System i

Conexión a System i - Desarrollo de conectores para System i Navigator

Versión 6 Release 1

System i

Conexión a System i - Desarrollo de conectores para System i Navigator

Versión 6 Release 1

Nota

Antes de utilizar esta información y el producto al que hace referencia, lea la información que figura en: “Avisos”, en la página 97.

Esta edición aplicación a la versión 6, release 1, modificación 0 de IBM i5/OS (5761–SS1) y a todos los releases y modificaciones siguientes hasta que se indique lo contrario en nuevas ediciones. Esta versión no se ejecuta en todos los modelos de sistema con conjunto reducido de instrucciones (RISC) ni tampoco se ejecuta en los modelos CISC.

Contenido

Desarrollo de conectores de System i Navigator

Con la característica de conectores de System i Navigator, puede integrar las tareas de administración del sistema y programas de cliente/servidor en un único entorno de aplicación.

Puede utilizar los conectores para consolidar aplicaciones de terceros y funciones especializadas escritas en C++, Visual Basic o Java en la interfaz de System i Navigator. Utilice esta colección de temas para ayudarle a comprender qué son los conectores, cómo crearlos o personalizarlos y cómo distribuirlos a los usuarios.

Nota: Al utilizar los ejemplos de código, aceptará los términos del "Información sobre licencia de código y exención de responsabilidad" en la página 95.

Novedades de V6R1

Lea acerca de la información nueva o cambiada significativamente de la colección de temas de Desarrollo de conectores de System i Navigator.

Nuevo campo en setup.ini

La sección de información del archivo setup.ini tiene un nuevo campo llamado EclipseHelp. Este campo indica si la aplicación de conector utiliza la plataforma de Eclipse para desarrollar la ayuda.

Cómo visualizar las novedades o cambios

Para ayudarle a ver dónde se han realizado cambios técnicos, el centro de información utiliza:

- La imagen » para marcar el inicio de información nueva o cambiada.
- La imagen « para marcar el final de la información nueva o cambiada.

En los archivos PDF es posible que vea marcas de revisión (|) en el margen izquierdo de la información nueva y modificada.

Para encontrar otra información relativa a las novedades o cambios de este release, consulte el Memorándum para los usuarios.

Archivo PDF para desarrollar conectores de System i Navigator

Esta vista le permite ver e imprimir un archivo PDF de esta información.

Para ver o descargar la versión PDF de este documento, seleccione Desarrollar conectores de System i Navigator (aproximadamente 960 KB).

Cómo guardar los archivos PDF

Si desea guardar un archivo PDF en su estación de trabajo para verlo o imprimirlo:

1. Pulse con el botón derecho del ratón sobre el enlace PDF en el navegador.
2. Pulse la opción que guarda el PDF localmente.
3. Navegue hasta el directorio en el que desea guardar el PDF.
4. Pulse **Guardar**.

Cómo descargar Adobe Reader

Necesita tener Adobe Reader instalado en el sistema para ver o imprimir estos PDF. Puede descargar una copia gratis en el sitio web de Adobe (www.adobe.com/products/acrobat/readstep.html)  .

Soporte de conectores en System i Navigator

El soporte de conectores de System i Navigator proporciona una manera cómoda de integrar sus propias funciones y aplicaciones en una única interfaz de usuario llamada System i Navigator.

Estas funciones y aplicaciones nuevas pueden tener diversas complejidades, desde incorporar sencillos comportamientos nuevos a crear aplicaciones completas. Independientemente de qué nuevas funciones proporcione el conector, su integración en System i Navigator proporciona varias ventajas importantes. Por ejemplo, crear empaquetar tareas comunes del sistema en una única ubicación en System i Navigator puede simplificar de forma significativa la administración común y las funciones operativas. Además, la interfaz gráfica de System i Navigator asegura que las funciones integradas puedan completarse fácilmente y sólo con requisitos previos mínimos de habilidades.

Qué se puede realizar con un conector

Los conectores son conjuntos de clases predefinidas y métodos que System i Navigator inicia en respuesta a una acción en particular del usuario.

Puede utilizar los conectores para agregar o modificar objetos y carpetas a la jerarquía de System i Navigator que representen herramientas y aplicaciones. Puede personalizar completamente el soporte de las carpetas y objetos añadiendo o modificando los siguientes elementos:

Menús de contexto
Utilice los menús de contexto para lanzar aplicaciones, presentar nuevos diálogos y añadir o modificar comportamientos.

Páginas de propiedades
Utilice las páginas de propiedades para dar soporte a atributos personalizados, como por ejemplo valores adicionales de seguridad. Puede añadir páginas de propiedades a cualquier objeto o carpeta que tenga una hoja de propiedades.

Barras de herramientas
Puede personalizar totalmente las barras de herramientas y los botones.

Carpetas y objetos personalizados
Puede agregar sus propias carpetas y objetos personalizados en la jerarquía de árbol de System i Navigator.

Funcionamiento de los conectores

Después de identificar el nuevo conector en el registro de Windows, System i Navigator encuentra el nuevo conector y lo instala en una nueva configuración. A continuación, el nuevo contenedor aparece en la jerarquía de System i Navigator. Cuando el usuario selecciona el contenedor, se llama al código del conector para obtener el contenido del contenedor.

System i Navigator se comunica con el conector llamando a métodos definidos en la interfaz ListManager. Esta interfaz habilita las aplicaciones para que proporcionen datos de lista al árbol de System i Navigator y para que listen vistas. Para integrar su aplicación en System i Navigator, puede crear una nueva clase que implemente esta interfaz. Los métodos en la nueva clase llaman a la aplicación existente para obtener los datos de lista.

Figura 1 muestra cómo un conector de Java que añade un nuevo contenedor al árbol de System i Navigator puede funcionar.

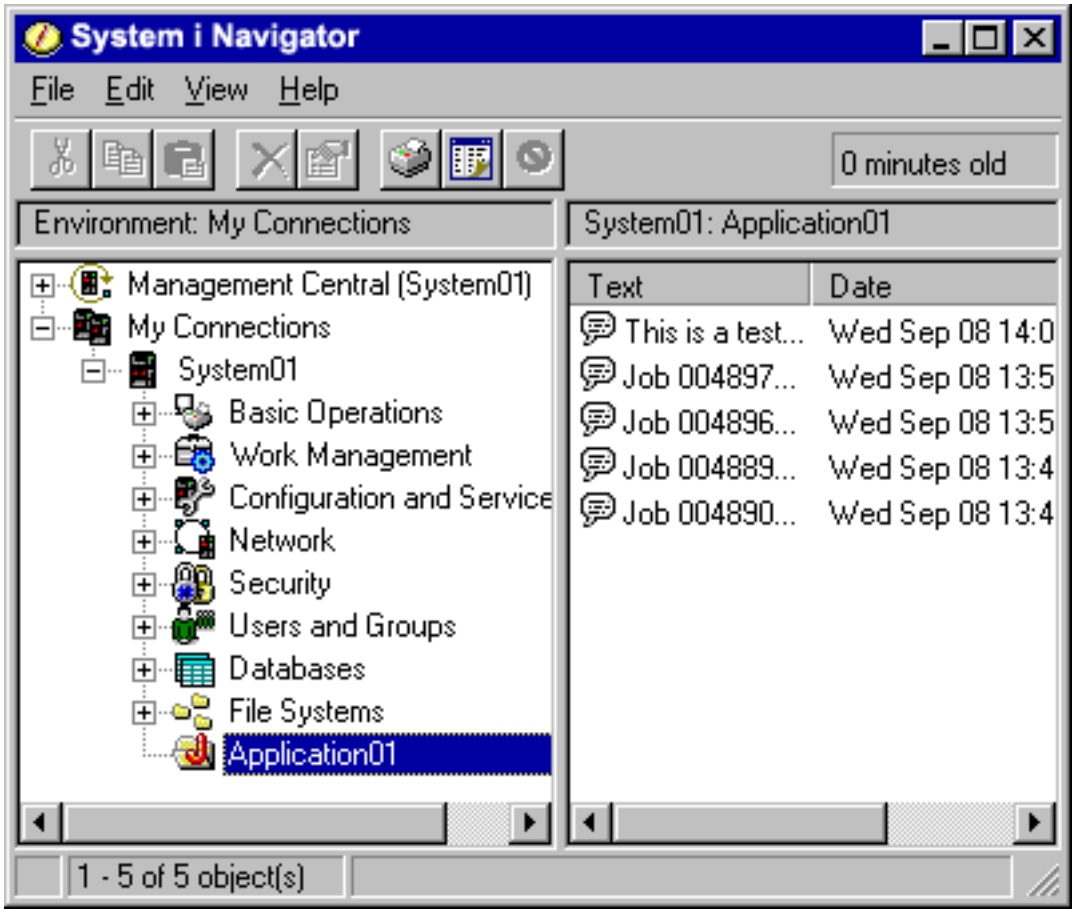

Figura 1. Diálogo de System i Navigator - mensajes en la cola de mensajes

La Figura 2 muestra cómo System i Navigator se comunica con el conector de Java para obtener los datos de lista.

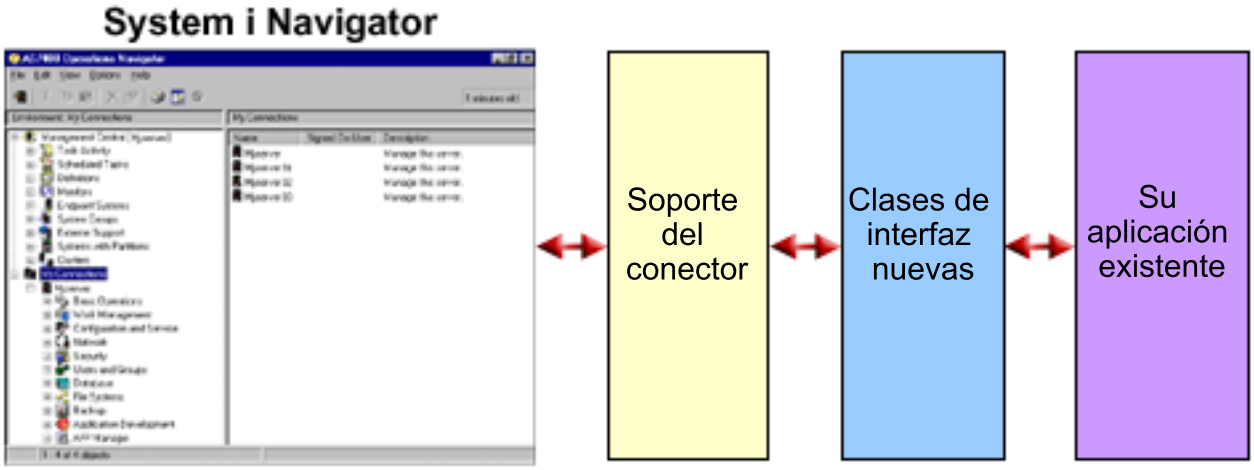

Figura 2. Cómo llama System i Navigator a una aplicación para obtener datos de lista

Utilice la interfaz ActionsManager Java para las funciones especializadas de la aplicación disponibles para los usuarios a través de System i Navigator. Cuando un usuario selecciona un elemento de menú, System i Navigator llama a otro método ActionsManager para que realice la acción (necesitará crear una nueva clase de Java para implementar esta interfaz). La implementación de ActionsManager llama a la aplicación Java existente, que muestra entonces un diálogo de confirmación o un panel de interfaz de usuario más complejo que ayude al usuario a realizar una tarea especializada.

La Figura 3 muestra qué sucede cuando un usuario pulsa con el botón derecho del ratón sobre un objeto de mensaje para mostrar su menú contextual.

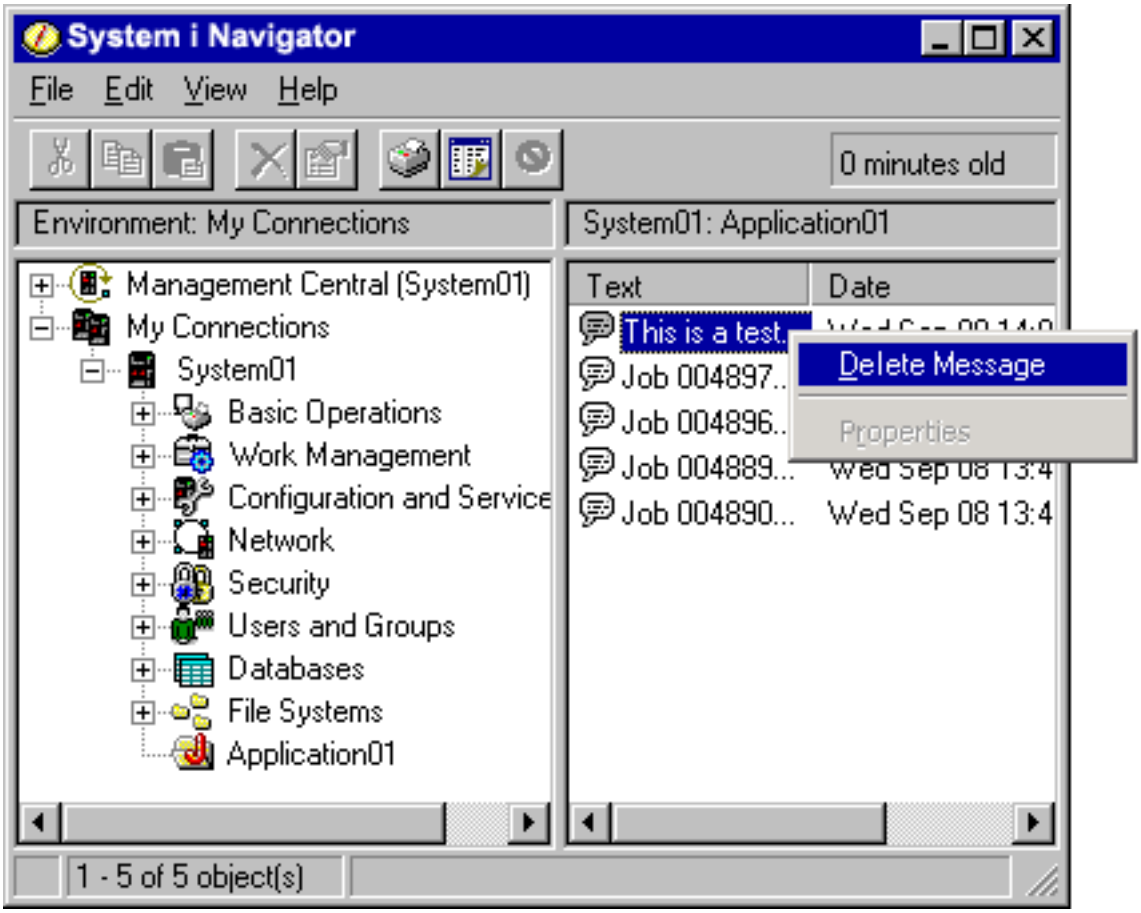

Figura 3. Menú contextual de objetos de System i Navigator

Cuando un usuario selecciona un elemento de menú, System i Navigator llama a otro método ActionsManager para realizar la acción. System i Navigator llama a un método predefinido en la interfaz ActionsManager de Java. Esta interfaz obtiene la lista de elementos de menú soportados para objetos de mensaje. La interfaz de usuario de System i Navigator está diseñada para ayudar a los usuarios a trabajar con los recursos del sistema. La arquitectura del conector refleja este diseño de la interfaz de usuario, definiendo interfaces para trabajar con listas de objetos en una jerarquía y definiendo acciones sobre dichos objetos. Una tercera interfaz, DropTargetManager, maneja las operaciones de arrastrar y solar.

Requisitos de los conectores

Los requisitos del conector de System i Navigator difieren de acuerdo con el lenguaje de programación que utilice.

Conectores C++

Los conectores que están desarrollados utilizando el lenguaje de programación Microsoft Visual C++ deben escribirse en la versión 4.2 o posterior.

Los conectores de C++ requieren también las siguientes API de System i Navigator.

Archivo de cabecera	Biblioteca de importación	Biblioteca de enlaces dinámicos
cwbun.h	cwbunapi.lib	cwbunapi.dll
cwbunpla.h (las API de Administración de Aplicaciones)	cwbapi.lib	cwbunpla.dll

Conectores Java

Los conectores de Java se ejecutan en el entorno de ejecución de IBM de Windows, Java Technology Edition. La siguiente tabla indica la versión de Java instalada con el programa con licencia de System i Access para Windows.

Release	JRE	Swing	JavaHelp
V6R1	5.0	N/A	1.1.1
V5R4	1.4.2	N/A	1.1.1
V5R3	1.4.1	N/A	1.1.1
V5R2	1.3.1	N/A	1.1.1

Todos los conectores de Java requieren una pequeña DLL de recursos de Windows que contiene información acerca de los conectores. Esto permite que System i Navigator represente la función en la jerarquía de objetos de System i Navigator sin tener que cargar la implementación del conector. La DLL de recursos de ejemplo fue creada utilizando Microsoft Visual C++ versión 4.2, pero puede utilizar cualquier compilador de C que soporte compilación y enlace de recursos de Windows.

System i Navigator proporciona una consola Java como una ayuda a la depuración. La consola se activa seleccionando un archivo del registro para escribir los indicadores de consola necesarios en el registro de Windows. Cuando se activa la consola, el compilador JIT se desactiva para permitir que los números de línea del código fuente aparezcan en el rastreo de la pila y las excepciones que se encuentren en la infraestructura Java de System i Navigator aparecerán en los recuadros de mensajes. Los archivos de registro para activar y desactivar la consola se proporcionan con el conector de ejemplo de Java en System i Access para Windows Toolkit.

La interfaz de usuario del ejemplo se ha desarrollado con la Caja de Herramientas Gráfica para Java, que forma parte del componente IBM Toolbox para Java. Toolbox es una característica de instalación opcional de System i Access para Windows. Puede instalarse con la instalación inicial del producto System i Access para Windows o selectivamente posteriormente utilización Agregar o eliminar programas en el Panel de control de System i Access para Windows.

Conectores Visual Basic

Los conectores Visual Basic se ejecutan en la versión 5.0 del entorno de ejecución Visual Basic.

Conceptos relacionados

"Instalar y ejecutar conectores de ejemplo" en la página 14
El juego de herramientas del programador suministra conectores de ejemplo de cada uno de los lenguajes de programación soportados.

Distribuir conectores

Puede entregar a los usuarios de System i Navigator el código de los conectores incluyendo el código en la aplicaciones de i5/OS.

El programa de instalación de la aplicación escribe los archivos binarios del código del conector, archivo de registro y recursos traducibles en una carpeta en el sistema de archivos integrados. Después de completar este proceso, los usuarios pueden instalar el conector pulsando sobre el mismo con el botón derecho del ratón y seleccionando **Mis conexiones → Instalar opciones → Instalar conectores**. El asistente Instalar conectores copia el código del conector a las estaciones de trabajo de los usuarios, descarga los recursos traducibles correspondientes dependiendo de los valores de idioma de las estaciones de trabajo de los usuarios y ejecuta el archivo de registro para escribir la información de registro del conector en el registro de Windows. Si no se ha instalado System i Access para Windows aún, los usuarios pueden instalar los conectores durante la instalación inicial utilizando el tipo de instalación personalizada.

Para este tipo de conector	Instálelo en este directorio	E incluya estos archivos
C++	/QIBM/USERDATA/OpNavPlugin/ <*proveedor*>.<*componente*>	• El archivo del registro del conector. • El archivo de instalación de System i Access para Windows del conector. • La DLL servidora ActiveX del conector, así como las DLL de código asociadas.
Java	/QIBM/USERDATA/OpNavPlugin/ <*proveedor*>.<*componente*>	• El archivo del registro del conector. • El archivo de instalación de System i Access para Windows del conector. • El archivo JAR de Java, que contiene todas las clases Java, AUIML, HTML, GIF, PDML, PCML y archivos de serialización.
Visual Basic	/QIBM/USERDATA/OpNavPlugin/ <*proveedor*>.<*componente*>	• El archivo del registro del conector. • El archivo de configuración de System i Access para Windows del conector. • La DLL servidora ActiveX del conector, así como las DLL de código asociadas.

Notas:

- El subdirectorio <*proveedor*>.<*componente*> debe coincidir con el especificado en el archivo del registro.
- System i Navigatorno proporciona soporte para la ubicación GUIPlugin. Deberá migrar los conectores en la ubicación GUIPlugin a la nueva ubicación OpNavPlugin.

Además, todos los conectores deberán crear al menos un directorio por debajo del subdirectorio <*proveedor*>.<*componente*> llamado MRI29*XX*, donde *XX* identifica un idioma soportado, por ejemplo MRI2924 (inglés). Este directorio debe contener la versión correcta de idioma de los elementos siguientes:

- La DLL de recursos del conector
- Los archivos de ayuda del conector
- El archivo de instalación de MRI del conector

Actualizar o desinstalar el conector

Una vez los usuarios han instalado el conector, puede elegir actualizarlo posteriormente o distribuir arreglos de errores. Una vez el código ha sido actualizado en el sistema, los usuarios podrán iniciar manualmente actualizaciones de conectores utilizando la opción **Actualizar o dar servicio a conectores** de System i Navigator.

System i Access para Windows proporciona soporte de desinstalación por lo que los usuarios pueden eliminar completamente el conector de sus estaciones de trabajo en cualquier momento. Los usuario pueden conocer qué conectores están instalados en sus estaciones de trabajo pulsando la pestaña **Conectores** en la página Propiedades de System i Navigator del sistema.

Restringir acceso al conector con Administración de aplicaciones

Puede utilizar el soporte Administración de aplicaciones basado en el sistema para controlar qué usuarios y grupos de usuarios de System i Navigator pueden acceder al conector.

Archivo setup.ini

El archivo setup.ini del conector proporciona el asistente de instalación con la información necesaria para instalar un conector de System i Navigator en una estación de trabajo cliente. También facilita información que permite al programa de comprobación del nivel de servicio determinar si el conector necesita una actualización o servicio.

El archivo de instalación debe llamarse SETUP.INI y debe residir en el directorio <*proveedor*>.<*componente*> del conector en el sistema.

El formato del archivo está en conformidad con el de un archivo de configuración estándar de Windows (.INI). El archivo está dividido en cuatro partes:

- Información del conector
- Servicio
- Secciones para identificar los archivos que deben instalarse en la estación de trabajo cliente
- Secciones para identificar los programas de salida que deben ejecutarse en la estación de trabajo cliente

Conceptos relacionados

"Directorio de archivos de conectores de ejemplo de Visual Basic" en la página 17
Estas tablas describen todos los archivos incluidos con el conector de ejemplo de Visual Basic.

Ejemplo: sección de información de setup.ini:

La primera sección del archivo setup.ini, Plug-in Info, contiene información global acerca del conector.

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

Campo de la sección [Plugin Info] de setup.ini	Descripción del campo
Name	Nombre en inglés del conector. Este nombre se visualiza durante la instalación del conector si no puede determinarse el nombre traducido.
NameDLL	Nombre de la DLL de recursos que contiene el nombre traducido del conector. Esta DLL se encuentra en los directorios de MRI del conector.
NameResID	ID de recurso del nombre traducido en la DLL de MRI. Este campo debe contener el mismo valor que el campo NameID definido en la clave primaria del registro para el conector.

Campo de la sección [Plugin Info] de setup.ini	Descripción del campo
Description	Descripción en inglés del conector. Esta descripción se visualiza durante la instalación del conector si no puede determinarse la descripción traducida.
DescriptionDLL	Nombre de la DLL de recursos que contiene la descripción traducida del conector. Esta DLL se encuentra en los directorios de MRI del conector.
DescriptionResID	ID de recurso de la descripción traducida en la DLL de MRI. Este campo debe contener el mismo valor que el campo DescriptionID definido en la clave primaria del registro para el conector.
Version	Valor numérico que indica el nivel de release del conector. Este valor se utiliza para determinar si el conector necesita ser actualizado en la estación de trabajo cliente. Este valor se incrementa en una cantidad con cada nuevo release del conector. El valor Version se compara con el valor de Version actual del conector instalado en la estación de trabajo cliente. Cuando este valor de Version es mayor que el que ya existe en la estación de trabajo cliente, el conector será actualizado a la nueva versión.
VendorID	La serie <*proveedor*>.<*componente*> utilizada para identificar el conector. Esta serie se utiliza para crear la clave de registro del conector en el árbol de registro de System i Access para Windows. VendorID debe ser idéntico a la parte <*proveedor*>.<*componente*> de la vía de acceso donde se instalará el componente en el sistema.
JavaPlugin	Un campo que indica si este es un conector de Java. Para los conectores de Java, todos los archivos JAR deben instalarse en el directorio \PLUGINS\<*proveedor*>.<*componente*>, y este valor será utilizado para determinar si el proceso de instalación deber hacer esto o no. Si este es un conector de Java y este valor está establecido en NO o no existe, el conector no podrá funcionar una vez ha sido instalado.

EclipseHelp	Un campo que indica si la aplicación de conector utiliza la plataforma Eclipse para desarrollar la ayuda. La ayuda de Eclipse se utiliza sólo en conectores Java. La ayuda del conector, si está habilitada para Eclipse, se encuentra contenida en un archivo comprimido especificado en el archivo setup.ini. Para cada idioma, el archivo comprimido se toma del directorio MRI29*xx* correcto y se extrae en el directorio [DirInstalación]\Eclipse\Plugins.
	
	Si esta entrada no existe el valor por omisión es NO.
	
	Si se especifica EclipseHelp=YES en el archivo setup.ini, deberá existir una sección EclipseHelp con la siguiente información:
	`[Eclipse]`
	`EclipseDirName=com.ibm.iSeries.`*`nombre conector ayuda`*`.help.doc`
	`EclipseZipFile=superzip.zip`

EclipseDirName	El directorio donde se extraen los archivos de ayuda. Este nombre de directorio sólo es necesario durante la desinstalación ya que el archivo comprimido ya contiene la estructura de directorio.
EclipseZipFile	El nombre del archivo comprimido que extraer en el directorio.
ExpressMinRelease	El release mínimo del sistema operativo en el que el conector está soportado (por ejemplo, V5R2M0).
ExpressMaxRelease	El release máximo del sistema operativo en el que el conector está soportado (por ejemplo, V6R1M0).

Ejemplo: sección de servicio (Service) de setup.ini:

La segunda sección del archivo setup.ini, Service, proporciona al programa Comprobar nivel de servicio la información que necesita para determinar si debe aplicarse un nuevo nivel de arreglo del conector a la estación de trabajo cliente.

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

Campo de la sección [Service] de setup.ini	Descripción del campo
FixLevel	Valor numérico que indica el nivel de servicio del conector. Este valor debe incrementarse con cada release de servicio para una versión determinada. El valor FixLevel se compara con el valor FixLevel actual del conector instalado en el sistema del cliente. Si este valor de FixLevel es mayor que el del conector instalado en la estación de trabajo cliente, el conector será actualizado al nuevo FixLevel cuando los usuarios seleccionen la opción **Actualizar o dar servicio a conectores** de System i Navigator. El valor debe ser restablecido a cero cuando se actualiza un conector a una nueva versión o nivel de release.
AdditionalSize	La cantidad de espacio de disco necesaria para almacenar cualquier archivo ejecutable nuevo o adicional que será añadida al conector durante la aplicación del servicio. La instalación utiliza este valor para determinar si la estación de trabajo dispone de espacio de disco suficiente para el conector.

Ejemplo: sección de identificación de archivos de setup.ini:

Esta parte del archivo setup.ini contiene la información que identifica los archivos que serán instalados en la estación de trabajo cliente.

La sección en que aparece un archivo identifica las ubicaciones del origen y el destino de cada archivo. Estas secciones del archivo se utilizan durante la instalación inicial o durante una actualización a una nueva versión o a un nuevo nivel de release.

El formato de las entradas de archivo de cada sección de archivos debe ser `n=archivo.ext`, donde `n` es el número del archivo en esa sección. La numeración debe empezar por uno (1) y aumentar en una (1) unidad hasta que se muestren todos los archivos de la sección. Por ejemplo:

```
[Base Files]
1=archivo1.dll
2=archivo2.dll
3=archivo3.dll
```

En todos los casos, únicamente debe especificarse el nombre de archivo. No especifique nombres de vía de acceso de directorio. Si una sección de archivo no contiene entradas, se ignorará dicha sección.

