



IBM Systems - iSeries

Conexión a iSeries

Desarrollo de conectores de iSeries Navigator

Versión 5 Release 4





IBM Systems - iSeries

Conexión a iSeries

Desarrollo de conectores de iSeries Navigator

Versión 5 Release 4

Nota

Antes de utilizar esta información y el producto al que hace referencia, lea la información que figura en: "Avisos", en la página 101.

Sexta edición (febrero de 2006)

Esta edición atañe a la versión 5, release 4, modificación 0 de i5/OS (5722-SS1) y a todos los releases y modificaciones posteriores 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.

© Copyright International Business Machines Corporation 2004. Reservados todos los derechos.

Contenido

Desarrollo de conectores de iSeries

Navigator 1

Novedades del release V5R4 1

Archivo PDF imprimible 1

Soporte para conectores en iSeries Navigator 2

 Qué se puede realizar con un conector. 2

 Funcionamiento de los conectores 2

 Requisitos de los conectores 4

 Distribuir conectores. 5

 Identificar conectores en iSeries Navigator 13

Instalar y ejecutar los conectores de ejemplo 13

 Configurar conectores C++ de ejemplo 14

 Configurar conectores Visual Basic de ejemplo. 15

 Configurar el conector Java de ejemplo 18

Información de consulta para la programación de conectores 21

 Información de consulta de C++ 21

 Información de consulta de Visual Basic. 71

 Información de consulta de Java 73

 Personalizar los archivos del registro de conectores 74

Información de licencia de código y declaración de limitación de responsabilidad 98

Apéndice. Avisos. 101

Información de la interfaz de programación 103

Marcas registradas. 103

Términos y condiciones 103

Desarrollo de conectores de iSeries Navigator

¿Está interesado en integrar los programas de cliente/servidor y las tareas de administración del servidor iSeries en un único entorno de aplicación? La función de conector para iSeries Navigator permite conseguirlo.

Puede utilizar conectores para consolidar aplicaciones de terceros y funciones especializadas escritas en C++, Visual Basic (VB) o Java en la interfaz de iSeries Navigator. Estos artículos le permitirán aprender 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 de licencia de código y declaración de limitación de responsabilidad” en la página 98.

Novedades del release V5R4

Información acerca de las novedades de este release

V5R4 incluye soporte para Java Runtime Environment (JRE) 1.4.2.

El desarrollo de conectores de iSeries Navigator tiene una sección ampliada en el archivo setup.ini con ejemplos nuevos para secciones de programa de salida.

Cómo visualizar las novedades o cambios

Para facilitar la visualización de los cambios técnicos, en esta información se utilizan:

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

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

Archivo PDF imprimible

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

Para ver o descargar la versión PDF de este documento, seleccione Desarrollo de conectores de iSeries Navigator (alrededor de 99 KB).

Cómo guardar los archivos PDF

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

1. Pulse el PDF con el botón derecho del ratón en el navegador (pulse el enlace anterior con el botón derecho del ratón).
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

Para poder ver o imprimir archivos PDF, debe instalar Adobe Reader en su sistema. Puede descargar una copia gratuita desde el sitio Web de Adobe (www.adobe.com/products/acrobat/readstep.html) .

Soporte para conectores en iSeries Navigator

El soporte para conectores de iSeries Navigator permite integrar de forma cómoda y sencilla las funciones y aplicaciones propias en una única interfaz de usuario: iSeries Navigator.

Estas funciones y aplicaciones nuevas pueden tener diversas complejidades, desde incorporar sencillos comportamientos nuevos a crear aplicaciones completas. Sea cual sea la nueva capacidad concreta que proporcione el conector, su integración en iSeries Navigator ofrece varias ventajas importantes. Por ejemplo, el hecho de empaquetar tareas del sistema comunes en una sola ubicación en iSeries Navigator puede simplificar de forma notable las funciones operativas y de administración habituales. Además, la interfaz GUI de iSeries Navigator garantiza que las funciones integradas se puedan llevar cabo de forma sencilla y con tan solo prerequisites mínimos en lo que a aptitudes se refiere.

Qué se puede realizar con un conector

Los conectores son conjuntos de clases y métodos predefinidos que iSeries Navigator iniciará en respuesta a una acción del usuario específica.

Puede utilizar conectores para añadir o modificar objetos y carpetas de la jerarquía de iSeries Navigator que representarán sus herramientas y aplicaciones. Puede personalizar por completo el soporte para las carpetas y objetos añadiendo o modificando lo siguiente:

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 añadir sus carpetas y objetos personalizados propios en la jerarquía de árbol de iSeries Navigator.

Funcionamiento de los conectores

Ejemplo detallado de cómo iSeries Navigator localizará un conector nuevo y lo instalará en una configuración nueva

La ilustración siguiente muestra cómo podría funcionar un conector Java que añada un nuevo contenedor al árbol de iSeries Navigator.

Tras identificar el nuevo conector en el registro de Windows, iSeries Navigator localizará el nuevo conector y lo instalará en una nueva configuración. Posteriormente el nuevo contenedor aparecerá en la jerarquía de iSeries Navigator. Cuando el usuario selecciona el contenedor, se efectúa una llamada al código Java del conector para obtener el contenido del contenedor (en este caso, una lista de mensajes de la cola de mensajes por omisión del usuario).

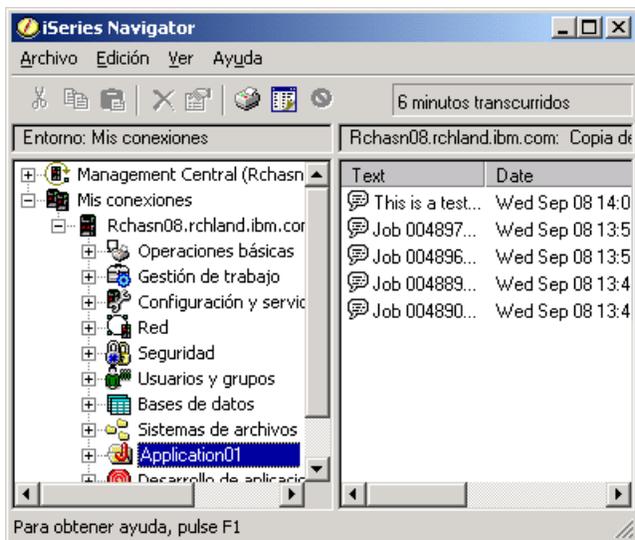


Figura 1. Diálogo de iSeries Navigator -- mensajes de la cola de mensajes

iSeries Navigator se comunica con el conector Java efectuando llamadas a los métodos definidos en una interfaz Java: ListManager. Esta interfaz permite a las aplicaciones Java proporcionar datos de lista a las vistas de árbol y lista de Navigator. Para integrar la aplicación en iSeries Navigator, debe crear una nueva clase Java que implemente esta interfaz. Los métodos de la nueva clase llaman a la aplicación Java existente para obtener los datos de lista, como se muestra más abajo.

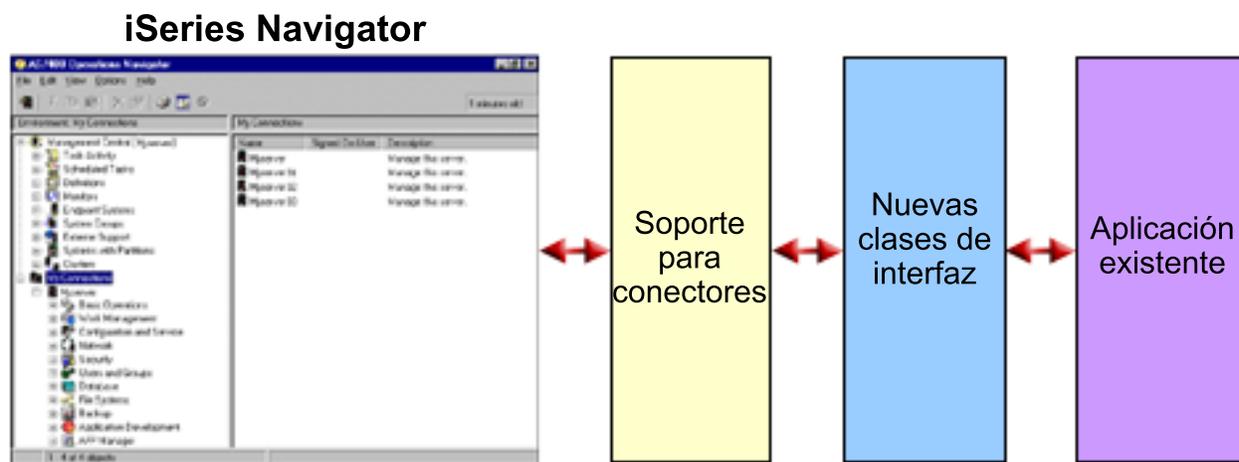


Figura 2. Cómo llama iSeries Navigator a una aplicación para obtener datos de lista

¿Qué sucede cuando el usuario desea llevar a cabo una acción en uno de los objetos? La ilustración siguiente muestra qué ocurre cuando el usuario pulsa el botón derecho del ratón sobre un objeto de mensaje para visualizar su menú de contexto.

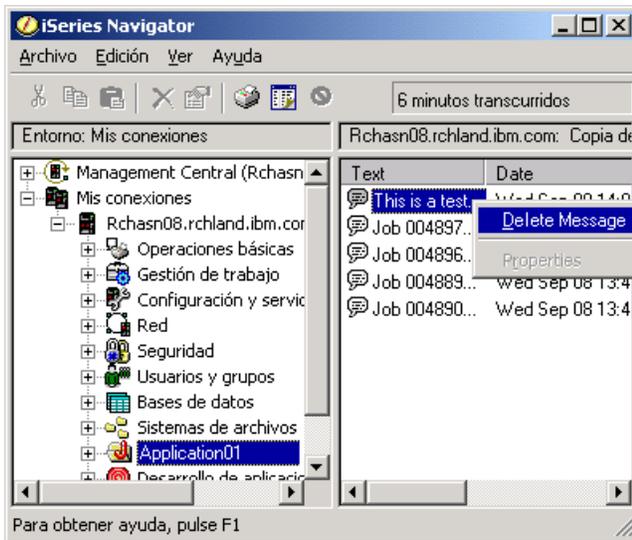


Figura 3. Menú de contexto de objeto de iSeries Navigator

iSeries Navigator efectúa una llamada a un método predefinido de otra interfaz Java: ActionsManager. Esta interfaz obtiene la lista de elementos de menú soportados para objetos de mensaje. Una vez más, debe crear una nueva clase Java que implemente esta interfaz. De esta forma conseguirá que las funciones especializadas de la aplicación estén disponibles para los usuarios mediante iSeries Navigator. Cuando el usuario selecciona el elemento de menú, iSeries Navigator efectúa una llamada a otro método de ActionsManager para llevar a cabo la acción. La implementación de ActionsManager llama a la aplicación Java existente, que a continuación puede visualizar un diálogo de confirmación o algún otro panel de interfaz de usuario más complejo que permite al usuario efectuar una tarea especializada. La interfaz de usuario de iSeries Navigator está diseñada para permitir a los usuarios trabajar con listas de recursos de servidor iSeries y llevar a cabo acciones en ellas. La arquitectura de la función de conector refleja el diseño de esta interfaz de usuario, tanto al definir interfaces para trabajar con listas de objetos de una jerarquía como al definir acciones sobre estos objetos. Una tercera interfaz, DropTargetManager, maneja las operaciones de arrastrar y soltar.

Requisitos de los conectores

Los requisitos de los conectores de iSeries Navigator varían en función del lenguaje de programación que se utilice.

Conectores C++

Los conectores que se desarrollan con el lenguaje de programación Visual C++ de Microsoft deben escribirse en la versión 4.2 o posterior.

Los conectores C++ también requieren las API de iSeries Navigator siguientes:

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

Conectores Java

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

Release	JRE	Swing	JavaHelp
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
V5R1	1.3.0	N/A	1.1.1
V4R5	1.1.8	1.1	N/A
V4R4	1.1.7	1.0.3	N/A

Todos los conectores Java requieren una pequeña DLL de recursos de Windows, que contiene determinada información sobre el conector. Esto permite a iSeries Navigator representar la función en la jerarquía de objetos de iSeries Navigator sin tener que cargar la implementación del conector. La DLL de recursos del ejemplo se ha creado con Visual C++ Versión 4.2 de Microsoft, pero puede emplearse cualquier compilador C que dé soporte a la compilación y al enlace de recursos de Windows.

iSeries Navigator proporciona una consola Java como herramienta de ayuda para 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 iSeries Navigator se visualizarán en recuadros de mensajes. Los archivos del registro que activan y desactivan la consola se suministran con el conector Java de ejemplo, que se encuentra en el juego de herramientas de iSeries Access para Windows.

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 un componente de instalación opcional de iSeries Access para Windows. Puede incluirse en la instalación inicial del producto iSeries Access para Windows o instalarse de forma selectiva más adelante, mediante el programa de instalación selectiva de iSeries 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 los conectores de ejemplo” en la página 13

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

Distribuir conectores

Puede entregar el código del conector a los usuarios de iSeries Navigator incluyéndolo con las aplicaciones i5/OS.

El programa de instalación de la aplicación escribe los binarios del código del conector, el archivo del registro y los recursos traducibles en una carpeta del sistema de archivos integrado (IFS) del servidor iSeries. Una vez finalizado este proceso, los usuarios finales pueden obtener el conector de la carpeta de iSeries Access para Windows (con la ayuda de una unidad de red correlacionada iSeries NetServer) utilizando el programa de instalación selectiva de iSeries Access para Windows. El programa de instalación selectiva copia el código del conector en la máquina del usuario, baja los recursos traducibles pertinentes según los valores de idioma del PC del usuario y ejecuta el archivo del registro para escribir la información del registro del conector en el registro de Windows. Si no se ha efectuado la instalación inicial, también puede instalar los conectores en la instalación inicial con la opción de personalización.

Para este tipo de conector...	Haga la instalación en este directorio...	E incluya estos archivos...
C++	/QIBM/USERDATA/OpNavPlugin/ <proveedor>.<componente> (para evitar la instalación sin iSeries Access para Windows)	<ul style="list-style-type: none"> • El archivo del registro del conector. • El archivo de instalación de iSeries Access para Windows del conector. • La DLL servidora ActiveX del conector, así como las DLL de código asociadas.
Java	/QIBM/USERDATA/OpNavPlugin/ <proveedor>.<componente> (los conectores Java requieren iSeries Access)	<ul style="list-style-type: none"> • El archivo del registro del conector. • El archivo de instalación de iSeries Access para Windows del conector. • El archivo JAR Java contiene todas las clases Java, HTML, .gif, PDML, PCML y archivos de serialización.
Visual Basic	/QIBM/USERDATA/OpNavPlugin/ <proveedor>.<componente> (los conectores VB requieren iSeries Access para Windows)	<ul style="list-style-type: none"> • El archivo del registro del conector. • El archivo de instalación de iSeries 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.
- El soporte para la ubicación GUIPlugin desaparecerá en un release futuro, por lo que debe migrar los conectores a la ubicación OpNavPlugin.

Asimismo, todos los conectores deben crear como mínimo un directorio bajo el subdirectorio <proveedor>.<componente> denominado MRI29XX, 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 que los usuarios hayan instalado el nuevo conector, puede elegir actualizarlo más adelante o distribuir arreglos para errores. Si se actualiza el código en el servidor iSeries, el programa de comprobación de versión de iSeries Access detectará que se ha producido este proceso y automáticamente bajará las actualizaciones a las máquinas de los usuarios. iSeries Access para Windows también proporciona soporte para la desinstalación, lo que permite a los usuarios eliminar por completo el conector de las máquinas siempre que lo deseen. Los usuarios pueden saber qué conectores están instalados en sus máquinas pulsando el botón del ratón sobre la pestaña Conectores de Propiedades de iSeries Navigator para un servidor iSeries.

Restringir el acceso al conector con políticas del sistema y Administración de aplicaciones

Si proporciona una plantilla de política de Windows con el conector, también puede aprovechar las políticas del sistema de Windows para controlar los usuarios de la red que pueden instalar el conector. Además, puede utilizar el soporte de Administración de aplicaciones basado en el servidor iSeries de

iSeries Navigator para controlar los usuarios y grupos de usuarios que pueden acceder al conector.

Archivo setup.ini

El archivo setup.ini del conector proporciona al asistente de instalación la información necesaria para instalar un conector de iSeries 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 debe denominarse SETUP.INI y debe residir en el directorio <proveedor>.<componente> principal del conector en el servidor iSeries.

