たなければならない。*決定ポイント (decision point)* も参照。

混合オブジェクト文書コンテンツ・アーキテクチャー (Mixed Object: Document Content Architecture (MO:DCA)). 交換環境内または環境間において、アプリケーション間でのオブジェクト・データの交換を可能にするために開発された IBM 体系。

混合オブジェクト文書コンテンツ・アーキテクチャー - プレゼンテーション (Mixed Object: Document Content Architecture-Presentation (MO:DCA-P)). MO:DCA のサブセットとなるアーキテクチャー。Content Manager for iSeries ワークステーションに表示または印刷のために送る文書を入れるエンベロープとして利用される。

コンテンツ・クラス (content class). オブジェクトのデータ形式 (たとえば、 MO:DCA、TIFF、ASCII など) を示す数値。

コンビニエンス・ワークステーション (convenience workstation). プリンターおよびスキャナーを備えている表示ワークステーション。

[サ行]

サーバー (server). ローカル・エリア・ネットワークにおいて、他のデータ・ステーションに機能を提供するデータ・ステーション。たとえば、ファイル・サーバー、プリント・サーバー、メール・サーバーなど。

先入れ先出し法 (FIFO) (first in first out (FIFO)). 次に取り出す項目がキューに最長時間存在していた項目である、キュー技法。

作業順序 (work order). ワーク・バスケット内の作業パッケージの順序。

作業プロセス (work process). ワーク・マネージメントにおいて、作業パッケージが流れる過程の一連のステップ、イベント、および規則。作業プロセスは、作業パッケージが流れる過程で通過する経路、コレクション・ポイント、および決定ポイントの組み合わせである。

索引 (index). 文書またはフォルダーを索引クラスに関連付けて、その索引クラスの必須キー・フィールドの値を指定すること。

索引クラス (index class). オブジェクトを保管および検索するためのカテゴリーであり、キー・フィールドと呼ばれる名前付きの属性セットで構成される。Content Manager for iSeries で項目を作成する際、アプリケーションは索引クラスを割り当てて、その索引クラスで要求されるキー・フィールド値を提供しなければならない。索引クラスによって、オブジェクトの自動処理要件および記憶要件が識別される。

サブシステム (subsystem). 2 次システムまたは従属システム。あるいは、通常、制御システムから独立してまたは非同期で実行することができる、システムのプログラミング・サポート部分。

磁気記憶装置 (magnetic storage). 特定の素材の磁気特性を使用する記憶装置。

磁気テープ (magnetic tape). データを記憶できる磁性表面層を持つテープ。

磁気テープ装置 (magnetic tape device). 磁気テープ上にデータを書き込んだり、磁気テープからデータを読み取る装置。

システム管理者 (system administrator). 光ディスク・ライブラリー・サブシステム、および部門プロセッサーを管理する人。システム管理者は、問題の判別および解決を援助する。*管理者 (administrator)* と同義。

システム管理ストレージ (system-managed storage (SMS)). ストレージ管理に対する Content Manager for iSeries のアプローチ。システムはオブジェクトの場所を決定し、オブジェクトのバックアップ、移動、空間、セキュリティーなどを自動的に管理する。

システム・サポート・プログラム (System Support Program (SSP)). 他のプログラムの実行ならびにディスプレイ装置およびプリンターなどの関連装置の操作を管理する IBM ライセンス・プログラムのグループ。SSP は、ディスケットからディスクへの情報のコピーなどの、共通タスクを実行するユーティリティー・プログラムも含む。

指定変更 (override). 前のパラメーターまたは値にとって代わるパラメーターまたは値。

処理項目 (process item). 作業プロセスで組み立てブロックとして使用される項目。

スキャナー (scanner). 空間パターンを一部分ずつ調べ、そのパターンに対応するアナログ信号またはディジタル信号を生成する装置。

スキャナー・ワークステーション (scanner workstation). スキャナーを備えている表示ワークステーション。

スキャン (scanning). 文書を Content Manager for iSeries ワークステーションに入力する物理処理。文書は、スキャン後に永続的に保管することができる。