Nota: El juego de herramientas del programador proporciona un archivo de instalación de ejemplo para tres conectores de ejemplo distintos: C++, Java y Visual Basic.

Sección de Setup.ini	Descripción
[Base Files]	Los archivos que se copian a \PLUGINS\<*proveedor*>.<*componente*> bajo el directorio de instalación de Client Access. Como norma general, en esta ubicación se encuentra la DLL servidora ActiveX (y las DLL de código asociadas) del conector.
[Shared Files]	Archivos que se copian en el directorio Shared de Client Access.
[System Files]	Archivos que se copian en el directorio \WINDOWS\SYSTEM o \WINNT\SYSTEM32.

Sección de Setup.ini	Descripción
[Core Files]	Archivos que se copian al directorio \WINDOWS\SYSTEM o \WINNT\SYSTEM32. Estos archivos son archivos comunes a más de una aplicación. Cada archivo común se asocia con un número que cuenta la cantidad de aplicaciones que usa el archivo. Cuando se elimina una aplicación que utiliza el archivo común, se reduce el número. Un archivo común no se elimina hasta que se desinstala la última aplicación que lo utiliza. Generalmente, estos archivos pueden ser redistribuidos.
[MRI Files]	Archivos que se copian de los directorios MRI del conector en el sistema a los directorios CLIENT ACCESS\MRI29XX\<*proveedor*>.<*componente*> en la estación de trabajo. Habitualmente es donde residen los recursos dependientes del entorno nacional de un conector. Esto incluye el nombre de la DLL de MRI de recursos.
[Java MRI29xx] (siendo 29xx el código de característica NLV de los archivos)	Archivos Java que se copian del directorio MRI29xx del conector en el sistema al mismo directorio en el que están instalados los archivos base. Esto suele ser donde residen los recursos JAR MRI29xx del conector. Para cada directorio MRI29xx soportado por el conector de Java, es necesario que una sección MRI29XX de Java liste dichos archivos. Esto sólo se utiliza con conectores de Java.
[Help files]	Los archivos .HLP y .CNT que se copian desde los directorios MRI del conector en el sistema en los directorios CLIENT ACCESS\MRI29XX\<*proveedor*>.<*componente*> en la estación de trabajo cliente. La vía de acceso de directorio de estos archivos se escribe en HKEY_LOCAL_MACHINE\SOFTWARE\MICROSOFT\WINDOWS\HELP en el registro de Windows.
[Registry files]	Archivo del registro de Windows que está asociado al conector.

Sección de Setup.ini	Descripción

[Dependencies]	La sección que define los subcomponentes que deben instalarse antes de poder instalar el conector. AS400_Client_Access_Express sólo es necesario si el conector requiere instalar otros subcomponentes, además del subcomponente de soporte base System i Navigator.
	
	AS400_Client_Access_Express
	• Los subcomponentes se especifican en una lista separada por comas. Un único subcomponente se especifica como un solo número (AS400_Client_Access_Express=3). El archivo de cabecera CWBAD.H contiene una lista de constantes que tienen el prefijo CWBAD_COMP_. Estas constantes proporcionan los valores numéricos que se utilizan en la lista delimitada por comas para AS400_Client_Access_Express. Estas constantes CWBAD_COMP_ identifican los subcomponentes de fonts PC5250 y no debe utilizarse en el valor AS400_Client_Access_Express:
	`//Subcomponentes de emulador de pantalla e impresora S5250`
	`#define CWBAD_COMP_PC5250_BASE_KOREAN (150)`
	`#define CWBAD_COMP_PC5250_PDFPDT_KOREAN (151)`
	`#define CWBAD_COMP_PC5250_BASE_SIMPCHIN (152)`
	`#define CWBAD_COMP_PC5250_PDFPDT_SIMPCHIN (153)`
	`#define CWBAD_COMP_PC5250_BASE_TRADCHIN (154)`
	`#define CWBAD_COMP_PC5250_PDFPDT_TRADCHIN (155)`
	`#define CWBAD_COMP_PC5250_BASE_STANDARD (156)`
	`#define CWBAD_COMP_PC5250_PDFPDT_STANDARD (157)`
	`#define CWBAD_COMP_PC5250_FONT_ARABIC (158)`
	`#define CWBAD_COMP_PC5250_FONT_BALTIC (159)`
	`#define CWBAD_COMP_PC5250_FONT_LATIN2 (160)`
	`#define CWBAD_COMP_PC5250_FONT_CYRILLIC (161)`
	`#define CWBAD_COMP_PC5250_FONT_GREEK (162)`
	`#define CWBAD_COMP_PC5250_FONT_HEBREW (163)`
	`#define CWBAD_COMP_PC5250_FONT_LAO (164)`
	`#define CWBAD_COMP_PC5250_FONT_THAI (165)`
	`#define CWBAD_COMP_PC5250_FONT_TURKISH (166)`
	`#define CWBAD_COMP_PC5250_FONT_VIET (167)`
	
	Nota: El valor AS400_Client_Access_Express se utiliza si existe; en caso contrario se ignora esta sección.

[Service Base Files]	Archivos que se copian en \PLUGINS\<*proveedor*>.<*componente*> bajo el directorio de instalación de System i Access para Windows.
[Service Shared Files]	Archivos que se copian al directorio Shared de System i Access para Windows.
[Service System Files]	Archivos que se copian en el directorio \WINDOWS\SYSTEM o \WINNT\SYSTEM32.
[Service Core Files]	Archivos que se copian al directorio \WINDOWS\SYSTEM o \WINNT\SYSTEM32. Estos archivos son archivos comunes a más de una aplicación. Cada archivo común se asocia con un número que cuenta la cantidad de aplicaciones que usa el archivo. Cuando se elimina una aplicación que utiliza el archivo común, se reduce el número. Un archivo común no se elimina hasta que se desinstala la última aplicación que lo utiliza. Generalmente, estos archivos pueden ser redistribuidos.
[Service Registry Files]	Archivo del registro de Windows que está asociado al conector.

Ejemplo: sección del programa de salida de setup.ini:

La parte final del archivo setup.ini contiene secciones que identifican los programas que se ejecutan en la estación de trabajo cliente antes o después de una instalación, actualización o desinstalación.

Los siguientes ejemplos muestran la sintaxis utilizada en secciones de los programas de salida para identificar y ejecutar los programas.

Ejemplo 1: programa opcional que será llamado antes de que los archivos sean instalados durante una instalación inicial

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

Campo de [PreInstallProgram]	Descripción
Program	Obligatorio. Si se especifica una vía de acceso, solo se utiliza el nombre de archivo. El programa debe residir en la vía de acceso del conector <*proveedor*>.<*componente*> en la fuente de instalación.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. El valor por omisión es `N`. Si este valor está establecido en `Y`, la instalación del conector no continuará si el código de retorno es distinto de cero. No aparecerá un mensaje si el programa devuelve un código de error distinto de cero, pero se registrará un mensaje en el archivo instlog.txt.
Wait	Opcional. Espera a que el programa finalice para continuar con la ejecución. El valor por omisión es `Y`. Si `CheckReturnCode=Y`, se utiliza `Wait=Y` independientemente del valor especificado aquí.

Ejemplo 2: programa opcional que será llamado después de instalar los archivos durante una instalación inicial

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

Campo de [PostInstallProgram]	Descripción
Program	Obligatorio. Si se especifica una vía de acceso, solo se utiliza el nombre de archivo. El programa debe residir en el directorio del conector <*proveedor*>.<*componente*> de la estación de trabajo.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. La sección `PostInstallProgram` soporta los mismos valores que aquellos en la sección `PreInstallProgram`. El valor `CheckReturnCode` se registra en el archivo instlog.txt para su consulta.
Wait	Opcional. Espera a que el programa finalice para continuar con la ejecución. La sección `PostInstallProgram` soporta los mismos valores que aquellos en la sección `PreInstallProgram`.

Ejemplo 3: programa opcional al que llamar antes de desinstalar los archivos

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

Campo de [UninstallProgram]	Descripción
Program	Obligatorio. Si se especifica una vía de acceso, solo se utiliza el nombre de archivo. El programa debe residir en el directorio del conector <*proveedor*>.<*componente*> de la estación de trabajo.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. El valor por omisión es `N`. Si este valor está establecido en `Y`, la desinstalación del conector no continuará si el código de retorno es distinto de cero. No aparecerá un mensaje si el código de error distinto de cero, pero se registrará un mensaje en el archivo instlog.txt.
Wait	Opcional. Espera a que el programa finalice para continuar con la ejecución. El valor por omisión es `Y`. Si `CheckReturnCode=Y`, se utiliza `Wait=Y` independientemente del valor especificado aquí.

Ejemplo 4: programa opcional al que debe llamarse antes de actualizar los archivos

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

Campo de [PreUpgradeProgram]	Descripción
Program	Obligatorio. Si se especifica una vía de acceso, solo se utiliza el nombre de archivo. El programa debe residir en la vía de acceso del conector <*proveedor*>.<*componente*> en la fuente de instalación.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. El valor por omisión es `N`. Si este valor está establecido en `Y`, la instalación del conector no continuará si el código de retorno es distinto de cero. No aparecerá un mensaje si el código de error distinto de cero, pero se registrará un mensaje en el archivo instlog.txt.
Wait	Opcional. Espera a que el programa finalice para continuar con la ejecución. El valor por omisión es `Y`. Si `CheckReturnCode=Y`, se utiliza `Wait=Y` independientemente del valor especificado aquí.

Ejemplo 5: programa opcional al que debe llamarse antes de actualizar los archivos

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

Campo de [PostUpgradeProgram]	Descripción
Program	Obligatorio. Si se especifica una vía de acceso, solo se utiliza el nombre de archivo. El programa debe residir en la vía de acceso del conector <*proveedor*>.<*componente*> en la fuente de instalación.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. La sección `PostUpgradeProgram` soporta los mismos valores que aquellos soportados en la sección `PreUpgradeProgram`. El valor `CheckReturnCode` se registra en el archivo instlog.txt para su consulta.

Campo de [PostUpgradeProgram]	Descripción
Wait	Opcional. Espera a que el programa finalice para continuar con la ejecución. El valor por omisión es `Y`. Si `CheckReturnCode=Y`, se utiliza `Wait=Y` independientemente del valor especificado aquí.

Archivo de instalación de MRI

El archivo de instalación de MRI proporciona al asistente Instalar conectores la información necesaria para instalar los recursos dependientes del entorno nacional asociados con un conector de System i Navigator en una estación de trabajo cliente.

Deberá denominar al archivo de instalación MRI MRISETUP.INI. Debe existir una versión de este archivo en el subdirectorio MRI29XX en el sistema de archivos de System i para cada idioma nacional soportado por el conector.

El formato del archivo está en conformidad con el de un archivo de configuración estándar de Windows (.INI). El archivo contiene una sola sección, MRI Info. La sección MRI Info proporciona el valor de la versión (Version) de la interfaz MRI del conector. La interfaz MRI del conector incluye todas las DLL de recursos, así como los archivos de ayuda (.HLP y .CNT) de un idioma determinado. Por ejemplo:

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

El asistente Instalar conectores comprueba el valor Version del archivo del archivo MRI. El valor Version de MRI en este archivo debe coincidir con el valor Version en el archivo SETUP.INI del conector. Cuando estos valores no coinciden, el conector no aparecerá listado en el asistente Instalar conectores y no se copiará ningún archivo a la estación de trabajo cliente. El juego de herramientas del programador proporciona un archivo de instalación de MRI de ejemplo junto con el conector de ejemplo.

Conceptos relacionados

"Directorio de archivos de conectores de ejemplo de Visual Basic" en la página 17
Estas tablas describen todos los archivos incluidos con el conector de ejemplo de Visual Basic.

Identificar conectores en System i Navigator

Los conectores se identifican a sí mismos en System i Navigator proporcionando información al registro de Windows cuando se instala el software de conector en las estaciones de trabajo de los usuarios.

Las entradas de registro especifican la ubicación del código de conector e identifican las clases que implementan las interfaces de System i Navigator especiales. Puede proporcionar información de registro adicional que System i Navigator utiliza para determinar si debe activarse la función del conector para un sistema en particular. Por ejemplo, un conector puede necesitar un nivel de release de i5/OS mínimo o puede especificar determinadas necesidades de un producto para que pueda ser instalado en el sistema para que funcione.

Cuando un usuario selecciona un sistema en el árbol de jerarquía de System i Navigator después de instalar un conector, System i Navigator examina el sistema para determinar si puede soportar el nuevo conector. Los prerrequisitos de software (especificados en las entradas de registro del conector) se comparan con el software instalado en el sistema. Si se satisfacen los requisitos del conector, aparecerá la nueva función en el árbol de jerarquías. si no se satisfacen los requisitos, la función del conector no aparece en dicho sistema.

Instalar y ejecutar conectores de ejemplo

El juego de herramientas del programador suministra conectores de ejemplo de cada uno de los lenguajes de programación soportados.

Estos ejemplos son un método excelente para aprender cómo funcionan los conectores y un punto de partida de gran eficacia para desarrollar sus propios conectores. Si todavía no tiene instalado el juego de herramientas del programador, tendrá que instalarlo antes de trabajar con cualquiera de los conectores de ejemplo. Puede instalar Toolkit a través de Agregar o eliminar programas en el Panel de control de System i Access para Windows.

Nota: Antes de comenzar a trabajar en cualquiera de los conectores de ejemplo, necesita conocer los requisitos exclusivos para desarrollar conectores en cada uno de los tres lenguajes.

Conceptos relacionados

"Requisitos de los conectores" en la página 4
Los requisitos del conector de System i Navigator difieren de acuerdo con el lenguaje de programación que utilice.

Configurar conectores C++ de ejemplo

Esta tarea supone crear y ejecutar la DLL servidora ActiveX de ejemplo.

El ejemplo proporciona un espacio de trabajo de Developer Studio funcional que puede utilizar para establecer puntos de interrupción y observar el comportamiento de un conector System i Navigator típico. Puede también utilizar el ejemplo para verificar que el entorno de Developer Studio está configurado correctamente para compilar y enlazar el código de conector.

Para configurar el conector de C++ de ejemplo en la estación de trabajo, siga estos pasos.

Bajar el conector C++	Bajar el archivo ejecutable cppsmppq.exe. Cuando ejecute el archivo, este extrae todos los archivos asociados con el conector. Cree un nuevo directorio, C:\MyProject, y copie todos los archivos en el mismo. Si crea un directorio distinto, tendrá que modificar el archivo de registro para especificar la ubicación correcta del conector.
Prepararse para crear una DLL servidora de ActiveX	1. Cree un nuevo directorio denominado "MyProject" en la unidad de disco duro local. En este ejemplo se presupone que la unidad local es la unidad C:. **Nota:** Si el nuevo directorio no es C:\MyProject, necesitará cambiar el archivo de registro. 2. Copie todos los archivos de ejemplo en este directorio. Puede descargar los ejemplos en la página web de conectores de Programmer's Toolkit - System i Navigator. 3. En Developer Studio, abra el menú **Archivo** y seleccione **Abrir área de trabajo**. 4. En el diálogo Abrir espacio de trabajo de proyecto, cambie al directorio **MyProject** y en **Tipos de archivo**, seleccione **Makefiles (*.mak)**. 5. Seleccione **sampext.mak** y pulse **Abrir**. 6. Abra el menú **Herramientas** y seleccione **Opciones**. 7. En la pestaña **Directorios**, compruebe que el directorio Include de Client Access aparezca en la parte superior de la vía de acceso de búsqueda de archivos Include. 8. En **Mostrar directorios para:**, seleccione **Archivos de biblioteca**. Compruebe que el directorio de biblioteca (Lib) de Client Access aparezca en la parte superior de la vía de acceso de búsqueda de archivos de biblioteca. 9. Pulse el botón de aceptar para guardar los cambios y, a continuación, cierre Developer Studio y vuelva a abrirlo. Esta es la única forma conocida de forzar a Developer Studio a guardar los cambios de la vía de acceso de búsqueda en el disco duro.
Crear la DLL servidora ActiveX	1. En Developer Studio, abra el menú **Generar** y seleccione **Establecer configuración por omisión**. 2. En el diálogo Configuración de proyecto predeterminada, seleccione **Configuración de depuración de Win32** de samptext. 3. Abra el menú **Generar** y seleccione **Regenerar todo** para compilar y enlazar la DLL. **Nota:** Si la DLL no se compila y enlaza sin errores, efectúe una doble pulsación en los mensajes de error de la ventana de compilación para localizar y corregir los errores. A continuación abra el menú **Generar** y seleccione **sampext.dll** para volver a iniciar el proceso de creación.

Crear la biblioteca de recursos	El ejemplo incluye una DLL de recursos que contiene las series de texto traducibles y otros recursos dependientes del entorno nacional para el conector. Esto significa que no es necesario que cree esta DLL por su cuenta. Aunque el conector solo dé soporte a un idioma, el código del conector debe cargar las series de texto y los recursos específicos del entorno nacional de esta biblioteca de recursos. Para crear la DLL de recursos, siga estos pasos: 1. En Developer Studio, abra el menú **Archivo**, seleccione **Abrir espacio de trabajo** y seleccione el directorio **MyProject**. 2. Especifique **Makefiles (*.mak)** en **Tipos de archivo**. 3. Seleccione **sampmri.mak** y pulse **Abrir**. 4. Abra el menú **Generar** y seleccione **Regenerar todo** para compilar y enlazar la DLL.
Registre la DLL servidora ActiveX	El archivo SAMPDBG.REG en el directorio MyProject contiene claves de registro que comunican la ubicación de los conectores de ejemplo en la estación de trabajo a System i Navigator. Para registrar la DLL servidora ActiveX, efectúe una doble pulsación sobre el archivo SAMPDBG.REG en el Explorador de Windows. El archivo de registro se ejecutará y las entradas en el archivo será escritas en el registro de Windows en la estación de trabajo. Si ha especificado un directorio distinto de C:\MyProject, complete los siguientes pasos antes de ejecutar el archivo de registro: 1. Abra el archivo SAMPDBG.REG en Developer Studio (o en su editor de texto preferido). 2. Sustituya todas las ocurrencias de C:\\MyProject\\ con *x*:\\<*dir*>\\, donde *x* es la letra de unidad donde reside el directorio y <*dir*> es el nombre del directorio. 3. Guarde el archivo.
Ejecute System i Navigator en el depurador	Para ejecutar System i Navigator y observar el conector de ejemplo en acción, siga estos pasos: 1. En Developer Studio, abra el menú **Generar** y seleccione **Depurar → Continuar**. 2. En el indicador que aparece, escriba la vía de acceso totalmente calificada del archivo ejecutable de System i Navigator en el directorio de instalación de System i Access para Windows en la estación de trabajo. La vía de acceso es C:\ARCHIVOS DE PROGRAMA\IBM\CLIENT ACCESS\CWBUNNAV.EXE, o algo similar. 3. Pulse **Aceptar**. Se abrirá la ventana principal de System i Navigator. Un diálogo en System i Navigator se solicitará que realice una exploración en busca del nuevo conector ya que acaba de registrar un nuevo conector al ejecutar el archivo SAMPDBG.REG. Una vez que finalice el indicador de progreso, pulse **Aceptar** en el diálogo que se muestra. Después de renovar la ventana de System i Navigator, aparecerá una nueva ventana (una carpeta de ejemplo de terceros) en la jerarquía bajo el sistema que seleccionado inicialmente. Ahora podrá interactuar con el conector en System i Navigator y observar su comportamiento en el depurador.

Información relacionada

IBM Client Access Express Toolkit - Página de ayuda de conectores de System i Navigator

Configurar conectores Visual Basic de ejemplo

El conector de ejemplo de Visual Basic añade una carpeta a la jerarquía de System i Navigator que proporciona una lista de bibliotecas de i5/OS e ilustra cómo implementar propiedades y acciones en dichos objetos de biblioteca.

Además de instalar el código del conector, el conector de ejemplo incluye un archivo Readme.txt y dos archivos del registro, uno para utilizarse durante el desarrollo y otro que se distribuye con la versión de minorista. Consulte el directorio de archivos del conecto de ejemplo de Visual Basic para obtener una descripción detallada de todos los archivos incluidos con el conector de Visual Basic.

Para configurar el conector de Visual Basic de ejemplo en la estación de trabajo, siga estos pasos.

Descargue el conector de Visual Basic	Bajar el archivo ejecutable vbopnav.exe. Cuando ejecute el archivo, este extrae todos los archivos asociados con el conector. Cree un nuevo directorio C:\VBSample y copie todos los archivos en el mismo. Si crea un directorio distinto, tendrá que modificar el archivo de registro para especificar la ubicación correcta del conector.
Cree el proyecto de Visual Basic	Abra vbsample.vpb en Visual Basic. En el diálogo Referencia, seleccione **IBM System i Access para WindowsBiblioteca de objetos ActiveX** y **System i Navigator Soporte de conectores de Visual Basic**. **Nota:** Si estas referencias no aparecen en el diálogo Referencias, seleccione **Examinar** y busque cwbx.dll y cwbunvbi.dll en el directorio compartido de System i Access para Windows. La biblioteca de objetos ActiveX de IBM System i Access contiene objetos de automatización OLE que la aplicación de ejemplo necesita para hacer llamadas de mandato remoto al sistema-. El soporte del conector de System i Navigator Visual Basic contiene clases e interfaces requeridos para crear un directorio de conectores de Visual Basic.
Crear la DLL servidora ActiveX	Seleccione **Generar** en el menú de archivo de Visual Basic para crear la DLL. Si no se compila o enlaza, encuentre y corrija los errores y, a continuación, vuelva a crear la DLL.
Crear la biblioteca de recursos	1. Abra Microsoft Developer Studio, abra el menú Archivo, seleccione Abrir espacio de trabajo y seleccione el directorio **VBSample\win32**. 2. En el campo **Tipo de archivo**, especifique **Makefiles** (*.mak). 3. Seleccione vbsmpmri.mak y pulse **Abrir**. 4. Abra el menú Generar y seleccione Regenerar todo para compilar y enlazar la DLL. **Nota:** No es necesario que cree esta DLL por su cuenta. El ejemplo incluye una DLL de recursos que contiene las series de texto traducibles; asimismo, contiene otros recursos dependientes del entorno nacional para el conector. Aunque el conector solo dé soporte a un idioma, el código del conector debe cargar las series de texto y los recursos específicos del entorno nacional de esta biblioteca de recursos.
Registrar el conector	Efectúe una doble pulsación en el archivo vbsmpdbg.reg para registrar el conector. Si no ha utilizado el directorio C:\VBSample, edite el archivo de registro y sustituya todas las ocurrencias de ″C:\\VBSample\\″ con la vía de acceso calificada al completo del código del conector. Debe utilizar barras inclinadas invertidas dobles en la vía de acceso.
Ejecute el conector en System i Navigator	Desde System i Navigator, expanda el sistema que desea explorar.System i Navigator detecta los cambios en el registro y solicita que explore el sistema para verificar que puede soportar el nuevo conector. Después de completar la exploración, System i Navigator muestra el nuevo conector en la jerarquía de árbol.

Información relacionada

IBM Client Access Express Toolkit - Página de ayuda de conectores de System i Navigator

Directorio de archivos de conectores de ejemplo de Visual Basic

Estas tablas describen todos los archivos incluidos con el conector de ejemplo de Visual Basic.

Archivo de proyecto de Visual Basic	Descripción
vbsample.vbp	Archivo de proyecto de Visual Basic 5.0

Formularios de Visual Basic	Descripción
authorty.frm	Formulario para establecer autorización
delete.frm	Formulario para confirmar supresión
propsht.frm	Formulario de hoja de propiedades
sysstat.frm	Formulario de estado del sistema

Formularios de Visual Basic	Descripción
wizard.frm	Formulario del asistente Crear biblioteca nueva

Módulos de Visual Basic	Descripción
global.bas	Declaraciones globales

Módulos de clases de Visual Basic	Descripción
actnman.cls	Clase de gestor SampleActions
dropman.cls	Clase de gestor de destino de acción de soltar de ejemplo
library.cls	Clase de biblioteca
listman.cls	Clase de gestor de lista de ejemplo

Archivos binarios de Visual Basic	Descripción
authorty.frx	Binario de formulario para establecer autorización
delete.frx	Binario de formulario para confirmar supresión
propsht.frx	Binario de formulario de hoja de propiedades
sysstat.frx	Binario de formulario de estado del sistema
wizard.frx	Binario de formulario del asistente Crear biblioteca nueva
vbsample.bin	Binario Vbsample

Valores de configuración	Descripción
mrisetup.ini	Información de instalación para recursos traducibles del conector
setup.ini	Información de instalación para archivos ejecutables del conector

Entradas del registro	Descripción
vbsmpdbg.reg	Archivo del registro, que se utilizará durante el desarrollo.
vbsmprls.reg	Archivo del registro, que se utiliza durante la instalación.

Archivos para crear la DLL de recursos	Descripción
vbsmpmri.mak	Archivo MAKE
vbsmpmri.rc	Archivo RC
vbsmpres.h	Archivo de cabecera

Imágenes	Descripción
compass.bmp	Icono de System i Navigator
lib.ico	
vbsmpflr.ico	Carpeta del conector Visual Basic de ejemplo en estado abierto y cerrado
vbsmplib.ico	Icono de biblioteca del conector Visual Basic de ejemplo

Conceptos relacionados

"Archivo de instalación de MRI" en la página 14
El archivo de instalación de MRI proporciona al asistente Instalar conectores la información necesaria para instalar los recursos dependientes del entorno nacional asociados con un conector de System i Navigator en una estación de trabajo cliente.

"Archivo setup.ini" en la página 7
El archivo setup.ini del conector proporciona el asistente de instalación con la información necesaria para instalar un conector de System i Navigator en una estación de trabajo cliente. También facilita información que permite al programa de comprobación del nivel de servicio determinar si el conector necesita una actualización o servicio.

Configurar los conectores de Java de ejemplo

Los conectores de Java de ejemplo funcionan con colas de mensajes en la biblioteca QUSRSYS en un sistema dado.

El primer conector permite ver, agregar y suprimir mensajes en la cola de mensajes por omisión, la misma que tiene el ID de usuario de System i. El segundo conector añade soporte para varias colas de mensajes. El tercer conector añade la posibilidad de arrastrar y soltar mensajes entre colas.

Además de instalar el código del conector, el conector de ejemplo incluye documentación Java, un archivo Readme.txt y dos archivos del registro, uno para utilizarse durante el desarrollo y otro que se distribuye con la versión de minorista. Consulte Directorio de archivos de los conectores Java de ejemplo para obtener una descripción detallada de todos los archivos incluidos junto con los conectores Java.

Para configurar los conectores de Java de ejemplo en la estación de trabajo, siga estos pasos.

Bajar los conectores Java de ejemplo	Bajar el archivo ejecutable jvopnav.exe. Cuando se ejecuta este archivo, extrae todos los archivos anteriormente mencionados. Debe permitir que el archivo ejecutable instale los archivos en el directorio por omisión: jvopnav\com\ibm\as400\opnav.
Identificar el conector para System i Navigator	1. Edite el archivo MsgQueueSampleX.reg en jvopnav\com\ibm\as400\opnav\ MsgQueueSampleX. (X=1, 2 ó 3, según el ejemplo que instale.) 2. Encuentre las líneas: ″NLS″=″C:\\jvopnav\\win32\\mri\\ MessageQueuesMRI.dll″ y ″JavaPath″=″C:\\jvopnav″ 3. Sustituya ″C:\\″ con la vía de acceso totalmente calificada del directorio jvopnav en la estación de trabajo. Debe utilizar barras inclinadas invertidas dobles en la vía de acceso. 4. Guarde los cambios. 5. Pulse dos veces sobre el archivo de registro MsgQueueSampleX.reg. El archivo de registro se ejecutará y se escribirán las entradas en el archivo en el registro de Windows de la estación de trabajo.