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 los conectores VB de ejemplo” en la página 16

Ejemplo: sección de información de setup.ini:

Descripciones de los campos de información global necesarios en el archivo setup.ini

La primera sección del archivo de instalación (Plug-in Info) contiene información global sobre el conector:

```
[Plugin Info]
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.
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.

Campo de la sección [Plugin Info] de setup.ini	Descripción del campo
Version	<p>Valor numérico que indica el nivel de release del conector. El programa Comprobar nivel de servicio utiliza este valor para determinar si el conector necesita actualizarse en la estación de trabajo cliente. Este valor se incrementa en una cantidad con cada nuevo release del conector.</p> <p>El valor de Version se compara con el valor actual de Version del conector instalado en la estación de trabajo cliente. Si este valor de Version es superior al que ya existe en la estación de trabajo cliente, el programa Comprobar nivel de servicio actualizará el conector a la nueva versión.</p>
VendorID	<p>Serie con el formato <PROVEEDOR>.<COMPONENTE> que se emplea para identificar el conector. Esta serie se utiliza para crear la clave del registro del conector en el árbol del registro de iSeries Access para Windows. El valor de VendorID debe ser idéntico a la parte <PROVEEDOR>.<COMPONENTE> de la vía de acceso donde se instalará el conector en el servidor iSeries.</p>
JavaPlugin	<p>JavaPlugin se utiliza para indicar si se trata de un conector Java. El proceso de instalación tiene que llevar a cabo algún proceso especial si el conector es un conector Java. Todos los archivos JAR deben estar instalados en el directorio \PLUGINS\<PROVEEDOR>.<COMPONENTE> y este valor permite determinar si el proceso de instalación debe efectuar este proceso. Si se trata de un conector Java y este valor está establecido en NO o no existe, puede que el conector no funcione una vez instalado.</p>

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

Los campos de esta sección del archivo setup.ini contienen la información utilizada por el programa Comprobar nivel de servicio

La segunda sección del archivo de instalación (Service) proporciona al programa Comprobar nivel de servicio la información que necesita para determinar si debe aplicarse un nuevo nivel de arreglo del conector en la estación de trabajo cliente:

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

A continuación se muestra el significado de cada uno de los campos:

Campo de la sección [Service] de setup.ini	Descripción del campo
FixLevel	<p>Valor numérico que indica el nivel de servicio del conector. El programa Comprobar nivel de servicio utiliza este valor para determinar si el conector necesita servicio. Este valor debe incrementarse en una cantidad con cada release de servicio de una versión determinada.</p> <p>El valor de FixLevel se compara con el valor actual de FixLevel del conector instalado en la máquina del cliente. Si este valor de FixLevel es superior al del conector instalado en la estación de trabajo cliente, el programa Comprobar nivel de servicio dará servicio al conector aplicándole el nuevo valor de FixLevel. El valor debe restablecerse a cero al actualizarse un conector a una nueva versión o a un nuevo nivel de release.</p>
AdditionalSize	<p>Cantidad de espacio DASD que se necesita para almacenar los archivos ejecutables nuevos o adicionales que se añadirán al conector durante la aplicación del servicio. El programa de instalación emplea este valor para determinar si la estación de trabajo tiene el espacio de disco adecuado para el conector.</p>

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

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

La tercera parte del archivo de instalación contiene secciones que identifican los archivos que se instalarán 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 y el conector. No especifique nombres de vía de acceso de directorio. Una sección de archivos que no contenga entradas se pasa por alto.

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]	Archivos que se copian en \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. Para C++ y Visual Basic , en esta ubicación se encuentra la DLL servidora ActiveX (y las DLL de código asociadas) del conector. Para Java , en esta ubicación residirá el nombre de archivo JAR de código.
[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.
[Core Files]	Archivos que se copian en el directorio \WINDOWS\SYSTEM o \WINNT\SYSTEM32, cuyo uso se contabiliza en el registro y nunca se eliminan. Normalmente son archivos redistribuibles.
[MRI Files]	Archivos que se copian de los directorios de MRI del conector del servidor iSeries en los directorios CLIENT ACCESS\MRI29XX\<<PROVEEDOR>.<COMPONENTE> de la estación de trabajo. Habitualmente es donde residen los recursos dependientes del entorno nacional de un conector. Incluirá el nombre de 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 del servidor iSeries al mismo directorio en que están instalados los archivos [Base Files]. Habitualmente es donde residen los recursos JAR MRI29xx del conector. Para cada directorio MRI29xx soportado por el conector Java, se necesita una sección [Java MRI29xx] donde figuren esos archivos. Solo utilizan esta sección los conectores Java.
[Help files]	Archivos .HLP y .CNT que se copian de los directorios de MRI del conector del servidor iSeries en los directorios CLIENT ACCESS\MRI29XX\<<PROVEEDOR>.<COMPONENTE> de la estación de trabajo. 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
[Dependencias]	<p>Define los subcomponentes que debe haber instalados para poder instalar el conector. AS400_Client_Access_Express solo se necesita si el conector requiere la instalación de otros subcomponentes, distintos del subcomponente de soporte base de iSeries Navigator.</p> <p>AS400_Client_Access_Express</p> <ul style="list-style-type: none"> Los subcomponentes se especifican en una lista delimitada 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 con el prefijo CWBAD_COMP_. Estas constantes proporcionan los valores numéricos que se utilizan en la lista delimitada por comas para AS400_Client_Access_Express. Hay varias constantes CWBAD_COMP_ que identifican subcomponentes de font PC5250. Estas constantes no pueden utilizarse en el valor AS400_Client_Access_Express y se indican a continuación: <pre>//Subcomponentes de emulador de pantalla e impresora 5250 #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)</pre> <p>Nota: El valor AS400_Client_Access_Express se utiliza si existe; de lo contrario, se hace caso omiso de esta sección.</p>
[Service Base Files]	Archivos que se copian en \PLUGINS\<<PROVEEDOR>.<COMPONENTE> bajo el directorio de instalación de iSeries Access para Windows.
[Service Shared Files]	Archivos que se copian en el directorio Shared de iSeries 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 en el directorio \WINDOWS\SYSTEM o \WINNT\SYSTEM32. Estos archivos, cuyo uso se contabiliza en el registro, nunca se eliminan y normalmente son archivos redistribuibles.
[Service Registry Files]	Archivo del registro de Windows que está asociado al conector.

| Ejemplo: sección de programas de salida del archivo setup.ini:

| La última parte del archivo de instalación contiene secciones que identifican los programas que deben ejecutarse en la estación de trabajo cliente antes o después de una instalación, actualización o desinstalación.

| A continuación figuran ejemplos de la sintaxis utilizada en estas secciones de programas de salida para identificar y ejecutar programas en la estación de trabajo cliente antes o después de una instalación, actualización o desinstalación.

| Ejemplo 1 - Programa opcional al que debe llamarse antes de instalar archivos. Solo se ejecuta 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 <proveedor>.<componente> del conector en el origen de instalación.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. El valor por omisión es N. La instalación de este conector no continuará si este valor se establece en Y y el retorno no es cero. No se visualizará un mensaje si el programa devuelve un código de retorno no cero, pero se anotará un mensaje en el archivo INSTLOG.TXT
Wait	Opcional. Esperar a que el programa termine antes de continuar la ejecución. El valor por omisión es Y. Si CheckReturnCode=Y, se utiliza Wait=Y independientemente del valor especificado aquí.

| Ejemplo 2 - Programa opcional al que debe llamarse después de instalar archivos. Solo se ejecuta 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 <proveedor>.<componente> del conector en el PC.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. En PostInstallProgram están soportados los mismos valores que en PreInstallProgram, pero, puesto que el conector ya está instalado en este momento, el valor de CheckReturnCode no es significativo.
Wait	Opcional. Esperar a que el programa termine antes de continuar la ejecución. De nuevo, en PostInstallProgram están soportados los mismos valores que en PreInstallProgram, pero, puesto que el conector ya está instalado en este momento, el valor de CheckReturnCode no es significativo.

| Ejemplo 3 - Programa opcional al que debe llamarse antes de desinstalar 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 <proveedor>.<componente> del conector en el PC.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. El valor por omisión es N. Si el valor de CheckReturnCode está establecido en Y, la desinstalación del conector no continuará si el valor de retorno no es cero. No se visualizará un mensaje si el programa devuelve un código de retorno no cero, pero se anotará un mensaje en el archivo INSTLOG.TXT
Wait	Opcional. Esperar a que el programa termine antes de continuar 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 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 <proveedor>.<componente> del conector en el origen de instalación.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. El valor por omisión es N. La instalación de este conector no continuará si este valor se establece en Y y el retorno no es cero. No se visualizará un mensaje si el programa devuelve un código de retorno no cero, pero se anotará un mensaje en el archivo INSTLOG.TXT
Wait	Opcional. Esperar a que el programa termine antes de continuar 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 después de actualizar 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 <proveedor>.<componente> del conector en el origen de instalación.
CmdLine	Opcional. Los mandatos necesarios para el programa específico.
CheckReturnCode	Opcional. En PostUpgradeProgram están soportados los mismos valores que en PreUpgradeProgram, pero, puesto que el conector ya está instalado en este momento, el valor de CheckReturnCode no es significativo.

Campo de [PostUpgradeProgram]	Descripción
Wait	Opcional. Esperar a que el programa termine antes de continuar 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

Proporciona al programa de instalación selectiva de iSeries Access para Windows la información que necesita para instalar los recursos dependientes del entorno nacional que están asociados a un conector de iSeries Navigator en un PC cliente.

Debe dar al archivo el nombre MRISSETUP.INI. Debe haber una versión de este archivo en el subdirectorio MRI29XX del servidor iSeries para cada uno de los idiomas a los que da soporte 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 programa de instalación selectiva de iSeries Access para Windows comprueba el valor de versión de la interfaz MRI durante una instalación inicial y durante una actualización del conector al incrementar la versión o el nivel de release del conector. El valor de la versión de MRI de este campo debe coincidir con el valor de versión del archivo SETUP.INI del conector durante la instalación o actualización. Si estos valores no coinciden, los archivos de MRI no se copiarán en el PC 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 los conectores VB de ejemplo” en la página 16

Identificar conectores en iSeries Navigator

Los conectores se identifican a sí mismos en iSeries Navigator proporcionando información en el registro de Windows al instalarse el software del conector en los escritorios Windows de los usuarios.

Las entradas del registro especifican la ubicación del código del conector e identifican las clases que implementan las interfaces de iSeries Navigator especiales. Puede facilitar información adicional sobre el registro, que permita a iSeries Navigator determinar si debe activarse la función del conector para un sistema iSeries específico. Por ejemplo, un conector puede requerir un release de i5/OS mínimo determinado o puede especificar que debe haber instalado un producto concreto en el servidor iSeries para que funcione.

Cuando un usuario pulsa en un servidor iSeries en el árbol de jerarquía de iSeries Navigator tras instalar un conector, iSeries Navigator examina el servidor iSeries para determinar si puede dar soporte al nuevo conector. Los prerequisites de software (especificados en las entradas del registro del conector) se cotejan con el software instalado en el servidor iSeries. Si los requisitos del conector se cumplen, la nueva función se visualizará en el árbol de jerarquía. Si no se cumplen los requisitos, no se mostrará la función del conector para ese servidor iSeries, salvo que el archivo del registro especifique lo contrario.

Instalar y ejecutar los 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 el juego de herramientas mediante el programa de instalación selectiva de iSeries Access para Windows.

Nota: Antes de empezar a trabajar en cualquiera de los conectores de ejemplo, es conveniente que sepa cuáles son los requisitos específicos para desarrollar conectores en cada uno de los tres lenguajes.

Conceptos relacionados

“Requisitos de los conectores” en la página 4

Los requisitos de los conectores de iSeries Navigator varían en función del lenguaje de programación que se utilice.

Configurar conectores C++ de ejemplo

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

El ejemplo proporciona un área de trabajo de Developer Studio que puede utilizar para establecer puntos de interrupción y observar el comportamiento de un conector de iSeries Navigator típico. También permite verificar que el entorno de Developer Studio esté bien configurado para compilar y enlazar el código de conector.

Para utilizar el conector C++ de ejemplo en el PC, debe seguir el procedimiento siguiente:

Bajar el conector C++	Bajar el archivo ejecutable cppsmppq.exe. Cuando ejecute el archivo, se extraerán todos los archivos asociados al conector. Cree un directorio nuevo, c:\MyProject, y copie en él todos los archivos. Si crea un directorio distinto, tendrá que modificar el archivo del registro para especificar la ubicación correcta del conector.
Prepararse para crear una .dll servidora ActiveX	<ol style="list-style-type: none"> 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 directorio nuevo no es c:\MyProject, tendrá que cambiar el archivo del registro. 2. Copie todos los archivos de ejemplo en este directorio. Puede bajar los ejemplos de la página Web de Programmer's Toolkit - iSeries Navigator Plug-ins. 3. En Developer Studio, abra el menú Archivo y seleccione Abrir área de trabajo. 4. En el diálogo Abrir área de trabajo de proyecto, vaya al directorio MyProject y en Tipo de archivo: seleccione Archivos MAKE (*.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	<ol style="list-style-type: none"> 1. En Developer Studio, abra el menú Generar y seleccione Establecer configuración predeterminada... 2. En el diálogo Configuración de proyecto predeterminada, seleccione Configuración de depuración de Win32 de sampext. 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.

<p>Crear la biblioteca de recursos</p>	<p>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.</p> <p>Para crear la DLL de recursos, siga estos pasos:</p> <ol style="list-style-type: none"> 1. En Developer Studio, abra el menú Archivo y seleccione Abrir área de trabajo... y, a continuación, el directorio MyProject. 2. Especifique Archivos MAKE (*.mak) en Tipo de archivo:. 3. Seleccione sampmri.mak y pulse Abrir. 4. Abra el menú Generar y seleccione Regenerar todo para compilar y enlazar la DLL.
<p>Registrar la .dll servidora ActiveX</p>	<p>El archivo SAMPDBG.REG del directorio MyProject contiene claves del registro que comunican a iSeries Navigator la ubicación del conector de ejemplo de la estación de trabajo. Si ha especificado un directorio distinto a c:\MyProject, siga los pasos que se describen a continuación.</p> <ol style="list-style-type: none"> 1. Abra el archivo SAMPDBG.REG en Developer Studio (o utilice el editor de texto que desee). 2. Sustituya todas las apariciones de "c:\MyProject\" por "x:\<dir>", donde x es la letra de la unidad en la que reside el directorio y <dir> es el nombre del directorio. 3. Guarde el archivo. 4. En el Explorador de Windows, efectúe una doble pulsación en el archivo SAMPDBG.REG. Con esta acción, las entradas del archivo del registro se escribirán en el registro de Windows de la máquina.
<p>Ejecutar iSeries Navigator en el depurador</p>	<p>Para ejecutar iSeries Navigator y observar el conector de ejemplo en acción, siga estos pasos.</p> <ol style="list-style-type: none"> 1. En Developer Studio, abra el menú Generar y seleccione Depurar → Continuar. 2. En la solicitud que aparece, escriba la vía de acceso totalmente calificada del ejecutable de iSeries Navigator en el directorio de instalación de iSeries Access para Windows de la estación de trabajo. La vía de acceso será C:\ARCHIVOS DE PROGRAMA\IBM\CLIENT ACCESS\CWBUNNAV.EXE o algo similar. 3. Pulse Aceptar. Se abrirá la ventana principal de iSeries Navigator. 4. Dado que acaba de registrar un nuevo conector de Navigator, un diálogo de iSeries Navigator le solicitará que lleve a cabo una exploración para localizar el nuevo conector. 5. Una vez que finalice el indicador de progreso, pulse Aceptar en el diálogo que se muestra. 6. Después de que se renueve la ventana de iSeries Navigator, aparece una carpeta nueva (carpeta de ejemplo de terceros) en la jerarquía bajo el servidor iSeries seleccionado inicialmente. Ahora puede interactuar con el conector en iSeries Navigator y observar su comportamiento en el depurador.

Conceptos relacionados

Página Web del Juego de herramientas del programador - Conectores de iSeries Navigator

Configurar conectores Visual Basic de ejemplo

El conector Visual Basic (VB) de ejemplo añade una carpeta a la jerarquía de iSeries Navigator que proporciona una lista de bibliotecas i5/OS e ilustra cómo implementar las propiedades y acciones en esos 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 conector VB de ejemplo para obtener una descripción detallada de todos los archivos incluidos con el conector VB.

Para utilizar el conector VB de ejemplo en el PC, debe seguir el procedimiento siguiente:

Bajar el conector VB	Bajar el archivo ejecutable vbopnav.exe. Cuando ejecute el archivo, se extraerán todos los archivos asociados al conector. Cree un directorio nuevo, c:\VBSample, y copie en él todos los archivos. Si crea un directorio distinto, tendrá que modificar el archivo del registro para especificar la ubicación correcta del conector.
Crear el proyecto VB	Abra vbsample.vpb en Visual Basic. En el diálogo de referencias, seleccione IBM iSeries Access para Windows ActiveX Object Library e iSeries Navigator Visual Basic Plug-in Support . Nota: Si alguna de estas referencias no aparece en el diálogo Referencias, seleccione Examinar y busque cwbx.dll y cwbnvbi.dll en el directorio Shared de iSeries Access para Windows. La biblioteca de objetos ActiveX de IBM iSeries Access (IBM iSeries Access ActiveX Object Library) contiene objetos de automatización OLE que la aplicación de ejemplo necesita para efectuar llamadas de mandato remotas al servidor iSeries. El soporte de conectores Visual Basic de iSeries Navigator (iSeries Navigator Visual Basic Plug-in Support) contiene clases e interfaces necesarias para crear un directorio para el conector Visual Basic.
Crear la DLL servidora ActiveX	Seleccione Generar en el menú de archivo de Visual Basic para crear la DLL. Si no se realiza el enlace y la compilación, localice los errores y corríjalos y, a continuación, vuelva a crear la DLL.
Crear la biblioteca de recursos	<ol style="list-style-type: none"> 1. Abra Microsoft Developer Studio, abra el menú Archivo, seleccione Abrir área de trabajo y después seleccione el directorio VBSample\win32. 2. En Tipo de archivo:, especifique Archivos MAKE (*.mak). 3. Seleccione vbsmpmri.mak y pulse Abrir. 4. Abra el menú Generar y seleccione Regenerar todo para compilar y enlazar la DLL. <p>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.</p>
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 del registro y sustituya todas las apariciones de "c:\VBSample\" por la vía totalmente calificada de acceso al código del conector. Debe utilizar barras inclinadas invertidas dobles en la vía de acceso.
Ejecutar el conector en iSeries Navigator	Inicie iSeries Navigator y pulse el botón del ratón en el signo "+" que figura junto a un servidor iSeries para expandir el árbol. iSeries Navigator detectará los cambios efectuados en el registro y le solicitará que explore el servidor iSeries para verificar que pueda dar soporte al nuevo conector. Tras finalizar la exploración, iSeries Navigator visualizará el nuevo conector en la jerarquía de árbol.

Conceptos relacionados

"Directorio de archivos de los conectores VB de ejemplo"

Tareas relacionadas

Bajar el archivo ejecutable vbopnav.exe

Directorio de archivos de los conectores VB de ejemplo

Las tablas siguientes describen todos los archivos incluidos junto con el conector VB de ejemplo.

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

Formularios de VB	Descripción
authority.frm	Formulario para establecer autorización
delete.frm	Formulario para confirmar supresión

Formularios de VB	Descripción
propsht.frm	Formulario de hoja de propiedades
sysstat.frm	Formulario de estado del sistema
wizard.frm	Formulario del asistente Crear biblioteca nueva

Módulos de VB	Descripción
global.bas	Declaraciones globales

Módulos de clase de VB	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

Binarios de VB	Descripción
authority.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 los recursos traducibles del conector
setup.ini	Información de instalación para los 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 iSeries 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

“Configurar conectores Visual Basic de ejemplo” en la página 15

El conector Visual Basic (VB) de ejemplo añade una carpeta a la jerarquía de iSeries Navigator que proporciona una lista de bibliotecas i5/OS e ilustra cómo implementar las propiedades y acciones en esos objetos de biblioteca.

“Archivo de instalación de MRI” en la página 13

Proporciona al programa de instalación selectiva de iSeries Access para Windows la información que necesita para instalar los recursos dependientes del entorno nacional que están asociados a un conector de iSeries Navigator en un PC cliente.

“Archivo setup.ini” en la página 7

El archivo setup.ini del conector proporciona al asistente de instalación la información necesaria para instalar un conector de iSeries 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 el conector Java de ejemplo

El conector Java de ejemplo trabaja con las colas de mensajes de QUSRSYS en un servidor iSeries determinado.

El primer conector permite ver, añadir y suprimir mensajes de la cola de mensajes por omisión, la que se denomina igual que el ID de usuario de iSeries. El segundo conector añade soporte para varias colas de mensajes. Por último, 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 el conector Java de ejemplo, siga este procedimiento:

Bajar los conectores Java de ejemplo	Bajar el archivo ejecutable jvopnav.exe. Cuando ejecute este archivo, se extraerán todos los archivos indicados anteriormente. Debe permitir al ejecutable instalar los archivos en el directorio por omisión: jvopnav\com\ibm\as400\opnav.
Identificar el conector en iSeries Navigator	<ol style="list-style-type: none">1. Edite el archivo MsgQueueSampleX.reg en jvopnav\com\ibm\as400\opnav\MsgQueueSampleX. (X=1, 2 ó 3, según el ejemplo que instale.)2. Localice las líneas: "NLS"="c:\jvopnav\win32\mri\MessageQueuesMRI.dll" y "JavaPath"="c:\jvopnav"3. Sustituya "c:\\" por la vía de acceso totalmente calificada al directorio jvopnav en el PC. Debe utilizar barras inclinadas invertidas dobles en la vía de acceso.4. Guarde los cambios y efectúe una doble pulsación en el archivo del registro.

