# C. WMI

## C.1 WMI Overview

El estándar Gestión Empresarial Basada en Web (WBEM-Web Based Enterprise Management) proporciona acceso uniforme a información de gestión. El Intrumental de Gestión de Windows (WMI-Windows Management Instrumentation) es la aplicación Microsoft de esta tecnología. Tal información de gestión incluye información sobre el estado de memoria del sistema, inventarios de aplicaciones del cliente actualmente instaladas, y otra información sobre estado del cliente. La tecnología WMI implementada por las plataformas Microsoft® Windows® permite representar los sistemas, aplicaciones, redes, y otros componentes gestionados utilizando al Modelo de Información Común (CIM- Common Information Model) diseñado por la Fuerza de Tarea de Gestión Distribuída (DMTF-Distributed Management Task Force). CIM puede modelar cualquier elemento en el ambiente gestionado sin tener en cuenta la localización de la fuente de datos.

Además para el modelamiento de datos, WMI ofrece un poderoso conjunto de servicios base que incluyen recuperación de información basada en consultas y notificaciónes de evento. El acceso a estos servicios y a los datos de gestión se hace con un solo Modelo de Objeto de Componente (COM- Component Object Model) para programar la interfase.

El Conjunto de Desarrollo Software WMI (WMI SDK- WMI Software Development Kit) incluye las herramientas para crear aplicaciones del cliente y proveedores de WMI. La apreciación global de la documentación WMI SDK se incluye en las secciones que se describen en la siguiente tabla.

| **Sección** | **Descripción** |
|---|---|
| Requerimientos del sistema | Configuraciones del sistema mínimas para WMI. |

| | |
|---|---|
| Aprecicación global de la arquitectura | Descripción de alto nivel de la iniciativa WMI y de los componentes que son parte de la tecnología WMI. |
| Terminos y conceptos | Los términos principales y conceptos que son centrales para WMI. |
| Localización de WMI | Calificadores localizados agregados a las clases de WMI. |
| Seguridad y Autenticación | Apoyo de seguridad WMI para los sistemas operativos de Windows. |

### C.1.1 Requerimientos del sistema

El Windows Management Instrumentation (WMI) SDK requiere las siguientes coniguraciones mínimas del sistema:

- Requerimientos Software
- Requerimientos del Compilador
- Requerimientos Hardware

- **Requerimientos Software**

Esta versión del Windows Management Instrumentation (WMI) SDK es soprtada por Microsoft® Windows NT® version 4.0 Service Pack 5 o una versión superior, incluyendo Microsoft® Windows® 2000. Tambien se necesita Microsoft® Internet Explorer version 5 o una versión superior.

El WMI SDK debe ser instalado por un usuario con cuenta de administrador sobre plataformas Windows NT y Windows 2000.

**Nota:** Los componentes centrales del sistema operativo de WMI son soportados sobre Windows NT 4.0 con Service Pack 4 o una versión superior, incluyendo Windows 2000, o sobre Windows 95 OSR2 y Windows 98.

- **Requerimientos del compilador**

El Microsoft WMI SDK trabaja con la mayoría de compiladores, Microsoft solo soporta el ambiente de desarrollo Microsoft® Visual C++® versión 6.0.

Los archivos .idl y .h incluídos en el WMI SDK son desarrollados para usarse con compiladores Visual C++ 6.0. Estos archivos requieren editarse antes de ser usados con compiladores de vendedores diferentes a Microsoft.

- **Requerimientos Hardware**

Las configuraciones mínimas que debe tener un computador existen y son basadas en el sistema operativo Microsoft usado. Para el desarrollo de aplicaciones utilizando el Windows Management Instrumentation (WMI) SDK, es recomendable tener un computador con las siguientes características:

- Computador Pentium
- 32 MB RAM
- 30 MB disponibles en el Disco duro
- Tarjeta de Video que soporte 256 colores con una resolución de 800 por 600
- Tarjeta de red

### C.1.2 Apreciación global de la arquitectura

La Gestión Empresarial Basada en Web es una iniciativa emprendida por la Fuerza de Tarea de Gestión Distribuída para proveer un estándar para la gestión de ambientes empresariales y

la solución económica a sus necesidades de gestión. La iniciativa de la Gestión Empresarial Basada en Web abarca una multitud de tareas y va desde la configuración de una simple estación de trabajo hasta la gestión máxima de la empresa a través de múltiples plataformas. La parte central a la iniciativa es El Modelo de Información Común (CIM), un modelo de datos extensible para representar objetos que existen en ambientes de dirección típicos y el Formato del Objeto Gestionado (MOF-Management Object Format), lenguaje para definir y guardar datos modelados.

WMI es una aplicación de la iniciativa WBEM para las plataformas de Microsoft® Windows® que extiende CIM para representar objetos que existen en ambientes de WMI y llevando a cabo una infraestructura de gestión para apoyar el lenguaje de MOF y una interfase de programación común, WMI permite a las diversas aplicaciones manejar una variedad de componentes de la empresa transparentemente.