Ejecutar el conector Java de ejemplo	1. Desde System i Navigator, expanda el sistema que desea explorar. 2. System i Navigator detecta los cambios en el registro y le solicita que explore el sistema para verificar que puede soportar el nuevo conector. Pulse **Explorar ahora**. 3. System i Navigator explora el sistema. Cuando la exploración termina, System i Navigator muestra una nueva carpeta en el árbol de jerarquías, Cola de mensajes de ejemplo de Java 1, 2 ó 3. Efectúe una nombre pulsación sobre la nueva carpeta. 4. El primer conector de ejemplo muestra el contenido de la cola de mensajes por omisión en la biblioteca QUSRSYS en el sistema. El segundo y tercer ejemplo muestran una lista de las colas de mensajes. Para añadir un nuevo mensaje, pulse con el botón derecho del ratón sobre la carpeta de cola y seleccione **Nuevo** → **Mensaje**. Escriba el texto del mensaje en el diálogo que muestra el conector. Para suprimir un mensaje, pulse con el botón derecho sobre un mensaje y seleccione **Suprimir**. 5. Si está utilizando el tercer conector de ejemplo, puede seleccionar un mensaje y arrastrarlo a otra cola. El conector moverá entonces el mensaje a la otra cola.

Información relacionada

IBM Client Access Express Toolkit - Página de ayuda de conectores de System i Navigator

Directorio de archivos de los conectores Java de ejemplo

Estas tablas describe todos los archivos incluidos con lo conectores de Java de ejemplo.

Para obtener más información, lea la documentación de javadocs del conector. Estos estarán instalados en el directorio jvopnav\com\ibm\as400\opnav\MsgQueueSample1\docs. Comience con el archivo Package-com.ibm.as400.opnav.MsgQueueSample1.html. El nombre del paquete de ejemplo es com.ibm.as400.opnav.MsgQueueSample1. Todos los nombres de clase tienen el prefijo Mq para diferenciarlos de clases parecidas en otros paquetes.

Archivos de código fuente Java; primer conector de ejemplo	**Descripción**
MqActionsManager.java	La implementación ActionsManager que maneja todos los menús de contexto del conector.
MqDeleteMessageBean.java	La implementación DataBean de UI para el diálogo Confirmar supresión.
MqMessage.java	Un objeto que representa un mensaje del sistema.
MqMessageQueue.java	Una colección de objetos de mensaje de sistema en una cola de mensajes.
MqMessagesListManager.java	ListManager para listas de mensajes.
MqNewMessageBean.java	La implementación DataBean de UI del diálogo Nuevo mensaje.

Archivos de código fuente Java; segundo conector de ejemplo	**Descripción**
MqActionsManager.java	La implementación ActionsManager que maneja todos los menús de contexto del conector.
MqDeleteMessageBean.java	La implementación DataBean de UI para el diálogo Confirmar supresión.
MqListManager.java	Implementación de ListManager maestro para el conector.
MqMessage.java	Un objeto que representa un mensaje del sistema.
MqMessageQueue.java	Una colección de objetos de mensaje de sistema en una cola determinada.

Archivos de código fuente Java; segundo conector de ejemplo	Descripción
MqMessageQueueList.java	Una colección de colas de mensajes de sistema.
MqMessageQueuesListManager.java	ListManager esclavo para listas de colas de mensajes.
MqMessagesListManager.java	ListManager esclavo para listas de mensajes.
MqNewMessageBean.java	La implementación DataBean de UI del diálogo Nuevo mensaje.

Archivos de código fuente Java; tercer conector de ejemplo	Descripción
MqActionsManager.java	La implementación ActionsManager que maneja todos los menús de contexto del conector.
MqDeleteMessageBean.java	La implementación DataBean de UI para el diálogo Confirmar supresión.
MqDropTargetManager.java	La implementación DropTargetManager que maneja las acciones arrastrar/soltar en el conector.
MqListManager.java	Implementación de ListManager maestro para el conector.
MqMessage.java	Un objeto que representa un mensaje del sistema.
MqMessageQueue.java	Una colección de objetos de mensaje de sistema en una cola determinada.
MqMessageQueueList.java	Una colección de colas de mensajes de sistema.
MqMessageQueuesListManager.java	ListManager esclavo para listas de colas de mensajes.
MqMessagesListManager.java	ListManager esclavo para listas de mensajes.
MqNewMessageBean.java	La implementación DataBean de UI del diálogo Nuevo mensaje.

Archivos PDML	Descripción
MessageQueueGUI.pdml	Contiene todas las definiciones de panel de interfaz de usuario Java para el conector.
MessageQueueGUI.java	Paquete de recursos Java asociado (subclases java.util.ListResourceBundle).

Archivos de ayuda en línea	Descripción
IDD_MSGQ_ADD.html	Esqueleto de la ayuda en línea del diálogo Nuevo mensaje.
IDD_MSGQ_CONFIRM_DELETE.html	Esqueleto de la ayuda en línea del diálogo Confirmar supresión.

Archivos serializados	Descripción
IDD_MSGQ_ADD.pdml.ser	Definición de panel serializada para el diálogo Nuevo mensaje.
IDD_MSGQ_CONFIRM_DELETE.pdml.ser	Definición de panel serializada para el diálogo Confirmar supresión. **Nota:** Si efectúa cambios en MessageQueueGUI.pdml, cambie el nombre de estos archivos. De lo contrario los cambios que haya realizado no se reflejarán en los paneles.

Entradas del registro	Descripción
MsgQueueSample1.reg MsgQueueSample2.reg MsgQueueSample3.reg	Entradas de registro Windows que comunican a System i Navigator que este conector existe e identifica sus clases de implementación de interfaz Java.

Entradas del registro	Descripción
MsgQueueSample1install.reg MsgQueueSample2install.reg MsgQueueSample3install.reg	Archivo del registro, que se distribuye con la versión de minorista del conector. Esta versión del archivo de registro no puede ser leída directamente por el sistema operativo Windows. Contiene variables de substitución que representan parte del directorio de instalación de System i Access para Windows. Cuando el usuario inicia el asistente Instalar conectores para instalar el conector desde el sistema, el asistente leer este archivo de registro, rellena las vías de acceso de directorios correspondientes y escribe las entradas en el registro de la estación de trabajo del usuario. Las entradas en este archivo, deben por lo tanto, estar sincronizadas con el archivo de registro utilizado durante el desarrollo.

Información de consulta para la programación de conectores

System i Navigator maneja los conectores en cada lenguaje de programación de distinta manera.

Puede utilizar los siguientes temas para aprender acerca del flujo de control en System i Navigator de cada tipo de conector, así como información de consulta específica relativa a los interfaces exclusivos de cada lenguaje.

Además de la información de consulta específica de cada lenguaje, cada conector requiere algo de personalización en los archivos del registro de Windows.

Información de consulta de C++

Los conectores de C++ tienen un flujo de control exclusivo en System i Navigator. Puede utilizar una variedad de API de System i Navigator para desarrollar conectores de C++. Cada conector puede implementar uno o más interfaces de COM (modelo de objeto de componentes).

Estructura y flujo de control de System i Navigator para conectores de C++

La arquitectura interna del producto System i Navigator está pensada para servir como punto de integración de operaciones extensibles y amplias en la plataforma de System i.

Cada componente funcional de la interfaz se empaqueta como una DLL servidora ActiveX. System i Navigator utiliza la tecnología de COM de Microsoft (modelo de objeto de componente) para activar sólo las implementaciones de componente necesarias en la actualidad para dar servicio a una solicitud de servicio. Esto evita el problema de tener que cargar todo el producto durante el arranque, lo cual puede consumir la mayoría de los recursos de Windows y afectar al rendimiento de todo el sistema. Varios sistemas puede registrar su solicitud para agregar elementos de menú y diálogos a un tipo de objeto determinado en la jerarquía de System i Navigator.

Los conectores funcionan respondiendo a las llamadas de método de System i Navigator que son generadas en respuesta a las acciones de usuario. Por ejemplo, cuando un usuario pulsa con el botón derecho del ratón sobre un objeto en la jerarquía de System i Navigator, System i Navigator construye un menú contextual para dicho objeto y muestra el menú en la pantalla. System i Navigator obtiene los elementos de menú llamando a cada conector que ha registrado su intención de proporcionar elementos de menú contextual para el tipo de objeto seleccionado.

Las funciones implementadas lógicamente por medio de un conector se agrupan en interfaces. Una interfaz es un conjunto de métodos relacionados lógicamente en una clase a la que System i Navigator puede llamar para realizar una función específica. La tecnología COM da soporte a la definición de interfaces en C++ mediante la declaración de una clase abstracta que define un conjunto de funciones virtuales puras. Las clases que llaman a la interfaz se denominan clases de implementación. Las clases de implementación subclasifican la definición de la clase abstracta y proporcionan el código C++ para cada una de las funciones definidas en la interfaz.

Una clase de implementación dada puede implementar tantas interfaces como desee el desarrollador. Al crear una nueva área de trabajo de proyecto para una DLL servidora ActiveX en Developer Studio, AppWizard genera macros que facilitan la implementación de interfaces. Cada una de las interfaces se declara como una clase anidada de una clase de implementación que las contiene. La clase anidada no tiene datos de miembro y únicamente utiliza las funciones definidas en su interfaz. Sus métodos normalmente llaman a funciones de la clase de implementación para obtener y establecer la información de estado y para llevar a cabo el trabajo real definido por la especificación de interfaz.

Interfaces COM de System i Navigator para C++

Las funciones implementadas por un conector de forma lógica se agrupan en interfaces COM (modelo de objetos de componente).

Una interfaz es un conjunto de métodos relacionados de forma lógica en una clase de System i Navigator que puede llamar a una función específica. Un conector puede implementar una o más interfaces COM, dependiendo del tipo de función que el desarrollador pretenda proporcionar. Por ejemplo, cuando un usuario pulsa con el botón derecho del ratón sobre un objeto en la jerarquía de árbol, System i Navigator construye un menú contextual para el objeto y lo muestra. System i Navigator obtiene los elementos de menú llamando a todos los conectores que hayan registrado que proporcionarán elementos de menú contextual para el tipo de objeto seleccionado. Los conectores pasan sus elementos de menú a System i Navigator cuando llama a su implementación del método **QueryContextMenu** en la interfaz **IContextMenu**.

Interfaz	Método	Descripción
IContextMenu	QueryContextMenu	Proporciona los elementos de menú contextual cuando un usuario pulsa con el botón derecho del ratón sobre un objeto.
	GetCommandString	Proporciona texto de ayuda para los elementos de menú de contexto y, según el estado del objeto, también indica si el elemento debe estar habilitado o inhabilitado.
	InvokeCommand	Visualiza el diálogo adecuado y lleva a cabo la acción solicitada. Se le llama cuando el usuario pulsa el botón del ratón sobre un elemento de menú determinado.
IPropSheetExt	AddPages	Crea las páginas de propiedades que se añaden utilizando las API de Windows estándar. A continuación añade las páginas efectuando una llamada a una función pasada como parámetro.
IDropTarget	DragEnter	En estado activo cuando el usuario arrastra un objeto sobre el área de soltar.
	DragLeave	En estado activo cuando el usuario arrastra un objeto fuera del área de soltar.
	DragOver	En estado activo cuando el usuario está sobre el área de soltar.
	Drop	En estado activo cuando el usuario suelta el objeto.

Interfaz	Método	Descripción
IPersistFile	Load	Se le llama para inicializar la extensión con el nombre de objeto totalmente calificado de la carpeta seleccionada.
IA4SortingHierarchyFolder	IsSortingEnabled	Indica si la ordenación está habilitada para una carpeta.
	SortOnColumn	Ordena la lista en la columna de la vista de lista especificada.
IA4FilteringHierarchyFolder	GetFilterDescription	Devuelve una descripción de texto de los criterios de inclusión actuales.
IA4PublicObjectHierarchyFolder	GetPublicListObject	Un conector lo implementa cuando desea que sus objetos de lista estén disponibles para el uso de otros conectores.
IA4ListObject	GetAttributes	Devuelve una lista de identificadores de atributo soportados y el tipo de datos asociado a cada uno de ellos.
	GetValue	Dado un identificador de atributo, devuelve el valor actual del atributo.
IA4TasksManager	QueryTasks	Devuelve una lista de tareas a las que este objeto da soporte.
	TaskSelected	Informa a la implementación de IA4TasksManager de que el usuario ha seleccionado una tarea determinada.

Interfaces IA4

Además de las interfaces COM de Microsoft, IBM proporciona las interfaces IA4HierarchyFolder e IA4PropSheetNotify.

La interfaz IA4PropSheetNotify notifica a páginas de propiedades de terceros cuando se cierra el diálogo principal. También define métodos que comunican información al conector. Por ejemplo, el método puede comunicar si el usuario cuyas propiedades están en pantalla ya existe o está siendo definido y si deben guardarse o descartarse los cambios.

La interfaz IA4HierarchyFolder permite que un conector agregue carpetas a la jerarquía de System i Navigator. El propósito de esta interfaz es proporcionar los datos que se utilizarán para rellenar el contenido de una nueva carpeta que el conector ha añadido a la jerarquía de System i Navigator. Asimismo, define métodos para especificar las columnas de la vista de lista y sus cabeceras y para definir una barra de herramientas personalizada asociada a una carpeta.

Descripción de la interfaz IA4HierarchyFolder:

La interfaz IA4HierarchyFolder describe un conjunto de funciones que el proveedor de software independiente implementará. IA4HierarchyFolder es una interfaz de COM (modelo de objeto de componente) definida por IBM con el único propósito de permitir a terceros añadir nuevas carpetas y objetos a la jerarquía de System i Navigator.

Para obtener una descripción de las interfaces COM de Microsoft, consulte el sitio Web de Microsoft.

El programa System i Navigator llama a los métodos en la interfaz IA4HierarchyFolder siempre que necesite comunicarse con el conector de terceros. El propósito principal de la interfaz es proporcionar a System i Navigator con datos de lista que serán utilizados cuando System i Navigator muestre el contenido de una carpeta definida por el conector. Los métodos en la interfaz permiten que System i Navigator enlace una carpeta de terceros en particular y liste su contenido. Existen métodos para devolver el número de columnas de la vista de detalles y sus cabeceras asociadas. Hay métodos adicionales que proporcionan las especificaciones para asociar una carpeta personalizada con la carpeta.

La implementación de la interfaz normalmente se compila y enlaza en una DLL (biblioteca de enlace dinámico) servidora ActiveX. System i Navigator conoce la existencia de la nueva DLL por medio de las entradas en el registro de Windows. Estas entradas especifican la ubicación de la DLL en el PC del usuario y el punto de unión en la jerarquía de objetos donde la o las nuevas carpetas serán insertadas. System i Navigator carga entonces la DLL en el momento adecuado y llama a los métodos en la interfaz IA4HierarchyFolder si es necesario.

El archivo de cabecera CWBA4HYF.H contiene declaraciones del prototipo de interfaz y las estructuras de datos y los códigos de retorno asociados.

Información relacionada

Sitio Web de Microsoft

Lista de especificaciones de la interfaz IA4HierarchyFolder:

Una entidad de datos de identificador de elemento que identifica todas las carpetas y objetos en el espacio de nombres de Windows. Los identificadores de elementos son como los nombres de archivo de un sistema de archivos jerárquico. El espacio de nombre de Windows es, de hecho, un espacio de nombres jerárquico en el Escritorio del Explorador de Windows.

Un identificador de elementos consiste en un campo de cuenta de 2 bytes, seguido de una estructura de datos binarios de longitud variable (consulte la estructura SHITEMID en el archivo de cabecera de Microsoft SHLOBJ.H). Este identificador de elemento describe de forma exclusiva un objeto en relación con la carpeta padre del objeto.

System i Navigator utiliza los identificadores de elementos que cumplen la siguiente estructura que debe ser devuelta por IA4HierarchyFolder::ItemAt.

```
<cb><nombre elemento>\x01<tipo elemento>\x02<índice
elemento>
```

siendo

`<cb>` es el tamaño en bytes del identificador de elemento, incluyendo el mismo campo de cuenta.

`<nombre elemento>` es el nombre traducido del objeto, adecuado para ser mostrado en pantalla al usuario.

`<tipo elemento>` una serie exclusiva independiente del lenguaje que identifica el tipo de objeto. Debe constar como mínimo de cuatro caracteres.

`<índice elemento>` es el índice basado en cero que identifica la posición del objeto en la lista de objetos de la carpeta padre.

IA4HierarchyFolder::Activate:

Esta especificación activa una instancia de IA4HierarchyFolder. Esta función también realiza cualquier proceso que sea necesario para preparar la carpeta para la enumeración, incluyendo llamar al sistema para preparar la antememoria de objetos de carpeta en el cliente.

La llamada a la función se efectúa desde una hebra de datos para que las operaciones de larga ejecución no incidan de forma negativa en el rendimiento de la interfaz de usuario. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

Ninguno.

Códigos de retorno

Devuelve NOERROR si es satisfactoria o E_FAIL si no puede obtener el contenido de la carpeta.

Comentarios

System i Navigator llama a esta función la primera vez que un usuario selecciona o expande una carpeta. Se vuelve a llamar después de una llamada para cerrar, cuando un usuario solicita una operación de renovación del contenido de la carpeta.

Puede llamarse a la función siempre que un puntero a la interfaz de carpeta necesite volver a establecerse como, por ejemplo, cuando un usuario selecciona una carpeta por segunda vez. Después de haber seleccionado otra carpeta, la función debe simplemente devolver TRUE si el proceso asociado ya ha sido realizado.

En el caso de listas sumamente grandes, puede elegir volver de Activate antes de que se cree por completo la lista, después de haber creado primero una hebra de trabajo para seguir generando la lista. En estas circunstancias, asegúrese de que la implementación de GetListSize devuelva la indicación correcta de si la lista se ha creado por completo.

IA4HierarchyFolder::BindToList:

Esta especificación devuelve una instancia de IA4HierarchyFolder que se corresponde con una carpeta en particular en la jerarquía de System i Navigator. Esta es una función de miembro requerida.

Sintaxis

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

Parámetros

hwnd El manejador de la ventana de vista que muestra la lista, que puede ser un control de árbol o de lista. Un componente debe utilizar este manejador para determinar si una lista de objetos de esta vista ya está almacenada en la antememoria en el cliente.

pidl Un puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) que identifica de forma exclusiva la carpeta que será enumerada.

riid Un identificador de la interfaz que devolver. Este parámetro apunta al identificador de interfaz IID_IA4HierarchyFolder.

ppvOut
Una dirección que recibe el puntero de la interfaz. Si se produce un error, debe devolverse un puntero NULL a esta dirección.

Códigos de retorno

Devuelve NOERROR si es satisfactoria o E_FAIL si se ha producido un error general.

Comentarios

Si ya existe una instancia de IA4HierarchyFolder para la carpeta especificada, esta función de miembro debería devolver la instancia en la antememoria en lugar de crear una instancia e

inicializar una instancia independiente. Sin embargo, si el manejador de la ventana asociado al objeto almacenado en la antememoria no coincide con el valor especificado en el parámetro hwnd, debe crearse una nueva instancia.

La función debe inicializar las variables de miembro de la clase de implementación a partir de los parámetros proporcionados.

IA4HierarchyFolder::DisplayErrorMessage:

Esta especificación se utiliza para mostrar un mensaje de error a un usuario final siempre que el método Activate devuelve un error. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

Ninguno.

Códigos de retorno

Devuelve NOERROR si es satisfactorio o un E_FAIL si no existe ningún mensaje que mostrar.

Comentarios

Ninguno.

IA4HierarchyFolder::GetAttributesOf:

Esta especificación devuelve los atributos de una carpeta determinada en la jerarquía de System i Navigator. Los indicadores de atributo son los mismos que los definidos para el método de la interfaz de Microsoft IShellFolder::GetAttributesOf. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

pidl Un puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) que identifica de forma exclusiva el objeto cuyos atributos se van a recuperar.

ulfInOut
Atributos del objeto devueltos. En la entrada, este parámetro se establece para indicar qué atributos de objetos recuperar.

Códigos de retorno

Devuelve NOERROR si es satisfactorio o E_FAIL si no puede localizar los atributos del objeto.

Comentarios

Consulte en el archivo include de Windows shlobj.h las constantes que definen los distintivos de bit.

System i Navigator llama de forma repetida a esta función cuando rellena una vista de árbol o lista. Por consiguiente deben evitarse las operaciones de larga ejecución.

IA4HierarchyFolder::GetColumnDataItem:

Esta especificación devuelve un campo de datos para que una carpeta u objeto aparezca en una columna en la vista de lista de System i Navigator. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

pidl Un puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) que identifica de forma exclusiva el objeto cuyos datos de columna serán obtenidos.

lParam
Valor asociado anteriormente a la columna para la que el componente solicita datos (consulte GetColumnInfo).

lpszColumnData
La dirección de la antememoria que recibe la serie de datos terminada en nulo.

cchMax
El tamaño de la antememoria que recibe la serie de datos terminados en nulo.

Códigos de retorno
Devuelve NOERROR si es satisfactorio o E_FAIL si no puede recuperar los datos de columna.

Comentarios

System i Navigator llama repetidamente a esta función cuando rellena una vista de lista. Por consiguiente deben evitarse las operaciones de larga ejecución.

IA4HierarchyFolder::GetColumnInfo:

Esta especificación devuelve una estructura de datos que describe las columnas necesarias para mostrar el contenido de una carpeta en particular en una vista detallada. Es una función de miembro opcional.

Sintaxis

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

Parámetros

ppvInfo

Estructura de datos devuelta. La estructura devuelta debe constar de una instancia de la estructura A4hyfColumnInfo. Esta estructura contiene una matriz de estructuras A4hyfColumnItem, una para cada una de las columnas de la vista de lista.

Cada estructura de elemento de columna proporciona la serie traducida para la cabecera de columna, la anchura por omisión de la columna y un valor entero que identifica de forma exclusiva el campo de datos que suministra datos para la columna. Consulte CWBA4HYF.H.

Códigos de retorno
Devuelve NOERROR si es satisfactoria y E_NOTIMPL si no puede implementar la función.

Comentarios

System i Navigator llama a esta función después de que se ha devuelto la llamada a Open para crear los encabezados de columna de una vista detallada.

Si esta función no se implementa, System i Navigator inserta dos columnas: Nombre y Descripción. La función GetColumnDataItem debe poder devolver datos para estos dos campos, identificados con valores enteros de 0 a 1.

Utilice la interfaz IMalloc de Windows para asignar memoria para las estructuras devueltas. System i Navigator es responsable de suprimir esta memoria.

IA4HierarchyFolder::GetIconIndexOf:

Esta especificación devuelve el índice a la DLL de recursos de componente que puede utilizarse para cargar el icono en la carpeta de jerarquía. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

pidl Un puntero a una estructura ITEMIDLIST (lista de identificadores de elementos) que identifican de forma exclusiva el objeto cuyo índice de iconos será recuperado.

uFlags La especificación del tipo de índice de icono que recuperar. Este parámetro debe ser cero, o debe contener el valor GIL_OPENICON, que indica que el icono que debe proporcionarse está en una carpeta abierta. GIL_OPENICON se define en el archivo include de Windows SHLOBJ.H.

piIndex
Un puntero a un entero que recibe el índice de icono.

Códigos de retorno
Devuelve NOERROR si es satisfactorio o E_FAIL si no es posible determinar el índice.

Comentarios

System i Navigator llama de forma repetida a esta función cuando rellena una vista de árbol o lista. Por consiguiente deben evitarse las operaciones de larga ejecución.

IA4HierarchyFolder::GetItemCount:

Esta especificación devuelve el número total de objetos contenidos en una carpeta en particular en la jerarquía de System i Navigator. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

pCount
Un puntero a un entero largo que recibe el número de elementos en la lista.

Códigos de retorno

- Devuelve A4HYF_OK_LISTCOMPLETE si la lista se crea por completo y el número total de elementos es conocido.
- Devuelve A4HYF_OK_LISTNOTCOMPLETE si la lista aún está siendo construida. En esta situación, el número de elementos representa el número de elementos en la lista parcialmente construida.
- Devuelve A4HYF_E_LISTDATAERROR si se encuentra un error mientras se está construyendo la lista. En esta situación, el número de elementos representa sólo aquellos elementos ya almacenados en la antememoria del cliente.

Comentarios

Tras un retorno satisfactorio del método Activate, System i Navigator llama a esta función para obtener el número de objetos de la carpeta que está a punto de ser rellenada. Tras la llamada a esta función, System i Navigator llama de forma repetida a ItemAt para obtener los identificadores de elementos de los objetos en la carpeta.

Para aquellas listas extremadamente largas, puede elegir devolver de la función Activate antes de haber almacenado toda la lista en la antememoria del cliente. Si este es el caso, necesitará devolver A4HYF_OK_LISTNOTCOMPLETE de la función GetItemCount. A partir de ese momento, System i Navigator llama a la función GetItemCount cada 10 segundos hasta que se devuelva A4HYF_OK_LISTCOMPLETE o A4HYF_E_LISTDATAERROR.

IA4HierarchyFolder::GetToolBarInfo:

Esta especificación devuelve una estructura que describe la barra de herramientas personalizada asociada con la carpeta específica en la jerarquía de System i Navigator. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

pidl Un puntero a una estructura ITEMIDLIST (lista de identificadores de elementos) que identifica de forma exclusiva el objeto del cual se recuperará información de la barra de herramientas.

ppvInfo
Estructura de datos devuelta. En este puntero debe devolverse una instancia de A4hyfToolBarInfo. Esta estructura proporciona el número de botones de barra de herramientas del objeto, la dirección de una matriz de estructuras TBBUTTON que contiene los atributos de cada botón y el handle de instancia del conector. Consulte el archivo de cabecera CWBA4HYF.H.

Códigos de retorno
Devuelve NOERROR si es satisfactorio o E_NOTIMPL si ha elegido no implementar la función.

Comentarios

Esta función se utiliza cada vez que un usuario selecciona una carpeta u objeto que pertenece a un conector de System i Navigator.

Utilice la interfaz IMalloc de Windows para asignar memoria para la estructura devuelta. System i Navigator es responsable de suprimir esta memoria.

Si esta función de miembro no está implementada, se utilizará la barra de herramientas de System i Navigator por omisión. Esta barra de herramientas contiene los botones Copiar, Pegar, Suprimir y Propiedades de las cuatro vistas de lista, y Renovar. System i Navigator llama a la implementación de IContextMenu::GetCommandString (con el distintivo GCS_VALIDATE establecido) que está en el producto para descubrir cuál de los botones de barra de herramienta debe habilitarse para los objetos.

IA4HierarchyFolder::GetListObject:

Dado un nombre de objeto calificado al completo, esta función devuelve un puntero a un objeto de proxy (creado por el conector) en la antememoria. Es una función de miembro opcional.

Sintaxis

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

Parámetros

lpszObjectName
El nombre del objeto totalmente calificado para el que se devolverá un objeto de lista.

ppvObj
Puntero devuelto a un objeto definido por la implementación. La rutina de llamada debe convertir este puntero en un tipo de objeto adecuado.

Códigos de retorno
Devuelve NOERROR si es satisfactorio o E_NOTIMPL si ha elegido no implementar la función.

Comentarios

La llamadas a esta función se producen siempre que un código de conector llama a la API cwbUN_GetListObjectFromName o cwbUN_GetListObjectFromPidl para obtener un objeto de proxy instanciado por el método Activate. El conector utiliza este objeto de proxy para acceder a los datos en el sistema o para realizar acciones en el mismo. Debido a que la implementación de IA4HierarchyFolder mantiene la antememoria de los objetos de proxy, el programa que realiza la llamada no debe suprimir el objeto.

IA4HierarchyFolder::ItemAt:

Esta especificación devuelve una estructura SHITEMID (identificador de elemento) para el objeto de carpeta en la posición especificada en la lista de contenidos de carpeta. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

ulIndex
Índice basado en cero del elemento para el que se solicita un identificador de elemento.

ppidl Dirección del puntero que recibe el identificador de elemento solicitado.