Ejecutar el conector Java de ejemplo	<ol style="list-style-type: none"> 1. Inicie iSeries Navigator y pulse el botón del ratón en el signo "+" que figura junto a un servidor iSeries para expandir el árbol. 2. iSeries Navigator detectará los cambios efectuados en el registro y le solicitará que explore el servidor iSeries para verificar que pueda dar soporte al nuevo conector. 3. Pulse Explorar ahora. 4. iSeries Navigator explorará el servidor iSeries. Cuando finalice, visualizará una carpeta nueva en el árbol de jerarquía (ejemplo de cola de mensajes Java 1, 2 ó 3). 5. Efectúe una doble pulsación en la carpeta nueva. 6. El primer conector de ejemplo visualizará el contenido de la cola de mensajes por omisión de QUSRSYS en el servidor iSeries. Los ejemplos segundo y tercero visualizarán una lista de colas de mensajes. 7. Añada un nuevo mensaje pulsando el botón derecho del ratón sobre la carpeta de la cola de mensajes y seleccionando Nuevo → Mensaje. 8. El conector visualiza un diálogo PDML que permite especificar el texto del mensaje. 9. Suprima un mensaje pulsando el botón derecho del ratón sobre un mensaje y seleccionando Suprimir. También puede hacerlo desde la barra de herramientas. 10. Si está utilizando el tercer conector de ejemplo, puede seleccionar un mensaje, arrastrarlo a otra cola y soltarlo. 11. El conector moverá el mensaje a la otra cola.
--------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Conceptos relacionados

"Directorio de archivos de los conectores Java de ejemplo"

Información relacionada

Bajar el archivo ejecutable jvopnav.exe

Directorio de archivos de los conectores Java de ejemplo

Las tablas siguientes describen todos los archivos incluidos con los conectores 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`. Empiece por el archivo `Package-com.ibm.as400.opnav.MsgQueueSample1.html`. El nombre de paquete del ejemplo es `com.ibm.as400.opnav.MsgQueueSample1`. Todos los nombres de clase tienen el prefijo "Mq" para distinguirlos de las clases con nombres similares de otros paquetes.

Archivos de código fuente Java; primer conector de ejemplo	Descripción
MqMessagesListManager.java	ListManager para listas de mensajes.
MqActionsManager.java	Implementación de ActionsManager que maneja todos los menús de contexto para el conector.
MqMessageQueue.java	Colección de objetos de mensaje del servidor iSeries en una cola de mensajes.
MqMessage.java	Objeto que representa un mensaje del servidor iSeries.
MqNewMessageBean.java	Implementación de DataBean de interfaz de usuario para el diálogo de nuevo mensaje.
MqDeleteMessageBean.java	Implementación de DataBean de interfaz de usuario para el diálogo de confirmación de supresión.

Archivos de código fuente Java; segundo conector de ejemplo	Descripción
MqListManager.java	Implementación de ListManager maestro para el conector.
MqMessageQueuesListManager.java	ListManager esclavo para listas de colas de mensajes.
MqMessagesListManager.java	ListManager esclavo para listas de mensajes.
MqActionsManager.java	Implementación de ActionsManager que maneja todos los menús de contexto para el conector.
MqMessageQueueList.java	Colección de colas de mensajes del servidor iSeries.
MqMessageQueue.java	Colección de objetos de mensaje del servidor iSeries en una cola determinada.
MqMessage.java	Objeto que representa un mensaje del servidor iSeries.
MqNewMessageBean.java	Implementación de DataBean de interfaz de usuario para el diálogo de nuevo mensaje.
MqDeleteMessageBean.java	Implementación de DataBean de interfaz de usuario para el diálogo de confirmación de supresión.

Archivos de código fuente Java; tercer conector de ejemplo	Descripción
MqListManager.java	Implementación de ListManager maestro para el conector.
MqMessageQueuesListManager.java	ListManager esclavo para listas de colas de mensajes.
MqMessagesListManager.java	ListManager esclavo para listas de mensajes.
MqActionsManager.java	Implementación de ActionsManager que maneja todos los menús de contexto para el conector.
MqDropTargetManager.java	Implementación de DropTargetManager que maneja las operaciones de arrastrar y soltar para el conector.
MqMessageQueueList.java	Colección de colas de mensajes del servidor iSeries.
MqMessageQueue.java	Colección de objetos de mensaje del servidor iSeries en una cola determinada.
MqMessage.java	Objeto que representa un mensaje del servidor iSeries.
MqNewMessageBean.java	Implementación de DataBean de interfaz de usuario para el diálogo de nuevo mensaje.
MqDeleteMessageBean.java	Implementación de DataBean de interfaz de usuario para el diálogo de confirmación de supresión.

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 para el diálogo Mensaje nuevo.
IDD_MSGQ_CONFIRM_DELETE.html	Esqueleto de la ayuda en línea para el diálogo Confirmar supresión.

Archivos serializados	Descripción
IDD_MSGQ_ADD.pdml.ser	Definición de panel serializada para el diálogo de nuevo mensaje.

Archivos serializados	Descripción
IDD_MSGQ_CONFIRM_DELETE.pdml.ser	Definición de panel serializada para el diálogo de confirmación de 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 del registro de Windows que indican a iSeries Navigator que este conector existe e identifican sus clases de implementación de interfaz Java.
MsgQueueSample1install.reg MsgQueueSample2install.reg MsgQueueSample3install.reg	Archivo del registro, que se distribuye con la versión de minorista del conector. Windows no puede leer directamente esta versión del archivo del registro. Contiene variables de sustitución que representan la vía de acceso del directorio de instalación de iSeries Access para Windows. Cuando el usuario invoca el programa de instalación selectiva de iSeries Access para Windows con el fin de instalar el conector desde el servidor iSeries, el programa de instalación selectiva lee este archivo del registro, rellena las vías de acceso de directorio correctas y escribe las entradas en el registro de la máquina del usuario. Por consiguiente, las entradas de este archivo deben mantenerse sincronizadas con el archivo del registro utilizado en el entorno de desarrollo.

Conceptos relacionados

“Configurar el conector Java de ejemplo” en la página 18

El conector Java de ejemplo trabaja con las colas de mensajes de QUSRSYS en un servidor iSeries determinado.

Información de consulta para la programación de conectores

iSeries Navigator maneja los conectores de cada uno de los lenguajes de programación de forma distinta.

Puede utilizar los temas siguientes para obtener información sobre el flujo de control en iSeries Navigator para cada tipo de conector, así como información de consulta específica en relación con las distintas interfaces para 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++

Utilice los temas siguientes para obtener información sobre el flujo de control en iSeries Navigator para C++.

Estructura de iSeries Navigator y flujo de control de los conectores C++

La arquitectura interna del producto iSeries Navigator está pensada para servir de punto de integración para una interfaz ampliable de una gran gama de operaciones para el servidor iSeries.

Cada uno de los componentes funcionales de la interfaz está empaquetado como una DLL de servidor ActiveX. iSeries Navigator utiliza la tecnología COM (Component Object Model) de Microsoft para activar únicamente las implementaciones de componentes que son necesarias en ese momento para atender una petición del usuario. Esto evita el problema de tener que cargar todo el producto en el momento del arranque, con lo que se consumen la mayoría de los recursos de Windows, y se compromete el rendimiento de todo el sistema. Varios servidores pueden registrar su petición de añadir elementos de menú y diálogos a un tipo de objeto determinado en la jerarquía de iSeries Navigator.

Los conectores actúan respondiendo a las llamadas de método procedentes de iSeries Navigator que se generan en respuesta a las acciones del usuario. Por ejemplo, cuando un usuario pulsa el botón derecho del ratón sobre un objeto de la jerarquía de iSeries Navigator, iSeries Navigator crea un menú de contexto para el objeto y visualiza el menú en la pantalla. Navigator obtiene los elementos de menú efectuando una llamada a cada conector que ha registrado su intención de proporcionar elementos de menú de contexto 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 iSeries Navigator puede llamar para llevar a cabo 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 determinada puede implementar tantas interfaces como elija 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 iSeries Navigator para C++

Las funciones implementadas por un conector se agrupan lógicamente en **interfaces COM (Component Object Model)**.

Una interfaz es un conjunto de métodos relacionados lógicamente en una clase a la que iSeries Navigator puede llamar para llevar a cabo una función específica. Un conector puede implementar una interfaz COM o varias, según el tipo de función que desee proporcionar el desarrollador. Por ejemplo, cuando un usuario pulsa el botón derecho del ratón sobre un objeto de la jerarquía de árbol, iSeries Navigator crea un menú de contexto para el objeto y visualiza el menú en la pantalla. Navigator obtiene los elementos de menú efectuando una llamada a cada conector que ha registrado su deseo de proporcionar elementos de menú de contexto para el tipo de objeto seleccionado. Los conectores pasan sus elementos de menú a Navigator cuando este llama a su implementación del método **QueryContextMenu** en la **interfaz IContextMenu**.

Interfaz	Método	Descripción
IContextMenu	QueryContextMenu	Proporciona elementos de menú de contexto cuando un usuario pulsa 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 (en color gris).
	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.

Interfaz	Método	Descripción
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.
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.

IA4PropSheetNotify envía una notificación a las 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 de iSeries cuyas propiedades están visualizándose ya existe o se está definiendo y si los cambios deben guardarse o descartarse.

IA4HierarchyFolder permite a un conector añadir nuevas carpetas a la jerarquía de iSeries Navigator. Esta interfaz tiene por finalidad proporcionar los datos utilizados para rellenar el contenido de una carpeta nueva añadida por el conector a la jerarquía de iSeries 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.

Consulte los temas siguientes para obtener más información:

Descripción de la interfaz IA4HierarchyFolder:

La interfaz IA4HierarchyFolder describe un conjunto de funciones que el proveedor de software independiente implementará. IA4HierarchyFolder es una interfaz COM (Component Object Model) definida por IBM para permitir a terceros añadir nuevas carpetas y objetos a la jerarquía de iSeries Navigator.

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

El programa iSeries Navigator llama a los métodos de la interfaz IA4HierarchyFolder cada vez que necesita comunicarse con el conector de terceros. El objetivo principal de la interfaz consiste en proporcionar a iSeries Navigator datos de lista que se utilizarán al visualizar el contenido de una carpeta que el conector ha definido. Los métodos de la interfaz permiten a iSeries Navigator enlazar con una carpeta de terceros concreta y mostrar 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 barra de herramientas personalizada a la carpeta.

La implementación de la interfaz normalmente se compila y enlaza en una DLL (biblioteca de enlace dinámico) servidora ActiveX. Navigator conoce la existencia de la nueva DLL por medio de las entradas del 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 deben insertarse las nuevas carpetas. Posteriormente Navigator carga la DLL en el momento adecuado y efectúa llamadas a los métodos de la interfaz IA4HierarchyFolder a medida que 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:

Un identificador de elemento, o una entidad de datos, identifica la totalidad de las carpetas y los objetos del espacio de nombres de Windows. Los identificadores de elementos son como los nombres de archivo de un sistema de archivos jerárquico. De hecho el espacio de nombres de Windows es un espacio de nombres jerárquico con su raíz en el Escritorio.

Un identificador de elemento está formado por un campo de cuenta de dos bytes seguido de una estructura de datos binarios de longitud variable (vea 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.

iSeries Navigator utiliza identificadores de elemento que siguen la siguiente estructura específica que IA4HierarchyFolder::ItemAt debe devolver.

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

siendo

<cb> el tamaño en bytes del identificador de elemento, incluido el propio campo de cuenta

<nombre elemento> el nombre traducido del objeto, adecuado para visualizarse 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.

Enlace con cualquiera de las siguientes especificaciones de IA4HierarchyFolder:

IA4HierarchyFolder::Activate: **Objetivo**

Coloca la instancia de IA4HierarchyFolder en estado activo. Esta función también lleva a cabo el proceso necesario para preparar una carpeta para su visualización, lo que incluye efectuar una llamada al servidor iSeries para preparar la antememoria de objetos de la 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 la ejecución es satisfactoria o E_FAIL si no se puede obtener el contenido de la carpeta.

Comentarios

iSeries Navigator efectúa una llamada a esta función la primera vez que un usuario selecciona o expande una carpeta. Se efectúa una nueva llamada a esta función, tras una llamada a la función de cierre (Close), cuando el usuario ha solicitado una renovación del contenido de la carpeta.

Pueden realizarse llamadas a esta función en otras ocasiones cuando deba volver a establecerse un puntero a la interfaz de la carpeta. Un ejemplo sería cuando el usuario selecciona una carpeta por segunda vez. Después de seleccionar otra carpeta, basta con que la función devuelva TRUE si el proceso asociado ya se ha efectuado.

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: **Objetivo**

Devuelve una instancia de IA4HierarchyFolder que corresponde a una carpeta determinada de la jerarquía de iSeries Navigator. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

hwnd Handle de la ventana de vista que visualizará la lista que puede ser un control de lista o árbol. Un componente utilizará este handle para determinar si ya hay almacenada en la antememoria del cliente una lista de objetos para esta vista.

pidl Puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) que identifica de forma exclusiva la carpeta que se mostrará.

riid Identificador de la interfaz que se devolverá. Este parámetro apunta al identificador de interfaz IID_IA4HierarchyFolder.

ppvOut

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

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o E_FAIL si se produce un error general.

Comentarios

Si ya existe una instancia de IA4HierarchyFolder para la carpeta especificada, esta función de miembro debe devolver la instancia almacenada en la antememoria en lugar de crear una instancia e inicializar una instancia aparte. Sin embargo, si el handle de 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: **Objetivo**

Se la llama para visualizar un mensaje de error al usuario final cada vez que Activate devuelve un error. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

Ninguno.

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o E_FAIL si no hay ningún mensaje para visualizar.

Comentarios

Ninguno.

IA4HierarchyFolder::GetAttributesOf: **Objetivo**

Devuelve los atributos de una carpeta determinada de la jerarquía de iSeries 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 Puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) que identifica de forma exclusiva el objeto cuyos atributos se recuperarán.

ulfInOut

Atributos del objeto devueltos. En la entrada, este parámetro se establecerá para indicar qué atributos de objeto se recuperarán.

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o E_FAIL si no se puede localizar los atributos del objeto.

Comentarios

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

iSeries Navigator efectúa repetidas llamadas a esta función al rellenar una vista de árbol o lista. Por consiguiente deben evitarse las operaciones de larga ejecución.

IA4HierarchyFolder::GetColumnDataItem: **Objetivo**

Devuelve un campo de datos para una carpeta o un objeto que se visualizará en una columna de la vista de lista de iSeries Navigator. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

pidl Puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) que identifica de forma exclusiva el objeto cuyos datos de columna se obtendrán.

lParam

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

lpszColumnData

Dirección del almacenamiento intermedio que recibirá la serie de datos terminada en nulo.

cchMax

Tamaño del almacenamiento intermedio que recibirá la serie de datos terminada en nulo.

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o E_FAIL si no es posible recuperar los datos de columna.

Comentarios

iSeries Navigator efectúa repetidas llamadas a esta función al rellenar una vista de lista. Por consiguiente deben evitarse las operaciones de larga ejecución.

IA4HierarchyFolder::GetColumnInfo: **Objetivo**

Devuelve una estructura de datos que describe las columnas necesarias para visualizar el contenido de una carpeta determinada en una vista de detalles. 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 *A4hyfColumnInfo*, 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 la ejecución es satisfactoria o E_NOTIMPL si no se puede implementar la función.

Comentarios

iSeries Navigator efectúa una llamada a esta función después de que vuelva la llamada a la función de apertura (Open) para crear las cabeceras de columna de una vista de detalles.

Si esta función no se implementa, iSeries Navigator insertará dos columnas: Nombre y Descripción. *GetColumnInfo* debe poder devolver datos para estos dos campos, que se identifican con los valores enteros 0 y 1 respectivamente.

Utilice la interfaz IMalloc de Windows para asignar memoria para las estructuras devueltas. iSeries Navigator se encargará de suprimir esta memoria.

IA4HierarchyFolder::GetIconIndexOf: **Objetivo**

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

Sintaxis

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

Parámetros

pidl Puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) que identifica de forma exclusiva el objeto cuyo índice de icono se recuperará.

uFlags Este parámetro puede ser cero o contener el valor GIL_OPENICON, que indica que el icono que debe proporcionarse es una carpeta abierta. GIL_OPENICON se define en el archivo include de Windows SHLOBJ.H.

piIndex

Puntero a un entero que recibe el índice de icono.

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o E_FAIL si no se puede determinar el índice.

Comentarios

iSeries Navigator efectúa repetidas llamadas a esta función al rellenar una vista de árbol o lista. Por consiguiente deben evitarse las operaciones de larga ejecución.

IA4HierarchyFolder::GetItemCount: **Objetivo**

Devuelve el número total de objetos incluidos en una carpeta determinada de la jerarquía de iSeries Navigator. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

pCount

Puntero a un entero largo que recibirá el número de elementos de 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 todavía está en proceso de creación; en esta situación el número de elementos representa el número de elementos de la lista creada parcialmente. Devuelve A4HYF_E_LISTDATAERROR si se encuentra un error al crear la lista. En esta situación el número de elementos representa únicamente los elementos que ya están almacenados en la antememoria en el cliente.

Comentarios

Tras un retorno satisfactorio de Activate, iSeries Navigator efectúa una llamada a esta función para obtener el número de objetos de la carpeta que se va a rellenar. A continuación de la llamada a esta función, iSeries Navigator efectúa repetidas llamadas a ItemAt para obtener los identificadores de elemento de los objetos de la carpeta.

En el caso de listas sumamente grandes, puede elegir volver de Activate antes de que se almacene toda la lista en la antememoria en el cliente. En estas circunstancias, GetItemCount devolverá A4HYF_OK_LISTNOTCOMPLETE. A partir de ese momento, iSeries Navigator efectuará una llamada a GetItemCount cada 10 segundos hasta que se devuelva A4HYF_OK_LISTCOMPLETE o A4HYF_E_LISTDATAERROR.

IA4HierarchyFolder::GetToolBarInfo: **Objetivo**

Devuelve una estructura que describe la barra de herramientas personalizada asociada a la carpeta especificada en la jerarquía de iSeries Navigator. Es una función de miembro obligatoria.

Sintaxis

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

Parámetros

pidl Puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) que identifica de forma exclusiva el objeto para el que 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 CWBA4HYF.H.

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o E_NOTIMPL si elige no implementar la función.

Comentarios

Se llama a esta función cada vez que un usuario selecciona una carpeta o un objeto que pertenece a un conector de iSeries Navigator.

Utilice la interfaz IMalloc de Windows para asignar memoria para la estructura devuelta. iSeries Navigator se encargará de suprimir esta memoria.

Si esta función de miembro no se implementa, se utilizará la barra de herramientas de iSeries Navigator por omisión. Esta barra de herramientas contiene Copiar, Pegar, Suprimir, Propiedades, botones para las cuatro vistas de lista y Renovar. iSeries Navigator efectúa una llamada a la implementación de IContextMenu::GetCommandString (con el distintivo GCS_VALIDATE establecido) que se encuentra en el producto para descubrir qué botón de la barra de herramientas debe habilitarse para los objetos.

IA4HierarchyFolder::GetListObject: **Objetivo**

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

Sintaxis

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

Parámetros

lpszObjectName

Nombre totalmente calificado del objeto 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 la ejecución es satisfactoria o E_NOTIMPL si elige no implementar la función.

Comentarios

Las llamadas a esta función se producen cuando el código del conector efectúa una llamada a **cwbUN_GetListObjectFromName** o **cwbUN_GetListObjectFromPidl** para obtener un objeto de "proxy" del que se ha creado una instancia con el método Activate. El conector utiliza este objeto de proxy para acceder a los datos del servidor iSeries o para llevar a cabo acciones en el servidor iSeries. Dado que la implementación de IA4HierarchyFolder mantiene la antememoria de los objetos de proxy, conviene que el programa de llamada no intente suprimir el objeto.

IA4HierarchyFolder::ItemAt: **Objetivo**

Devuelve una estructura SHITEMID (identificador de elemento) para el objeto de carpeta en la posición especificada en la lista de contenido de la 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 recibirá el identificador de elemento solicitado.

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o E_FAIL si el elemento no está disponible. Devuelve E_OUTOFMEMORY si no hay suficiente memoria disponible para el identificador de elemento.

Comentarios

iSeries Navigator efectúa repetidas llamadas a esta función para rellenar una carpeta en tiempo real. Por consiguiente deben evitarse las operaciones de larga ejecución. Consulte en CWBA4HYF.H el formato de los identificadores de elemento de iSeries Navigator. Utilice la interfaz IMalloc de Windows para asignar memoria para el identificador de elemento.

IA4HierarchyFolder::ProcessTerminating: **Objetivo**

Se llama a esta función cuando el usuario cierra la ventana de iSeries Navigator para permitir al conector guardar los datos persistentes. 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: **Objetivo**

Destruye los objetos de la carpeta almacenados en la antememoria y vuelve a crear la antememoria con los datos nuevos obtenidos del servidor iSeries. 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

iSeries Navigator efectúa una llamada a esta función cada vez que lleva a cabo una renovación global de la ventana principal de iSeries 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 definida por IBM para permitir a terceros añadir nuevas páginas de propiedades a cualquier hoja de propiedades que iSeries Navigator defina para un usuario de servidor iSeries.

El programa iSeries Navigator llama a los métodos de la interfaz IA4PropSheetNotify cada vez que necesita comunicarse con el conector de terceros. El objetivo de la interfaz consiste en proporcionar una notificación cuando se cierra el diálogo Propiedades principal de un usuario de iSeries. 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. Navigator conoce la existencia de la nueva DLL por medio de las entradas del registro de Windows. Estas entradas especifican la ubicación de la DLL en el PC del usuario. Posteriormente Navigator carga la DLL en el momento adecuado y efectúa llamadas a los métodos de la interfaz IA4PropSheetNotify a medida que es 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 que son necesarias al añadir páginas de propiedades adicionales a una de las hojas de propiedades de usuarios y grupos.