La infraestructura de WMI consta de los siguientes componentes:

- El software de WMI (Winmgmt.exe) el cual es un componente que proporciona a las aplicaciones un acceso uniforme para gestionar datos.

- El repositorio CIM que es una área central de almacenamiento para gestionar datos.

## ❑ Arquitectura de WMI

La arquitectura de la tecnología WMI consta de:

- Aplicaciones de gestión
- Objetos gestionados
- Proveedores
- Infraestructura de gestión la cual contiene el software de WMI (Winmgmt.exe) y el repositorio CIM.

Las aplicaciones de gestión son aplicaciones basadas en Microsoft® Windows® o servicios Microsoft® Windows NT®/Windows 2000 que procesan o despliegan datos de los objetos gestionados. Una aplicación de gestión puede realizar una variedad de tareas como medir desempeño, informar detenciones y correlacionar datos.

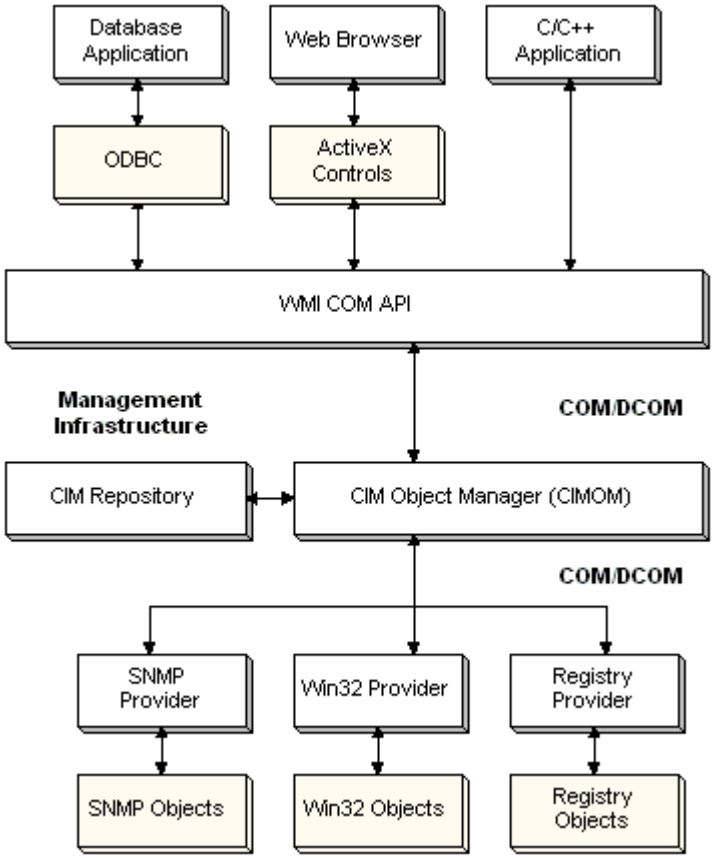
Los objetos gestionados son componentes lógicos o físicos de la empresa. Se modelan los objetos gestionados utilizando el CIM y ellos son accedidos por aplicaciones de gestión a través de WMI. Un objeto gestionado puede ser cualquier componente de la empresa, desde un pequeño pedazo de hardware como un cable, hasta una gran aplicación de software como un sistema de base de datos.

Los proveedores usan la API de COM para proporcionar a WMI datos de los objetos gestionados, manejar demandas en nombre de las aplicaciones de gestión y para generar notificaciones de eventos. El WMI SDK proporciona varios proveedores normales, como un proveedor del registro, para acceder información desde el registro del sistema. Además el WMI SDK también proporciona un Proveedor de eventos Windows NT/Windows 2000 que permite a las aplicaciones recibir notificaciones de los eventos Windows NT/Windows 2000 y acceder la información guardada en el registro de eventos Windows NT/Windows 2000. Los vendedores que desarrollan para este ambiente, llamados vendedores de terceras partes, pueden crear proveedores usuales para interactuar con objetos gestionados específicos a su ambiente.

La infraestructura de gestión consiste de WMI y el Repositorio CIM. WMI permite a los usuarios manejar comunicaciones entre las aplicaciones de gestión y proveedores, de esta forma los usuarios guardan sus datos estáticos en el almacén de CIM. Las aplicaciones y proveedores se comunican a través de WMI que usa una interfase de programación común (API de COM). El API de COM proporciona la notificación de eventos y requerimientos que procesan servicios y está disponible en lenguajes de programación como C y C++ .

El Repositorio CIM incluye datos de gestión estáticos. Los datos estáticos son aquellos que no cambian regularmente. WMI también soporta datos dinámicos los cuales deben generarse por demanda porque varían frecuentemente. Pueden ponerse datos en el Repositorio CIM a través de WMI o de administradores de red.
Los diseñadores de terceras partes pueden poner información en el repositorio CIM con el MOF y su compilador o por medio del API de COM.