Códigos de retorno
Devuelve NOERROR si no es satisfactorio o E_FAIL si el elemento no está disponible. Devuelve E_OUTOFMEMORY si no hay suficiente memoria disponible para el identificador de elemento.

Comentarios

System i Navigator llamada de forma repetida a esta función para rellenar una carpeta en tiempo real. Por consiguiente deben evitarse las operaciones de larga ejecución. Consulte el archivo CWBA4HYF.H para obtener el formato de los identificadores de elementos de System i Navigator. Utilice la interfaz IMalloc de Windows para asignar memoria para el identificador de elemento.

IA4HierarchyFolder::ProcessTerminating:

Esta función se utiliza cuando el usuario cierra la ventana de System i Navigator. Permite que el conector guarde datos persistentes. Esta es una función de miembro opcional.

Sintaxis

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

Códigos de retorno
Devuelve NOERROR si la ejecución es satisfactoria o E_NOTIMPL si elige no implementar la función. Los valores de retorno de error se pasan por alto.

Comentarios

Ninguno.

IA4HierarchyFolder::Refresh:

Esta especificación destruye cualquier objeto de carpeta almacenado en la antememoria y reconstruye la antememoria utilizando nuevos datos obtenidos del sistema. Es una función de miembro obligatoria.

Sintaxis

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

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o A4HYF_E_LISTDATAERROR si se produce un error al acceder a los objetos de la carpeta.

Comentarios

System i Navigator llama a esta función siempre que esté realizando una renovación global de la ventana principal de System i Navigator.

Descripción de la interfaz IA4PropSheetNotify:

Al igual que la interfaz IA4HierarchyFolder, la interfaz IA4PropSheetNotify describe un conjunto de funciones que el proveedor de software independiente implementará. IA4PropSheetNotify es una interfaz COM (modelo de objeto de componente) que IBM ha definido para permitir que terceros puedan añadir páginas a cualquier hoja de propiedades que System i Navigator defina para un usuario.

El programa System i Navigator llama a los métodos en la interfaz IA4PropSheetNotify siempre que necesite comunicarse con el conector de terceros. El propósito de la interfaz es proporcionar una notificación cuando el diálogo principal Propiedades de un usuario se esté cerrando. La notificación indica si los cambios efectuados por el usuario deben guardarse o descartarse. La intención es que la interfaz se añada a la misma clase de implementación que se utiliza para IPropSheetExt.

La implementación de la interfaz se compila y se enlaza en la DLL servidora ActiveX para el conector. System i Navigator conoce la existencia de la nueva DLL por medio de entradas en el registro de Windows. Estas entradas especifican la ubicación de la DLL en el PC del usuario. System i Navigator carga entonces la DLL en el momento adecuado, llamando a los métodos en la interfaz IA4PropSheetNotify según sea necesario.

CWBA4HYF.H contiene declaraciones del prototipo de interfaz y las estructuras de datos y los códigos de retorno asociados.

Lista de especificaciones de la interfaz IA4PropSheetNotify:

La interfaz IA4PropSheetNotify proporciona notificaciones a la implementación de IShellPropSheetExt. Estas notificaciones son necesarias para añadir páginas de propiedades a una de las hojas de propiedades Usuarios y grupos.

Estas notificaciones son necesarias porque es posible que crear y destruir hojas de propiedades Usuarios y grupos muchas veces antes de que el usuario pulse **Aceptar** en el diálogo Propiedades principal. La interfaz IA4PropSheetNotify informa a la implementación IShellPropSheetExt cuando el usuario realiza cambios que deben ser guardados.

System i Navigator conoce la existencia de una implementación IA4PropSheetNotify por medio de las entradas normales de registro definidas para los conectores de System i Navigator. Además, cuando se registra un manejador de hoja de propiedades del componente Usuarios y grupos, se da soporte a un valor de registro especial, que permite que el conector especifique qué hoja de propiedades añadirá páginas.

Conceptos relacionados

"Páginas de propiedades para un manejador de hojas de propiedades" en la página 89
Las clases de la biblioteca Microsoft Foundation Class (MFC) no dan soporte a la creación de páginas de propiedades para un manejador de hojas de propiedades. No obstante, IBM suministra CExtPropertyPage in en lugar de la clase MFC CPropertyPage.

IA4PropSheetNotify::ApplyChanges:

A esta función se la llama para informar a la implementación de que los datos pertenecientes al usuario deben ahora ser guardados.

Sintaxis

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

Parámetros

pszNewUserName
El nombre del nuevo usuario si se está creando el usuario por primera vez; por ejemplo, si InformUserState especifica un valor distinto de IUS_USEREXISTS.

Códigos de retorno
Devuelve NOERROR si es satisfactorio o E_FAIL si se ha producido un error general.

Comentarios
Ninguno.

IA4PropSheetNotify::GetErrorMessage:

esta función se utiliza cuando se devuelven errores en la función ApplyChanges para recuperar el texto del mensaje de error de la implementación.

Sintaxis

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

Parámetros

pszErrMsg
Una dirección del almacenamiento intermedio que recibe el mensaje de error terminado en nulo.

cchMax
El tamaño de la antememoria que recibe el mensaje de error terminado en nulo.

Códigos de retorno
Devuelve NOERROR si es satisfactorio o E_FAIL si no puede recuperar el texto del mensaje o si el texto del mensaje es demasiado largo para el almacenamiento intermedio.

Comentarios
Ninguno.

IA4PropSheetNotify::InformUserState:

Esta función se llama inmediatamente a continuación de la creación de la instancia de IShellPropSheetExt. Informa a la implementación de si este usuario ya existe en el sistema o si está siendo creado por primera vez.

Sintaxis

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

Parámetros

wUserState

El estado actual del usuario. El sistema proporciona estos valores que se excluyen mutuamente:

- IUS_NEWUSER

  Se crea un usuario en base a los atributos proporcionados por el usuario de System i Navigator.
- IUS_NEWUSERBASEDON

  Se crea un usuario en base a los atributos de un usuario existente.
- IUS_USEREXISTS

  El usuario ya existe en el sistema.

Códigos de retorno

Devuelve NOERROR si es satisfactorio o E_FAIL si se ha producido un error error.

Comentarios

Ninguno.

API de System i Navigator

Las API de System i Navigator ayudan a los desarrolladores de conectores a obtener y gestionar ciertos tipos de información global.

Listado de API de System i Navigator:

La tabla lista las API de System i Navigator agrupadas por función.

Función	API de System i Navigator
Valores del sistema: esta API permite que el desarrollador del conector obtenga el valor actual de un valor del sistema.	"cwbUN_GetSystemValue" en la página 36
Manejador del sistema: estas API muestra al desarrollador del conector cómo obtener y liberar el valor actual de un manejador de objeto del sistema que contiene propiedades de conexión incluyendo valores de capa de sockets segura (SSL) para ser utilizados por el sistema especificado.	"cwbUN_GetSystemHandle" en la página 37 "cwbUN_ReleaseSystemHandle" en la página 38
Validación de entrada del usuario: estas API permiten que el desarrollador del conector compruebe si el usuario actual tiene autorización sobre un objeto de System i en particular. Las API también permiten que el desarrollador determine si el usuario tiene una o más autorizaciones especiales.	"cwbUN_CheckObjectAuthority" en la página 38 "cwbUN_CheckSpecialAuthority" en la página 39
Comprobación de la autorización del usuario: esta API permite que el desarrollador del conector compruebe si determinados tipos de series proporcionadas por el usuario son válidas antes de transmitirlas al sistema.	"cwbUN_CheckAS400Name" en la página 39
Atributos del perfil de usuario: esta API permite que el desarrollador del conector obtenga el valor de cualquiera de los atributos de perfil de usuario del usuario de System i Navigator actual.	"cwbUN_GetUserAttribute" en la página 40

Función	API de System i Navigator
Gestión de datos: Los objetos que el usuario ha seleccionado se identifican ante el conector de terceros mediante dos entidades de datos, la lista de identificadores de elemento y el nombre de objeto. Con las API de gestión de datos el desarrollador de conectores puede extraer información de estas estructuras.	"cwbUN_ConvertPidlToString" en la página 41 "cwbUN_GetDisplayNameFromItemId" en la página 42 "cwbUN_GetDisplayNameFromName" en la página 42 "cwbUN_GetDisplayPathFromName" en la página 43 "cwbUN_GetIndexFromItemId" en la página 44 "cwbUN_GetIndexFromName" en la página 44 "cwbUN_GetIndexFromPidl" en la página 44 "cwbUN_GetListObject" en la página 45 "cwbUN_GetParentFolderNameFromName" en la página 45 "cwbUN_GetParentFolderPathFromName" en la página 46 "cwbUN_GetParentFolderPidl" en la página 47 "cwbUN_GetSystemNameFromName" en la página 47 "cwbUN_GetSystemNameFromPidl" en la página 48 "cwbUN_GetTypeFromName" en la página 48 "cwbUN_GetTypeFromPidl" en la página 49
Renovación de la ventana de System i Navigator: a continuación de la operación en nombre del usuario, estas API permiten la ejecución de una solicitud por parte del conector de renovación del árbol o para colocar un mensaje en la barra de estado de System i Navigator.	"cwbUN_RefreshAll" en la página 49 "cwbUN_RefreshList" en la página 50 "cwbUN_RefreshListItems" en la página 50 "cwbUN_UpdateStatusBar" en la página 51
Conexiones de ODBC: estas API permiten que el desarrollador del conector reutilice y termine el manejador de una conexión ODBC que ya ha sido obtenido por el componente Base de datos de System i Navigator.	"cwbUN_GetODBCConnection" en la página 51 "cwbUN_EndODBCConnections" en la página 52
Acceso a los iconos de System i Navigator: estas API permiten que el desarrollador del conector acceda a las listas de imágenes de objetos que aparecen en la jerarquía de objetos de System i Navigator.	"cwbUN_GetIconIndex" en la página 52 "cwbUN_GetSharedImageList" en la página 53
Administración de Aplicaciones: Estas API permiten al desarrollador de conectores determinar programáticamente si se debe denegar o permitir a un usuario el uso de una función administrable. Una *función administrable* es cualquier función que puede ser controlada a través del subcomponente Administración de aplicaciones de System i Navigator.	"cwbUN_GetAdminCacheState" en la página 55 "cwbUN_GetAdminCacheStateEx" en la página 56 "cwbUN_GetAdminValue" en la página 53 "cwbUN_GetAdminValueEx" en la página 54
Instalación: esta API permite que el desarrollador del conector determine si un componente de System i Navigator está instalado.	"cwbUN_IsSubcomponentInstalled" en la página 57

Función	API de System i Navigator
Servicios de directorio: estas API proporcionan información acerca del servidor LDAP (protocolo ligero de acceso a directorios) en una plataforma System i. Estas API también proporcionan funciones para conectarse al servidor. Las funciones de conexión permiten conectar un servidor utilizando información que System i Access para Windows almacena en la antememoria, como nombres distinguidos y una contraseña. Las funciones de conexión utilizan el cliente LDAP incluido con System i Access para Windows (LDAP.LIB y LDAP.DLL) y, por lo tanto, requieren que la aplicación utilice dicho cliente. Las funciones que utilizan estas series están disponibles en versiones ANSI (American National Standards Institute) y Unicode. Las funciones que devuelven nombres distinguidos y otras series para ser utilizadas con API de clientes LDAP se proporcionan en una versión UTF-8 para ser utilizadas con los servidores LDAP versión 3.	"cwbUN_FreeLdapPublishing" en la página 61 "cwbUN_OpenLdapPublishing" en la página 60 "cwbUN_GetLdapPublishCount" en la página 62 "cwbUN_GetLdapPublishParentDn" en la página 65 "cwbUN_GetLdapPublishPort" en la página 64 "cwbUN_GetLdapPublishServer" en la página 63 "cwbUN_GetLdapPublishType" en la página 62 "cwbUN_GetLdapSvrPort" en la página 58 "cwbUN_GetLdapSvrSuffixCount" en la página 59 "cwbUN_GetLdapSvrSuffixName" en la página 60 "cwbUN_FreeLocalLdapServer" en la página 58 "cwbUN_OpenLocalLdapServer" en la página 57

cwbUN_GetSystemValue:

Esta API devuelve una serie que contiene un valor del sistema.

Sintaxis

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

Parámetros

const char * szSystemValueId - entrada
Un valor numérico que identifica el valor del sistema que recuperar. Las definiciones de las constantes de valor del sistema se encuentran en el archivo de cabecera CWBA4SVL.H.

char * szSystemValue - salida
Una dirección del almacenamiento intermedio que recibe la serie de valor del sistema terminado en nulo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe la serie del valor terminado en nulo.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_INTERNAL_ERROR
No se ha podido recuperar el valor del sistema.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización

El valor devuelto por esta API no es una serie NLS (soporte de idioma nacional) si no está traducido. Por ejemplo, se devolverá '*NONE' en lugar de 'Ninguno'.

cwbUN_GetSystemHandle:

Esta API devuelve un manejador del sistema que contiene los valores de seguridad, ID de usuario y contraseña utilizados por el sistema. El manejador del sistema tiene los valores configurados en System i Navigator para el nombre del sistema de entrada.

Si el nombre de la aplicación se establece en nulo (NULL), el handle del sistema devuelto será exclusivo. Si el nombre de la aplicación está establecido, se devolverá el mismo handle del sistema que coincide con el nombre de la aplicación.

Si una aplicación necesita un trabajo exclusivo de i5/OS para un sistema, deberá pasarse NULL o un nombre exclusivo como nombre de aplicación.

Si una aplicación necesita compartir un trabajo de i5/OS, todos los llamadores de dicha función deberán pasar el mismo nombre de aplicación.

Sintaxis

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

Parámetros

char * szSystemName - entrada
Un puntero a una serie ASCIIZ que contiene el nombre del sistema para el que desea crear un manejador del sistema.

char * szAppName - entrada
Un puntero a una serie ASCIIZ de no más de 12 caracteres. Identifica de forma exclusiva la aplicación que compartirá un único handle del sistema.

cwbCO_SysHandle * systemHandle - salida
Un puntero al manejador del sistema de este nombre del sistema.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_NULL_PARM
El nombre del sistema es nulo (NULL).

CWBUN_INVALID_NAME_PARM
El nombre del sistema no es válido.

CWB_NON_REPRESENTABLE_UNICODE_CHAR
Uno o varios caracteres UNICODE de entrada no tienen representación en la página de códigos que se utiliza.

CWB_API_ERROR
No es posible devolver el handle del sistema.

Utilización

Esta función debe ser utilizada por todas las aplicaciones de terceros que deseen soportar SSL utilizando las API de System i Access para Windows. Por ejemplo, todas las API de comunicaciones de System i Access para Windows requieren un manejador del sistema para soportar SSL.

Cuando el llamador de esta función ya no necesite el handle del sistema para las comunicaciones, se podrá liberar el handle efectuando una llamada a la función **cwbUN_ReleaseSystemHandle**.

Todos los manejadores son liberados cuando termina la aplicación de System i Navigator (cwbunnav.exe).

cwbUN_ReleaseSystemHandle:

Esta API libera un manejador del sistema que contiene los valores de seguridad que utilizar para el sistema. El manejador del sistema se obtiene utilizando la función cwbUN_GetSystemHandle. Si el llamador de esta función tiene la última referencia al manejador, los recursos del manejador serán destruidos.

Sintaxis

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

Parámetros

cwbCO_SysHandle * systemHandle - entrada
Un puntero al manejador del sistema que ha sido obtenido a partir de una llamada cwbUN_GetSystemHandle.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_API_ERROR
No es posible liberar el handle del sistema.

Utilización

Cuando el llamador de esta función ya no necesite el handle del sistema para las comunicaciones, se podrá liberar el handle.

cwbUN_CheckObjectAuthority:

Esta API devuelve una indicación de si el usuario de System i Navigator tiene la autoridad de un objeto en particular en el sistema.

Sintaxis

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

Parámetros

const char * szObjectPath - entrada
La vía de acceso de objeto de System i cuya autoridad debe comprobarse.

const char * szObjectType - entrada
El tipo de objeto de System i del objeto cuya autoridad debe comprobarse como, por ejemplo, *DTAQ.

const char * szAuthorityType - entrada
La autorización del objeto de System i que será comprobada.

Si se va a comprobar más de una autorización, las autorizaciones deben ir concatenadas (por ejemplo, *OBJMGT*OBJEXIST). Pueden especificarse hasta once tipos de autorización en una sola llamada. La función devuelve CWB_OK sólo si el usuario tiene todas las autorizaciones especificadas en el objeto.

const char * szSystemName - entrada
El nombre del sistema en el que realizar la comprobación.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
El usuario tiene la autorización especificada para el objeto.

CWBUN_USER_NOT_AUTHORIZED
El usuario no tiene la autorización especificada.

CWBUN_OBJECT_NOT_FOUND
No se ha podido comprobar el objeto especificado.

CWBUN_INTERNAL_ERROR
No se ha podido comprobar la autorización sobre objeto.

Utilización
Si se especifica *EXCLUDE como autorización, no puede especificarse ningún otro tipo de autorización. *AUTLMGT es sólo válida si szObjectType es *AUTL.

cwbUN_CheckSpecialAuthority:

Esta API devuelve una indicación de si el usuario de System i Navigator tiene una autorización determinada en el sistema.

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

Parámetros

const char * szSpecialAuthority - entrada
La autorización especial de System i que debe ser comprobada.

const char * szSystemName - entrada
El nombre del sistema en el que realizar la comprobación.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
El usuario tiene la autorización especial especificada.

CWBUN_USER_NOT_AUTHORIZED
El usuario no tiene la autorización especificada.

CWBUN_INTERNAL_ERROR
No se ha podido comprobar la autorización especial.

Utilización
Ninguna.

cwbUN_CheckAS400Name:

Esta API devuelve una indicación de si una serie es un parámetro de nombre válido en el sistema.

Sintaxis

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

Parámetros

const char * szAS400Name - entrada
El nombre del sistema cuya validez será comprobada.

const char * szSystemName - entrada
El nombre del sistema en el que realizar la comprobación.

USHORT usTypeId - entrada
Valor numérico que indica cómo debe interpretarse la serie de entrada: como un nombre de objeto largo, como un nombre de objeto corto, como un nombre de comunicaciones o como una serie (las constantes de tipo están definidas más arriba).

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_NAME_TOO_LONG
El nombre es demasiado largo.

CWBUN_NAME_NULLSTRING
La serie está vacía: no se ha encontrado ningún carácter.

CWBUN_NAME_INVALIDCHAR
Carácter no válido.

CWBUN_NAME_STRINGTOOLONG
La serie es demasiado larga.

CWBUN_NAME_MISSINGENDQUOTE
Falta la comilla final.

CWBUN_NAME_INVALIDQUOTECHAR
Carácter no válido para serie entrecomillada.

CWBUN_NAME_ONLYBLANKS
Se ha encontrado una serie que únicamente contiene blancos.

CWBUN_NAME_STRINGTOOSHORT
La serie es demasiado corta.

CWBUN_NAME_TOOLONGFORIBM
Serie correcta, pero demasiado larga para mandatos IBM.

CWBUN_NAME_INVALIDFIRSTCHAR
El primer carácter no es válido.

Utilización

Ninguna.

cwbUN_GetUserAttribute:

Esta API devuelve una serie que contiene el valor de un atributo de perfil de usuario del usuario de System i Navigator actual.

Sintaxis

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

Parámetros

USHORT usAttributeId - entrada
Un valor numérico que identifica el valor del atributo de usuario que recuperar. Definiciones de las constantes de atributos de usuario que se encuentran en el archivo de cabecera CWBA4USR.H.

const char * szSystemName - entrada
El nombre del sistema desde el que recuperar el atributo de usuario.

char * szValue - salida
Dirección del almacenamiento intermedio que recibe la serie del valor de atributo terminado en nulo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe la serie del valor terminado en nulo.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_INTERNAL_ERROR
No se ha podido recuperar el valor de atributo.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización

El valor devuelto por esta API no es una serie NLS y no está traducido. Por ejemplo, se devolverá '*NONE' en lugar de 'Ninguno'.

cwbUN_ConvertPidlToString:

Esta API convierte una de identificadores de elemento en System i Navigator en un nombre de objeto totalmente calificado.

Sintaxis

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

Parámetros

LPCITEMIDLIST pidl - entrada
Un puntero a la estructura ITEMIDLIST (lista de identificadores de elementos) que será convertida.

char * szObjectName - salida
Una dirección de la antememoria que recibe el nombre del objeto terminado en nulo.

UINT cchMax - entrada
El tamaño de la antememoria que recibe el nombre del objeto terminado en nulo.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
La lista de identificadores de elemento especificada no es válida.

WB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización

Ninguna.

cwbUN_GetDisplayNameFromItemId:

Esta API extrae el campo de nombre de elemento de un identificador de elemento Unity.

Sintaxis

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

Parámetros

const char * szItemId - entrada
El identificador de elemento Unity del que se extrae el nombre de elemento.

char * szItemName - salida
La dirección del almacenamiento intermedio que recibe el nombre de elemento calificado al completo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe el nombre de elemento terminado en nulo.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
El identificador de elemento especificado no es válido.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización

Ninguna.

cwbUN_GetDisplayNameFromName:

Esta API extrae el campo de nombre de elemento de un nombre de objeto Unity totalmente calificado.

Sintaxis

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

Parámetros

const char * szObjectName - entrada
El nombre de objeto Unity del cual se extraerá el nombre de elemento.

char * szItemName - salida
La dirección del almacenamiento intermedio que recibe el nombre de elemento calificado al completo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe el nombre de elemento terminado en nulo.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
El nombre de objeto especificado no es válido.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización
Ninguna.

cwbUN_GetDisplayPathFromName:

Esta API convierte un nombre de objeto Unity calificado al completo en un nombre de vía de acceso totalmente calificada adecuada para mostrarse al usuario.

Sintaxis

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

Parámetros

const char * szObjectName - entrada
El nombre de objeto Unity del cual deriva el nombre de vía de acceso.

char * szPathName - salida
Una dirección del almacenamiento intermedio que recibe un nombre de vía de acceso terminado en nulo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe el nombre de vía de acceso terminado en nulo.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
El nombre de objeto especificado no es válido.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización
Ninguna.

cwbUN_GetIndexFromItemId:

La API extrae el campo de índice de elemento de un identificador de elementos Unity.

Sintaxis

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

Parámetros

const char * szItemId - entrada
Identificador de elementos Unity del que se extrae el elemento de índice.

ULONG* piIndex - salida
Una dirección de un entero largo sin signo que recibirá el elemento de índice.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
El identificador de elemento especificado no es válido.

Utilización

Ninguna.

cwbUN_GetIndexFromName:

Esta API extrae el campo de índice de elemento de un nombre de objeto totalmente calificado Unity.

Sintaxis

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

Parámetros

const char * szObjectName - entrada
El nombre de objeto Unity del que se extraerá el índice de elemento.

ULONG* piIndex - salida
Una dirección de un entero largo sin signo que recibirá el elemento de índice.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
El nombre de objeto especificado no es válido.

Utilización

Ninguna.

cwbUN_GetIndexFromPidl:

Esta API extrae el campo de índice de elemento de una lista de identificadores de elementos Unity totalmente calificados.

Sintaxis

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

Parámetros

LPCITEMIDLIST pidl - entrada
Un puntero a una estructura ITEMIDLIST (lista de identificadores de elementos) de la que se extraerá el índice de elemento.

ULONG* piIndex - salida
Una dirección de un entero largo sin signo que recibirá el elemento de índice.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
La lista de identificadores de elemento especificada no es válida.

Utilización
Ninguna.

cwbUN_GetListObject:

Esta API obtiene un puntero al objeto asociado con el nombre de objeto de lista especificado.

Sintaxis

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

Parámetros

const char * szFileName - entrada
El nombre de objeto Unity a partir del cual se encontrará y devolverá el puntero del objeto.

LPVOID pListObject - salida
Una dirección de un puntero al objeto Unity de solicitud.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

Utilización
Ninguna.

cwbUN_GetParentFolderNameFromName:

Esta API extrae el nombre de la carpeta padre de un objeto de un nombre de objeto Unity totalmetne calificado.

Sintaxis

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

Parámetros

const char * szObjectName - entrada
El nombre de objeto Unity del cual se extrae el nombre de la carpeta padre.

char * szParentFolderPath - salida
Una dirección del almacenamiento intermedio que recibe un nombre de carpeta padre terminada en nulo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe el nombre de la carpeta padre terminado en nulo.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
El nombre de objeto especificado no es válido.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización
Ninguna.

cwbUN_GetParentFolderPathFromName:

Dado un nombre de objeto Unity totalmente calificado, esta API devuelve el nombre de objeto totalmente calificado de la carpeta padre del objeto.

Sintaxis

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

Parámetros

const char * szObjectName - entrada
El nombre de objeto Unity del cual se extra el nombre de objeto de carpeta padre.

char * szParentFolderPath - salida
Una dirección del almacenamiento intermedio que recibe el nombre de objeto de la carpeta padre terminado en nulo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe el nombre del objeto de carpeta padre terminado en nulo.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
El nombre de objeto especificado no es válido.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización
Ninguna.

cwbUN_GetParentFolderPidl:

Dada una lista de identificadores de elementos Unity totalmente calificada, esta API devuelve la lista de identificadores de elementos totalmente calificada de la carpeta padre del objeto.

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

Parámetros

LPCITEMIDLIST pidl - entrada
Un puntero a un estructura ITEMIDLIST (lista de identificadores de elementos) de la cual se extrae la lista de identificadores de elementos de carpeta padre.

LPITEMIDLIST* ppidl - salida
Una dirección de un puntero de lista de identificador de elementos que recibe la lista de identificadores de elementos de carpeta padre.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
La lista de identificadores de elemento especificada no es válida.

Utilización
Ninguna.

cwbUN_GetSystemNameFromName:

Esta API extrae el nombre del sistema de un nombre de objeto Unity totalmente calificado.

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

Parámetros

const char * szObjectName - entrada
El nombre de objeto Unity del que se extrae el nombre del sistema.

char * szSystemName - salida
Una dirección de la antememoria que recibe el nombre del sistema terminado en nulo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe el nombre de sistema terminado en nulo.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
El nombre de objeto especificado no es válido.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización
Ninguna.

cwbUN_GetSystemNameFromPidl:

Esta API extrae el nombre del sistema de una lista de identificadores de elementos Unity totalmente calificados.

Sintaxis

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

Parámetros

LPCITEMIDLIST pidl - entrada
Un puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) de la que extraer el nombre del sistema.

char * szSystemName - salida
Una dirección de la antememoria que recibe el nombre del sistema terminado en nulo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe el nombre de sistema terminado en nulo.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
La lista de identificadores de elemento especificada no es válida.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización
Ninguna.

cwbUN_GetTypeFromName:

Esta API extrae el campo de tipo de elemento de un nombre de objeto Unity totalmente calificado.

Sintaxis

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

Parámetros

const char * szObjectName - entrada
El nombre de objeto Unity del que se extraerá el índice de elemento.

char * szType - salida
La dirección del almacenamiento intermedio que recibe el tipo de elemento calificado al completo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe el elemento de tipo terminado en nulo.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
El nombre de objeto especificado no es válido.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización
Ninguna.

cwbUN_GetTypeFromPidl:

Esta API extrae el campo de índice de elemento de una lista de identificadores de elementos Unity totalmente calificados.

Sintaxis

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

Parámetros

LPCITEMIDLIST pidl - entrada
Un puntero a una estructura ITEMIDLIST (lista de identificadores de elementos) de la que se extraerá el índice de elemento.

char * szType - salida
La dirección del almacenamiento intermedio que recibe el tipo de elemento calificado al completo.

UINT cchMax - entrada
El tamaño del almacenamiento intermedio que recibe el elemento de tipo terminado en nulo.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_FORMAT_NOT_VALID
La lista de identificadores de elemento especificada no es válida.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio es demasiado pequeño para contener la serie devuelta.

Utilización
Ninguna.

cwbUN_RefreshAll:

Esta API renueva el contenido de la ventana de árbol y la ventana de lista de System i Navigator.