Estas notificaciones son necesarias ya que la creación y destrucción de hojas de propiedades de Usuarios y Grupos puede producirse muchas veces antes de que el usuario pulse **Aceptar** en el diálogo Propiedades principal. IA4PropSheetNotify informa a la implementación de IShellPropSheetExt cuándo deben guardarse los cambios efectuados por el usuario.

iSeries Navigator conoce la existencia de una implementación de IA4PropSheetNotify por medio de las entradas normales del registro que están definidas para los conectores de iSeries Navigator. Además,

cuando se registra un manejador de hojas de propiedades del componente Usuarios y Grupos, se da soporte a un valor especial del registro que permite al conector especificar a qué hoja de propiedades desea añadir páginas.

Enlace con cualquiera de las siguientes especificaciones de la interfaz IA4PropSheetNotify:

Conceptos relacionados

“Páginas de propiedades para un manejador de hojas de propiedades” en la página 91
Las clases de la biblioteca Microsoft Foundation Classes no pueden utilizarse para crear páginas de propiedades para un manejador de hojas de propiedades. No obstante, IBM suministra **CExtPropertyPage**, que puede emplearse en lugar de la clase **CPropertyPage** de MFC.

IA4PropSheetNotify::InformUserState: **Objetivo**

Se le llama inmediatamente después de crear la instancia **IShellPropSheetExt** para informar a la implementación de si este usuario ya existe en el servidor **iSeries** o se crea por primera vez.

Sintaxis

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

Parámetros

wUserState

Estado actual del usuario de **iSeries**. Proporciona uno de tres valores posibles, que se excluyen mutuamente:

IUS_NEWUSER

Se crea un usuario según los atributos proporcionados por el usuario de **iSeries Navigator**.

IUS_NEWUSERBASEDON

Se crea un usuario según los atributos de un usuario de **iSeries** ya existente.

IUS_USEREXISTS

El usuario ya existe en el **iSeries**.

Códigos de retorno

Devuelve **NOERROR** si la ejecución es satisfactoria o **E_FAIL** si se produce un error general.

Comentarios

Ninguno.

IA4PropSheetNotify::ApplyChanges: **Objetivo**

Se le llama para informar a la implementación de que deben guardarse los datos pertenecientes al usuario.

Sintaxis

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

Parámetros

pszNewUserName

Proporciona el nombre del nuevo usuario de iSeries si se crea el usuario por primera vez, como por ejemplo si InformUserState especifica un valor distinto a IUS_USEREXISTS.

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o E_FAIL si se produce un error general.

Comentarios

Ninguno.

IA4PropSheetNotify::GetErrorMessage: **Objetivo**

Se le llama cuando se devuelven errores en ApplyChanges para recuperar el texto del mensaje de error de la implementación.

Sintaxis

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

Parámetros

pszErrMsg

Dirección del almacenamiento intermedio que recibirá el mensaje de error terminado en nulo.

cchMax

Tamaño del almacenamiento intermedio que recibirá el mensaje de error terminado en nulo.

Códigos de retorno

Devuelve NOERROR si la ejecución es satisfactoria o E_FAIL si no se puede recuperar el texto del mensaje o si el texto del mensaje es demasiado grande para poder ubicarse en el almacenamiento intermedio.

Comentarios

Ninguno.

Las API de iSeries Navigator

Las API de iSeries Navigator ayudan a los desarrolladores de conectores a obtener y gestionar determinados tipos de información global.

Lista de las API de iSeries Navigator:

Se proporcionan las API de iSeries Navigator en una lista, agrupadas por su función.

Función	API de iSeries Navigator
Valores del sistema: esta API permite al desarrollador de conectores obtener el valor actual de un valor del sistema de iSeries.	"cwbUN_GetSystemValue" en la página 36

Función	API de iSeries Navigator
<p>Handles del sistema: Estas API permiten al desarrollador de conectores obtener y liberar el valor actual de un handle de objeto del sistema iSeries que contiene las propiedades de conexión, incluidos los valores de la capa de sockets segura (SSL) que hay que utilizar para el sistema iSeries especificado.</p>	<p>“cwbUN_GetSystemHandle” en la página 37</p> <p>“cwbUN_ReleaseSystemHandle” en la página 38</p>
<p>Validación de la entrada del usuario: Estas API permiten al desarrollador de conectores comprobar si el usuario actual tiene autorización sobre un objeto iSeries determinado. Estas API también permiten al desarrollador determinar si el usuario tiene una autorización especial o varias.</p>	<p>“cwbUN_CheckObjectAuthority” en la página 39</p> <p>“cwbUN_CheckSpecialAuthority” en la página 40</p>
<p>Comprobación de la autorización del usuario: Esta API permite al desarrollador de conectores comprobar si determinados tipos de series proporcionadas por el usuario son válidos antes de transmitirlos al servidor iSeries.</p>	<p>“cwbUN_CheckAS400Name” en la página 40</p>
<p>Atributos del perfil de usuario: Esta API permite al desarrollador de conectores obtener el valor de cualquiera de los atributos de perfil de usuario para el usuario actual de iSeries Navigator.</p>	<p>“cwbUN_GetUserAttribute” en la página 41</p>
<p>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.</p>	<p>“cwbUN_ConvertPidlToString” en la página 42</p> <p>“cwbUN_GetDisplayNameFromItemId” en la página 43</p> <p>“cwbUN_GetDisplayNameFromName” en la página 43</p> <p>“cwbUN_GetDisplayPathFromName” en la página 44</p> <p>“cwbUN_GetIndexFromItemId” en la página 45</p> <p>“cwbUN_GetIndexFromName” en la página 45</p> <p>“cwbUN_GetIndexFromPidl” en la página 46</p> <p>“cwbUN_GetListObject” en la página 46</p> <p>“cwbUN_GetParentFolderNameFromName” en la página 47</p> <p>“cwbUN_GetParentFolderPathFromName” en la página 47</p> <p>“cwbUN_GetParentFolderPidl” en la página 48</p> <p>“cwbUN_GetSystemNameFromName” en la página 49</p> <p>“cwbUN_GetSystemNameFromPidl” en la página 49</p> <p>“cwbUN_GetSystemNameFromPidl” en la página 49</p> <p>“cwbUN_GetTypeFromName” en la página 50</p> <p>“cwbUN_GetTypeFromPidl” en la página 51</p>

Función	API de iSeries Navigator
<p>Renovación de la ventana de iSeries Navigator: Tras la finalización de una operación en nombre del usuario, estas API habilitan la ejecución de una petición del conector para renovar las vistas de árbol y lista o para colocar un mensaje en la barra de estado de Navigator.</p>	<p>“cwbUN_RefreshAll” en la página 51</p> <p>“cwbUN_RefreshList” en la página 52</p> <p>“cwbUN_RefreshListItems” en la página 52</p> <p>“cwbUN_UpdateStatusBar” en la página 53</p>
<p>Conexiones ODBC: Estas API permiten al desarrollador de conectores reutilizar y finalizar el handle de una conexión ODBC que ya se ha obtenido mediante el componente Base de datos de iSeries Navigator.</p>	<p>“cwbUN_GetODBCConnection” en la página 53</p> <p>“cwbUN_EndODBCConnections” en la página 54</p>
<p>Acceso a los iconos de iSeries Navigator: Estas API permiten al desarrollador de conectores acceder a las listas de imágenes de icono de los objetos que aparecen en la jerarquía de objetos de Navigator.</p>	<p>“cwbUN_GetIconIndex” en la página 54</p> <p>“cwbUN_GetSharedImageList” en la página 55</p>
<p>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. Función administrable es cualquier función cuyo uso se puede controlar mediante el subcomponente Administración de Aplicaciones de iSeries Navigator.</p>	<p>“cwbUN_GetAdminValue” en la página 56</p> <p>“cwbUN_GetAdminValueEx” en la página 57</p> <p>“cwbUN_GetAdminCacheState” en la página 58</p> <p>“cwbUN_GetAdminCacheStateEx” en la página 59</p>
<p>Instalación: Esta API permite al desarrollador de conectores determinar si está instalado un subcomponente de iSeries Navigator.</p>	<p>“cwbUN_IsSubcomponentInstalled” en la página 59</p>
<p>Servicios de directorio: Estas API facilitan información sobre el servidor LDAP (Servicios de directorio) en un sistema iSeries y las funciones para conectarse al servidor. Las funciones de conexión permiten conectarse a un servidor utilizando información (nombre distinguido, contraseña, etc.) almacenada en la antememoria por iSeries Access para Windows. Las funciones de conexión utilizan el cliente LDAP suministrado con iSeries Access para Windows (LDAP.LIB y LDAP.DLL) y, por consiguiente, requieren que la aplicación utilice ese cliente.</p> <p>Las funciones que utilizan series están disponibles en versiones ANSI y Unicode.</p> <p>Las funciones que devuelven nombres distinguidos y otras series que se utilizan con las API de cliente LDAP también se proporcionan en una versión UTF-8 para poder emplearse con servidores LDAP Versión 3.</p>	<p>“cwbUN_OpenLocalLdapServer” en la página 60</p> <p>“cwbUN_FreeLocalLdapServer” en la página 61</p> <p>“cwbUN_GetLdapSvrPort” en la página 61</p> <p>“cwbUN_GetLdapSvrSuffixCount” en la página 62</p> <p>“cwbUN_GetLdapSvrSuffixName” en la página 62</p> <p>“cwbUN_OpenLdapPublishing” en la página 64</p> <p>“cwbUN_FreeLdapPublishing” en la página 64</p> <p>“cwbUN_GetLdapPublishCount” en la página 65</p> <p>“cwbUN_GetLdapPublishType” en la página 65</p> <p>“cwbUN_GetLdapPublishServer” en la página 66</p> <p>“cwbUN_GetLdapPublishPort” en la página 67</p> <p>“cwbUN_GetLdapPublishParentDn” en la página 68</p>

cwbUN_GetSystemValue: Objetivo

Devuelve una serie que contiene el valor que corresponde a un valor del sistema iSeries.

Sintaxis

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

Parámetros

const char * szSystemValueId - entrada

Valor numérico que identifica el valor del sistema iSeries que se recuperará. Las definiciones de las constantes de valor del sistema se encuentran en el archivo de cabecera 'CWBA4SVL.H'.

char * szSystemValue - salida

Dirección del almacenamiento intermedio que recibirá la serie de valor del sistema terminada en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá la serie de valor terminada 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 que esta API devuelve **no** es una serie NLS (soporte de idioma nacional) y no se traduce. Por ejemplo, se devolverá '*NONE' en lugar de 'Ninguno'.

cwbUN_GetSystemHandle: Objetivo

Devuelve un handle del sistema que contiene los valores de seguridad (SSL), ID de usuario y contraseña que se utilizarán para este sistema iSeries. El handle del sistema tendrá los valores que se configuraron en iSeries Navigator para el nombre del sistema iSeries 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 requiere un trabajo OS/400 exclusivo para un sistema, debe pasarse NULL o un nombre exclusivo para el nombre de aplicación.

Si una aplicación requiere compartir un trabajo OS/400, todos los llamadores de esta función deben pasar el mismo nombre de aplicación.

Sintaxis

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

Parámetros

char * szSystemName - entrada

Puntero a una serie ASCIIZ que contiene el nombre del sistema para el que desea que se cree un handle del sistema.

char * szAppName - entrada

Puntero a una serie ASCIIZ de 12 caracteres como máximo. Identifica de forma exclusiva la aplicación que compartirá un único handle del sistema.

cwbCO_SysHandle * systemHandle - salida

Puntero al handle del sistema para este nombre de 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

Todas las aplicaciones de terceros que deseen dar soporte a SSL utilizando las API de iSeries Access para Windows deben utilizar esta función. Por ejemplo, todas las API de comunicaciones de iSeries Access para Windows requieren un handle del sistema para dar soporte a 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 handles se liberarán cuando se termine la aplicación iSeries Navigator (cwbunnav.exe).

cwbUN_ReleaseSystemHandle: Objetivo

Libera un handle del sistema que contiene los valores de seguridad (SSL) que se utilizarán para este sistema iSeries^(TM). El handle del sistema se obtiene mediante la función **cwbUN_GetSystemHandle**.

Si el llamador de esta función tiene la última referencia al handle, los recursos del handle se destruirán.

Sintaxis

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

Parámetros

cwbCO_SysHandle * systemHandle - entrada

Puntero al handle del sistema obtenido en 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: Objetivo

Devuelve una indicación de si el usuario de iSeries Navigator tiene autorización sobre un objeto determinado del servidor iSeries.

Sintaxis

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

Parámetros

const char * szObjectPath - entrada

Vía de acceso del objeto iSeries para el que se comprobará la autorización.

const char * szObjectType - entrada

Tipo de objeto iSeries del objeto para el que se va a comprobar la autorización, como por ejemplo *DTAQ.

const char * szAuthorityType - entrada

Autorización sobre el objeto iSeries que se va a comprobar. Los valores posibles son:

Si se va a comprobar más de una autorización, las autorizaciones deben ir concatenadas (por ejemplo, *OBJMGT*OBJEXIST). Es posible especificar hasta once tipos de autorización en una sola llamada. La función únicamente devolverá CWB_OK si el usuario tiene todas las autorizaciones especificadas para el objeto.

const char * szSystemName - entrada

Nombre del sistema iSeries en el que se efectuará 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. La autorización *AUTLMGT solo es válida si 'szObjectType' es *AUTL.

cwbUN_CheckSpecialAuthority: Objetivo

Devuelve una indicación de si el usuario de iSeries Navigator tiene una autorización especial determinada en el servidor iSeries.

Sintaxis

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

Parámetros

const char * szSpecialAuthority - entrada

Autorización especial del servidor iSeries que se va a comprobar. Los valores posibles son:

const char * szSystemName - entrada

Nombre del sistema iSeries en el que se efectuará 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: Objetivo

Devuelve una indicación de si una serie específica es un parámetro de nombre válido en el servidor iSeries.

Sintaxis

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

Parámetros

const char * szAS400Name - entrada

Nombre de iSeries cuya validez se va a comprobar.

const char * szSystemName - entrada

Nombre del sistema iSeries en el que se efectuará 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: Objetivo

Devuelve una serie que contiene el valor de un atributo de perfil de usuario para el usuario actual de iSeries Navigator.

Sintaxis

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

Parámetros

USHORT usAttributeId - entrada

Valor numérico que identifica el valor del atributo de usuario que se recuperará. Las definiciones de las constantes de atributo de usuario se encuentran en el archivo de cabecera 'CWBA4USR.H'.

const char * szSystemName - entrada

Nombre del sistema iSeries del que se recuperará el atributo de usuario.

char * szValue - salida

Dirección del almacenamiento intermedio que recibirá la serie de valor de atributo terminada en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá la serie de valor terminada 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 que esta API devuelve **no** es una serie NLS y no se traduce. Por ejemplo, se devolverá '*NONE' en lugar de 'Ninguno'.

cwbUN_ConvertPidlToString: Objetivo

Convierte una lista de identificadores de elemento de iSeries 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**

Puntero a la estructura ITEMIDLIST (lista de identificadores de elemento) que se va a convertir.

char * szObjectName - salida

Dirección del almacenamiento intermedio que recibirá el nombre de objeto terminado en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá el nombre de 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: Objetivo

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

Identificador de elemento Unity del que se extraerá el nombre de elemento.

char * szItemName - salida

Dirección del almacenamiento intermedio que recibirá el nombre de elemento terminado en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá 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: Objetivo

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

Nombre de objeto Unity del que se extraerá el nombre de elemento.

char * szItemName - salida

Dirección del almacenamiento intermedio que recibirá el nombre de elemento terminado en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá 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: Objetivo

Convierte un nombre de objeto Unity totalmente calificado en un nombre de vía de acceso totalmente calificado adecuado para visualizarse al usuario.

Sintaxis

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

Parámetros

const char * szObjectName - entrada

Nombre de objeto Unity del que se derivará el nombre de vía de acceso.

char * szPathName - salida

Dirección del almacenamiento intermedio que recibirá el nombre de vía de acceso terminado en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá 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: Objetivo

Extrae el campo de índice de elemento de un identificador de elemento Unity.

Sintaxis

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

Parámetros

const char * szItemId - entrada

Identificador de elemento Unity del que se extraerá el índice de elemento.

ULONG* piIndex - salida

Dirección de un entero largo sin signo que recibirá el índice de elemento.

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: Objetivo

Extrae el campo de índice de elemento de un nombre de objeto Unity totalmente calificado.

Sintaxis

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

Parámetros

const char * szObjectName - entrada

Nombre de objeto Unity del que se extraerá el índice de elemento.

ULONG* piIndex - salida

Dirección de un entero largo sin signo que recibirá el índice de elemento.

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: Objetivo

Extrae el campo de índice de elemento de una lista de identificadores de elemento Unity totalmente calificada.

Sintaxis

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

Parámetros

LPCITEMIDLIST pidl - entrada

Puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) de la que se extraerá el índice de elemento.

ULONG* piIndex - salida

Dirección de un entero largo sin signo que recibirá el índice de elemento.

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: Objetivo

Obtiene un puntero al objeto asociado al nombre de objeto de lista especificado.

Sintaxis

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

Parámetros

const char * szFileName - entrada

Nombre de objeto Unity a partir del cual se encontrará y devolverá el puntero de objeto.

LPVOID pListObject - salida

Dirección de un puntero al objeto Unity de petición.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK

Finalización satisfactoria.

Utilización

Ninguna.

cwbUN_GetParentFolderNameFromName: Objetivo

Extrae de un nombre de objeto Unity totalmente calificado el nombre de la carpeta padre de un objeto.

Sintaxis

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

Parámetros

const char * szObjectName - entrada

Nombre de objeto Unity del que se extraerá el nombre de carpeta padre.

char * szParentFolderPath - salida

Dirección del almacenamiento intermedio que recibirá el nombre de carpeta padre terminado en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá el nombre 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_GetParentFolderPathFromName: Objetivo

Dado un nombre de objeto Unity totalmente calificado, 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

Nombre de objeto Unity del que se extraerá el nombre de objeto de la carpeta padre.

char * szParentFolderPath - salida

Dirección del almacenamiento intermedio que recibirá el nombre de objeto de la carpeta padre terminado en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá el nombre de objeto 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_GetParentFolderPid: Objetivo

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

Sintaxis

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

Parámetros

LPCITEMIDLIST pidl - entrada

Puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) de la que se extraerá la lista de identificadores de elemento de la carpeta padre.

LPITEMIDLIST* ppidl - salida

Dirección de un puntero de lista de identificadores de elemento que recibirá la lista de identificadores de elemento de la 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: Objetivo

Extrae el nombre del sistema iSeries 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**

Nombre de objeto Unity del que se extraerá el nombre del sistema.

char * szSystemName - salida

Dirección del almacenamiento intermedio que recibirá el nombre del sistema terminado en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá el nombre del 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: Objetivo

Extrae el nombre del sistema iSeries de una lista de identificadores de elementos Unity totalmente calificada.

Sintaxis

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

Parámetros

LPCITEMIDLIST *pidl* - entrada

Puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) de la que se extraerá el nombre del sistema.

char * *szSystemName* - salida

Dirección del almacenamiento intermedio que recibirá el nombre del sistema terminado en nulo.

UINT *cchMax* - entrada

Tamaño del almacenamiento intermedio que recibirá el nombre del 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: Objetivo

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

Nombre de objeto Unity del que se extraerá el índice de elemento.

char * *szType* - salida

Dirección del almacenamiento intermedio que recibirá el tipo de elemento terminado en nulo.

UINT *cchMax* - entrada

Tamaño del almacenamiento intermedio que recibirá el tipo 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_GetTypeFromPidl: Objetivo

Extrae el campo de índice de elemento de una lista de identificadores de elemento Unity totalmente calificada.

Sintaxis

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

Parámetros

LPCITEMIDLIST pidl - entrada

Puntero a una estructura ITEMIDLIST (lista de identificadores de elemento) de la que se extraerá el índice de elemento.

char * szType - salida

Dirección del almacenamiento intermedio que recibirá el tipo de elemento terminado en nulo.

UINT cchMax - entrada

Tamaño del almacenamiento intermedio que recibirá el tipo de elemento 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: Objetivo

Renueva el contenido de la ventana de árbol y la ventana de lista de iSeries 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 iSeries Navigator tras llevar a cabo una acción solicitada por el usuario.

cwbUN_RefreshList: Objetivo

Renueva el contenido de la ventana de la vista de lista de iSeries 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: Objetivo

Renueva los elementos seleccionados actualmente en la ventana de la vista de lista de iSeries 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: Objetivo

Inserta una serie de texto en la barra de estado de la ventana de iSeries 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: Objetivo

Devuelve el handle a una conexión ODBC en el sistema servidor iSeries especificado. Si no existe ninguna conexión con el servidor iSeries especificado, la interfaz API obtiene un nuevo handle.

Sintaxis

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

Parámetros

const char * szSystemName - entrada

Nombre del sistema iSeries en el que se recuperará una conexión ODBC.

HDBC *phDBC - salida

Dirección para devolver el handle de conexión ODBC.

Códigos de retorno

La lista siguiente muestra valores de retorno comunes:

CWB_OK

Finalización satisfactoria.

Utilización

Ninguna.

cwbUN_EndODBCConnections: Objetivo

Finaliza todas las conexiones ODBC abiertas anteriormente con **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: Objetivo

Obtiene el índice de la lista de imágenes del icono especificado.

Sintaxis

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

Parámetros

LPCITEMIDLIST *pidl* - entrada

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

Especificación del tipo de índice de icono que se va a recuperar (definido más arriba). Los tipos de distintivo permitidos son los siguientes:

int * *piIndex* - salida

Dirección del entero que recibirá 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: Objetivo

Recupera la lista de imágenes de icono asociada a iSeries Navigator.

Sintaxis

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

Parámetros

UINT *uFlags* - entrada

Especificación del tipo de lista de imágenes que se va a recuperar (definida más arriba). Los tipos de distintivo permitidos son los siguientes:

HIMAGELIST* *phImageList* -