スタンドアロン (stand-alone). 他の装置、プログラム、またはシステムとは独立した操作に関する用語。

ステージング (staging). 通常、システムまたはユーザーからの要求時に、保管されているオブジェクトをオフラインまたは優先順位の低いデバイスから優先順位の高いデバイスに移動するプロセス。ユーザーが永続ストレージに保管されているオブジェクトを要求すると、作業コピーがステージング域 *(staging area)* に書き込まれる。

ストレージ (storage). データを記憶装置に収める操作。

ストレージ方式 (storage method). 光ディスクに保管するために文書をまとめる手段。

ストレージ・クラス (storage class). ストレージ・クラスは、光ディスク・システム ID と組み合わされて、文書を保管する光ディスク・ボリュームのセットを定義する。同じストレージ・クラスおよび光ディスク・システム ID を持つ文書は、同じ光ディスク・ボリューム上に保管される。

ストレージ・システム (storage system). Content Manager for iSeries におけるストレージの総称用語。

スプール・ファイル (spool file). プログラムによって印刷されるのを待っている出力データまたは処理されるのを待っている入力データが入っているファイル。

スロット (slot). (1) 取り外し可能な記憶メディアのために使用される、装置内の場所。 (2) 光ディスク・カートリッジが格納される、光ディスク・ライブラリー内のスペース。*光ディスク・カートリッジ (optical cartridge)* を参照。

制御ファイル (control files). オペレーターが実行する作業のカテゴリーとシステムが認識する文書のタイプを管理するファイルのこと。

属性 (attribute). Content Manager for iSeries API で使用される、項目 (文書またはフォルダー) に関連づけられる 1 つの値。各索引クラスは、最大 8 つの属性を持つことができる。

[タ行]

注釈 (annotation). 本文に追加された説明のためのコメントまたは注。

中断 (suspend). 規定の基準が満たされるまで、作業パッケージをワーク・バスケットに保持しておくこと。作業パッケージを中断する基準は複数指定できるため、1 つの作業パッケージについて複数の中断要求が存在する場合もある。文書作業パッケージは、特定の日付により中断することができる。フォルダー作業パッケージは、特定の日付または索引クラスを条件に指定して中断することができる。

直接アクセス記憶装置 (direct access storage device (DASD)). アクセス時間が実質上データの記憶場所に依存しない装置。

テープ (tape). *磁気テープ (magnetic tape)* を参照。

テープ・カートリッジ (tape cartridge). *カートリッジ (cartridge)* を参照。

随時経路 (ad hoc route). 定義されたプロセスの一部ではない経路。*随時経路* が開始されるのは、ユーザーが項目をワーク・バスケットに直接割り当てるときである。ユーザーは項目を割り当てなおすことにより、手操作で項目をあるワーク・バスケットから別のワーク・バスケットに経路指定する。

特権 (privilege). ユーザーが、Content Manager for iSeries に保管されているオブジェクトにアクセスしたり、そのオブジェクトに対して特定のタスクを実行できる権限認可。この特権はシステム管理者が付与する。

特権セット (privilege set). Content Manager for iSeries において、システムの構成要素および機能を用いて作業するための*特権* の集合。システム管理者は、特権セットをユーザー (ユーザー ID) およびユーザー・グループに割り当てる。

[ナ行]

ネットワーク (network). 情報の送受信のために接続されるプログラムおよび装置の編成。

ネットワーク・テーブル・ファイル (network table file). インストール時に作成されるテキスト・ファイルで、各 Content Manager for iSeries サーバーのノードごとのシステム固有の構成情報が入っている。各サーバーには、この情報を識別するネットワーク・テーブル・ファイルがなければならない。ネットワーク・テーブル・ファイルの名前は常に FRNOLNT.TBL である。

[ハ行]

バイナリー・ラージ・オブジェクト (binary large object) (BLOB). 1 つのオブジェクトとして処理される、大きなストリームのバイナリー・データ。