Sintaxis

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

Parámetros

const char * pszStatusText - entrada
Serie terminada en nulo que se debe colocar en la ventana de la barra de estado al finalizar. Este parámetro puede ser nulo (NULL).

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_WINDOW_NOTAVAIL
No se han podido encontrar las ventanas de vista.

Utilización

Utilice esta función para renovar todo el contenido de System i Navigator después de que el sistema realice una acción solicitada por el usuario.

cwbUN_RefreshList:

Esta API renueva el contenido de la ventana de vista de lista para System i Navigator.

Sintaxis

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

Parámetros

const char * pszStatusText - entrada
Serie terminada en nulo que se debe colocar en la ventana de la barra de estado al finalizar. Este parámetro puede ser nulo (NULL).

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_WINDOW_NOTAVAIL
No se ha podido encontrar la ventana de la vista de lista.

Utilización

Utilice esta función para renovar el contenido de la ventana de lista tras llevar a cabo una acción solicitada por el usuario.

cwbUN_RefreshListItems:

Esta API renueva el elemento o elementos seleccionados en la actualidad en la ventana de vista de lista de System i Navigator.

Sintaxis

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

Parámetros

const char * pszStatusText - entrada
Serie terminada en nulo que se debe colocar en la ventana de la barra de estado al finalizar. Este parámetro puede ser nulo (NULL).

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_WINDOW_NOTAVAIL
No se ha podido encontrar la ventana de la vista de lista.

Utilización
Utilice esta función para renovar los elementos seleccionados en la ventana de lista tras llevar a cabo una acción solicitada por el usuario.

cwbUN_UpdateStatusBar:

Esta API inserta una serie de texto en la barra de estado de la ventana de System i Navigator.

Sintaxis

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

Parámetros

const char * pszStatusText - entrada
Serie terminada en nulo que se debe colocar en la ventana de la barra de estado al finalizar.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_WINDOW_NOTAVAIL
No es posible encontrar la ventana de la barra de estado.

Utilización
Utilice esta función para informar al usuario de que una acción solicitada pulsando el botón Aceptar de un diálogo se ha completado satisfactoriamente.

cwbUN_GetODBCConnection:

Esta API devuelve el manejador para una conexión ODBC (Open Database Connectivity) en el sistema especificado. Si no existe ninguna conexión con el sistema especificado, la API obtendrá un nuevo manejador.

Sintaxis

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

Parámetros

const char * szSystemName - entrada
El nombre del sistema en el que recuperar una conexión ODBC.

HDBC *phDBC - salida
La dirección a la que devolver el manejador de la conexión ODBC.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

Utilización

Ninguna.

cwbUN_EndODBCConnections:

Esta API finaliza todas las conexiones de ODBC (Open Database Connectivity) abiertas anteriormente por la API cwbUN_GetODBCConnection.

Sintaxis

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

Parámetros

Ninguno.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
El handle no se ha creado.

Utilización

Es importante recordar que la función **EndODBCConnections** únicamente cierra las conexiones abiertas mediante la función **GetODBCConnection**. La función **EndODBCConnections** no tiene conocimiento de las conexiones ODBC abiertas directamente o utilizando otras interfaces.

Asimismo, asegúrese de que el destructor de la carpeta de su extensión de aplicación invoque **EndODBCConnections** si algún código de la extensión utiliza **GetODBCConnection**.

Consulte también **cwbUN_GetODBCConnection**.

cwbUN_GetIconIndex:

Esta API obtiene el índice en la lista de imágenes del icono especificado.

Sintaxis

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

Parámetros

LPCITEMIDLIST pidl - entrada
Un puntero a la estructura ITEMIDLIST (lista de identificadores de elemento) que se utiliza para identificar el icono al que se hará referencia.

UINT uFlags - entrada
La especificación del índice de tipo de icono que recuperar (definido anteriormente).

int * piIndex - salida
Una dirección del entero que recibe el índice de icono.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_INVALID_FLAG_VALUE
El valor de distintivo no es un valor soportado válido.

Utilización
Ninguna.

cwbUN_GetSharedImageList:

Esta API recupera la lista de imágenes de iconos asociadas con System i Navigator.

Sintaxis

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

Parámetros

UINT uFlags - entrada
La especificación de la lista de tipos de imágenes que recuperar (definida anteriormente).

HIMAGELIST* phImageList -
Dirección de la variable que recibirá el manejador de lista de imágenes.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWBUN_INVALID_FLAG_VALUE
El valor de distintivo no es un valor soportado válido.

CWBUN_CANT_GET_IMAGELIST
Se ha producido una anomalía al intentar obtener la lista de imágenes de icono.

Utilización
Ninguna.

cwbUN_GetAdminValue:

Esta API devuelve información sobre si el usuario actual de System i Navigator en un sistema especificado tiene permiso o no en una función administrativa. Una *Función administrativa* es cualquier función cuya utilización pueda ser controlada a través del subcomponente de Administración de System i Navigator.

Por ejemplo, un administrador puede utilizar Administración de aplicaciones para controlar si un usuario tiene acceso a varias de las funciones en System i Navigator. Una de estas funciones es la gestión de trabajos. Puede utilizar la API de cwbUN_GetAdminValue para determinar programáticamente si el usuario actual de System i Navigator puede utilizar la gestión de trabajos especificando el nombre de la función administrativa que se corresponde a la gestión del trabajo. Consulte el archivo de cabecera cwbunpla.h para obtener una lista de nombres de función administrables soportadas en System i Navigator.

Sintaxis

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

Parámetros

const char * szSystemName
El nombre del sistema en el que realizar la comprobación.

char* adminFunction
Puntero a una serie ASCII que contiene el nombre de la función administrable. La serie debe estar terminada en nulo y puede tener una longitud máxima de 30 bytes + 1 byte para el terminador nulo (NULL). Consulte en cwbunpla.h una lista de los valores de entrada soportados.

cwbUN_Usage & usageValue
Este valor solo es válido si se devuelve el código de retorno CWB_OK. Se devolverá uno de los dos valores posibles:

- cwbUN_granted -- Se permite al usuario el uso de la función.
- cwbUN_denied -- Se deniega al usuario el uso de la función.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
La API ha finalizado de forma satisfactoria.

CWBSY_USER_CANCELLED
El usuario ha cancelado la solicitud de ID de usuario y contraseña que la API ha visualizado.

Utilización

Esta API determina si el usuario actual de System i Navigator del sistema especificado puede utilizar la función especificada. Si no hay ningún usuario actualmente conectado al sistema especificado, la API inicia la sesión del usuario, posiblemente mostrando un indicador de ID de usuario y contraseña.

Esta API sólo puede ser utilizara para comprobar funciones administrativas en System i Navigator o en la categoría Aplicaciones cliente.

cwbUN_GetAdminValueEx:

Esta API devuelve una indicación de si el usuario actual del sistema especificado tiene permiso o no para utilizar una función administrable específica. Una *función administrable* es cualquier función que puede ser controlada a través del subcomponente Administración de aplicaciones de System i Navigator.

Nota: Los conectores de System i Navigator deben utilizar la API de cwbUN_GetAdminValue en lugar de cwbUN_GetAdminValueEx.

Un administrador puede utilizar Administración de de aplicaciones para controlar si un usuario tiene acceso a varias funciones en System i Navigator. Una de estas funciones es la gestión de trabajos. La API de cwbUN_GetAdminValueEx API puede ser utilizada para determinar programáticamente si el usuario actual puede utilizar la función de gestión de trabajos especificando el el nombre de la función administrable que corresponde a la gestión de trabajos. Consulte el archivo de cabecera CWBUNPLA.H para obtener una lista de nombres de funciones administrables soportadas en System i Navigator.

Esta API proporciona la misma función que cwbUN_GetAdminValue, con la diferencia de que está diseñada para admitir un handle de objeto del sistema en lugar de un nombre de sistema.

Sintaxis

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

Parámetros

cwbCO_SysHandle* pSysHandle
Puntero a un handle de objeto del sistema. El nombre de sistema debe ser especificado en el objeto antes de llamar a esta API. El comportamiento de la API cwbUN_GetAdminValueEx depende de si el objeto ha obtenido un inicio de sesión en el sistema:

No conectado->
cwbUN_GetAdminValueEx inicia sesión en el sistema. Se descargarán los valores de Administración de aplicaciones más recientes del usuario del sistema si no están ya almacenados en la estación de trabajo cliente.

Conectado->
Si el objeto del sistema ha iniciado sesión en un sistema que especifica que el ID de usuario y contraseña de System i deben ser validados (modalidad de validación), la API cwbUN_GetAdminValueEx utiliza una instantánea de los valores de Administración de aplicaciones exactos tras completarse el inicio de sesión. Si el inicio de sesión se realizó sin validar el ID de usuario y la contraseña, es posible que cwbUN_GetAdminValueEx esté utilizando una copia de los valores de Administración de aplicaciones cuya antigüedad sea como máximo de 24 horas.

char* adminFunction
Puntero a una serie ASCII que contiene el nombre de la función administrable. La serie debe estar terminada en nulo y puede tener una longitud máxima de 30 bytes + 1 byte para el terminador nulo (NULL). Consulte en CWBUNPLA.H una lista de los valores de entrada soportados.

cwbUN_Usage& usageValue
Este valor solo es válido si se devuelve el código de retorno CWB_OK. Se devolverá uno de los dos valores posibles:

cwbUN_granted
Se permite al usuario el uso de la función.

cwbUN_denied
Se deniega al usuario el uso de la función.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
La API ha finalizado de forma satisfactoria.

CWBSY_USER_CANCELLED
El usuario ha cancelado la solicitud de ID de usuario y contraseña que la API ha visualizado.

Utilización

Esta API determina si el usuario del sistema actual (tal como está definido por el objeto del sistema de entrada) puede utilizar la función especificada. Si no hay ningún usuario actualmente conectado al sistema especificado, la API inicia la sesión del usuario, posiblemente mostrando un indicador de ID de usuario y contraseña.

Esta API puede sólo ser utilizada para comprobar las funciones administrativas que se encuentran en System i Navigator o en la categoría funcional Aplicaciones cliente.

cwbUN_GetAdminCacheState:

Esta API indica si la siguiente invocación de la API cwbUN_GetAdminValue será de larga ejecución. La API de cwbUN_GetAdminValue almacena datos en la antememoria en la estación de trabajo. Si la

antememoria no es actual, cwbUN_GetAdminValue puede presentar un indicador de inicio de sesión o realizar otros procesos para actualizar su antememoria.

Sintaxis

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

Parámetros

const char * szSystemName
El nombre del sistema en el que realizar la comprobación.

cwbUN_State& adminState
Un parámetro que indica si la siguiente invocación de la API cwbUN_GetAdminValue es de larga ejecución o si utiliza su antememoria interna para devolver sin acceder al sistema principal.

Se devolverá uno de los siguientes valores:

cwbUN_logon
No hay ningún usuario actual para el sistema especificado. La API cwbUN_GetAdminValue puede presentar un indicador de inicio de sesión.

cwbUN_refresh
cwbUN_GetAdminValue accede al sistema para actualizar su antememoria interna.

cwbUN_cache
cwbUN_GetAdminValue tiene una antememoria actualizada y no será de larga ejecución.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
La API ha finalizado de forma satisfactoria.

Utilización

Los usuarios de cwbUN_GetAdminValue pueden utilizar esta API para determinar si la siguiente invocación de cwbUN_GetAdminValue es de larga ejecución.

cwbUN_GetAdminCacheStateEx:

Esta API indica si la siguiente invocación de la API cwbUN_GetAdminValueEx es de larga invocación. La API cwbUN_GetAdminValueEx almacena datos en la antememoria en la estación de trabajo. Si la antememoria no esta actualizada, la API cwbUN_GetAdminValueEx puede presentar un indicador de inicio de sesión o realizar otros procesos para actualizar su antememoria.

Sintaxis

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

Parámetros

cwbCO_SysHandle* pSysHandle - entrada
Puntero a un handle de objeto del sistema. El nombre de sistema debe especificarse en el objeto del sistema antes de efectuar una llamada a esta API.

cwbUN_State& adminState
Un parámetro que indica si la siguiente invocación de la API cwbUN_GetAdminValue es de larga ejecución o si utiliza su antememoria interna para devolver sin acceder al sistema principal.

Se devolverá uno de los siguientes valores:

cwbUN_logon
No hay ningún usuario actual para el sistema especificado. La API cwbUN_GetAdminValue puede presentar un indicador de inicio de sesión.

cwbUN_refresh
cwbUN_GetAdminValue accede al sistema para actualizar su antememoria interna.

cwbUN_cache
cwbUN_GetAdminValue tiene una antememoria actualizada y no será de larga ejecución.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
La API ha finalizado de forma satisfactoria.

Utilización
Los usuarios de cwbUN_GetAdminValueEx puede utilizar esta API para determinar si la siguiente invocación de cwbUN_GetAdminValueEx es de larga ejecución.

cwbUN_IsSubcomponentInstalled:

Esta API determina si un subcomponente de System i Navigator está instalado en la estación de trabajo.

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

Parámetros

UNIT uOption
Este parámetro especifica el subcomponente de System i Navigator que comprobar. En el prólogo de la API que encontrará en cwbun.h puede consultar una lista de los valores soportados.

Códigos de retorno
Devuelve un valor booleano.

TRUE Si el subcomponente está instalado.

FALSE
Si el subcomponente no está instalado.

Utilización
Ninguna.

cwbUN_OpenLocalLdapServer:

Esta API crea un manejador que puede utilizarse para acceder a la información de configuración acerca del servidor LDAP (protocolo ligero de acceso a directorios) en el sistema.

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

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

Parámetros

LPCSTR system - entrada
Un puntero al nombre del sistema.

cwbUN_ldapSvrHandle *pHandle - salida
De retorno, contiene un handle que puede utilizarse con las siguientes API:

- cwbUN_FreeLocalLdapServer
- cwbUN_GetLdapSvrPort
- cwbUN_GetLdapSvrSuffixCount
- cwbUN_GetLdapSuffixName

Nota: Este manejador debe liberarse con una llamada a cwbUN_FreeLocalLdapServer.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_PARAMETER
Se ha especificado un parámetro no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

CWBUN_LDAP_NOT_AVAIL
Los servicios de directorio no están instalados o el servidor no está configurado.

Utilización
Ninguna.

cwbUN_FreeLocalLdapServer:

Esta api libera recursos asociados con el manjeador de entrada.

Sintaxis

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

Parámetros

cwbUN_ldapSvrHandle handle - entrada
Handle para el que se liberarán recursos.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
El handle no se ha creado mediante cwbUN_OpenLocalLdapServer().

Utilización
El handle se obtiene mediante una llamada a cwbUN_OpenLocalLdapServer.

cwbUN_GetLdapSvrPort:

Esta API devuelve el número de puerto que será utilizado por el servidor LDAP (protocolo ligero de acceso a directorio).

Sintaxis

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

Parámetros

cwbUN_ldapSvrHandle handle - entrada
Handle obtenido anteriormente mediante una llamada a cwbUN_OpenLocalLdapServer().

int * port - salida
Número de puerto utilizado para las conexiones LDAP.

int * sslPort - salida
Número de puerto utilizado para las conexiones SSL.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
Handle no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

Utilización
Ninguna.

cwbUN_GetLdapSvrSuffixCount:

Esta API devuelve el número de sufijos configurados para este servidor. Un sufijo es el nombre distinguido (DN) de un punto de inicio en el árbol de directorios.

Sintaxis

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

Parámetros

cwbUN_ldapSvrHandle handle - entrada
Handle obtenido anteriormente mediante una llamada a cwbUN_OpenLocalLdapServer().

int * count - salida
Devuelve el número de sufijos presentes en el servidor.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
Handle no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

Utilización
Ninguna.

cwbUN_GetLdapSvrSuffixName:

Esta API devuelve el nombre distinguido del sufijo.

Sintaxis

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

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

int cwbUN_GetLdapSuffixName8 /* devuelve el sufijo en UTF-8 */
    ( cwbUN_ldapSvrHandle      handle,
      int                      index,
      LPSTR                    suffix,
      int                      *length
    );
```

Parámetros

cwbUN_ldapSuffixHandle handle - entrada
Handle obtenido anteriormente mediante una llamada a cwbUN_OpenLocalLdapServer().

int index - entrada
Índice basado en cero del sufijo. Este valor debe ser inferior al número devuelto por cwbUN_GetLdapSvrSuffixCount().

LPSTR suffix - salida
Un puntero al almacenamiento intermedio que contiene el nombre distinguido del sufijo.

int * length - entrada/salida
Un puntero a la longitud del almacenamiento intermedio del sufijo. Si el almacenamiento intermedio es demasiado pequeño para contener la serie, incluyendo espacio para el NULL de terminación, el tamaño de la antememoria se completará en este parámetro.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
Handle no válido.

CWB_INVALID_API_PARAMETER
Índice no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio de sufijo no es suficientemente grande para albergar el resultado completo.

Utilización

Ninguna.

cwbUN_OpenLdapPublishing:

Esta API crea un manejador que puede ser utilizado para acceder a información de configuración acerca de la información que será publicada por el sistema en los directorios LDAP (protocolo ligero de acceso a directorios).

Sintaxis

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

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

Parámetros

LPCSTR system - entrada
Un puntero al nombre del sistema.

cwbUN_ldapSvrHandle *pHandle - salida
De retorno, contiene un manejador que puede ser utilizado con API.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_PARAMETER
Se ha especificado un parámetro no válido.

CWB_INVALID_API_HANDLE
Handle no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

CWBUN_LDAP_NOT_AVAIL
Los servicios de directorio (Directory Services) no están instalados o el servidor no ha sido configurado.

Utilización

Ninguna.

cwbUN_FreeLdapPublishing:

Esta api libera recursos asociados con el manjeador de entrada.

Sintaxis

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

Parámetros

cwbUN_ldapPubHandle handle - entrada
Handle para el que se liberarán recursos.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
El handle no se ha creado mediante cwbUN_OpenLdapPublishing().

Utilización
El handle se obtiene mediante una llamada a cwbUN_OpenLdapPublishing().

cwbUN_GetLdapPublishCount:

Esta API devuelve el número de registros de publicación configurados para el servidor. Un registro de publicación identifica una categoría de información que será publicada y cómo y dónde será publicada.

Sintaxis

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

Parámetros

cwbUN_ldapPubHandle handle - entrada
Handle obtenido anteriormente mediante una llamada a cwbUN_OpenLdapPublishing().

int * count - salida
El número de registros de publicación configurados en el servidor.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
Handle no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

Utilización
Ninguna.

cwbUN_GetLdapPublishType:

Esta API devuelve el tipo de información de registro.

Sintaxis

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

Parámetros

cwbUN_ldapPubHandle handle - entrada
Handle obtenido anteriormente mediante una llamada a cwbUN_OpenLdapPublishing().

int index - entrada
Índice basado en cero del registro de publicación. Este valor debe ser inferior al número devuelto por cwbUN_GetLdapPublishCount().

cwbUN_LdapPubCategories * information - salida
Tipo de información al que corresponde este registro de publicación. Los valores posibles son:

CWBUN_LDAP_PUBLISH_USERS
Información del usuario.

CWBUN_LDAP_PUBLISH_COMPUTERS
Plataformas de System i.

CWBUN_LDAP_PUBLISH_NETWORK_INVENTORY
NetFinity.

CWBUN_LDAP_PUBLISH_PRINTERS
Impresoras conectadas a la plataforma de System i.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
Handle no válido.

CWB_INVALID_API_PARAMETER
Índice no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

Utilización
Ninguna.

cwbUN_GetLdapPublishServer:

Esta API devuelve el nombre del servidor en el que se publicará esta información.

Sintaxis

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

int cwbUN_GetLdapPublishServerA
    ( cwbUN_ldapPubHandle  handle,
      int                  index,
      LPSTR                server,
      int                  *length
    );
```

Parámetros

cwbUN_ldapPubHandle handle - entrada
Handle obtenido anteriormente mediante una llamada a cwbUN_OpenLdapPublishing().

int index - entrada
Índice basado en cero del registro de publicación. Este valor debe ser inferior al número devuelto por cwbUN_GetLdapPublishCount().

LPSTR server - salida
Un puntero al almacenamiento intermedio que contiene el nombre del servidor.

int * length - entrada/salida
Un puntero a la longitud del almacenamiento intermedio del servidor. Si el almacenamiento intermedio es demasiado pequeño para contener la serie, incluyendo espacio para el NULL de terminación, el tamaño de la antememoria se completará en este parámetro.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
Handle no válido.

CWB_INVALID_API_PARAMETER
Índice no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio de sufijo no es suficientemente grande para albergar el resultado completo.

Utilización

Ninguna.

cwbUN_GetLdapPublishPort:

Esta API devuelve el número de puerto del servidor utilizado para publicar esta información.

Sintaxis

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

Parámetros

cwbUN_ldapPubHandle handle - entrada
Handle obtenido anteriormente mediante una llamada a cwbUN_OpenLdapPublishing().

int index - entrada
Índice basado en cero del registro de publicación. Este valor debe ser inferior al número devuelto por cwbUN_GetLdapPublishCount().

int * port - salida
Número de puerto utilizado para conectarse al servidor.

cwbUN_LdapCnnSecurity * connectionSecurity - salida
Tipo de conexión utilizado para conectarse al servidor. Indica el tipo de conexión que puede establecerse a través del puerto asociado. Este parámetro permite estos valores:

CWBUN_LDAPCNN_NORMAL
Se utiliza una conexión normal.

CWBUN_LDAPCNN_SSL
Se utiliza una conexión SSL.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
Handle no válido.

CWB_INVALID_API_PARAMETER
Índice no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

Utilización
Ninguna.

cwbUN_GetLdapPublishParentDn:

Esta API devuelve el nombre distinguido del padre de los objetos publicados.

Por ejemplo, si el nombre distinguido padre (parentDN) para los usuarios de la publicación fuera `cn=users,o=ace industry,c=us` y la información de usuario se publicara para John Smith, el nombre distinguido del objeto publicado podría ser `cn=john smith,cn=users,ou=ace industry,c=us`.

Sintaxis

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

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

int cwbUN_GetLdapPublishParentDn8 /* devuelve parentDn en UTF-8 */
    ( cwbUN_ldapPubHandle  handle,
      int                  index,
      LPSTR                parentDn,
      int                  *length
    );
```

Parámetros

cwbUN_ldapPubHandle handle - entrada
Handle obtenido anteriormente mediante una llamada a cwbUN_OpenLdapPublishing().

int index - entrada
Índice basado en cero del registro de publicación. Este valor debe ser inferior al número devuelto por cwbUN_GetLdapPublishCount().

LPSTR parentDn - salida
Un puntero al almacenamiento intermedio que contiene el nombre de parentDn.

int * length - entrada/salida
Un puntero a la longitud del almacenamiento intermedio de parentDn. Si el almacenamiento intermedio es demasiado pequeño para contener la serie, incluyendo espacio para el NULL de terminación, el tamaño de la antememoria se completará en este parámetro.

Códigos de retorno
La lista siguiente muestra valores de retorno comunes:

CWB_OK
Finalización satisfactoria.

CWB_INVALID_API_HANDLE
Handle no válido.

CWB_INVALID_API_PARAMETER
Índice no válido.

CWB_INVALID_POINTER
Se ha especificado un puntero NULL.

CWB_BUFFER_OVERFLOW
El almacenamiento intermedio de sufijo no es suficientemente grande para albergar el resultado completo.

Utilización
Ninguna.

Códigos de retorno exclusivos de las API de System i Navigator

System i Navigator tiene un conjunto de códigos de retorno específicos. Cada código tiene su propio significado.