Dirección de la variable que recibirá el handle 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: Objetivo

Esta API devuelve una indicación de si se ha permitido o denegado al usuario actual de iSeries Navigator en el servidor iSeries especificado el uso de una función administrable determinada. Una **función administrable** es cualquier función cuyo uso puede controlarse mediante el subcomponente Administración de aplicaciones de iSeries Navigator.

Por ejemplo, el subcomponente Administración de Aplicaciones permite a un administrador controlar si un usuario puede acceder a varias funciones de iSeries Navigator. Una de estas funciones es la de Gestión de trabajos. Puede utilizarse la API `cwbUN_GetAdminValue` para determinar por programación si el usuario actual de iSeries Navigator puede utilizar la función Gestión de trabajos especificando el nombre de la función administrable que corresponde a Gestión de trabajos. Consulte el archivo de cabecera `cwbunpla.h` para obtener una lista de los nombres de las funciones administrables que reciben soporte en iSeries Navigator.

Sintaxis

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

Parámetros

const char * szSystemName

Nombre del sistema iSeries en el que se efectuará 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 se permite al usuario actual de iSeries Navigator para el servidor iSeries especificado emplear la función especificada. Si en ese momento no hay ningún usuario conectado al servidor iSeries especificado, la API conectará al usuario, posiblemente con la visualización de una solicitud de ID de usuario y contraseña.

Esta API solo puede utilizarse para comprobar las funciones administrables que se encuentran en iSeries Navigator o en la categoría de funciones de Aplicaciones de cliente.

cwbUN_GetAdminValueEx: Objetivo

Esta API devuelve una indicación de si se ha permitido o denegado al usuario actual del servidor iSeries especificado el uso de una función administrable determinada. Una **función administrable** es cualquier función cuyo uso puede controlarse mediante el subcomponente Administración de aplicaciones de iSeries Navigator.

Nota: Los conectores de iSeries Navigator deben utilizar la API `cwbUN_GetAdminValue` en lugar de `cwbUN_GetAdminValueEx`.

Por ejemplo, el subcomponente Administración de Aplicaciones permite a un administrador controlar si un usuario puede acceder a varias funciones de iSeries Navigator. Una de estas funciones es la de Gestión de trabajos. Puede utilizarse la API `cwbUN_GetAdminValueEx` para determinar por programación si el usuario actual puede utilizar la función Gestión de trabajos especificando el nombre de la función administrable que corresponde a Gestión de trabajos. Consulte el archivo de cabecera `CWBUNPLA.H` para obtener una lista de los nombres de las funciones administrables que reciben soporte en iSeries 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 especificarse en el objeto del sistema antes de efectuar una llamada a esta API. El comportamiento de la API `cwbUN_GetAdminValueEx` varía en función de si el objeto del sistema ha obtenido una conexión al servidor iSeries:

No conectado->

`cwbUN_GetAdminValueEx` efectuará la conexión al servidor iSeries. Se bajarán los últimos valores de Administración de Aplicaciones para el usuario desde el servidor iSeries si todavía no se han almacenado en la antememoria del PC cliente.

Conectado->

Si el objeto del sistema se ha conectado al servidor iSeries especificando que deben validarse el ID de usuario y la contraseña de iSeries (modalidad de validación), la API `cwbUN_GetAdminValueEx` utilizará una instantánea de los valores de Administración de Aplicaciones exactos en el momento de completarse la conexión. Si la conexión se ha efectuado sin validar el ID de usuario y la contraseña, es posible que `cwbUN_GetAdminValueEx` utilice una copia de los valores de Administración de Aplicaciones cuya antigüedad puede ser 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 se permite al usuario actual de iSeries (definido por el objeto del sistema de entrada) emplear la función especificada. Si en ese momento no hay ningún usuario conectado al servidor iSeries especificado, la API conectará al usuario, posiblemente con la visualización de una solicitud de ID de usuario y contraseña.

Esta API solo puede utilizarse para comprobar las funciones administrables que se encuentran en iSeries Navigator o en la categoría de funciones de Aplicaciones de cliente.

cwbUN_GetAdminCacheState: Objetivo

Esta API indica si la siguiente invocación de la API `cwbUN_GetAdminValue` será de larga ejecución. La API `cwbUN_GetAdminValue` almacena datos en la antememoria en el PC. Si la antememoria no está actualizada, `cwbUN_GetAdminValue` puede presentar una solicitud de conexión, o llevar a cabo otro proceso, para actualizar la antememoria.

Sintaxis

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

Parámetros**const char * szSystemName**

Nombre del sistema iSeries en el que se efectuará la comprobación.

cwbUN_State& adminState

Indica si la siguiente invocación de la API `cwbUN_GetAdminValue` será de larga ejecución o utilizará la antememoria interna para volver sin acceder al servidor iSeries del sistema principal.

Se devolverá uno de tres valores posibles:

cwbUN_logon

No hay ningún usuario actual para el servidor iSeries especificado. La API `cwbUN_GetAdminValue` puede visualizar una solicitud de conexión.

cwbUN_refresh

`cwbUN_GetAdminValue` accederá al servidor iSeries para actualizar la 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` será de larga ejecución.

cwbUN_GetAdminCacheStateEx: Objetivo

Esta API indica si la siguiente invocación de la API `cwbUN_GetAdminValueEx` será de larga ejecución. La API `cwbUN_GetAdminValueEx` almacena datos en la antememoria en el PC. Si la antememoria no está actualizada, la API `cwbUN_GetAdminValueEx` puede presentar una solicitud de conexión, o llevar a cabo otro proceso, para actualizar la 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

Indica si la siguiente invocación de la API `cwbUN_GetAdminValue` será de larga ejecución o utilizará la antememoria interna para volver sin acceder al servidor iSeries del sistema principal.

Se devolverá uno de tres valores posibles:

cwbUN_logon

No hay ningún usuario actual para el servidor iSeries especificado. La API `cwbUN_GetAdminValue` puede visualizar una solicitud de conexión.

cwbUN_refresh

`cwbUN_GetAdminValue` accederá al servidor iSeries para actualizar la 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` pueden utilizar esta API para determinar si la siguiente invocación de `cwbUN_GetAdminValueEx` será de larga ejecución.

cwbUN_IsSubcomponentInstalled: Objetivo

Determina si un subcomponente de iSeries Navigator está instalado en el PC.

Sintaxis

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

Parámetros

UNIT uOption

Especifica el subcomponente de iSeries Navigator que se debe 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: Objetivo

Creación de un handle que puede utilizarse para acceder a información de configuración sobre el servidor LDAP (Servicios de directorio) en el servidor iSeries.

Sintaxis

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

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

Parámetros

LPCSTR system - entrada

Puntero al nombre del sistema iSeries.

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 handle 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: Objetivo

Libera recursos asociados al handle 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: Objetivo

Devuelve el número de puerto utilizado por el servidor LDAP.

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: Objetivo

Devuelve el número de sufijos configurados para este servidor. Sufijo es el nombre distinguido (DN) de un punto de partida 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: Objetivo

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

Puntero al almacenamiento intermedio que contendrá el nombre distinguido del sufijo.

int * length - entrada/salida

Puntero a la longitud del almacenamiento intermedio de sufijo. Si el almacenamiento intermedio es demasiado pequeño para contener la serie, incluyendo el espacio para el valor nulo (NULL) de terminación, el tamaño del almacenamiento intermedio necesario 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: Objetivo

Crea un handle que puede utilizarse para acceder a información de configuración sobre la información publicada por este servidor iSeries en directorios LDAP.

Sintaxis

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

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

Parámetros

LPCWSTR system - entrada

Puntero al nombre del sistema iSeries.

cwbUN_ldapSvrHandle *pHandle - salida

De retorno, contiene un handle que puede utilizarse con las siguientes 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 no están instalados o el servidor no está configurado.

Utilización

Ninguna.

cwbUN_FreeLdapPublishing: Objetivo

Libera recursos asociados al handle 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: Objetivo

Devuelve el número de registros de publicación configurados para este servidor iSeries. Un registro de publicación identifica una categoría de información que se publicará e indica cómo y dónde se publicará.

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

Número de registros de publicación configurados en el servidor iSeries.

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: Objetivo

Devuelve el tipo de información al que corresponde este registro de publicación.

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 de usuario.

CWBUN_LDAP_PUBLISH_COMPUTERS

Sistemas iSeries

CWBUN_LDAP_PUBLISH_NETWORK_INVENTORY

NetFinity.

CWBUN_LDAP_PUBLISH_PRINTERS

Impresoras iSeries

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: Objetivo

Devuelve el nombre del servidor en el que se publica 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

Puntero al almacenamiento intermedio que contendrá el nombre del servidor.

int * length - entrada/salida

Puntero a la longitud del almacenamiento intermedio de servidor. Si el almacenamiento intermedio es demasiado pequeño para contener la serie, incluyendo el espacio para el valor nulo (NULL) de terminación, el tamaño del almacenamiento intermedio necesario 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: Objetivo

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. Los valores posibles son:

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: Objetivo

Devuelve el nombre distinguido 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_IldapPubHandle 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

Puntero al almacenamiento intermedio que contendrá el nombre de parentDn.

int * length - entrada/salida

Puntero a la longitud del almacenamiento intermedio de nombre distinguido padre (parentDn). Si el almacenamiento intermedio es demasiado pequeño para contener la serie, incluyendo el espacio para el valor nulo (NULL) de terminación, el tamaño del almacenamiento intermedio necesario 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 iSeries Navigator

Lista de los códigos de retorno de las API de iSeries Navigator y sus significados

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	

6011 CWBUN_NULL_PIDL_RETURNED
 Parámetro de PIDL no válido.
 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 consulta de Visual Basic

Utilice los temas siguientes para obtener información sobre el flujo de control en iSeries Navigator para Visual Basic.

Estructura de iSeries Navigator y flujo de control de los conectores Visual Basic

Para los conectores Visual Basic, iSeries Navigator proporciona un servidor ActiveX incorporado que gestiona la comunicación entre iSeries Navigator y la implementación del conector.

Los programadores de Visual Basic que desarrollan conectores de iSeries Navigator utilizan los recursos que proporciona Visual Basic 5.0 de Microsoft para crear sus clases de conectores y empaquetarlas en una DLL servidora ActiveX.

Los conectores actúan respondiendo a las llamadas de método procedentes de iSeries Navigator que se generan en respuesta a las acciones del usuario. Por ejemplo, cuando un usuario pulsa el botón derecho del ratón sobre un objeto de la jerarquía de iSeries Navigator, iSeries Navigator crea un menú de contexto para el objeto y visualiza el menú en la pantalla. iSeries Navigator obtiene los elementos de menú efectuando una llamada a cada conector que ha registrado su intención de proporcionar elementos de menú de contexto 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 iSeries Navigator puede llamar para llevar a cabo una función específica. Hay tres interfaces definidas para los conectores Visual Basic:

- ListManager
- ActionsManager
- DropTargetManager

Información de iSeries Navigator para los conectores Visual Basic

Cuando iSeries Navigator efectúa una llamada a una función implementada por un conector, la petición generalmente supone utilizar uno o varios objetos que el usuario ha seleccionado en la ventana principal de iSeries Navigator. El conector debe poder determinar qué objetos se han seleccionado. El conector recibe esta información como una lista de nombres de objeto totalmente calificados. Para los conectores Visual Basic hay una clase ObjectName definida que proporciona información sobre los objetos

seleccionados. Los conectores que añaden carpetas a la jerarquía de objetos deben devolver los elementos de la carpeta a iSeries Navigator con el formato de "identificadores de elemento". Para los conectores Visual Basic hay una clase ItemIdentifier definida que el conector emplea para devolver la información solicitada.

Servicios de iSeries Navigator para los conectores Visual Basic

En ocasiones un conector de iSeries Navigator tendrá que incidir en el comportamiento de la ventana principal de iSeries Navigator. Por ejemplo, tras la finalización de una operación de usuario, puede ser necesario renovar la vista de lista de iSeries Navigator o insertar texto en el área de estado de iSeries Navigator. En el entorno Visual Basic se suministra una clase de utilidades denominada UIServices que proporciona los servicios necesarios. Los conectores Visual Basic también pueden utilizar las API C++ del archivo de cabecera cwbn.h para obtener resultados similares. Para obtener descripciones detalladas de esta clase y sus métodos, consulte la ayuda en línea facilitada con la DLL de soporte para conectores Visual Basic de iSeries Navigator (cwbnvbi.dll y cwbnvbi.hlp).

Conceptos relacionados

"Clase de interfaz ListManager de iSeries Navigator"

La **clase de interfaz ListManager** se utiliza para el servicio de datos en iSeries Navigator. Por ejemplo, si debe crearse una vista de lista y rellenarse con objetos, iSeries Navigator efectuará una llamada a los métodos de la clase ListManager para ello.

"Clase de interfaz ActionsManager de iSeries Navigator" en la página 73

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, si un usuario pulsa el botón derecho del ratón sobre un objeto de lista Visual Basic en iSeries Navigator, se efectuará una llamada al método queryActions de la clase de interfaz ActionsManager para devolver las series de los elementos de menú de contexto.

"Clase de interfaz DropTargetManager de iSeries Navigator" en la página 73

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

Interfaces Visual Basic de iSeries Navigator

Un conector Visual Basic debe implementar una o varias clases de interfaz de iSeries Navigator, según el tipo de función que el desarrollador desee proporcionar a iSeries Navigator.

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

Hay tres clases de interfaz de iSeries Navigator:

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

Clase de interfaz ListManager de iSeries Navigator:

La **clase de interfaz ListManager** se utiliza para el servicio de datos en iSeries Navigator. Por ejemplo, si debe crearse una vista de lista y rellenarse con objetos, iSeries Navigator efectuará una llamada a los métodos de la clase ListManager para ello.

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 tiene que rellenar listas de componentes de iSeries Navigator.

Para obtener descripciones detalladas de esta clase y sus métodos, consulte la ayuda en línea facilitada con la DLL de soporte para conectores Visual Basic de iSeries Navigator (cwbnvbi.dll y cwbnvbi.hlp).

Conceptos relacionados

“Estructura de iSeries Navigator y flujo de control de los conectores Visual Basic” en la página 71
Para los conectores Visual Basic, iSeries Navigator proporciona un servidor ActiveX incorporado que gestiona la comunicación entre iSeries Navigator y la implementación del conector.

Clase de interfaz ActionsManager de iSeries 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, si un usuario pulsa el botón derecho del ratón sobre un objeto de lista Visual Basic en iSeries Navigator, se efectuará una llamada al método queryActions de la clase de interfaz ActionsManager para devolver las series de los elementos de menú de contexto.

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 facilitada con la DLL de soporte para conectores Visual Basic de iSeries Navigator (archivos cwbnvbi.dll y cwbnvbi.hlp).

Conceptos relacionados

“Estructura de iSeries Navigator y flujo de control de los conectores Visual Basic” en la página 71
Para los conectores Visual Basic, iSeries Navigator proporciona un servidor ActiveX incorporado que gestiona la comunicación entre iSeries Navigator y la implementación del conector.

Clase de interfaz DropTargetManager de iSeries Navigator:

La **clase de interfaz DropTargetManager** se utiliza para manejar las operaciones de arrastrar y soltar en iSeries 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 descripciones detalladas de esta clase y sus métodos, consulte la ayuda en línea facilitada con la DLL de soporte para conectores Visual Basic de iSeries Navigator (cwbnvbi.dll y cwbnvbi.hlp).

Conceptos relacionados

“Estructura de iSeries Navigator y flujo de control de los conectores Visual Basic” en la página 71
Para los conectores Visual Basic, iSeries Navigator proporciona un servidor ActiveX incorporado que gestiona la comunicación entre iSeries Navigator y la implementación del conector.

Información de consulta de Java

Utilice los temas siguientes para obtener información sobre el flujo de control en iSeries Navigator para Java.

Estructura de iSeries Navigator y flujo de control de los conectores Java

Para los conectores Java, iSeries Navigator proporciona un servidor ActiveX incorporado que gestiona la comunicación entre iSeries Navigator y las clases Java del conector.

El componente servidor utiliza la API JNI (Java Native Interface) para crear los objetos del conector y llamar a sus métodos. Por consiguiente, los programadores de Java que desarrollan conectores de iSeries Navigator no tienen que preocuparse de los detalles de la implementación del servidor ActiveX.

Cuando un usuario interactúa con los conectores Java de iSeries Navigator, se generarán llamadas a las distintas clases de interfaz Java registradas para la implementación de la petición específica.

Los conectores actúan respondiendo a las llamadas de método procedentes de iSeries Navigator que se generan en respuesta a las acciones del usuario. Por ejemplo, cuando un usuario pulsa el botón derecho del ratón sobre un objeto de la jerarquía de iSeries Navigator, iSeries Navigator crea un menú de contexto para el objeto y visualiza el menú en la pantalla. iSeries Navigator obtiene los elementos de menú efectuando una llamada a cada conector que ha registrado su intención de proporcionar elementos de menú de contexto 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 iSeries Navigator puede llamar para llevar a cabo una función específica. Para los conectores Java están definidas las tres **interfaces Java** siguientes:

- ListManager
- ActionsManager
- DropTargetManager

Arquitectura del producto para los conectores de iSeries Navigator

La arquitectura interna del producto iSeries Navigator refleja que está pensado para servir de punto de integración para una interfaz ampliable de una gran gama de operaciones para el servidor iSeries. Cada uno de los componentes funcionales de la interfaz está empaquetado como un servidor ActiveX. iSeries Navigator conoce la existencia de un componente servidor determinado por medio de las entradas del registro de Windows. Varios servidores pueden registrar su petición de añadir elementos de menú y diálogos a un tipo de objeto determinado en la jerarquía de iSeries Navigator.

Nota: Para que los conectores Java de terceros estén disponibles para los usuarios de iSeries Navigator, los usuarios de iSeries Access para Windows deben tener instalado en el PC iSeries Access para Windows Versión 4 Release 4 Nivel de modificación 0.

Datos de iSeries Navigator para conectores Java

Cuando iSeries Navigator efectúa una llamada a una función implementada por un conector, la petición generalmente supone utilizar uno o varios objetos que el usuario ha seleccionado en la ventana principal de iSeries Navigator. El conector debe poder determinar qué objetos se han seleccionado. El conector recibe esta información como una lista de nombres de objeto totalmente calificados. Para los conectores Java hay una clase ObjectName definida que proporciona información sobre los objetos seleccionados. Los conectores que añaden carpetas a la jerarquía de objetos deben devolver los elementos de la carpeta a iSeries Navigator con el formato de "identificadores de elemento". Para los conectores Java hay una clase ItemIdentifier definida que el conector emplea para devolver la información solicitada.

En ocasiones un conector de iSeries Navigator tendrá que incidir en el comportamiento de la ventana principal de iSeries Navigator. Por ejemplo, tras la finalización de una operación de usuario, puede ser necesario renovar la vista de lista de iSeries Navigator o insertar texto en el área de estado de iSeries Navigator. En el paquete com.ibm.as400.opnav se suministran clases de utilidades que proporcionan los servicios necesarios.

Personalizar los archivos del registro de conectores

Tras modificar los conectores de ejemplo, será necesario que efectúe algunas modificaciones en los archivos del registro. Este tema proporciona información acerca de los archivos del registro para cada tipo de conector y recomienda algunas modificaciones.

Los archivos del registro identifican los conectores ante iSeries Navigator, describen sus funciones y especifican los prerequisites para utilizarlos.

Los conectores de ejemplo incluyen dos archivos del registro: una copia legible por Windows para utilizar durante el desarrollo y una copia para la distribución en el servidor iSeries. Será necesario que efectúe

algunas modificaciones en estos archivos del registro después de desarrollar el conector. Para ayudarle a hacer estos cambios, este tema ofrece una visión general de los archivos del registro, así como descripciones detalladas de las secciones necesarias de cada uno de los archivos del registro.

iSeries Navigator utiliza los archivos del registro para conocer la existencia de los conectores, así como sus requisitos y funciones. Para proporcionar esta información, cada conector debe especificar como mínimo la información siguiente:

- Una clave "primaria" del registro, que facilita información global sobre el conector. Esta sección incluye el identificador programático (ProgID) que especifica el proveedor y el nombre de componente para el conector, además de especificar el nombre de la carpeta donde reside el conector en el servidor iSeries. El identificador ProgID debe tener el formato <proveedor>.<componente>, por ejemplo, IBM.Sample.
- Las claves del registro que identifican los tipos de objeto en la jerarquía de iSeries Navigator para los que un conector va a proporcionar funciones adicionales.
- Una clave del registro aparte para la raíz de cada uno de los subárboles de objetos que un conector añade a la jerarquía de objetos. Esta clave contiene información sobre la carpeta raíz del subárbol.

Descripciones de las secciones necesarias de los archivos del registro, y cambios recomendados:

- Archivos del registro C++
- Archivos del registro VB
- Archivos del registro Java

Consideraciones especiales para los archivos del registro

- Manejar las hojas de propiedades en C++
- Manejar las hojas de propiedades en VB
- Soporte SSL en conectores

Personalizar los valores del registro de C++

El conector de ejemplo incluye dos archivos del registro: SAMDBG.REG, un archivo del registro legible por Windows para utilizar durante el desarrollo, y SAMPRLS.REG, un archivo del registro para distribuir en el servidor iSeries.

La tabla siguiente describe las secciones de estos archivos del registro y recomienda algunos cambios al desarrollar un conector propio.

Clave primaria del registro