パン (pan). イメージが横に移動する視覚印象を与えるために、表示イメージ全体を徐々に移動させること。

光ディスク (optical disk). 光学式技法で読み込むことが可能なディジタル・データを含んだディスク。ディジタル光ディスクと同義。

光ディスク記憶サポート (Optical Storage Support). スタンドアロン光ディスク・ドライブ、光ディスク・ライブラリー、および Content Manager for iSeries との間の通信をサポートするソフトウェア。このソフトウェアは、光ディスク制御機構として機能するシステム/36™ 5363 装置上で実行される。

光ディスク・カートリッジ (optical cartridge). 保護カバーに入った光ディスクからなる記憶装置。*カートリッジ (cartridge)* も参照。

光ディスク・システム (optical systems). イメージ・データを光ディスク・プラッターに保管するためのハードウェア。光ディスク・ライブラリーは、直接接続された光ディスク・システムのみに入れられる。

光ディスク・システム・プロファイル (optical system profile). 文書の光ディスク記憶に使用する光ディスク制御機構を定義するために使用されるファイル。

光ディスク・ドライブ (optical drive). 光ディスクにあるデータのシーク、読み取り、書き込みをするときに使用する機構。光ディスク・ドライブは、光ディスク・ライブラリーの中の装置として、またはスタンドアロン装置として存在することができる。

光ディスク・ボリューム (optical volume). 光ディスク格納データを収めた両面の光ディスクの片面。

光ディスク・ライブラリー (optical libraries). イメージ・データを光ディスク・プラッターに保管するためのソフトウェア。光ディスク・ライブラリーは、直接接続された光ディスク・システムのみに入れられる。

光ディスク・ライブラリー・サブシステム (optical library subsystem). イメージ・データを長期保管できるようにするハードウェアおよびソフトウェア。*イメージ・ホスト (image host)* も参照。

表示テキスト・オブジェクト・コンテンツ体系 (Presentation Text Object Content Architecture (PTOCA)). 表示テキスト・データの交換を可能にするために開発されたアーキテクチャー。

表示ワークステーション (display workstation). 既に iSeries システムにスキャンまたはインポートされた文書の表示に主に使用されるイメージ処理ワークステーション。

フォルダー (folder). Content Manager for iSeries において、他のフォルダーまたは文書を入れることのできるオブジェクト。

フォルダーの平衡化 (folder balancing). iSeries において、文書をシステム内の使用可能なフォルダーに均等に分散する処理。

フォルダー・マネージャー (folder manager). Content Manager for iSeries 以外の IBM Content Manager for iSeries システムにおいて、データ・モデルおよび API のサブセットを記述するために使用される用語。Content Manager for iSeries では、この用語は Content Manager for iSeries API の全セットを指す。

プラッター (platter). *光ディスク (optical disk)* を参照。

プリンター・ワークステーション (printer workstation). プリンターを備えている表示ワークステーション。

プログラム一時修正 (program temporary fix (PTF)). IBM 技術員が、プログラムの現行リリースにある未訂正の欠陥が原因であると診断した問題を、一時的に解決または回避すること。

プロセス (process). 作業パッケージが順次流される一連のステップ、イベント、および規則。プロセスは、定義済みタイプの作業パッケージが流れる過程で通過する経路、コレクション・ポイント、および決定ポイントの組み合わせである。

例えば、「新規口座の開設」というプロセスの内容は、以下のようになる。

- 新規口座の開設に関連する作業パッケージが従う必要のあるステップ
- 新規口座の作業パッケージをシステム内の別のポイントへ経路指定するときの条件となるイベント (信用情報の検査など)
- その特定の顧客に関する情報 (信用格付けが高いか低いかなど) に基づいて、新規口座を開設するかどうかを決定する判断

プロファイル (profile). システムによって実行される処理のカテゴリーと、システムによって認識されるユーザーのタイプを管理するファイル。

分散データ管理機能 (distributed data management (DDM)). システム・サポート・プログラムの機能の 1 つで、リモート・システムに常駐するファイルに対してアプリケーション・プログラムを実行することができる。

文書 (document). (1) 1 つ以上の基本部分が入っている項目。 (2) 名前付きのテキストの構造上の単位で、個々の単位として、システムやユーザーの間で保管、検索、および交換を行うことができる。オブジェクト とも呼ばれる。1 つの文書には、複数の異なるタイプの基本部分 (テキスト、イメージのほか、スプレッドシート・ファイルなどのオブジェクト) を入れることができる。