```
6000    CWBUN_BAD_PARAMETER
          Un parámetro de entrada no es válido.
6001    CWBUN_FORMAT_NOT_VALID
          El nombre de objeto de entrada no es válido.
6002    CWBUN_WINDOW_NOTAVAIL
          No se ha encontrado la ventana de vista.
6003    CWBUN_INTERNAL_ERROR
          Se ha producido un error de proceso.
6004    CWBUN_USER_NOT_AUTHORIZED
          El usuario no tiene la autorización especificada.
6005    CWBUN_OBJECT_NOT_FOUND
          Objeto no encontrado en el iSeries.
6006    CWBUN_INVALID_ITEM_ID
          Parámetro de ID de elemento no válido.
6007    CWBUN_NULL_PARM
          Se ha pasado un parámetro nulo (NULL).
6008    CWBUN_RTN_STR_TOO_LONG
          Serie demasiado larga para almacenamiento intermedio de retorno.
6009    CWBUN_INVALID_OBJ_NAME
          Parámetro de nombre de objeto no válido.
6010    CWBUN_INVALID_PIDL
          Parámetro de PIDL no válido.
6011    CWBUN_NULL_PIDL_RETURNED
          El PIDL de la carpeta padre es nulo (NULL).
6012    CWBUN_REFRESH_FAILED
          La operación de renovar lista ha fallado.
6012    CWBUN_UPDATE_FAILED
          La operación de actualizar barra de herramientas ha fallado.
6013    CWBUN_INVALID_NAME_TYPE
          Tipo de nombre de iSeries no válido.
6014    CWBUN_INVALID_AUTH_TYPE
          Tipo de autorización no válido.
6016    CWBUN_HOST_COMM_ERROR
          Error de comunicaciones de iSeries.
6017    CWBUN_INVALID_NAME_PARM
          Parámetro de nombre no válido.
6018    CWBUN_NULL_DISPLAY_STRING
          Se ha devuelto una serie de visualización nula.
6019    CWBUN_GENERAL_FAILURE
          Error general de funcionamiento de iSeries.
6020    CWBUN_INVALID_SYSVAL_ID
          ID de valor del sistema no válido.
6021    CWBUN_INVALID_LIST_OBJECT
          No es posible obtener objeto de lista a partir de nombre.
6022    CWBUN_INVALID_IFS_PATH
          La vía de acceso IFS especificada no es válida.
6023    CWBUN_LANG_NOT_FOUND
          La extensión no da soporte a ninguno de los idiomas instalados.
6024    CWBUN_INVALID_USER_ATTR_ID
          ID de atributo de usuario no válido.
```

6025 CWBUN_GET_USER_ATTR_FAILED
No es posible recuperar el atributo de usuario.
6026 CWBUN_INVALID_FLAG_VALUE
El valor de parámetro de distintivo establecido no es válido.
6027 CWBUN_CANT_GET_IMAGELIST
No es posible obtener la lista de imágenes de icono.

Los códigos de retorno siguientes corresponden a las API de comprobación de nombres:

6050 CWBUN_NAME_TOO_LONG
El nombre es demasiado largo.
6051 CWBUN_NAME_NULLSTRING
La serie está vacía: no se ha encontrado ningún carácter.
6054 CWBUN_NAME_INVALIDCHAR
Carácter no válido.
6055 CWBUN_NAME_STRINGTOOLONG
La serie es demasiado larga.
6056 CWBUN_NAME_MISSINGENDQUOTE
Falta la comilla final.
6057 CWBUN_NAME_INVALIDQUOTECHAR
Carácter no válido para serie entrecomillada.
6058 CWBUN_NAME_ONLYBLANKS
Se ha encontrado una serie que únicamente contiene blancos.
6059 CWBUN_NAME_STRINGTOOSHORT
La serie es demasiado corta.
6060 CWBUN_NAME_TOOLONGFORIBM
Serie correcta, pero demasiado larga para mandato IBM.
6011 CWBUN_NAME_INVALIDFIRSTCHAR
El primer carácter no es válido.
6020 CWBUN_NAME_CHECK_LAST
Rango reservado.

Los códigos de retorno siguientes corresponden a las API relacionadas con LDAP:

6101 CWBUN_LDAP_NOT_AVAIL
LDAP no está instalado o configurado.
6102 CWBUN_LDAP_BIND_FAILED
El enlace LDAP ha fallado.

Los códigos de retorno siguientes corresponden a las API de comprobación de nombres de iSeries(TM):

1001 CWBUN_NULLSTRING
La serie está vacía.
1004 CWBUN_INVALIDCHAR
Carácter no válido.
1005 CWBUN_STRINGTOOLONG
La serie es demasiado larga.
1006 CWBUN_MISSINGENDQUOTE
Falta comilla final para serie entrecomillada.
1007 CWBUN_INVALIDQUOTECHAR
Carácter no válido para serie entrecomillada.
1008 CWBUN_ONLYBLANKS
La serie solo contiene blancos.
1009 CWBUN_STRINGTOOSHORT
La serie es más corta que el mínimo definido.
1011 CWBUN_TOOLONGFORIBM
Serie correcta, pero demasiado larga para mandatos IBM.
1012 CWBUN_INVALIDFIRSTCHAR
El primer carácter no es válido.
1999 CWBUN_GENERALFAILURE
Error no especificado.

Información de referencia de Visual Basic

Los conectores de Visual Basic tienen un flujo de control exclusivo en System i Navigator. Además, los conectores de Visual Basic deben implementarse en al menos una clase de interfaz de System i Navigator.

Estructura y flujo de control de conectores de System i Navigator para conectores de Visual Basic

Para los conectores de Visual Basic, System i Navigator proporciona un servidor ActiveX incorporado que gestiona las comunicaciones entre System i Navigator y el conector.

Los programadores de Visual Basic que estén desarrollando conectores de System i Navigator pueden crear los recursos proporcionados por Microsoft Visual Basic 5.0 para crear sus clases de conectores y empaquetarlas en una DLL servidora ActiveX.

Los conectores funcionan respondiendo a llamadas de método desde System i Navigator generadas en respuesta a acciones de usuarios. Por ejemplo, cuando un usuario pulsa con el botón derecho del ratón sobre un objeto en la jerarquía de System i Navigator, System i Navigator construye un menú contextual para el objeto y muestra el menú en la pantalla. System i Navigator obtiene los elementos del menú llamando a cada conector que haya registrado su intención de proporcionar elementos de menú contextual para el tipo de objeto seleccionado.

Las funciones implementadas por un conector se agrupan lógicamente en **interfaces**. Una interfaz es un conjunto de métodos relacionados lógicamente en una clase a la que System i Navigator puede llamar para realizar una función determinada. Hay tres interfaces definidas para los conectores Visual Basic:

- ListManager
- ActionsManager
- DropTargetManager

Datos de System i Navigator para conectores de Visual Basic

Cuando System i Navigator llama a una función implementada por un conector, la solicitud suele implicar un objeto u objetos que el usuario ha seleccionado en la ventana principal de System i Navigator. El conector debe poder determinar qué objetos se han seleccionado. El conector recibe esta información como una lista de nombres de objetos totalmente calificados. Para conectores de Visual Basic, se define una clase ObjectName que proporciona información acerca de los objetos seleccionados. Los conectores que añaden carpetas a la jerarquía de objetos deben devolver elementos en la carpeta a System i Navigator en forma de identificadores de elementos. Para los conectores de Visual Basic, se define una clase ItemIdentifier y el conector utiliza dicha clase para devolver la información solicitada.

Servicios de System i Navigator para conectores de Visual Basic

Los conectores de System i Navigator a veces necesitan afectar al comportamiento de la ventana de System i Navigator principal. Por ejemplo, después de la finalización de una operación de usuario, puede que sea necesario renovar la vista de lista de System i Navigator o insertar texto en el área de estado de System i Navigator. Se proporciona una clase de utilidad llamada UIServices, que proporciona los servicios requeridos, en el entorno de Visual Basic. Un conector de Visual Basic puede también utilizar las API de C++ APIs en el archivo de cabecera cwbun.h para conseguir resultados similares. Para obtener instrucciones detalladas sobre esta clase y sus métodos, consulte la ayuda en línea proporcionada con la DLL de soporte de Visual Basic de System i Navigator (cwbunvbi.dll y cwbunvbi.hlp).

Conceptos relacionados

"System i NavigatorClase de interfaz ListManager" en la página 69
La clase de interfaz **ListManager** se utiliza para servir datos en System i Navigator. Por ejemplo, cuando es necesario crear una lista y rellenarla con objetos, System i Navigator llamará a los métodos en la clase ListManager para hacerlo.

"Clase de interfaz ActionsManager de System i Navigator"
La **clase de interfaz ActionsManager** se utiliza para crear menús de contexto e implementar los mandatos de las acciones del menú de contexto. Por ejemplo, cuando un usuario efectúa una doble pulsación en un objeto de lista de Visual Basic en System i Navigator, se llamará al método queryActions en la clase de interfaz ActionsManager para devolver las series de elementos de menú contextual.

"Clase de interfaz de System i Navigator DropTargetManager" en la página 70
La clase de interfaz **DropTargetManager** se utiliza para manejar las operaciones de arrastrar y soltar en System i Navigator.

Interfaces de System i Navigator Visual Basic

Un conector de Visual Basic debe implementar una o más clases de interfaces de System i Navigator, dependiendo del tipo de función que el desarrollador pretenda proporcionar a System i Navigator.

El juego de herramientas del programador contiene un enlace al archivo de ayuda de definición de interfaces Visual Basic.

Hay tres clases de interfaces de System i Navigator:

- Clase de interfaz ActionsManager de System i Navigator
- Clase de interfaz DropTargetManager de System i Navigator
- Clase de interfaz ListManager de System i Navigator

Nota: No es necesario que la aplicación implemente las tres clases de interfaz.

System i NavigatorClase de interfaz ListManager:

La clase de interfaz **ListManager** se utiliza para servir datos en System i Navigator. Por ejemplo, cuando es necesario crear una lista y rellenarla con objetos, System i Navigator llamará a los métodos en la clase ListManager para hacerlo.

El conector Visual Basic de ejemplo ofrece un ejemplo de esta clase en el archivo listman.cls. Debe tener una clase ListManager si el conector necesita rellenar listas de componentes System i Navigator.

Para obtener instrucciones detalladas sobre esta clase y sus métodos, consulte la ayuda en línea proporcionada con la DLL de soporte de conectores de System i Navigator Visual Basic (cwbunvbi.dll y cwbunvbi.hlp).

Conceptos relacionados

"Estructura y flujo de control de conectores de System i Navigator para conectores de Visual Basic" en la página 68
Para los conectores de Visual Basic, System i Navigator proporciona un servidor ActiveX incorporado que gestiona las comunicaciones entre System i Navigator y el conector.

Clase de interfaz ActionsManager de System i Navigator:

La **clase de interfaz ActionsManager** se utiliza para crear menús de contexto e implementar los mandatos de las acciones del menú de contexto. Por ejemplo, cuando un usuario efectúa una doble pulsación en un objeto de lista de Visual Basic en System i Navigator, se llamará al método queryActions en la clase de interfaz ActionsManager para devolver las series de elementos de menú contextual.

El conector Visual Basic de ejemplo ofrece un ejemplo de esta clase en el archivo **actnman.cls**. Debe definir una clase de interfaz ActionsManager para cada uno de los tipos de objeto exclusivos a los que dé soporte el conector. Puede especificar la misma clase de interfaz ActionsManager para tipos de objeto distintos, pero la lógica del código debe admitir que se le llame con varios tipos de objetos.

Para obtener descripciones detalladas de esta clase y sus métodos, consulte la ayuda en línea proporcionada con la DLL de soporte del conector de System i Navigator Visual Basic (archivos cwbunvbi.dll y cwbunvbi.hlp).

Conceptos relacionados

"Estructura y flujo de control de conectores de System i Navigator para conectores de Visual Basic" en la página 68
Para los conectores de Visual Basic, System i Navigator proporciona un servidor ActiveX incorporado que gestiona las comunicaciones entre System i Navigator y el conector.

Clase de interfaz de System i Navigator DropTargetManager:

La clase de interfaz **DropTargetManager** se utiliza para manejar las operaciones de arrastrar y soltar en System i Navigator.

Si un usuario selecciona un objeto de lista Visual Basic y lleva a cabo operaciones de arrastrar y soltar utilizando el ratón, se efectuarán llamadas a los métodos de esta clase para efectuar las operaciones de arrastrar y soltar.

Para obtener instrucciones detalladas sobre esta clase y sus métodos, consulte la ayuda en línea proporcionada con la DLL de soporte de conectores de System i Navigator Visual Basic (cwbunvbi.dll y cwbunvbi.hlp).

Conceptos relacionados

"Estructura y flujo de control de conectores de System i Navigator para conectores de Visual Basic" en la página 68
Para los conectores de Visual Basic, System i Navigator proporciona un servidor ActiveX incorporado que gestiona las comunicaciones entre System i Navigator y el conector.

Información de consulta Java

Los conectores de Java tienen un flujo de control exclusivo en System i Navigator.

Estructura y flujo de control de System i Navigator para conectores de Java

Para los conectores de Java, System i Navigator proporciona un servidor ActiveX incorporado que gestiona las comunicaciones entre System i Navigator y las clases Java de conectores.

El componente servidor utiliza la API JNI (Java Native Interface) para crear los objetos del conector y llamar a sus métodos. Por lo tanto, los programadores de Java que estén desarrollando conectores de System i Navigator deben familiarizarse con los detalles de la implementación del servidor ActiveX.

Cuando un usuario está interactuando con los conectores System i Navigator de Java, las llamadas serán generadas a las clases de interfaz de Java distintas para la implementación de la solicitud específica.

Los conectores funcionan respondiendo a llamadas de método desde System i Navigator generadas en respuesta a acciones de usuarios. Por ejemplo, cuando un usuario pulsa con el botón derecho del ratón sobre un objeto en la jerarquía de System i Navigator, System i Navigator construye un menú contextual para el objeto y muestra el menú en la pantalla. System i Navigator obtiene los elementos del menú llamando a cada conector que haya registrado su intención de proporcionar elementos de menú contextual para el tipo de objeto seleccionado.

Las funciones que son implementadas por un conector lógicamente se agrupan en interfaces. Una interfaz es un conjunto de métodos relacionados lógicamente en una clase a la que System i Navigator puede llamar para realizar una función determinada. Para los conectores Java están definidas las tres **interfaces Java** siguientes:

- ActionsManager
- DropTargetManager

- ListManager

Arquitectura de productos de conectores de System i Navigator

La arquitectura interna del producto System i Navigator refleja que debe servir como punto de integración de una interfaz de operaciones extensibles y amplia para la plataforma de System i. Cada uno de los componentes funcionales de la interfaz está empaquetado como un servidor ActiveX. System i Navigator aprende acerca de la existencia de un componente de servidor determinado por medio de entradas en el registro de Windows. Varios servidores pueden registrar sus solicitudes para agregar elementos de enmú y diálogos a un tipo de objeto dado en la jerarquía de System i Navigator.

Nota: Para que los conectores Java de terceros estén disponibles para usuarios de System i Navigator, los usuarios de System i Access para Windows deben contar con la Versión 4 Release 4 Nivel de modificación 0 de System i Access para Windows instalada en sus PC.

Datos de System i Navigator para conectores Java

Cuando System i Navigator llama a una función implementada por un conector, la solicitud suele implicar un objeto u objetos que el usuario ha seleccionado en la ventana principal de System i Navigator. El conector debe poder determinar qué objetos se han seleccionado. El conector recibe esta información como una lista de nombres de objetos totalmente calificados. Para conectores Java, se define una clase ObjectName. Proporciona información acerca de los objetos seleccionados. Los conectores que añaden carpetas a la jerarquía de objetos deben devolver elementos en la carpeta a System i Navigator en forma de identificadores de elementos. Para conectores Java, se define una clase ItemIdentifier. El conector la emplea para devolver la información solicitada.

Los conectores de System i Navigator a veces necesitan afectar al comportamiento de la ventana de System i Navigator principal. Por ejemplo, después de la finalización de una operación de usuario, puede que sea necesario renovar la vista de lista de System i Navigator o insertar texto en el área de estado de System i Navigator. Las clases de utilidades que proporcionan los servicios requeridos se incluyen en el paquete com.ibm.as400.opnav.

Personalización los archivos de registro de conectores

Los conectores de ejemplo incluyen dos archivos de registros: una copia legible por Windows para ser utilizada durante el desarrollo y una copia para su distribución en el sistema. Debe modificar estos archivos de registro para desarrollar su conector. Este tema proporciona una visión general de los archivos de registro y descripciones detalladas de las secciones requeridas de cada archivo de registro.

System i Navigator utiliza los archivos de registro para aprender acerca de la existencia del conector, sus requisitos y funciones. Para proporcionar dicha información, cada conector debe especificar como mínimo la siguiente información:

- Una clave de registro principal que proporcione información global acerca del conector.

  Este apartado incluye el identificador programático (ProgID) que especifica el nombre de proveedor y componente del componente y nombra la carpeta en la que reside el conector en el sistema. El ProgID debe seguir el formato <*proveedor*>.<*componente*>; como, por ejemplo, IBM.Sample.
- Claves de registro que identifique los tipos de objetos en la jerarquía de System i Navigator para la cual un conector pretende proporcionar una función adicional.
- Una clave de registro independiente para la raíz de cada subárbol de objetos que añada el conector a la jerarquía de objetos.

  Esta clave contiene información acerca de la carpeta raíz del subárbol.

Personalizar los valores de registro de C++

El conector de ejemplo de incluye dos archivos de registro: SAMDBG.REG, un archivo de registro para utilizarse durante el desarrollo, y SAMPRLS.REG, un archivo de registro para su distribución en el

sistema. Ambos archivos pueden ser leídos por el sistema operativo Windows. Puede personalizar los archivos del registro de ejemplo para sus propios conectores.

Un archivo de registro de conector consiste en varias secciones. Cuando se desarrollan sus propios conectores, necesitará personalizar cada sección tal como se describe en esta información.

Clave de registro primaria:

La clave primaria del registro define un conjunto de campos que especifican información global para el conector. Esta información es necesaria.

```
;-------------------------------------------------------------------
; Definir la clave primaria del registro para el conector
; NOTA: los nombres de DLL ServerEntryPoint y NLS no pueden contener vías de directorio calificadas

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

Cambio de clave de registro primaria	Descripción de campo
Type	Si el conector añade nuevas carpetas a la jerarquía de System i Navigator, el valor de este campo debe ser PLUGIN. De lo contrario, debe ser EXT.
NLS	Identifica el nombre de la DLL de recursos que contiene los recursos dependientes del entorno nacional para el conector. En la versión de desarrollo del archivo de registro, este puede ser un nombre de vía de acceso totalmente calificado.
NameID	Una doble palabra que contiene el identificador de recursos de la serie de texto en la DLL de recursos que será utilizada para identificar el conector en la interfaz de usuario de System i Navigator.
DescriptionID	Palabra doble que contiene el identificador de recurso de la serie de texto en la DLL de recursos. Esta DLL de recurso se utiliza para describir la función del conector en la interfaz de usuario de System i Navigator.
MinimumIMPIRelease	Una serie de 6 caracteres que identifica el release mínimo de i5/OS que debe ejecutarse que requiere el conector. La serie debe tener el formato vvrrmm, donde vv es la versión de i5/OS, rr es el release y mm es el nivel de modificación. Por ejemplo, si el conector requiere Versión 3 Release 2 Nivel de modificación 0, el valor de este campo debe ser ″030200″. Si el conector no soporta ningún release de i5/OS que se ejecute en hardware IMPI (releases anteriores a la Versión 3 Release 6), el valor en este campo debe ser ″NONE.″ Si el conector puede dar soporte a cualquier release de OS/400 que se ejecute en el hardware IMPI, el valor de este campo debe ser ″ANY″.

Cambio de clave de registro primaria	Descripción de campo
MinimumRISCRelease	Una serie de 6 caracteres que identifica el release mínimo de i5/OS que se ejecuta en hardware RISC que requiere el conector. La serie debe tener el formato vvrrmm, donde vv es la versión de i5/OS, rr es el release y mm es el nivel de modificación. Por ejemplo, si el conector requiere la Versión 3 Release 7 Nivel de modificación 1, el valor de este campo debería ser ″030701″. Si el conector no soporta ningún release de i5/OS que se ejecute en hardware RISC (Versión 3 Release 6 y superiores), el valor de este campo debe ser ″NONE.″ Si el conector puede dar soporte a cualquier release de OS/400 que se ejecute en el hardware RISC, el valor de este campo debe ser ″ANY″.
ProductID	Una serie de 7 caracteres que especifica el ID de producto de un programa bajo licencia de prerrequisito System i requerido por el conector. Si el conector no requiere que dicho programa con licencia esté instalado en el sistema, el valor de este campo debe ser NONE. Pueden especificarse varios productos separados por comas si existen varios ID para el mismo producto.
ServerEntryPoint	Nombre de la DLL de código que implementa el punto de entrada de servidor. Este punto de entrada es llamado por System i Navigator cuando necesita determinar si el conector está soportado en un sistema determinado. Si el conector no implementa el punto de entrada, el valor de este campo debe ser ″NONE″. En la versión de desarrollo del archivo de registro, este puede ser un nombre de vía de acceso totalmente calificado.
JavaPath	Serie de vía de acceso de clase que identifica la ubicación de las clases Java del conector. Durante el desarrollo del conector, este campo puede contener las vías de acceso de directorio de los directorios donde residen los archivos de clase. En la versión de producción del archivo de registro, este debe identificar los nombres de archivo JAR relativos a la vía de acceso de instalación de System i Access para Windows, cada una precedida por la variable de sustitución System i Access para Windows que representa la vía de acceso de instalación.
JavaMRI	Nombres base de los archivos JAR que contienen los recursos dependientes del entorno nacional para el conector. System i Navigator buscará cada archivo JAR después de añadir un sufijo al nombre con los identificadores de lenguaje Java y país correspondientes. Si no existen ningún archivo JAR de MRI para un entorno nacional determinado, System i Navigator esperará que el MRI del entorno nacional base (generalmente inglés) resida en el código de los archivos JAR.

Implementación del servidor de datos:

Este apartado registra la implementación IA4HierarchyFolder para cada nueva carpeta añadida a la jerarquía de System i Navigator.

```
;------------------------------------------------------------------------
; Esta sección registrará una implementación IA4HierarchyFolder
para cada nueva
; carpeta añadida a la jerarquía de System i Navigator.

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

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

Si el conector añade más de una nueva carpeta a la jerarquía, deberá duplicar esta sección del archivo de registro para cada carpeta adicional. Compruebe que genera un GUID (identificador exclusivo global) para cada carpeta. Si el conector no añade ninguna carpeta podrá eliminar esta sección.

Si duplica SAMPDATA.CPP tal como se muestra a continuación, todas las nuevas carpetas contendrán inicialmente objetos de biblioteca:

1. Cambie el nombre de la DLL de modo que coincida con el nombre de la DLL generada por la nueva área de trabajo del proyecto.
2. Genere y copie un nuevo GUID. Consulte el apartado “Cambios globales de los archivos de registro de conectores de C++” en la página 77.
3. Sustituya ambas ocurrencias del identificador de clase (CLSID) en esta sección del registro con la nueva serie GUID que acaba de generar.
4. Busque la serie IMPLEMENT_OLECREATE en su versión del archivo SAMPDATA.CPP.
5. Pegue el nuevo GUID sobre el CLSID existente en la línea de comentarios y, a continuación, cambie el CLSID en la llamada a la macro IMPLEMENT_OLECREATE para que coincida con los valores hexadecimales en su nuevo GUID. Sustituya la palabra Sample con el nombre de su nueva carpeta.
6. Cree dos nuevos archivos fuente para cada GUID nuevo utilizando como base una copia redenominada de SAMPDATA.H y SAMPDATA.CPP.

   Nota: El archivo de cabecera (.H) contiene la declaración de clase de la nueva clase de implementación. El archivo de implementación (.CPP) contiene el código que obtiene los datos de la nueva carpeta.
7. Sustituya las ocurrencias del nombre de clase CSampleData en los dos archivos fuentes con un nombre de clase que sea significativo en el contexto del conector.
8. Para agregar los nuevos archivos de implementación al espacio de trabajo del proyecto, abra el menú **Insertar** y seleccione **Archivos en proyecto**.

Clase de implementación de conector de shell:

Esta sección registra la clase de implementación de conector de shell. Cada uno de los conectores c++ debe utilizar esta sección.

```
;------------------------------------------------------------------
; Esta sección registrará la clase de implementación de conector de shell.
; Un conector de shell añade elementos de menú de contexto y páginas de
; propiedades a objetos nuevos o existentes de la jerarquía.

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

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

;------------------------------------------------------------------
; Aprobar conector de shell (necesario en Windows NT(R))
```

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

Para personalizar esta sección para sus propios conectores, siga estos pasos:

1. Cambie el nombre de la DLL de modo que coincida con el nombre de la DLL generada por la nueva área de trabajo del proyecto.
2. Genere y copie un nuevo GUID (identificador exclusivo global). Consulte el apartado "Cambios globales de los archivos de registro de conectores de C++" en la página 77.
3. Sustituya todas las ocurrencias del identificador de clase (CLSID) en las entradas con el nuevo GUID que acaba de generar.
4. Busque la serie IMPLEMENT_OLECREATE en su versión del archivo EXTINTFC.CPP.
5. Copie el nuevo GUID en la línea de comentarios CLSID existente y, continuación, cambie el CLSID en la llamada de macro IMPLEMENT_OLECREATE para que coincida con los valores hexadecimales en el nuevo GUID.

Implementación de conectores de shell para objetos:

La sección final del registro especifica qué objetos en la jerarquía de System i Navigator se ven afectados por la implementación del conector.

```
;------------------------------------------------------------------
; Registrar un manejador de menú de contexto para la carpeta nueva y sus objetos

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

;------------------------------------------------------------------
; Registrar un manejador de hojas de propiedades para la carpeta nueva y sus objetos

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

;------------------------------------------------------------------
; Registrar el manejador de hojas de propiedades de renovación automática para la
; (esto permitirá que la carpeta se aproveche de la función de System i Navigator
; de renovación automática).

[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY plug-inS\IBM.Sample\shellex\Sample\*
\PropertySheetHandlers\{5E44E520-2F69-11d1-9318-0004AC946C18}]

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

; Registrar manejadores de menú de contexto de arrastrar y soltar

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

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

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

; Registrar un manejador de acciones de soltar para aceptar las acciones de soltar objetos

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

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

; Registrar que este conector da soporte a conexiones SSL (Capa de Sockets Segura)
; Nota:  "Support Level"=dword:00000001   indica que el conector ofrece soporte SSL
```

```
; Nota:  "Support Level"=dword:00000000   indica que el conector no ofrece soporte SSL

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

Para personalizar esta sección para sus propios conectores, siga estos pasos:

1. Sustituya el identificador de clase (CLSID) en este apartado con los identificadores exclusivos globalmente (GUID).
2. Si el conector no añade páginas de propiedades adicionales a una hoja de propiedades de una carpeta u objeto, elimine la entrada de registro del manejador de hoja de propiedades.
3. Si el conector no es un manejador de acciones de soltar de objetos, elimine el manejador de menú contextual y las entradas de registro de manejador de soltar.
4. Edite las subclaves \Sample*\. Para obtener más información, consulte el apartado “Conectores de shell”.
5. Edite o elimine de su versión de EXTINTFC.CPP el código que realiza las comprobaciones de los tipos de objeto definidos por el ejemplo.

   Verá las carpetas, los elementos de menú de contexto, las páginas de propiedades y las acciones de soltar del ejemplo, según cuántas funciones del ejemplo haya decidido conservar.

   Nota: El archivo de código basado en el archivo de ejemplo EXTINTFC.CPP contiene el código al que se llamará para los menús de contexto, las páginas de propiedades y las acciones de soltar. El código de ejemplo contiene comprobaciones para los tipos de objeto que define el ejemplo. Debe editar este archivo y eliminar estas comprobaciones o cambiarlas de modo que realice comprobaciones para los tipos de objeto para los que desea proporcionar nuevas funciones.

Conectores de shell:

Estas claves del registro correlacionan un nodo o conjunto de nodos concreto de la jerarquía con el tipo de función que proporciona el conector y con el CLSID de la clase de implementación que implementa la función.

Recuerde que cualquier número de conectores de shell puede registrar su intención de añadir funciones a un tipo de objeto determinado en la jerarquía de System i Navigator. El conector nunca debe suponer que es el único componente servidor que proporciona funciones para un tipo de objeto determinado. Esto es válido no solo para los tipos de objeto existentes, sino también para los objetos nuevos que un conector pueda definir. Si el conector se usa de forma generalizada, nada puede evitar que otro proveedor amplíe los tipos de objeto definidos por el conector.

Identificadores de tipo de objeto

En este nivel de la jerarquía de subclaves siempre se espera un par de identificadores de tipo de objeto, las subclaves \Sample*\.

El primer identificador en el par especifica la carpeta raíz de un componente de System i Navigator. En el caso de los conectores que añaden nuevas carpetas, este identificador siempre debe coincidir con el nombre de clave del registro de una carpeta raíz especificada en la sección anterior. Para aquellos conectores que añadan comportamientos a tipos de objetos existentes, esta subclave debe generalmente ser el tipo de objeto del primer nivel bajo un objeto de contenedor de System i. Estas series de tipo están definidas en el registro bajo HKEY_CLASSES_ROOT\IBM.AS400.Network\TYPES.

El segundo identificador del par identifica el tipo de objeto específico que el conector desea modificar. Si se especifica *, se efectuará una llamada al conector para el tipo de carpeta que se ha identificado en la subclave padre, más la totalidad de carpetas y objetos que aparecen en la jerarquía bajo esa carpeta. De lo contrario, debe especificarse un identificador de tipo específico y solo se efectuará una llamada al conector para ese tipo de objeto.

Comprobaciones de los tipos de objeto

Cuando realiza comprobaciones de tipos objetos, necesitará utilizar los identificadores de tipo de 3 caracteres definidos bajo la clave HKEY_CLASSES_ROOT\IBM.AS400.Network\TYPES en el registro. Al efectuar comprobaciones para los tipos de objeto nuevos definidos por el conector, emplee una clave del registro. Utilice la clave de registro que identifica la carpeta que ha especificado en el punto de unión o el tipo que devuelva a System i Navigator cuando sirva datos de una carpeta definida por el conector.

Cambios globales de los archivos de registro de conectores de C++:

Cuando desarrolle sus propios conectores, necesitará realizar algunos cambios globales en los archivos de registro de conectores de ejemplo. Debe especificar un identificador programático exclusivo (ProgID) e identificadores exclusivos globales (GUID) para utilizarlos en todo el archivo del registro de conector.

Definir un ProgID exclusivo para el conector

El ProgID debe coincidir con la serie de texto <*proveedor*>.<*componente*>, donde *proveedor* identifica el nombre del proveedor que desarrolló el conector y *componente* describe la función que proporciona. En el conector de ejemplo, la serie IBM.Sample identifica IBM como el proveedor y Sample como la descripción de la función proporcionada por el conector. Esto se utiliza en todo el archivo de registro. Nombre el directorio donde reside el conector en el sistema y en la estación de trabajo. Sustituya todas las instancias de IBM.Sample en el archivo de registro con su ProgID.

Genera nuevos GUID y sustituir los valores CLSID en el archivo de registro

Para que el conector de System i Navigator C++ funcione correctamente, deberá sustituir los identificadores de clase específicos (CLSID) en el nuevo archivo de registro con los GUID que genere.

El modelo de objeto de componente de Microsoft utiliza enteros hexadecimales de 16 bits para identificar de forma exclusiva clases e interfaces de implementación de ActiveX. Estos enteros se conocen como GUID. Los GUID que identifican clases de implementación se llaman CLSID. System i Navigator utiliza el soporte de tiempo de ejecución de ActiveX de Windows para cargar los componentes de un conector y para obtener un puntero a una instancia de la implementación del conector de una interfaz en particular. Un CLSID en el registro identifica de forma exclusiva una clase de implementación específica que reside en una DLL servidora ActiveX específica. La primera parte de esta correlación, la que hace corresponder el CLSID con el nombre y la ubicación de la DLL servidora, se lleva a cabo mediante una entrada del registro. Por lo tanto, un conector de System i Navigator debe registrar un CLSID para cada clase implementación que proporcione.