```
; -----  
; 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  
"MinimumMPIRelease"="NONE"  
"MinimumRISRelease"="030701"  
"ProductID"="NONE"  
"ServerEntryPoint"="sampext.dll"
```

Consulte el tema Ejemplo de clave primaria del registro para obtener una descripción de cada uno de los campos y los valores recomendados.

Implementación del servidor de datos

```
-----  
; Esta sección registrará una implementación de IA4HierarchyFolder para cada  
; carpeta nueva que se añada a la jerarquía de iSeries 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 añadirá más de una carpeta nueva a la jerarquía, debe duplicar esta sección del archivo del registro para cada carpeta adicional y asegurarse de generar un GUID aparte para cada carpeta. Si el conector no añade ninguna carpeta, puede eliminar esta sección.

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 la sección de los cambios globales al final de esta página).
3. Sustituya ambas apariciones del identificador CLSID de esta sección del registro por 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; a continuación cambie el CLSID en la llamada de macro IMPLEMENT_OLECREATE para que coincida con los valores hexadecimales del nuevo GUID. Sustituya la palabra "Sample" por el nombre de la nueva carpeta.
6. Cree dos nuevos archivos fuente para cada GUID nuevo utilizando como base una copia red denominada 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 todas las apariciones del nombre de clase "CSampleData" en los dos archivos fuente por un nombre de clase que sea significativo en el contexto del conector.
8. Para añadir los nuevos archivos de implementación al área de trabajo del proyecto, abra el menú **Insertar** y seleccione **Archivos del proyecto...**
9. Al duplicar SAMPDATA.CPP de esta forma, todas las carpetas nuevas inicialmente contendrán objetos de biblioteca.