文書コンテンツ・アーキテクチャー (document content architecture (DCA)). オフィス・システムのネットワーク内でやりとりされる文書に対して情報の整合性を保証するアーキテクチャー。DCA は、文書の形式と意味を指定するときのルールを提供する。また、変更可能テキスト (変更が可能) と最終形式テキスト (変更が不可) を定義している。

ページ (page). 1 つの物理的な媒体。たとえば、縦 11 インチ横 8.5 インチの 1 枚の紙。

ページ・イメージ (page image). 物理ページの片面を電子的に表現したもの。ページ・イメージの境界は、スキャン装置の電気機械的特性、ならびに受信側データ処理システムのイメージ収集アプリケーションの仕様によって決定される。

ページ・スキャン (page scan). 物理ページ (紙) をスキャンして、そのページのビット・イメージを作成するための電気機械的プロセス。

アーカイブ (archiving). バックアップ・ファイルおよび関連したジャーナルを、通常はある一定期間保管しておくこと。

ボリューム (volume). 1 つの単位として容易に取り扱うことができる、データの特定の部分。データ・キャリアを含む。

［マ行］

マシン生成データ構造 (Machine-Generated Data Structure (MGDS)). イメージから抽出され、汎用データ・ストリーム (GDS) 形式に設定されたデータ。

［ヤ行］

ユーザー (user). Content Manager for iSeries のサービスを必要とする人。この用語は通常、Content Manager for iSeries API を使用するアプリケーション開発者ではなく、クライアント・アプリケーションのユーザーを指す。

ユーザー ID プロファイル (user ID profile). 各ユーザーごとに 1 つの項目が入っているファイル。項目には処理の適格性などの情報が含まれる。

ユーザー出口 (user exit). (1) ユーザー出口ルーチンに制御権が与えられる IBM 提供のプログラム内の 1

地点。 (2) IBM ソフトウェア製品から提供されるプログラミング・サービス。ユーザー指定のイベントが後に生じた場合にアプリケーション・プログラムに制御権を戻すサービスのため、アプリケーション・プログラムの処理の間に要求される。

ユーザー出口ルーチン (user exit routine). IBM が提供するプログラムのユーザー出口で、制御を受け取るためにユーザーが作成するルーチン。

優先順位 (priority). (1) システム・リソースを受け取る優先順位を決める、タスクに割り当てられるランク。(2) Content Manager for iSeries ワークフローでは、実行する作業の優先順位のことを指す。この優先順位により、作業パッケージの処理順序が決定する。数値が大きいほど、優先順位が高くなる。

横並び (side by side). 文書表示ウィンドウの機能であり、複数ページの文書の中から隣合わせた 2 ページを表示する。

[ラ行]

ライブラリー・サーバー (library server). 1 つ以上のオブジェクト・サーバー に保管されている項目に関する索引情報が入っている、Content Manager for iSeries の構成装置。

リリース (release). 作業パッケージの中断条件を取り除き、作業パッケージを処理できるようにすること。保留状態の作業パッケージが保留を解除されるのは、特定の条件を満たしたとき、または権限のあるユーザーが条件を変更して、中断要求の設定を手操作により解除したときである。

レンダリング (render). 通常であればイメージとは無関係なデータを、イメージとして捕らえて表示すること。Content Manager for iSeries では、ワード処理文書を表示のためにイメージとしてレンダーすることができる。

ローカル・エリア・ネットワーク (local area network (LAN)). 地理的に限定された区域内のユーザーの構内に置かれているコンピューター・ネットワーク。

[ワ行]

ワークステーション (workstation). ユーザーが入力、索引付け、および印刷を行うことができるコンピューター処理装置、イメージ表示装置、スキャナー、およびプリンター。