Para generar los GUID, siga estos pasos:

1. Desde la barra de tareas de Windows, seleccione **Inicio** y, a continuación, **Ejecutar**.
2. Escriba `GUIDGEN` y pulse **Aceptar**.
3. Compruebe que está seleccionado Formato del registro.
4. Para generar un nuevo valor GUID, seleccione **Nuevo GUID**.
5. Para copiar el nuevo valor GUID en el portapapeles, seleccione **Copiar**.

Personalizar los valores de registro de los conectores de Visual Basic

El conector de ejemplo contiene dos archivos: VBSMPDBG.REG, un archivo de registro para utilizarse durante el desarrollo, y VBSMPRLS.REG, un archivo de registro para ser utilizado en el sistema. Ambos archivos pueden ser leídos por el sistema operativo Windows. Puede personalizar los archivos del registro de ejemplo para sus propios conectores.

Un archivo de registro de conector consiste en varias secciones. Cuando se desarrollan sus propios conectores, necesitará personalizar cada sección tal como se describe en esta información.

Clave de registro primaria:

La clave primaria del registro define un conjunto de campos que especifican información global para el conector. Esta información es necesaria.

Nota: El nombre de la subclave debe coincidir con el ID de programa (ProgID) del conector.

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

"Type"="Plugin"

"NLS"="vbsmpmri.dll"

"NameID"=dword:00000080

"DescriptionID"=dword:00000081

"MinimumIMPIRelease"="NONE"

"MinimumRISCRelease"="040200"

"ProductID"="NONE"

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

Cambio de clave de registro primaria	Descripción de campo
Type	Si el conector añade nuevas carpetas a la jerarquía de System i Navigator, el valor de este campo debe ser PLUGIN. De lo contrario, debe ser EXT.
NLS	Identifica el nombre de la DLL de recursos que contiene los recursos dependientes del entorno nacional para el conector. En la versión de desarrollo del archivo de registro, este puede ser un nombre de vía de acceso totalmente calificado.
NameID	Una doble palabra que contiene el identificador de recursos de la serie de texto en la DLL de recursos que será utilizada para identificar el conector en la interfaz de usuario de System i Navigator.
DescriptionID	Palabra doble que contiene el identificador de recurso de la serie de texto en la DLL de recursos. Esta DLL de recurso se utiliza para describir la función del conector en la interfaz de usuario de System i Navigator.
MinimumIMPIRelease	Una serie de 6 caracteres que identifica el release mínimo de i5/OS que debe ejecutarse que requiere el conector. La serie debe tener el formato vvrrmm, donde vv es la versión de i5/OS, rr es el release y mm es el nivel de modificación. Por ejemplo, si el conector requiere Versión 3 Release 2 Nivel de modificación 0, el valor de este campo debe ser ″030200″. Si el conector no soporta ningún release de i5/OS que se ejecute en hardware IMPI (releases anteriores a la Versión 3 Release 6), el valor en este campo debe ser ″NONE.″ Si el conector puede dar soporte a cualquier release de OS/400 que se ejecute en el hardware IMPI, el valor de este campo debe ser ″ANY″.

Cambio de clave de registro primaria	Descripción de campo
MinimumRISCRelease	Una serie de 6 caracteres que identifica el release mínimo de i5/OS que se ejecuta en hardware RISC que requiere el conector. La serie debe tener el formato vvrrmm, donde vv es la versión de i5/OS, rr es el release y mm es el nivel de modificación. Por ejemplo, si el conector requiere la Versión 3 Release 7 Nivel de modificación 1, el valor de este campo debería ser ″030701″. Si el conector no soporta ningún release de i5/OS que se ejecute en hardware RISC (Versión 3 Release 6 y superiores), el valor de este campo debe ser ″NONE.″ Si el conector puede dar soporte a cualquier release de OS/400 que se ejecute en el hardware RISC, el valor de este campo debe ser ″ANY″.
ProductID	Una serie de 7 caracteres que especifica el ID de producto de un programa bajo licencia de prerrequisito System i requerido por el conector. Si el conector no requiere que dicho programa con licencia esté instalado en el sistema, el valor de este campo debe ser NONE. Pueden especificarse varios productos separados por comas si existen varios ID para el mismo producto.
ServerEntryPoint	Nombre de la DLL de código que implementa el punto de entrada de servidor. Este punto de entrada es llamado por System i Navigator cuando necesita determinar si el conector está soportado en un sistema determinado. Si el conector no implementa el punto de entrada, el valor de este campo debe ser ″NONE″. En la versión de desarrollo del archivo de registro, este puede ser un nombre de vía de acceso totalmente calificado.
JavaPath	Serie de vía de acceso de clase que identifica la ubicación de las clases Java del conector. Durante el desarrollo del conector, este campo puede contener las vías de acceso de directorio de los directorios donde residen los archivos de clase. En la versión de producción del archivo de registro, este debe identificar los nombres de archivo JAR relativos a la vía de acceso de instalación de System i Access para Windows, cada una precedida por la variable de sustitución System i Access para Windows que representa la vía de acceso de instalación.
JavaMRI	Nombres base de los archivos JAR que contienen los recursos dependientes del entorno nacional para el conector. System i Navigator buscará cada archivo JAR después de añadir un sufijo al nombre con los identificadores de lenguaje Java y país correspondientes. Si no existen ningún archivo JAR de MRI para un entorno nacional determinado, System i Navigator esperará que el MRI del entorno nacional base (generalmente inglés) resida en el código de los archivos JAR.

Para personalizar esta sección la clave de registro principal para sus propios conectores, siga estos pasos:

1. Cambie el nombre ″vbsample.dll″ de la clave ServerEntryPoint de modo que coincida con el nombre de la DLL servidora ActiveX del conector.

2. Cambie el nombre ″vbsmpmri.dll″ de la clave NLS de modo que coincida con el nombre de la DLL de recursos de MRI C++ para el conector. Cada uno de los conectores Visual Basic debe tener un nombre de DLL de MRI C++.

Nota: No incluya la vía de acceso en ninguno de estos cambios.

Conceptos relacionados

“Cambios globales de los archivos de registro de conectores de Visual Basic” en la página 83
Cuando desarrolle sus propios conectores, necesitará definir un identificador programa exclusivo (ProgID) para el conector. Deberá especificar un ProgID exclusivo para que sea utilizado en todo el archivo.

Clase de implementación de conector de Visual Basic:

Esta sección registra una implementación de clase ListManager de Visual Basic para cada nueva carpeta añadida a la jerarquía de System i Navigator.

Si el conector no añade ninguna nueva carpeta a la jerarquía de System i Navigator, suprima esta sección.

La clase ListManager de Visual Basic es la interfaz principal para servir datos a la carpeta del conector.

El ejemplo coloca la carpeta de ejemplo de Visual Basic en el nivel raíz de un sistema llamado AS4 en la jerarquía de System i Navigator. Si desea que su carpeta aparezca en otro punto de la jerarquía, deberá cambiar el valor de la clave Parent.

```
[HKEY_CLASSES_ROOT\IBM.AS400.Network\

3RD PARTY EXTENSIONS\IBM.VBSample\

folders\SampleVBFolder]

"Parent"="AS4"

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

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

"VBClass"="vbsample.SampleListManager"

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

"NameID"=dword:00000082

"DescriptionID"=dword:00000083

"DefaultIconIndex"=dword:00000001

"OpenIconIndex"=dword:00000001
```

Para personalizar esta sección para sus propios conectores, siga estos pasos:

1. Cambie todas las apariciones del nombre ″SampleVBFolder″ del archivo del registro por un nombre exclusivo que identificará el objeto de carpeta. El nombre especificado en el archivo del registro debe coincidir con el nombre de objeto especificado en las clases ListManager y ActionsManager de Visual Basic. Para el conector de ejemplo estos archivos fuente de Visual Basic son **listman.cls** y **actnman.cls**.
2. Cambie el nombre ″vbsample.SampleListManager″ de la clave VBClass de modo que coincida con el nombre de identificador de programa de la clase ListManager. Por ejemplo, si la DLL servidora ActiveX se denomina foo.dll y la clase de implementación ListManager es MiListManager, el identificador de programa es ″foo.MiListManager″. Este nombre es sensible a las mayúsculas y minúsculas.

3. Cambie el valor de la clave ″VBInterface″ por el ID de interfaz de la clase de implementación ListManager.

Valores del campo Parent:

Un ID de tres caracteres se utiliza para identificar el padre de la carpeta que será añadida. System i Navigator proporciona un conjunto de ID para el valor de la clave padre.

Puede especificar uno de los siguientes ID:

AS4	Carpeta del sistema
BKF	Carpeta Copia de seguridad
BOF	Carpeta Operaciones básicas
CFG	Carpeta Configuración y servicio
DBF	Carpeta Base de datos
FSF	Carpeta Sistemas de archivos
JMF	Carpeta Gestión de trabajos
MCN	Carpeta Management Central
MCS	Carpeta Configuración y servicio de Management Central
MDF	Carpeta Definiciones de Management Central
MST	Tareas planificadas de Management Central
MSM	Supervisores de Management Central
MTA	Actividad de tareas de Management Central
MXS	Soporte completo de Management Central
NSR	Carpeta Servidores de red
NWF	Carpeta Red
SCF	Carpeta Seguridad
UGF	Carpeta Usuarios y grupos

Conceptos relacionados

“Ejemplo: nueva clave del registro de carpeta”
Debe definir una clave de registro independiente para la raíz de cada subárbol de objetos que un conector añada a la jerarquía de objetos. Esta clave contiene información específica de la carpeta raíz del subárbol. Este tema describe cada nuevo campo de registro de nueva carpeta y sus posible valores.

Ejemplo: nueva clave del registro de carpeta:

Debe definir una clave de registro independiente para la raíz de cada subárbol de objetos que un conector añada a la jerarquía de objetos. Esta clave contiene información específica de la carpeta raíz del subárbol. Este tema describe cada nuevo campo de registro de nueva carpeta y sus posible valores.

Asigne a la clave del registro un nombre de carpeta que tenga sentido y que conste como mínimo de cuatro caracteres.

```
;-------------------------------------------------------------------
; Registrar una carpeta nueva

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

Parent	ID de 3 caracteres que identifica el elemento padre de la carpeta que se añadirá.

Attributes	Campo binario de 4 bytes que contiene los atributos de la carpeta, con los bytes de indicador en el orden inverso. Consulte los distintivos de atributo de carpeta definidos para el método IShellFolder::GetAttributesOf en el archivo include de Microsoft SHLOBJ.H.
CLSID	El CLSID de la implementación IA4HierarchyFolder que debe ser llamado por System i Navigator para obtener el contenido de la carpeta. Para los **conectores Java**, el CLSID siempre debe ser: 1827A856-9C20-11d1-96C3-00062912C9B2. Para los **conectores Visual Basic**, el CLSID siempre debe ser: 040606B1-1C19-11d2-AA12-08005AD17735}.
JavaClass	El nombre de clase Java totalmente calificado de la implementación **ListManager** que debe ser llamado por System i Navigator para obtener el contenido de la carpeta. Este campo debe omitirse si el conector no es un conector Java.
VBClass	El identificador de programa (ProgID) de la clase de implementación **ListManager** que debe ser llamado por System i Navigator para obtener el contenido de la carpeta.
VBInterface	GUID de la interfaz de la clase de implementación **ListManager**.
NameID	Una doble palabra que contiene el ID de recurso de la serie que debe aparecer como el nombre de la carpeta en la jerarquía de System i Navigator.
DescriptionID	Una palabra doble que contiene el ID de recurso de la serie que debe aparecer como la descripción de la carpeta en la jerarquía de System i Navigator.
DefaultIconIndex	Una palabra doble que contiene el índice en la DLL de recursos NLS del conector del icono que debe aparecer en la carpeta en la jerarquía de System i Navigator. Es un índice basado en cero en la DLL de recursos, no el ID de recurso del icono. Para que la indexación funcione correctamente, los ID de recurso de icono deben asignarse de forma secuencial.
OpenIconIndex	Una palabra doble que contiene el índice de la DLL de recursos NLS del conector del icono que debe aparecer en la jerarquía de System i Navigator siempre que sea seleccionado por el usuario.
AdminItem	Serie que contiene el ID de la función de Administración de Aplicaciones que controla el acceso a la carpeta. Si se omite este campo, ninguna función de Administración de Aplicaciones controla el acceso a la carpeta. Si se especifica, debe ser el ID de una función de grupo o administrable. No puede ser el ID de una función de producto.

Conceptos relacionados

"Valores del campo Parent" en la página 81
Un ID de tres caracteres se utiliza para identificar el padre de la carpeta que será añadida. System i Navigator proporciona un conjunto de ID para el valor de la clave padre.

Objetos de implementación de conector de Visual Basic:

La sección final del registro especifica qué objetos en la jerarquía de System i Navigator se ven afectados por la implementación del conector de Visual Basic.

En muchos de los métodos de las clases ActionsManager, ListManager y DropTargetManager, se le pasarán elementos u objetos. Para determinar a qué objeto de carpeta se hace referencia, utilice la serie de tipo de objeto definida en el registro de Windows.

Pueden seguir añadiéndose hojas de propiedades al conector por medio de un elemento de menú de contexto. No puede utilizar una clave del registro para una hoja de propiedades que sea el mecanismo utilizado para un conector C++. Los manejadores de hojas de propiedades que incluyen el manejador de hojas de propiedades de renovación automática (Auto Refresh) no están soportados para los conectores Visual Basic.

```
;-------------------------------------------------------------------
; Registrar un manejador de menú de contexto para la carpeta nueva y sus objetos

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

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

;-------------------------------------------------------------------
; Registrar manejadores de menú de contexto de arrastrar y soltar

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

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

;-------------------------------------------------------------------
; Registrar un manejador de acciones de soltar para aceptar las acciones de soltar objetos

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

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

Para personalizar esta sección para sus propios conectores, siga estos pasos:

1. Asegúrese de que el identificador de clase (CLSID) en las entradas anteriores siempre tiene la siguiente serie: {040606B2-1C19-11d2-AA12-08005AD17735}.
2. La clave VBClass contiene el identificador de programa (ProgID) de la clase de implementación de Visual Basic.
3. La clave VBInterface contiene el ID de interfaz de clase de implementación de Visual Basic.
4. Si el conector no es un manejador de acciones de soltar de objetos, elimine el manejador de menú contextual y las entradas de registro de manejador de soltar.
5. Cambie el nombre de las subclaves \SampleVBFolder*\ y utilice una serie exclusiva para identificar el objeto de carpeta. El nombre es el tipo de objeto que se utiliza en el código fuente de Visual Basic para identificar cuándo se toman acciones en esta carpeta en System i Navigator.
6. En el archivo que ha creado a partir de la interfaz ActionsManager, edite el código que efectúa las comprobaciones de los tipos de objeto que define el ejemplo de modo que refleje el nombre del nuevo objeto de carpeta. La interfaz ActionsManager del ejemplo se encuentra en actnman.cls.

Cambios globales de los archivos de registro de conectores de Visual Basic:

Cuando desarrolle sus propios conectores, necesitará definir un identificador programa exclusivo (ProgID) para el conector. Deberá especificar un ProgID exclusivo para que sea utilizado en todo el archivo.

El ProgID debe coincidir con la serie de texto <*proveedor*>.<*componente*>, donde *proveedor* identifica el nombre del proveedor que desarrolló el conector y *componente* describe la función que proporciona. En el conector de ejemplo, la serie IBM.Sample identifica IBM como el proveedor y Sample como la descripción de la función proporcionada por el conector. Esto se utiliza en todo el archivo de registro. Nombre el directorio donde reside el conector en el sistema y en la estación de trabajo. Sustituya todas las instancias de IBM.Sample en el archivo de registro con su ProgID.

Sustituya todas las instancias de IBM.VBSample con el nuevo [proveedor].ProgID.

Nota: System i Navigator proporciona DLL servidoras ActiveX incorporadas que gestionan los conectores escritos en Java y en Visual Basic. Por consiguiente, todos los conectores Java Visual Basic registran sus propios CLSID respectivos. Los archivos del registro que se facilitan con los ejemplos de programación ya contienen estos CLSID predefinidos.

Conceptos relacionados

"Clave de registro primaria" en la página 78
La clave primaria del registro define un conjunto de campos que especifican información global para el conector. Esta información es necesaria.

Archivo del registro Java de ejemplo

Cada uno de los conectores de ejemplo escritos en Java proporciona su propio archivo del registro.

Las secciones siguientes describen las partes importantes del archivo del registro e ilustran cómo crear las entradas correspondientes para los conectores propios. Los ejemplos se toman del ejemplo adecuado que ilustra la función descrita.

Identificador programático (ProgID)

El conector se identifica de forma exclusiva en System i Navigator por medio de una serie de texto con el formato <*proveedor*>.<*componente*>, donde *proveedor* identifica el proveedor que ha desarrollado el conector y *componente* describe la función proporcionada. En los siguientes ejemplos, la serie `IBM.MsgQueueSample3` identifica IBM como el proveedor y MsgQueueSample3 como la descripción de la función proporcionada por el conector. Esta serie se conoce como el identificador programático (ProgID). Se utiliza en todo el archivo de registro cuando se especifica la función que el conector proporciona. También nombra el directorio donde reside el conector en el sistema y en la estación de trabajo cliente.

Identificadores exclusivos globalmente (GUID)

El modelo de objetos de componentes de Microsoft utiliza enteros hexadecimales de 16 bytes para identificar de forma exclusiva las clases de implementación e interfaces de ActiveX. Estos enteros se denominan *identificadores exclusivos globalmente* (*GUID*). Los GUID que identifican clases de implementación se llaman CLSID (ID de clase).

Para aquellos componentes de System i Navigator escritos en Java, no debe definir nuevos GUID. Todos los conectores Java emplean un conjunto de GUID estándar que especifican el componente servidor ActiveX incorporado que gestiona los conectores Java. Los CLSID estándar que se utilizarán se proporcionan en los ejemplos siguientes.

Definir los atributos primarios del conector

```
;------------------------------------------------------------------
; Definir clave primaria del registro para el ejemplo de cola de mensajes 3.

[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY EXTENSIONS\IBM.MsgQueueSample3]
"Type"="PLUGIN"
"NLS"="MessageQueuesMRI.dll"
"NameID"=dword:00000001
"DescriptionID"=dword:00000002
"MinimumIMPIRelease"="NONE"
"MinimumRISCRelease"="ANY"
"ProductID"="NONE"
"ServerEntryPoint"="NONE"
"JavaPath"="MsgQueueSample3.jar"
"JavaMRI"="MsgQueueSample3MRI.jar"
```

Type Si el conector añade nuevas carpetas a la jerarquía de System i Navigator, el valor de este campo debe ser PLUGIN. De lo contrario, debe ser EXT.

NLS Identifica el nombre de la DLL de recurso que contiene recursos dependientes del entorno nacional del conector. En la versión de desarrollo del archivo de registro, este puede ser un nombre de vía de acceso totalmente calificado.

NameID
Una doble palabra que contiene el identificador de recursos de la serie de texto en la DLL de recursos que será utilizada para identificar el conector en la interfaz de usuario de System i Navigator.

DescriptionID
Palabra doble que contiene el identificador de recurso de la serie de texto en la DLL de recursos. Esta DLL de recurso se utiliza para describir la función del conector en la interfaz de usuario de System i Navigator.

MinimumIMPIRelease
Una serie de 6 caracteres que identifica el release mínimo de i5/OS que se ejecuta en hardware IMPI que requiere el conector. La serie debe tener el formato `vvrrmm`, donde vv es la versión de i5/OS, rr es el release y mm es el nivel de modificación. Por ejemplo, si el conector requiere la Versión 3 Release 2 Nivel de modificación 0, el valor de este campo debe ser 030200.

Si el conector no soporta ningún release de i5/OS que se ejecute en hardware IMPI (releases anteriores a la Versión 3 Release 6), el valor en este campo debe ser ″NONE.″ Si el conector puede dar soporte a cualquier release que se ejecute en hardware IMPI, el valor de este campo debe ser ″ANY″.

MinimumRISCRelease
Una serie de 6 caracteres que identifica el release mínimo de i5/OS ejecutándose en hardware RISC requerido por el conector. La serie debe tener el formato `vvrrmm`, donde vv es la versión de i5/OS, rr es el release y mm es el nivel de modificación. Por ejemplo, si el conector requiere la Versión 3 Release 7 Nivel de modificación 1, el valor de este campo debe ser 030701.

Si el conector no tiene soporte para ningún release de i5/OS que se ejecute en hardware RISC (Versión 3 Release 6 y superior), el valor de este campo debe ser NONE. Si el conector puede dar soporte a cualquier release de OS/400 que se ejecute en el hardware RISC, el valor de este campo debe ser ″ANY″.

ProductID
Una serie de 7 caracteres que especifica el ID de producto de un programa bajo licencia de prerrequisito System i requerido por el conector. Si el conector no requiere que dicho programa con licencia esté instalado en el sistema, el valor de este campo debe ser NONE.

Pueden especificarse varios productos separados por comas si existen varios ID para el mismo producto.

ServerEntryPoint
Nombre de la DLL de código que implementa el punto de entrada de servidor. Este punto de entrada es llamado por System i Navigator cuando necesita determinar si el conector está soportado en un sistema determinado. Si el conector no implementa el punto de entrada, el valor de este campo deberá ser NONE. En la versión de desarrollo del archivo de registro, este puede ser un nombre de vía de acceso calificado al completo.

JavaPath
Serie de vía de acceso de clase que identifica la ubicación de las clases Java del conector. Durante el desarrollo del conector, este campo puede contener las vías de acceso de directorio de los directorios donde residen los archivos de clase. En la versión de producción del archivo del registro, debe identificar los archivos JAR. Los nombres de archivos JAR no deben ser calificados con ningún nombre de directorio, System i Navigator los calificará automáticamente al construir la serie vía de acceso de clase que pasará a la VM de Java.

JavaMRI
Nombres base de los archivos JAR que contienen los recursos dependientes del entorno nacional para el conector. System i Navigator busca todos los archivos JAR después de añadir primero un sufijo al nombre con los identificadores de lenguaje Java y de país adecuados. En la versión de

desarrollo del archivo de registro, este campo puede contener una serie vacía porque los recursos en el entorno nacional de base (generalmente en inglés de EE.UU) deben residir en el JAR de código.

Definir carpetas nuevas

```
;-------------------------------------------------------------------
; Registrar una carpeta nueva

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

Type Cada nueva carpeta que el conector añada a la jerarquía de System i Navigator tiene un tipo lógico exclusivo. En el ejemplo anterior, la serie `Sample3` es el tipo que será utilizado para identificar la carpeta seleccionada actualmente cuando se pase el control al conector durante la ejecución.

Parent ID de 3 caracteres que identifica el elemento padre de la carpeta que se añadirá. Puede especificar uno de los siguientes ID:

ADF	Carpeta de Desarrollo de aplicaciones
AS4	Carpeta del sistema
BKF	Carpeta Copia de seguridad
BOF	Carpeta Operaciones básicas
CFG	Carpeta Configuración y servicio
DBF	Carpeta Base de datos
FSF	Carpeta Sistemas de archivos
MCN	Carpeta Management Central
MCS	Carpeta Configuración y servicio de Management Central
MDF	Carpeta Definiciones de Management Central
MMN	Supervisores de Management Central
MST	Tareas planificadas de Management Central
MTA	Actividad de tareas de Management Central
MXS	Soporte completo de Management Central
NSR	Carpeta Servidores de red
NWF	Carpeta Red
SCF	Carpeta Seguridad
UGF	Carpeta Usuarios y grupos
WMF	Carpeta Gestión de trabajos

Attributes
Campo binario de 4 bytes que contiene los atributos de la carpeta, con los bytes de indicador en el orden inverso. Consulte los distintivos de atributos de carpeta definidos para el método IShellFolder::GetAttributesOf en el archivo include de Microsoft SHLOBJ.H. Para indicar que la carpeta tiene un área de tareas, utilice 0x00000008.

CLSID
El CLSID de la implementación IA4HierarchyFolder que debe ser llamado por System i Navigator para obtener el contenido de la carpeta. Para los conectores de Java este CLSID siempre debe ser `{1827A856-9C20-11d1-96C3-00062912C9B2}`.

JavaClass
El nombre de clase de Java totalmente calificado de la implementación **ListManager** que debe ser llamada por System i Navigator para obtener el contenido de la carpeta.

NameID
Una doble palabra que contiene el ID de recurso de la serie que debe aparecer como el nombre de la carpeta en la jerarquía de System i Navigator.

DescriptionID
Una palabra doble que contiene el ID de recurso de la serie que debe aparecer como la descripción de la carpeta en la jerarquía de System i Navigator.

DefaultIconIndex
Una palabra doble que contiene el índice en la DLL de recursos NLS del conector del icono que debe aparecer en la carpeta en la jerarquía de System i Navigator. Es un índice basado en cero en la DLL de recursos, no el ID de recurso del icono. Para que la indexación funcione correctamente, los ID de recurso de icono deben asignarse de forma secuencial.

OpenIconIndex
Una palabra doble que contiene el índice de la DLL de recursos NLS del conector del icono que debe aparecer en la jerarquía de System i Navigator siempre que sea seleccionado por el usuario. Este puede ser el mismo que el icono por omisión del índice.

AdminItem
Serie que contiene el ID de la función de Administración de Aplicaciones que controla el acceso a la carpeta. Si se omite este campo, ninguna función de Administración de Aplicaciones controla el acceso a la carpeta. Si se especifica, debe ser el ID de una función de grupo o administrable. No puede ser el ID de una función de producto.

TaskpadNameID
Una palabra doble que contiene el ID de recurso de la serie que debe aparecer como nombre del área de tareas en la jerarquía de System i Navigator.

TaskpadDescriptionID
Palabra doble que contiene el identificador de recurso de la serie de texto en la DLL de recursos. Esta DLL de recursos se utiliza para describir la función del área de tareas en la interfaz de usuario de System i Navigator.

Añadir elementos de menú de contexto

```
;------------------------------------------------------------------
; Registrar un manejador de menú de contexto para la carpeta nueva y sus objetos

[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY EXTENSIONS\IBM.MsgQueueSample3\
   shellex\Sample3\*\ContextMenuHandlers\{1827A857-9C20-11d1-96C3-00062912C9B2}]
"JavaClass"="com.ibm.as400.opnav.MsgQueueSample3.MqActionsManager"

;------------------------------------------------------------------------------
; Registrar un manejador de menú cont. arrastrar/soltar para la carpeta nueva y objetos

[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY EXTENSIONS\IBM.MsgQueueSample3\
   shellex\Sample3\*\DragDropHandlers\{1827A857-9C20-11d1-96C3-00062912C9B2}]
"JavaClass"="com.ibm.as400.opnav.MsgQueueSample3.MqActionsManager"
```

Añadir tareas de área de tareas

```
;------------------------------------------------------------------
; Registrar un manejador de tareas para la carpeta nueva y sus objetos

[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY EXTENSIONS\IBM.MsgQueueSample5\
   shellex\Sample5\*\TaskHandlers\{1827A857-9C20-11d1-96C3-00062912C9B2}]
"JavaClass"="com.ibm.as400.opnav.MsgQueueSample5.MqTasksManager"
"JavaClassType"="TasksManager"
```

Proporcionar soporte para arrastrar/soltar

```
;-------------------------------------------------------------------
; Registrar un manejador de acciones de soltar para la carpeta nueva y sus objetos

[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY EXTENSIONS\IBM.MsgQueueSample3\
    shellex\Sample3\*\DropHandler]
@="{1827A857-9C20-11d1-96C3-00062912C9B2}"
"JavaClass"="com.ibm.as400.opnav.MsgQueueSample3.MqDropTargetManager"
```

Especificar los objetos que se gestionarán

Se necesita un par de identificadores de tipo de objeto en la clave `shellex`. El primer identificador en el par especifica la carpeta raíz de un componente de System i Navigator. En el caso de las carpetas nuevas que añade el conector, este identificador debe coincidir con el tipo lógico de la carpeta que ha especificado como punto de unión. Para las carpetas existentes, esta subclave debe generalmente ser el tipo de objeto de la carpeta de primer nivel bajo un objeto de contenedor de System i. Estas series de tipo están definidas bajo `HKEY_CLASSES_ROOT\IBM.AS400.Network\TYPES` en el registro.

El segundo identificador del par identifica el tipo de objeto específico que el conector desea modificar. Si se especifica ″*″, se efectuará una llamada al conector para el tipo de carpeta que se ha identificado en el primer identificador, más la totalidad de carpetas y objetos que aparecen en la jerarquía bajo esa carpeta. De lo contrario, debe especificarse un identificador de tipo específico y solo se efectuará una llamada al conector cuando se realice una acción en un objeto de ese tipo.

Recuerde que cualquier número de conectores puede registrar su intención de añadir funciones a un tipo de objeto dado en la jerarquía de System i Navigator. El conector nunca debe asumir que es el único componente del sistema que está proporcionando funciones para un tipo de objeto determinado. Esto se aplica no sólo a tipos de objetos existentes, sino también a cualquier nuevo objeto que pueda definir un conector. Si el conector se usa de forma generalizada, nada puede evitar que otro proveedor amplíe los tipos de objeto definidos por el conector.

CLSIDs
: Los CLSID mostrados en los ejemplos anteriores especifican el componente servidor ActiveX incorporado que gestiona los conectores Java. En el caso de todas las funciones no relacionadas con carpetas, este CLSID siempre debe ser `{1827A857-9C20-11d1-96C3-00062912C9B2}`.

JavaClass
: El nombre de clase Java totalmente calificada de la implementación de interfaz que debe ser llamada por System i Navigator para dar soporte a la función designada.

Soporte SSL

Si las comunicaciones de un conector con el sistema se establecen utilizando la API Sockets o algún otro servicio de comunicaciones de bajo nivel, es responsabilidad del conector dar soporte a SSL, si se ha solicitado SSL. Si el conector no proporciona este soporte, debe especificar `"Support Level"=dword:00000000`. Esto indica que el conector no soporta SL. En este caso, la función del conector estará inhabilitada si el usuario ha solicitado una conexión segura.