Implementación de conector de shell

```
-----  
; 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"
```

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

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 la sección de los cambios globales al final de esta página).
3. Sustituya todas las apariciones del CLSID de las entradas que se muestran en el ejemplo anterior por el nuevo GUID que acaba de generar.
4. Busque la serie "IMPLEMENT_OLECREATE" en su versión del archivo EXTINTFC.CPP.
5. Pegue el nuevo GUID sobre el CLSID existente en la línea de comentarios; a continuación cambie el CLSID en la llamada de macro IMPLEMENT_OLECREATE para que coincida con los valores hexadecimales del nuevo GUID.

Implementación de conector de shell para objetos

```

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

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

;-----
; Registrar el manejador de hojas de propiedades de renovación automática para la
; nueva carpeta y sus objetos (esto permitirá a la carpeta utilizar la
; función de renovación automática de iSeries Navigator).

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

[HKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY plug-in\IBM.Sample\shell\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-in\IBM.Sample\shell\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

```

La última sección del registro especifica los objetos de la jerarquía de iSeries Navigator que se ven afectados por la implementación del conector.

1. Sustituya el CLSID de esta sección por los nuevos GUID.
2. Si el conector no añadirá páginas de propiedades adicionales a una hoja de propiedades para una carpeta u objeto, elimine la entrada de registro del manejador de hojas de propiedades.
3. Si el conector no será un manejador de acciones de soltar para objetos, elimine las entradas de registro del manejador de menú de contexto de arrastrar y soltar y el manejador de acciones de soltar.
4. Edite las subclaves \Sample**. Para obtener más información, consulte 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.

Cambios globales

Debe especificar un identificador ProgID exclusivo e identificadores GUID que se utilizarán a lo largo del archivo del registro del conector.

Defina un identificador programático (ProgID) exclusivo para el conector:

El identificador ProgID debe coincidir con la serie de texto <proveedor>.<componente>, donde proveedor identifica el nombre del proveedor que ha desarrollado el conector y componente describe la función que proporciona. En el conector de ejemplo, la serie "IBM.Sample" identifica IBM como proveedor y "Sample" como descripción de la función que proporciona el conector. Esta información se utilizará a lo largo del archivo del registro y dará nombre al directorio donde residirá el conector tanto en el servidor iSeries como en la estación de trabajo. Sustituya todas las apariciones de "IBM.Sample" en el archivo del registro por el identificador ProgID que utilice.

Genere nuevos GUID y sustituya los valores de CLSID del archivo del registro:

Para que el conector C++ de iSeries Navigator funcione correctamente, debe sustituir determinados CLSID del nuevo archivo del registro por los GUID que genere.

COM (Component Object Model) para Microsoft utiliza enteros hexadecimales de 16 bytes para identificar de forma exclusiva las interfaces y clases de implementación ActiveX. Estos enteros se denominan GUID (identificadores exclusivos globalmente). Los GUID que identifican clases de implementación se llaman CLSID. iSeries Navigator utiliza el soporte de ejecución ActiveX de Windows para cargar los componentes de un conector y obtener un puntero a una instancia de la implementación del conector de una interfaz determinada. 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 consiguiente, un conector de iSeries Navigator debe registrar un CLSID para cada clase de implementación que proporcione.

Siga estos pasos para generar identificadores GUID:

1. En 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**.

Ejemplo: clave primaria del registro:

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-in\IBM.Sample]
"Type"="PLUGIN"
"NLS"="sampmri.dll"
"NameID"=dword:00000080
"DescriptionID"=dword:00000081
"MinimumIMPIRelease"="NONE"
"MinimumRISRelease"="030701"
"ProductID"="NONE"
"ServerEntryPoint"="sampext.dll"

```

Campo de clave primaria del registro	Descripción del campo
Type	Si el conector añade nuevas carpetas a la jerarquía de iSeries 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 del registro, este valor puede ser un nombre de vía de acceso totalmente calificado.
NameID	Palabra doble que contiene el identificador de recurso de la serie de texto de la DLL de recursos, que se utilizará para identificar el conector en la interfaz de usuario de iSeries Navigator.
DescriptionID	Palabra doble que contiene el identificador de recurso de la serie de texto en la DLL de recursos. Esta DLL de recursos se emplea para describir la función del conector en la interfaz de usuario de iSeries Navigator.
MinimumIMPIRelease	<p>Serie de 6 caracteres que identifica el release de i5/OS mínimo que se ejecuta en el hardware IMPI que requiere el conector. La serie debe tener el formato vvrrmm, siendo vv la versión de i5/OS, rr el release y mm 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".</p> <p>Si el conector no da soporte a ningún release de i5/OS que se ejecute en el hardware IMPI (releases anteriores a la versión 3 release 6), 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 IMPI, el valor de este campo debe ser "ANY".</p>

Campo de clave primaria del registro	Descripción del campo
MinimumRISCRelease	<p>Serie de 6 caracteres que identifica el release de i5/OS que se ejecuta en el hardware RISC que requiere el conector. La serie debe tener el formato vvrmm, siendo vv la versión de i5/OS, rr el release y mm el nivel de modificación. Por ejemplo, si el conector requiere Versión 3 Release 7 Nivel de modificación 1, el valor de este campo debe ser "030701".</p> <p>Si el conector no da soporte a ninguna versión de i5/OS que se ejecute en el hardware RISC (Versión 3 Release 6 y posteriores), el valor de este campo debe ser "NONE". Si el conector puede dar soporte a cualquier versión de OS/400 que se ejecute en el hardware RISC, el valor de este campo debe ser "ANY".</p>
ProductID	<p>Serie de 7 caracteres que especifica el ID de producto de un programa bajo licencia del servidor iSeries que el conector precisa como prerequisite. Si el conector no requiere que haya instalado un programa bajo licencia determinado en el servidor iSeries, el valor de este campo debe ser "NONE".</p> <p>Es posible especificar varios ID de producto separados por comas si existen varios ID para el mismo producto.</p>
ServerEntryPoint	<p>Nombre de la DLL de código que implementa el punto de entrada de servidor. iSeries Navigator efectúa una llamada a este punto de entrada cuando tiene que determinar si el conector está soportado en un servidor iSeries 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 del registro, este valor puede ser un nombre de vía de acceso totalmente calificado.</p>
JavaPath	<p>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 nombres de archivo JAR en relación con la vía de acceso de instalación de iSeries Access para Windows, cada uno de ellos precedido por la variable de sustitución de iSeries Access para Windows que representa la vía de acceso de instalación.</p>
JavaMRI	<p>Nombres base de los archivos JAR que contienen los recursos dependientes del entorno nacional para el conector. iSeries Navigator buscará cada uno de los archivos JAR tras primero añadir como sufijo al nombre los identificadores adecuados de idioma y país para Java. Si no existe ningún archivo JAR de MRI para un entorno nacional determinado, iSeries Navigator esperará que la interfaz MRI del entorno local base (por lo general el inglés estadounidense) resida en los archivos JAR de código.</p>

Tareas relacionadas

“Personalizar los valores del registro de conectores VB”

El conector de ejemplo incluye dos archivos del registro: VBSMPDBG.REG, un archivo del registro legible por Windows para utilizar durante el desarrollo, y VBSMPRLS.REG, un archivo del registro para distribuir en el servidor iSeries.

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 iSeries 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 del par especifica la carpeta raíz de un componente de iSeries 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. En el caso de los conectores que añaden comportamientos a tipos de objeto existentes, esta subclave normalmente debe ser el tipo de objeto de la carpeta de primer nivel bajo un objeto contenedor del servidor iSeries. 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

Al llevar a cabo comprobaciones para los tipos de objeto existentes, debe utilizar los identificadores de tipo de 3 caracteres que están definidos en el registro bajo la clave `HKEY_CLASSES_ROOT\IBM.AS400.Network\TYPES`. Al efectuar comprobaciones para los tipos de objeto nuevos definidos por el conector, emplee una clave del registro. Use la clave de registro que identifica la carpeta que haya especificado como punto de unión, o el tipo que devuelva a iSeries Navigator al servir datos para una carpeta definida por el conector.

Personalizar los valores del registro de conectores VB

El conector de ejemplo incluye dos archivos del registro: VBSMPDBG.REG, un archivo del registro legible por Windows para utilizar durante el desarrollo, y VBSMPRLS.REG, un archivo del registro para distribuir en el servidor iSeries.

La tabla siguiente describe las secciones de este archivo del registro y recomienda algunos cambios al desarrollar un conector propio.

Clave primaria del registro

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.

Consulte el tema Ejemplo de clave primaria del registro para obtener una descripción de cada campo.

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

"Type"="Plugin"

"NLS"="vbsmpmri.dll"

"NameID"=dword:00000080

"DescriptionID"=dword:00000081

"MinimumIMPIRelease"="NONE"

"MinimumRISCRRelease"="040200"

"ProductID"="NONE"

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

Cambios recomendados:

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.

Registrar una carpeta nueva

Esta sección registrará una implementación de la clase ListManager del conector Visual Basic para cada una de las carpetas nuevas que se añadan a la jerarquía de iSeries Navigator. Si el conector no añade ninguna carpeta nueva a la jerarquía de iSeries Navigator, suprima esta sección y continúe con la tarea siguiente.

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

El ejemplo coloca la carpeta de Visual Basic de ejemplo en el nivel raíz de un nombre de sistema de servidor iSeries en la jerarquía de iSeries Navigator. Si desea que la carpeta aparezca en alguna otra ubicación de la jerarquía, debe cambiar el valor de la clave "Parent". Consulte Valores del campo Parent para obtener una lista de los valores posibles.

Consulte el tema Ejemplo: nueva clave del registro de carpeta para obtener una descripción de cada uno de los campos y los valores posibles.

```
[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
```

Cambios recomendados:

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.

Registrar los objetos del conector VB

La última sección del registro especifica los objetos de la jerarquía de iSeries Navigator que se ven afectados por la implementación del conector 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}"

```

Cambios recomendados:

1. El CLSID de las entradas anteriores siempre debe tener lo siguiente: "{040606B2-1C19-11d2-AA12-08005AD17735}".
2. La clave "VBClass" contiene el identificador de programa (ProgID) de la clase de implementación Visual Basic.
3. La clave "VBInterface" contiene el ID de interfaz de la clase de implementación Visual Basic.
4. Si el conector no será un manejador de acciones de soltar para objetos, elimine las entradas de registro del manejador de menú de contexto de arrastrar y soltar y el manejador de acciones de soltar.
5. Cambie el nombre de las subclaves \SampleVBFolder** y utilice una serie exclusiva para identificar el objeto de carpeta. Este nombre es el tipo de objeto que se utilizará en el fuente Visual Basic para identificar cuándo se efectúan acciones en esta carpeta en iSeries 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:

Defina un identificador programático exclusivo (ProgID) para el conector. El identificador ProgID debe coincidir con la serie de texto <proveedor>.<componente>, donde proveedor identifica el nombre del proveedor que ha desarrollado el conector y componente describe la función que proporciona. En el conector de ejemplo, la serie "IBM.Sample" identifica a IBM como proveedor y "Sample" como descripción de la función que proporciona el conector. Esta información se utilizará a lo largo del archivo del registro y dará nombre al directorio donde residirá el conector tanto en el servidor iSeries como en la estación de trabajo.

Sustituya todas las instancias de "IBM.VBSample" por su nuevo [proveedor].ProgID.

Nota: iSeries Navigator proporciona DLL servidoras ActiveX incorporadas que gestionan los conectores escritos en Java y 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.

Consulte el tema **Ejemplo: clave primaria del registro**.

Conceptos relacionados

“Ejemplo: clave primaria del registro” en la página 79

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

Valores del campo Parent:

Lista de códigos de ID para las carpetas padre

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

AS4	Carpeta del servidor iSeries
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”

Hay que definir una clave del registro aparte para la raíz de cada uno de los subárboles de objetos que un conector añade a la jerarquía de objetos. Esta clave contiene información específica de la carpeta raíz del subárbol.

Ejemplo: nueva clave del registro de carpeta:

Hay que definir una clave del registro aparte para la raíz de cada uno de los subárboles de objetos que un conector añade a la jerarquía de objetos. Esta clave contiene información específica de la carpeta raíz del subárbol.