ワークフロー (workflow). 企業において作業プロセスおよび作業環境を定義して、作業の流れを自動化しビジネス・プロセスを制御するためのシステム。

ワーク・バスケット (workbasket). 作業パッケージを保持するコンテナー。ワーク・バスケットは、プロセス定義または随時経路の一部として使用できる。Content Manager for iSeries では、さらに処理を待つために作業パッケージを割り当てることができる、Content Manager for iSeries システム内の論理位置。

ワーク・バスケットの定義には、その内容の表示、状況、およびセキュリティーを管理する規則が含まれている。

作業パッケージ (work package). ある場所から別の場所に経路指定される作業。ユーザーは、ワーク・バスケットを介して、作業パッケージをアクセスおよび処理する。

[数字]

1 次プロセッサー (primary processor). 処理装置のグループの中の主処理装置およびその内部記憶装置で、それを介して他のすべての装置が通信する。

1 バイト文字セット (single-byte character set (SBCS)). 各文字が 1 バイトで表される文字セット。

2 次プロセッサー (secondary processor). 処理装置のグループの中で、1 次装置以外の処理装置。

2 バイト文字セット (DBCS) (double-byte character set (DBCS)). 各文字が 2 バイトで表される文字セット。日本語、中国語、韓国語などの言語の場合は、256 コード・ポイントによって表すことができる数より多くのシンボルが含まれていて、2 バイト文字セットが必要になる。各文字が 2 バイトを必要とするので、DBCS 文字の入力、表示、および印刷には、DBCS をサポートするハードウェアおよびソフトウェアが必要になる。

A

APAR. プログラム診断依頼書 (Authorized Program Analysis Report)。

API. アプリケーション・プログラミング・インターフェース (Application programming interface)。

APPC. 拡張プログラム間通信機能 (Advanced program-to-program communications)。

APPN®. 拡張対等通信ネットワーク機能 (Advanced Peer-to-Peer Networking)®。

AS/400®. Application System/400®。

D

DASD. 直接アクセス記憶装置 (Direct access storage device)。

DBCS. 2 バイト文字セット (Double-byte character set)。

DDM. 分散データ管理機能 (Distributed data management)。

H

HTML. ハイパーテキスト・マークアップ言語 (Hypertext markup language)。

I

iSeries オブジェクト・ディレクトリー・プロファイル (iSeries object directory profile). イメージ文書記憶に使用される iSeries オブジェクト・ディレクトリーを識別するために、Content Manager for iSeries で使用される制御ファイル。

L

LAN. ローカル・エリア・ネットワーク (Local area network)。

LIFO (後入れ先出し法) (LIFO (last in, first out)). キュー技法の 1 つで、キューに最も新しく入れられた項目から先に取り出す方法。

LU 6.2. システム・ネットワーク体系 (SNA) におけるセッション・タイプの 1 つで、分散処理環境で SNA 文字ストリングまたは構造化フィールド・データ・ストリームを使用して、2 つのアプリケーション・プログラム間で行われるもの。たとえば、iSeries アプリケーションとの CICS® 通信を使用したアプリケーション・プログラムなど。

M

MGDS. マシン生成データ構造 (Machine-Generated Data Structure)。

MO:DCA. 混合オブジェクト文書コンテンツ・アーキテクチャー (Mixed Object: Document Content Architecture)。

MO:DCA-P. 混合オブジェクト文書コンテンツ・アーキテクチャー - プレゼンテーション (Mixed Object: Document Content Architecture-Presentation)。

MRI. 機械可読情報 (Machine-readable information)。

N

NLS. 各国語サポート (National language support)。

O

OS/2. Operating System/2®。

OS/400. Operating System/400®。

P

PDF. ポータブル文書形式 (Portable document format)。

PTF. プログラム一時修正 (Program temporary fix)。

PTOCA. 表示テキスト・オブジェクト・コンテンツ体系 (Presentation Text Object Content Architecture)。

S

SBCS. 1 バイト文字セット (Single-byte character set)。

SMS. システム管理ストレージ (System-managed storage)。

索引

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

[ア行]

[カ行]

[サ行]

[タ行]

[ハ行]

[マ行]

[ヤ行]

[ラ行]

[ワ行]

A

C

D

W

プログラム番号: 5722-VI1

Printed in Japan

SC88-4007-01

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