```
;-------------------------------------------------------------------------
; Indicar que este conector da soporte a SSL.

[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY EXTENSIONS\IBM.MsgQueueSample3\SSL]
"Support Level"=dword:00000001
```

Support Level
: Si el conector soporta SSL, este valor debe ser 1. En caso contrario, debe ser 0.

Páginas de propiedades para un manejador de hojas de propiedades

Las clases de la biblioteca Microsoft Foundation Class (MFC) no dan soporte a la creación de páginas de propiedades para un manejador de hojas de propiedades. No obstante, IBM suministra CExtPropertyPage in en lugar de la clase MFC CPropertyPage.

Las páginas de propiedades implementadas por los conectores de System i Navigator deben tener la subclase CExtPropertyPage. La declaración de clase puede encontrarse- en el archivo de cabecera PROPEXT.H y la implementación está en el archivo PROPEXT.CPP. Ambos archivos se facilitan como parte del conector de ejemplo.

Nota: Es necesario incluir PROPEXT.CPP en el área de trabajo de proyecto del conector.

Si un conector requiere que una hoja de propiedades esté asociada a uno de sus propios tipos de objeto, el distintivo SFGAO_HASPROPSHEET debe volver como parte de los atributos del objeto. Cuando este distintito está activo, System i Navigator añade automáticamente Propiedades al menú contextual del objeto y llama a cualquier manejador de hojas de propiedades registrado para que añada páginas a la hoja de propiedades es el elemento de menú contextual está seleccionado.

En algunos casos, un conector puede implementar un elemento de menú contextual Propiedades definido para uno de sus propios tipos de objetos como un diálogo estándar de Windows en lugar de como una hoja de propiedades. Para esta situación se define un distintivo. Puede devolverse a System i Navigator en llamadas a IContextMenu::QueryContextMenu. Si se devuelve el distintivo, no se lleva a cabo ningún proceso automático para Propiedades y es el conector el que debe añadir el elemento de menú de contexto e implementar el diálogo asociado. Este distintivo se documenta en Descripción de los distintivos de QueryContextMenu.

Si un conector tiene la intención de añadir páginas de propiedades a las hojas de propiedades de un usuario, la clave que especifica el CLSID del manejador de hojas de propiedades deberá especificar un campo PropSheet. Este campo identifica la hoja de propiedades a la que el manejador especificado añadirá las páginas. Este es un ejemplo.

```
;------------------------------------------------------------------
;
Registre un manejador de hojas de propiedades para la hoja de propiedades Red para los usuarios de System i
[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY plug-inS\IBM.Sample\shellex\Users
y Groups\User\PropertySheetHandlers\{3D7907A1-9080-11d0-82BD-08005AA74F5C}]
"PropSheet"="Redes"
```

Los valores válidos del campo PropSheet son:

Valores válidos del campo PropSheet				
Grupos	**Personal**	**Seguridad o posibilidades**	**Trabajos**	**Redes**
Groups-Before-All Groups-After-Info	Personal-Before-All Personal-After-Name Personal-After-Location Personal-After-Mail	Capabilities-Before-All Capabilities-After-Privileges Capabilities-After-Auditing Capabilities-Before-Other Capabilities-After-Other	Jobs-Before-All Jobs-After-General Jobs-After-Startup Jobs-After-Display Jobs-After-Output Jobs-After-International	Networks-Before-All Networks-After-Servers Networks-After-General

Para añadir páginas a una hoja de propiedades de un usuario del sistema, el conector deberá implementar la interfaz IA4PropSheetNotify (consulte el listado de especificaciones de la interfaz IA4PropSheetNotify).

Restricción: Las hojas de propiedades de los objetos de usuario System i tienen esta restricción en la actualidad. No es posible implementar varios manejadores de hojas de propiedades para las distintas hojas de propiedades asociadas con un usuario del sistema en la misma clase de implementación. Cada una de las hojas de propiedades necesita un CLSID aparte.

Conceptos relacionados

"Lista de especificaciones de la interfaz IA4PropSheetNotify" en la página 32
La interfaz IA4PropSheetNotify proporciona notificaciones a la implementación de IShellPropSheetExt. Estas notificaciones son necesarias para añadir páginas de propiedades a una de las hojas de propiedades Usuarios y grupos.

Descripción de los distintivos de QueryContextMenu:

System i Navigator ha mejorado su soporte de la interfaz IContextMenu.

Orden de los elementos de menú de contexto

System i Navigator ha ampliado el soporte de la interfaz IContextMenu para obtener un control más preciso sobre el orden en el que los elementos de menú se añaden al menú para una carpeta u objeto en particular. System i Navigator estructura sus menús contextuales en tres secciones. Esta estructura garantiza que, cuando más de un componente añada elementos al menú de contexto de un objeto, los elementos seguirán apareciendo en el orden correcto que se haya definido para la interfaz de usuario de Windows.

La primera sección contiene las acciones específicas del tipo de objeto, como por ejemplo Reorganizar en el caso de una tabla de base de datos. La segunda sección contiene elementos de ″creación de objetos″; estos elementos son los tipos de objeto que penden en cascada del elemento de menú Nuevo. Por último se encuentran los denominados elementos de menú ″estándar″ de Windows, como por ejemplo Suprimir o Propiedades. Puede elegir añadir elementos de menú a cualquier sección del menú de contexto.

System i Navigator llama al método QueryContextMenu para un componente tres veces consecutivas, una por cada sección del menú. Los distintivos adicionales siguientes se definen en el parámetro uFlags para permitirle determinar con qué sección del menú de contexto se trabaja.

UNITY_CMF_CUSTOM
Este distintivo indica que debe añadir acciones específicas del objeto al menú.

UNITY_CMF_NEW
Este distintivo indica que debe añadir elementos de creación de objetos al menú.

UNITY_CMF_STANDARD
Este distintivo indica que debe añadir acciones estándar al menú.

UNITY_CMF_FILEMENU

Este distintivo cambia UNITY_CMF_STANDARD. Indica la creación del menú desplegable Archivo para el objeto, en lugar del menú que se visualiza cuando el usuario pulsa sobre un objeto con el botón 2 del ratón.

Los elementos del menú desplegable Archivo se organizan de forma algo distinta. Si añade Propiedades al menú, debe evitar insertar un separador como normalmente se hace antes de este elemento. Asimismo, no es conveniente añadir acciones de edición tales como Copiar o Pegar al menú Archivo, ya que aparecen en el menú desplegable Edición. (System i Navigator llama al conector de shell en el momento correspondiente para obtener los elementos del menú Editar y no establecer UNITY_CMF_FILEMENU).

Diálogos de propiedad exclusivos

En algunos casos, un conector puede querer implementar un elemento de menú de contexto Propiedades definido para uno de sus propios tipos de objeto como diálogo estándar de Windows en lugar de una hoja de propiedades. Se define un distintivo para esta situación que puede devolverse a System i Navigator en llamadas a IContextMenu::QueryContextMenu cuando el distintivo UNITY_CMF_STANDARD está establecido. Este distintivo, A4HYF_INFO_PROPERTIESADDED, debe establecerse en el valor de HRESULT devuelto por QueryContextMenu.

El hecho de devolver este distintivo significa que no se realiza el proceso automático de Propiedades. En este caso, el conector debe añadir el elemento de menú de contexto y crear el diálogo asociado.

Ejemplo: crear páginas de propiedades Visual Basic para un manejador de hojas de propiedades

No puede utilizar una clave de registro para especificar páginas de propiedades implementadas por conectores de System i Navigator Visual Basic. Debe añadir un elemento de menú de contexto de página de propiedades específico en la clase ListManager para implementar una página de propiedades. No puede añadir una página de propiedades a ningún objeto de hoja de propiedades existente.

En el conector de ejemplo de Visual Basic, una página de propiedades está soportada para las bibliotecas en la lista de System i Navigator. Esto se lleva a cabo siguiendo el procedimiento que se describe a continuación:

1. En listman.cls, el tipo de objeto de biblioteca (Library) especifica una página de propiedades en el método getAttributes:

```
' Devuelve los atributos de un objeto de la lista.
Public Function ListManager_getAttributes(ByVal item As Object) As Long
    Dim uItem As ItemIdentifier
    Dim nAttributes As ObjectTypeConstants

    If Not IsEmpty(item) Then
        Set uItem = item
    End If

    If uItem.getType = "SampleVBFolder" Then
        nAttributes = OBJECT_ISCONTAINER
    ElseIf item.getType = "SampleLibrary" Then
        nAttributes = OBJECT_IMPLEMENTSPROPERTIES
    Else
        nAttributes = 0
    End If

    ListManager_getAttributes = nAttributes
End Function
```

2. En actnman.cls, el método queryActions especifica que las propiedades deben mostrarse en el menú de contexto del objeto de biblioteca (Library).

```
Public Function ActionsManager_queryActions(ByVal flags As Long) As Variant
        .
        .

        ' Añadir elementos de menú a una biblioteca de ejemplo
        If selectedFolderType = "SampleLibrary" Then
             ' Acciones estándar
             If (flags And STANDARD_ACTIONS) = STANDARD_ACTIONS Then
                   ReDim actions(0)

                   ' Propiedades
                   Set actions(0) = New ActionDescriptor
                   With actions(0)
                         .Create
                         .setID IDPROPERTIES
                         .SetText m_uLoader.getString(IDS_ACTIONTEXT_PROPERTIES)
```

```
                        .setHelpText m_uLoader.getString(IDS_ACTIONHELP_PROPERTIES)
                        .setVerb "PROPERTIES"
                        .setEnabled True
                        .setDefault True
                    End With

                    ' Propiedades solo pueden seleccionarse si SOLO hay 1 objeto seleccionado
                    If Not IsEmpty(m_ObjectNames) Then
                        If UBound(m_ObjectNames) > 0 Then
                            actions(2).setEnabled False
                        End If
                    End If
                End If
            End If
            .
            .
    End Function
```

3. En actnman.cls, el método actionsSelected visualiza un formulario de propiedades cuando se selecciona el menú de contexto de propiedades.

```
    Public Sub ActionsManager_actionSelected(ByVal action As Integer, ByVal owner As Long)
        .
        .
        Select Case action
            .
            .
            Case IDPROPERTIES
                If (Not IsEmpty(m_ObjectNames)) Then
                    ' Pasar el nombre del sistema a un campo oculto del formulario para uso posterior
                    frmProperties.lblSystemName = m_ObjectNames(0).getSystemName

                    ' Pasar el nombre de visualización del objeto seleccionado
                    ' a un campo oculto del formulario
                    frmProperties.lblLibName = m_ObjectNames(0).getDisplayName

                    ' Mostrar las propiedades
                    frmProperties.Show vbModal
                End If
            .
            .
            Case Else
                'Do Nothing
        End Select

      .
    End Sub
```

Nota: El código para crear y visualizar la hoja de propiedades puede verse en **propsht.frm**.

Manejar las hojas de propiedades en Java

Puede añadir páginas de propiedades a las hojas de propiedades del conector de Java. Puede entonces crear nombres de objetos, mostrar propiedades, compartir objetos con terceros y mezclar código C++ y Java en el mismo conector.

Para utilizar páginas de propiedades, debe crear la interfaz del gestor de propiedades, que proporciona los métodos siguientes:

- Initialize

  Identifica el objeto contenedor de las propiedades.
- getPages

  Crea y proporciona un vector de objetos PanelManager.
- CommitHandlers

  Devuelve un vector de manejadores a los que se efectuará una llamada tras Commit.

- CancelHandlers

  Devuelve un vector de manejadores a los que se efectuará una llamada tras Cancel.

A continuación habilite el menú de propiedades haciendo que el método getAttributes de ListManager devuelva ListManager.OBJECT_HASPROPERTIES.

Por último, cree una entrada del registro que identifique PopertiesManagerInterface. Por ejemplo:

```
[HKEY_CLASSES_ROOT\IBM.AS400.Network\AS/400 Network\*
\shellex\PropertySheetHandlers\{1827A857-9C20-11d1-96C3-00062912C9B2}]
"JavaClass"="com.ibm.as400.opnav.TestPages.TestPropertiesManager"
"JavaClassType"="PropertiesManager"
```

Nota: Pueden registrarse varias implementaciones de PropertiesManager para proporcionar páginas de propiedades para un tipo de objeto determinado. No suponga que su entidad es la única que proporciona páginas ni el orden en que se añadirán las páginas.

Ejemplo: gestor de propiedades Java:

Este ejemplo muestra un ejemplo de código del gestor de propiedades de Java.

Nota: Al utilizar los ejemplos de código, aceptará los términos del "Información sobre licencia de código y exención de responsabilidad" en la página 95.

```
package com.ibm.as400.opnav.Sample;

import com.ibm.as400.opnav.*;

import java.awt.Frame;

import com.ibm.as400.ui.framework.java.*;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

public class SamplePropertiesManager implements
PropertiesManager
{

// La lista de objetos seleccionados.
ObjectName[] m_objectNames;

// Guardar la matriz de nombres de objetos seleccionados
//
public void initialize(ObjectName[] objectNames)
{

m_objectNames = objectNames;

}

// Devolver una matriz de gestores de paneles
//
public PanelManager[] getPages()
{

// Crear una instancia de los beans de datos
MyDataBean dataBean = new MyDataBean();
dataBean.load();
AnotherDataBean dataBean2 = new AnotherDataBean();
dataBean2.load();

DataBean[] dataBeans = { dataBean };
DataBean[] dataBeans2 = { dataBean2 };
```

```
// Crear el panel
PanelManager pm = null;
PanelManager pm2 = null; try
{

pm = new PanelManager("com.ibm.as400.opnav.Sample.Sample",
"PAGE1",
dataBeans);

pm2 = new PanelManager("com.ibm.as400.opnav.Sample.Sample",
"PAGE2",
dataBeans2);


}

catch (com.ibm.as400.ui.framework.java.DisplayManagerException
e)
{

Monitor.logError("SamplePropertiesManager: Exception when
creating pages "+e);

}

pm.setTitle("Primera página Java");
pm2.setTitle("Segunda página Java");

PanelManager[] PMArray = {pm, pm2};

return PMArray;

}

// Devolver una lista de objetos ActionListener que notificar cuando
se procese la confirmación

public ActionListener[] getCommitListeners()
{

ActionListener[] al = new ActionListener[1];
al[0] = new ActionListener()
{

public void actionPerformed(ActionEvent evt)
{

Monitor.logError("SamplePropertiesManager: Processing Commit
Listener");

}

};
return al;

}
// Devolver una lista de objetos ActionListener que notificar cuando
se seleccione Cancelar
public ActionListener[] getCancelListeners()
{

ActionListener[] al = new ActionListener[1];
al[0] = new ActionListener()
{

public void actionPerformed(ActionEvent evt)
{
```

```
Monitor.logError("SamplePropertiesManager: Processing Cancel
Listener");

}

};
return al;

}

}
```

Entrada de registros de capa de sockets segura (SSL)

Los usuarios de System i Navigator pueden solicitar una conexión segura con un sistema seleccionando el recuadro **Utilizar Capa de sockets segura** en la página Conexión de la hoja de propiedades de los objetos System i. Cuando se hace esto, sólo los componentes de System i Navigator que puedan soportar comunicaciones de capa de sockets segura (SSL) podrán ser activados por el usuario.

Si todas las comunicaciones de un conector con el sistema son gestionadas utilizando el manejador del sistema System i Access para Windows (mediante cwbCO_SysHandle) o utilizando la clase com.ibm.as400.access.AS400 en el caso de un conector de Java, el conector deberá indicar que soporta conexiones seguras con el sistema. Para los conectores C++, cwbCO_SysHandle se obtiene efectuando una llamada a la API cwbUN_GetSystemHandle. Cuando el usuario solicita una conexión segura, System i Navigator automáticamente habilita SSL. En el caso de los conectores de Java, el objeto de System i obtenido llamando al método getSystemObject en la clase com.ibm.as400.opnav.ObjectName es, en realidad, una instancia de com.ibm.as400.access.SecureAS400.

Nota: Si está ejecutando Java sobre SSL y creando su propio certificado de CA, necesitará el paquete de servicio GA de System i Access para Windows.

Cuando se establecen las comunicaciones de un conector con el sistema utilizando la API Sockets o algún otro servicio de comunicaciones de bajo nivel, es responsabilidad del conector dar soporte a SSL, si se ha solicitado SSL. Si el conector no proporciona este soporte, el conector deberá indicar que no soporta SSL tal como se muestra en el siguiente ejemplo. En este caso, la función del conector está inhabilitada, si el usuario a solicitado una conexión.

Ejemplo: añadir una clave del registro para habilitar SSL

La clave es SSL bajo `[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY EXTENSIONS\IBM.Sample\SSL]` `"Support Level"=dword:00000001` donde `IBM.Sample` es el componente de producto suministrado por el conector.

Nota: `"Support Level"=dword:00000001` = da soporte a SSL, y `"Support Level"=dword:00000000` = NO da soporte a SSL.

```
 ;-------------------------------------------------------------------------

; Clave de registro de ejemplo que
  indica que este conector da soporte a SSL
 {HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY EXTENSIONS\IBM.Sample\SSL}
 "Support Level"=dword:00000001
```

Información sobre licencia de código y exención de responsabilidad

IBM le otorga una licencia de copyright no exclusiva para utilizar todos los ejemplos de código de programación, a partir de los que puede generar funciones similares adaptadas a sus necesidades específicas.

SUJETO A LAS GARANTÍAS ESTATUTARIAS QUE NO PUEDAN EXCLUIRSE, IBM Y LOS DESARROLLADORES Y SUMINISTRADORES DE PROGRAMAS DE IBM NO OFRECEN NINGUNA GARANTÍA NI CONDICIÓN, YA SEA IMPLÍCITA O EXPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS O CONDICIONES IMPLÍCITAS DE COMERCIALIZACIÓN, ADECUACIÓN A UN PROPÓSITO DETERMINADO Y NO VULNERACIÓN CON RESPECTO AL PROGRAMA O AL SOPORTE TÉCNICO, SI EXISTE.

BAJO NINGUNA CIRCUNSTANCIA, IBM Y LOS DESARROLLADORES O SUMINISTRADORES DE PROGRAMAS DE IBM SE HACEN RESPONSABLES DE NINGUNA DE LAS SIGUIENTES SITUACIONES, NI SIQUIERA EN CASO DE HABER SIDO INFORMADOS DE TAL POSIBILIDAD:

1. PÉRDIDA DE DATOS O DAÑOS CAUSADOS EN ELLOS;
2. DAÑOS ESPECIALES, ACCIDENTALES, DIRECTOS O INDIRECTOS, O DAÑOS ECONÓMICOS DERIVADOS;
3. PÉRDIDAS DE BENEFICIOS, COMERCIALES, DE INGRESOS, CLIENTELA O AHORROS ANTICIPADOS.

ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN O LA LIMITACIÓN DE LOS DAÑOS DIRECTOS, ACCIDENTALES O DERIVADOS, POR LO QUE PARTE DE LAS LIMITACIONES O EXCLUSIONES ANTERIORES, O TODAS ELLAS, PUEDE NO SER PROCEDENTE EN SU CASO.

Apéndice. Avisos

Esta información se ha escrito para productos y servicios ofrecidos en Estados Unidos de América.

Es posible que en otros países IBM no ofrezca los productos, los servicios o las características que se describen en este documento. El representante local de IBM le puede informar acerca de los productos y servicios que actualmente están disponibles en su localidad. Las referencias hechas a productos, programas o servicios de IBM no pretenden afirmar ni dar a entender que únicamente puedan utilizarse dichos productos, programas o servicios de IBM. Puede utilizarse en su lugar cualquier otro producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patente pendientes de aprobación que cubran los temas descritos en este documento. La posesión de este documento no le otorga ninguna licencia sobre dichas patentes. Puede enviar las consultas sobre licencias, por escrito, a la siguiente dirección:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
Estados Unidos de América

Para consultas sobre licencias relativas a la información de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM en su país o envíe las consultas, por escrito, a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japón

El párrafo siguiente no es de aplicación en el Reino Unido ni en ningún otro país en el que tales disposiciones sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGUNA CLASE, EXPLÍCITA O IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN Y DE COMERCIALIZACIÓN O IDONEIDAD PARA UN PROPÓSITO DETERMINADO. Algunas legislaciones no contemplan la declaración de limitación de responsabilidad, ni implícitas ni explícitas, en determinadas transacciones, por lo que cabe la posibilidad de que esta declaración no se aplique en su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. La información incluida en este documento está sujeta a cambios periódicos, que se incorporarán en nuevas ediciones de la publicación. IBM puede efectuar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento y sin previo aviso.

Las referencias hechas en esta publicación a sitios Web que no son de IBM se proporcionan únicamente por cortesía y de ningún modo deben interpretarse como promoción de dichos sitios Web. Los materiales de estos sitios Web no forman parte de los materiales de IBM para este producto, y el usuario será responsable del uso que se haga de estos sitios Web.

IBM puede utilizar o distribuir la información que usted le suministre del modo que IBM considere conveniente sin incurrir por ello en ninguna obligación para con usted.

Los licenciatarios de este programa que deseen obtener información acerca del mismo con el fin de: (i) intercambiar la información entre programas creados independientemente y otros programas (incluido este) y (ii) utilizar mutuamente la información que se ha intercambiado, deben ponerse en contacto con:

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
Estados Unidos de América

Esta información puede estar disponible, sujeta a los términos y condiciones pertinentes, e incluir en algunos casos el pago de una cantidad.

El programa bajo licencia descrito en esta información, así como todo el material bajo licencia disponible para él, lo proporciona IBM bajo los términos del Acuerdo de Cliente de IBM, el Acuerdo Internacional de Programas bajo Licencia de IBM, el Acuerdo de Licencia para Código Máquina de IBM o cualquier otro acuerdo equivalente entre ambas partes.

Los datos de rendimiento incluidos aquí se determinaron en un entorno controlado. Por lo tanto, los resultados que se obtengan en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse realizado en sistemas que estén en fase de desarrollo y no existe ninguna garantía de que esas mediciones vayan a ser iguales en los sistemas disponibles en el mercado. Además, es posible que algunas mediciones se hayan estimado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información concerniente a productos que no son de IBM se ha obtenido de los suministradores de dichos productos, de sus anuncios publicados o de otras fuentes de información pública disponibles. IBM no ha comprobado dichos productos y no puede afirmar la exactitud en cuanto a rendimiento, compatibilidad u otras características relativas a productos no IBM. Las consultas acerca de las prestaciones de los productos que no son de IBM deben dirigirse a los suministradores de tales productos.

Todas las declaraciones relativas a la dirección o intención futura de IBM están sujetas a cambios o anulación sin previo aviso y representan únicamente metas y objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlas de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con los nombres y direcciones utilizados por una empresa real es mera coincidencia.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que muestran técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir los programas de ejemplo de cualquier forma, sin tener que pagar a IBM, con intención de desarrollar, utilizar, comercializar o distribuir programas de aplicación que estén en conformidad con la interfaz de programación de aplicaciones (API) de la plataforma operativa para la que están escritos los programas de ejemplo. Los ejemplos no se han probado minuciosamente bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni dar por sentada la fiabilidad, la facilidad de mantenimiento ni el funcionamiento de los programas.

Cada copia o parte de estos programas de ejemplo, así como todo trabajo derivado, debe incluir un aviso de copyright como el siguiente:

© (nombre de su empresa) (año). Algunas partes de este código se derivan de programas de ejemplo de IBM Corp. © Copyright IBM Corp. _escriba el año o años_. Reservados todos los derechos.

Si está viendo esta información en copia software, es posible que las fotografías y las ilustraciones en color no aparezcan.

Información de la interfaz de programación

Esta publicación de desarrollo de conectores de System i Navigator documenta las interfaces de programación cuya finalidad es escribir programas para obtener servicios de IBM i5/OS.

Marcas registradas

Los términos siguientes son marcas registradas de International Business Machines Corporation en Estados Unidos y/o en otros países:

i5/OS
IBM
IBM (logotipo)
iSeries
System i

Adobe, el logotipo de Adobe logo, PostScript y el logotipo de PostScript son marcas registradas de Adobe Systems Incorporated en los Estados Unidos y/o en otros países.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en Estados Unidos y/o en otros países.

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc., en Estados Unidos y/o en otros países.

Otros nombres de empresas, productos o servicios pueden ser marcas registradas o de servicio de terceros.

Términos y condiciones

Los permisos para utilizar estas publicaciones están sujetos a los siguientes términos y condiciones.

Uso personal: puede reproducir estas publicaciones para uso personal (no comercial) siempre y cuando incluya una copia de todos los avisos de derechos de autor. No puede distribuir ni visualizar estas publicaciones ni ninguna de sus partes, como tampoco elaborar trabajos que se deriven de ellas, sin el consentimiento explícito de IBM.

Uso comercial: puede reproducir, distribuir y visualizar estas publicaciones únicamente dentro de su empresa, siempre y cuando incluya una copia de todos los avisos de derechos de autor. No puede elaborar trabajos que se deriven de estas publicaciones, ni tampoco reproducir, distribuir ni visualizar estas publicaciones ni ninguna de sus partes fuera de su empresa, sin el consentimiento explícito de IBM.

Aparte de la autorización que se concede explícitamente en este permiso, no se otorga ningún otro permiso, licencia ni derecho, ya sea explícito o implícito, sobre las publicaciones, la información, los datos, el software o cualquier otra propiedad intelectual contenida en ellas.

IBM se reserva el derecho de retirar los permisos aquí concedidos siempre que, según el parecer del fabricante, se utilicen las publicaciones en detrimento de sus intereses o cuando, también según el parecer del fabricante, no se sigan debidamente las instrucciones anteriores.

No puede bajar, exportar ni reexportar esta información si no lo hace en plena conformidad con la legislación y normativa vigente, incluidas todas las leyes y normas de exportación de Estados Unidos.

IBM NO PROPORCIONA NINGUNA GARANTÍA SOBRE EL CONTENIDO DE ESTAS PUBLICACIONES. LAS PUBLICACIONES SE PROPORCIONAN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPLÍCITA O IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN, NO VULNERACIÓN E IDONEIDAD PARA UN FIN DETERMINADO.

Impreso en España