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-in\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. Consulte Valores del campo Parent para obtener una lista de los valores posibles.
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	CLSID de la implementación de IA4HierarchyFolder a la que iSeries Navigator debe llamar 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	Nombre de clase Java totalmente calificado de la implementación de ListManager a la que iSeries Navigator debe llamar para obtener el contenido de la carpeta. Este campo debe omitirse si el conector no es un conector Java.
VBClass	Identificador de programa (ProgID) de la clase de implementación ListManager a la que iSeries Navigator debe llamar para obtener el contenido de la carpeta.
VBInterface	GUID de la interfaz de la clase de implementación ListManager .
NameID	Palabra doble que contiene el ID de recurso de la serie que debe aparecer como nombre de la carpeta en la jerarquía de iSeries Navigator.
DescriptionID	Palabra doble que contiene el ID de recurso de la serie que debe aparecer como descripción de la carpeta en la jerarquía de iSeries Navigator.
DefaultIconIndex	Palabra doble que contiene el índice en la DLL de recursos NLS del conector para el icono que debe visualizarse para la carpeta en la jerarquía de iSeries 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	Palabra doble que contiene el índice en la DLL de recursos NLS del conector para el icono que debe visualizarse para la carpeta en la jerarquía de iSeries Navigator cada vez que el usuario selecciona ese elemento.
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 85
Lista de códigos de ID para las carpetas padre

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 iSeries Navigator mediante una serie de texto con el formato <proveedor>.<componente>, donde proveedor identifica el proveedor que ha desarrollado el conector y componente describe la función que proporciona. En los ejemplos siguientes, la serie IBM.MsgQueueSample3 identifica IBM como proveedor y "MsgQueueSample3" como descripción de la función que proporciona el conector. Esta serie se conoce como *identificador programático* (ProgID). Se utiliza a lo largo del archivo del registro al especificar la función que proporciona el conector y también da nombre al directorio donde residirá el conector tanto en el servidor iSeries como en la estación de trabajo cliente.

Identificadores exclusivos globalmente (GUID)

COM (Component Object Model) de Microsoft utiliza enteros hexadecimales de 16 bytes para identificar de forma exclusiva las interfaces y clases de implementación ActiveX. Estos enteros se denominan *identificadores exclusivos globalmente* (GUID). Los GUID que identifican clases de implementación se llaman CLSID (ID de clase).

Para los componentes de iSeries Navigator escritos en Java, no debe definir nuevos identificadores 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"  
"MinimumRISCRRelease"="ANY"  
"ProductID"="NONE"  
"ServerEntryPoint"="NONE"  
"JavaPath"="MsgQueueSample3.jar"  
"JavaMRI"="MsgQueueSample3MRI.jar"
```

Type Si el conector añade nuevas carpetas a la jerarquía de iSeries 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 del registro, este valor puede ser un nombre de vía de acceso totalmente calificado.

NameID

Palabra doble que contiene el identificador de recurso de la serie de texto de la DLL de recursos, que se utilizará para identificar el conector en la interfaz de usuario de iSeries Navigator.

DescriptionID

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

MinimumIMPIRelease

Serie de 6 caracteres que identifica el release de i5/OS mínimo que se ejecuta en el hardware IMPI que requiere el conector. La serie debe tener el formato vvrmmm, siendo vv la versión de

i5/OS, rr el release y mm 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 da soporte a ningún release de i5/OS que se ejecute en el hardware IMPI (releases anteriores a la versión 3 release 6), 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 IMPI, el valor de este campo debe ser "ANY".

MinimumRISCRelease

Serie de 6 caracteres que identifica el release de i5/OS mínimo que se ejecuta en el hardware RISC que requiere el conector. La serie debe tener el formato vvrmmm, siendo vv la versión de i5/OS, rr el release y mm el nivel de modificación. Por ejemplo, si el conector requiere Versión 3 Release 7 Nivel de modificación 1, el valor de este campo debe ser "030701".

Si el conector no da soporte a ningún release de i5/OS que se ejecute en el hardware RISC (Versión 3 Release 6 y posteriores), 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

Serie de 7 caracteres que especifica el ID de producto de un programa bajo licencia del servidor iSeries que el conector precisa como prerrequisito. Si el conector no requiere que haya instalado un programa bajo licencia determinado en el servidor iSeries, el valor de este campo debe ser "NONE".

Es posible especificar varios ID de producto 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. iSeries Navigator efectúa una llamada a este punto de entrada cuando tiene que determinar si el conector está soportado en un servidor iSeries 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 del registro, este valor 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 del registro, debe identificar los archivos JAR. Los nombres de archivo JAR no deben calificarse con nombres de directorio; iSeries Navigator los calificará automáticamente al crear la serie de vía de acceso de clase que se pasará a la máquina virtual Java..

JavaMRI

Nombres base de los archivos JAR que contienen los recursos dependientes del entorno nacional para el conector. iSeries Navigator buscará cada uno de los archivos JAR tras primero añadir como sufijo al nombre los identificadores adecuados de idioma y país para Java. En la versión de desarrollo del archivo del registro, este campo puede contener una serie vacía, ya que los recursos del entorno nacional base (por lo general el inglés estadounidense) deben residir en los archivos 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 carpeta nueva que el conector añade a la jerarquía de iSeries Navigator tiene un tipo lógico exclusivo. En el ejemplo anterior, la serie **Sample3** es el tipo que se empleará para identificar la carpeta seleccionada actualmente cuando se pase el control al conector en la ejecución.

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

ADF	Carpeta de Desarrollo de aplicaciones
AS4	Carpeta del servidor iSeries
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 atributo 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

CLSID de la implementación de `IA4HierarchyFolder` a la que iSeries Navigator debe llamar para obtener el contenido de la carpeta. Para los conectores Java, este CLSID siempre debe ser `{1827A856-9C20-11d1-96C3-00062912C9B2}`.

JavaClass

Nombre de clase Java totalmente calificado de la implementación de `ListManager` a la que iSeries Navigator debe llamar para obtener el contenido de la carpeta.

NameID

Palabra doble que contiene el ID de recurso de la serie que debe aparecer como nombre de la carpeta en la jerarquía de iSeries Navigator.

DescriptionID

Palabra doble que contiene el ID de recurso de la serie que debe aparecer como descripción de la carpeta en la jerarquía de iSeries Navigator.

DefaultIconIndex

Palabra doble que contiene el índice en la DLL de recursos NLS del conector para el icono que

debe visualizarse para la carpeta en la jerarquía de iSeries 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

Palabra doble que contiene el índice en la DLL de recursos NLS del conector para el icono que debe visualizarse para la carpeta en la jerarquía de iSeries Navigator cada vez que el usuario selecciona ese elemento. Puede coincidir con el índice de icono por omisión.

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

Palabra doble que contiene el ID de recurso de la serie que debe aparecer como nombre del área de tareas en la jerarquía de iSeries 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 emplea para describir la función del área de tareas en la interfaz de usuario de iSeries 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\  
shell\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\  
shell\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\  
shell\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\  
shell\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 shell. El primer identificador del par especifica la carpeta raíz de un componente de iSeries 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. En el caso de las carpetas existentes, esta subclave normalmente debe ser el tipo

de objeto de la carpeta de primer nivel bajo un objeto contenedor del servidor iSeries. 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 determinado en la jerarquía de iSeries 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.

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

Nombre de clase Java totalmente calificado de la implementación de la interfaz a la que iSeries Navigator debe llamar para dar soporte a la función designada.

Soporte SSL

Si las comunicaciones de un conector con el servidor iSeries se llevan a cabo utilizando la API de Sockets o algún otro servicio de comunicaciones de bajo nivel, el conector es el responsable de dar soporte a SSL si se ha solicitado. Si el conector no proporciona este soporte, debe indicar que no da soporte a SSL como se describe más abajo. En ese caso, la función del conector se inhabilitará 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 da soporte a SSL, este valor debe ser 1. De lo contrario, debe ser 0.

Páginas de propiedades para un manejador de hojas de propiedades

Las clases de la biblioteca Microsoft Foundation Classes no pueden utilizarse para crear páginas de propiedades para un manejador de hojas de propiedades. No obstante, IBM suministra **CExtPropertyPage**, que puede emplearse en lugar de la clase CPropertyPage de MFC.

Las páginas de propiedades implementadas por los conectores de iSeries Navigator deben ser una subclase de CExtPropertyPage. La declaración de clase puede encontrarse en el archivo de cabecera PROPEXT.H y la implementación se incluye 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. Si este distintivo está activo, iSeries Navigator automáticamente añadirá Propiedades al menú de contexto del

objeto. Además, si este distintivo está activo, iSeries Navigator llamará a cualquiera de los manejadores de hojas de propiedades registrados para añadir páginas a la hoja de propiedades cuando se seleccione el elemento del menú de contexto.

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. Hay definido un distintivo para esta situación, que puede devolverse a iSeries Navigator en las 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 intención de añadir páginas de propiedades a una de las hojas de propiedades para un usuario de iSeries, la clave que especifica el CLSID del manejador de hojas de propiedades debe especificar un campo PropSheet que identifique la hoja de propiedades a la que añadirá páginas el manejador especificado. Vea el ejemplo siguiente.

```

;-----
;
Registrar un manejador de hojas de propiedades para la hoja de
propiedades Red para los usuarios de iSeries
[HKKEY_CLASSES_ROOT\IBM.AS400.Network\3RD PARTY plug-in\IBM.Sample\shell\Users
and 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	Personal-Before-All	Capabilities-Before-All	Jobs-Before-All	Networks-Before-All
Groups-After-Info	Personal-After-Name	Capabilities-After-Privileges	Jobs-After-General	Networks-After-Servers
	Personal-After-Location	Capabilities-After-Auditing	Jobs-After-Startup	Networks-After-General
	Personal-After-Mail	Capabilities-Before-Other	Jobs-After-Display	
		Capabilities-After-Other	Jobs-After-Output	
		Jobs-After-International		

Para añadir páginas a una hoja de propiedades para un usuario de iSeries, el conector debe implementar la interfaz IA4PropSheetNotify (consulte Lista de especificaciones de la interfaz IA4PropSheetNotify).

Restricción:

La restricción que se describe a continuación es válida para las hojas de propiedades de los objetos de usuario de iSeries:

- No es posible implementar múltiples manejadores de hojas de propiedades para las diversas hojas de propiedades asociadas a un usuario de iSeries en la misma clase de implementación.
- Cada una de las hojas de propiedades necesita un CLSID aparte.

Conceptos relacionados

“Descripción de los distintivos de QueryContextMenu” en la página 93
 Añadir SHORT DESCRIPTION.

“Lista de especificaciones de la interfaz IA4PropSheetNotify” en la página 32

La interfaz IA4PropSheetNotify proporciona notificaciones a la implementación de IShellPropSheetExt que son necesarias al añadir páginas de propiedades adicionales a una de las hojas de propiedades de usuarios y grupos.

Descripción de los distintivos de QueryContextMenu:

Añadir SHORT DESCRIPTION.

iSeries Navigator da soporte a las siguientes mejoras efectuadas en la interfaz IContextMenu:

Orden de los elementos de menú de contexto

iSeries Navigator ha ampliado la interfaz IContextMenu para obtener un control más preciso del orden en que se añaden elementos de menú al menú para una carpeta o un objeto determinado.

iSeries Navigator estructura los menús de contexto 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.

iSeries Navigator efectúa una llamada al método QueryContextMenu para un componente tres veces seguidas, una para 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. (iSeries Navigator efectúa una llamada al conector de shell en el momento adecuado para obtener los elementos del menú Edición y no establece 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. Puede devolverse un distintivo definido para esta situación a iSeries Navigator en las llamadas a IContextMenu::QueryContextMenu cuando se establece el distintivo UNITY_CMF_STANDARD. 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.

Conceptos relacionados

“Páginas de propiedades para un manejador de hojas de propiedades” en la página 91

Las clases de la biblioteca Microsoft Foundation Classes no pueden utilizarse para crear páginas de propiedades para un manejador de hojas de propiedades. No obstante, IBM suministra `CExtPropertyPage`, que puede emplearse en lugar de la clase `CPropertyPage` de MFC.

Ejemplo: crear páginas de propiedades Visual Basic para un manejador de hojas de propiedades

Las páginas de propiedades implementadas por los conectores Visual Basic de iSeries Navigator no pueden utilizar una clave del registro para especificar páginas de propiedades. 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 los objetos de hojas de propiedades ya existentes.

En el conector Visual Basic de ejemplo, se da soporte a una página de propiedades para bibliotecas en la lista de iSeries 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
            End With
        End If
    End If
End Function
```

```

        .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 hojas de propiedades de conectores Java. Esto permite crear nombres de objeto, visualizar propiedades, compartir objetos con terceros y combinar 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 `PropertiesManagerInterface`. Por ejemplo:

```
[HKEY_CLASSES_ROOT\IBM.AS400.Network\AS/400 Network\*
\shell\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.

Para obtener más información, consulte el ejemplo de gestor de propiedades.

Conceptos relacionados

“Ejemplo: gestor de propiedades Java”

A continuación figura un ejemplo del código del gestor de propiedades Java.

Ejemplo: gestor de propiedades Java:

A continuación figura un ejemplo del código del gestor de propiedades Java.

```
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: Excepción al crear
páginas "+e);}

pm.setTitle("Primera página Java");
pm2.setTitle("Segunda página Java");

PanelManager[] PMArray = {pm, pm2};

return PMArray;

}

// Devolver lista de objetos ActionListener para
// enviar notificación al procesar compromiso
public ActionListener[] getCommitListeners()
{

ActionListener[] al = new ActionListener[1];
al[0] = new ActionListener()
{

public void actionPerformed(ActionEvent evt)
{

Monitor.logError("SamplePropertiesManager: Procesando escuchador
de compromiso");

}

};
return al;

}

// Devolver lista de objetos ActionListener para enviar
// notificación al seleccionar cancelación
public ActionListener[] getCancelListeners()
{

ActionListener[] al = new ActionListener[1];
al[0] = new ActionListener()
{

public void actionPerformed(ActionEvent evt)
{

Monitor.logError("SamplePropertiesManager: Procesando
escuchador de cancelación");

}

};
return al;
}

```

```
}  
}
```

Conceptos relacionados

“Manejar las hojas de propiedades en Java” en la página 95

Puede añadir páginas de propiedades a hojas de propiedades de conectores Java. Esto permite crear nombres de objeto, visualizar propiedades, compartir objetos con terceros y combinar código C++ y Java en el mismo conector.

Entrada de capa de sockets segura (SSL) del registro

Los usuarios de iSeries Navigator pueden solicitar una conexión segura a un servidor iSeries marcando el recuadro de selección **Utilizar Capa de Sockets Segura (SSL)** de la pestaña **Conexión** de la hoja de propiedades de los objetos iSeries. Si este recuadro está seleccionado, únicamente los componentes de iSeries Navigator que pueden dar soporte a las comunicaciones SSL están habilitados para que el usuario los active.

Si todas las comunicaciones de un conector con el servidor iSeries se gestionan utilizando el handle del sistema de iSeries Access para Windows (especifique `cwbCO_SysHandle`), o la clase `com.ibm.as400.access.AS400` en el caso de un conector Java, debe indicar que da soporte a las conexiones seguras con el servidor iSeries. Para los conectores C++, `cwbCO_SysHandle` se obtiene efectuando una llamada a la API `cwbUN_GetSystemHandle`. Cuando el usuario seleccione una conexión segura, iSeries Navigator automáticamente habilitará SSL. En el caso de los conectores Java, el objeto de servidor iSeries obtenido mediante una llamada al método `getSystemObject` de la clase `com.ibm.as400.opnav.ObjectName` en realidad será una instancia de `com.ibm.as400.access.SecureAS400`.

Nota: Si está ejecutando Java sobre SSL y crea su propio certificado de autoridad certificadora, se necesita el paquete de servicio GA de iSeries Access para Windows.

Si las comunicaciones de un conector con el servidor iSeries se llevan a cabo utilizando la API de Sockets o algún otro servicio de comunicaciones de bajo nivel, el conector es el responsable de dar soporte a SSL si se ha solicitado. Si el conector no proporciona este soporte, debe indicar que no da soporte a SSL como se describe más abajo. En ese caso, la función del conector se inhabilitará si el usuario ha solicitado una conexión segura.

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 de licencia de código y declaración de limitación 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, LOS
| DESARROLLADORES Y LOS SUMINISTRADORES DE PROGRAMAS 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, 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, EN CUYO CASO, ALGUNAS O TODAS LAS
| LIMITACIONES O EXCLUSIONES ANTERIORES PUEDEN NO SER PROCEDENTES 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 licenciarios 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 está pensada a efectos de planificación. La información que aquí se incluye está sujeta a cambios antes de que los productos descritos estén disponibles.

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 los 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 iSeries Navigator documenta las interfaces de programación cuya finalidad es permitir al cliente escribir programas para obtener los 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
iSeries
NetServer
OS/400

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.

Los demás nombres de compañías, productos y 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 de IBM, las publicaciones se utilicen en detrimento de sus intereses o cuando, también según el parecer de IBM, 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