El siguiente diagrama muestra la relación entre los componentes en la tecnología de WMI. En lo alto del diagrama se muestran ejemplos de aplicaciones de gestión. Algunas de estas aplicaciones acceden al API de COM directamente para actuar recíprocamente con WMI y el repositorio CIM para hacer peticiones de gestión. Otras aplicaciones usan métodos de acceso como el "Open Database Connectivity" (ODBC) y el "Hypertext Markup Language" (HTML) para hacer sus peticiones.



fuente: Tutorial WMISDK

En la parte baja del diagrama se ilustran ejemplos de objetos gestionados y sus proveedores correspondientes. Hay una variedad de métodos de acceso usados para la comunicación.

El protocolo usado para comunicación entre los componentes locales y remotos es El Modelo del Objeto Componente Distribuído (DCOM - Distributed Component Object Model).

- **Aplicaciones de gestión**

Una aplicación de gestión es una aplicación o un servicio Microsoft® Windows NT®/Microsoft® Windows® 2000 que usa la información originada desde uno o más objetos gestionados. Para acceder a la información, una aplicación de gestión hace una petición a WMI a través de uno de los siguientes métodos: el API de COM o el Scripting API.

WMI permite a las aplicaciones de gestión llevar a cabo una amplia gama de características como reportes de producción en un inventario de red, desplegar información del sistema, responde a eventos, empezar o detener servicios del sistema y envía un comando de expulsión a una unidad de disco con medios de comunicación reemplazables. Estas características son más fáciles de implementar con WMI, aunque es posible llevarlos a cabo sin WMI.

WMI soporta un número de estrategias para llevar a cabo aplicaciones de gestión. Las aplicaciones pueden acceder a WMI directamente a través del API de COM o Scripting API o indirectamente con uno de los siguientes métodos de acceso:

- Los Web browsers que pueden usar un conjunto de controles Microsoft® ActiveX® para controlar la apariencia, relaciones y conducta de datos que relacionan a los

objetos gestionados. Los controles son personalizables y pueden ser incluidos en la definición del esquema.

- Los Web browsers también pueden usar HTML, el estándar de páginas HTML y el paradigma hyperlink. HTML es soportado a través de un API para Servidor de Internet (ISAPI) que interactúa con WMI.

- Las aplicaciones de bases de datos pueden usar el adaptador WMI ODBC para unir las capacidades de la base de datos de ODBC con las capacidades de gestión de WMI. Con el adaptador ODBC, una aplicación puede usar una amplia gama de paquetes basados en ODBC y herramientas como Microsoft® Excel y Microsoft® Access.

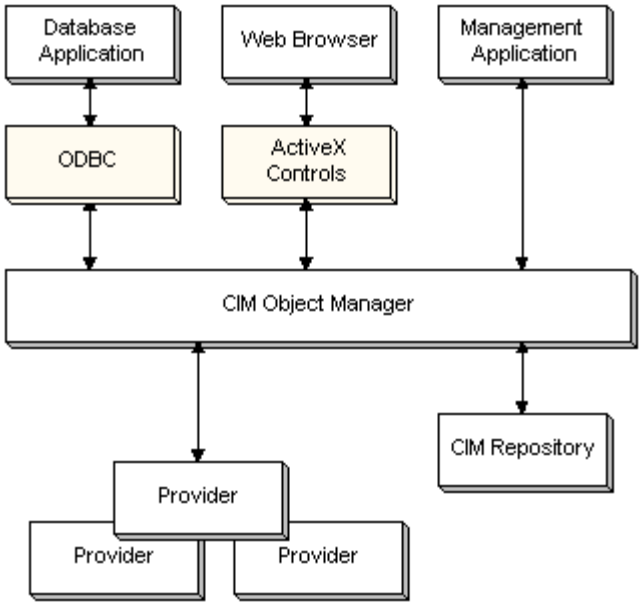
- Las aplicaciones de servicio de directorio pueden usar la extensión WMI llamada "Active Directory Service Interface" (ADSI) para integrar el servicios y gestión de datos.

El API de COM está directamente disponible para programadores de C/C++.

Se puede usar el Scripting API para desarrollar script y aplicaciones basadas en Microsoft® Visual Basic® que se pueden usar para visualizar o controlar los objetos gestionados. WMI también proporciona soporte scripting para los siguientes lenguajes:

- Visual Basic
- Visual Basic para Aplicaciones
- Visual Basic Scripting Edition
- Microsoft® JScript®

La siguiente figura muestra aplicaciones de gestión de varios tipos que se comunican con objetos gestionados CIM de WMI. Dos de las aplicaciones usan una de las estrategias de acceso; la otra se comunica directamente con Objeto Gestionador CIM de WMI.




- **WMI, Directorio Activo, y MMC**

Una aplicación de gestión completa a nivel empresarial típicamente comprende tres componentes importantes: WMI, el Directorio Activo de Microsoft (Microsoft® Active Directory™) y la consola de Gestión de Microsoft (MMC- Microsoft Management Console). WMI proporciona información individual del computador, como los dispositivos que se instalan y el software que se está ejecutando además el Directorio Activo proporciona la vista de la empresa y se enfoca en la información que necesita estar disponible para la red. Se pueden usar WMI y el Directorio Activo al mismo tiempo para proporcionar una vista completa del ambiente. Puesto que ambas fuentes de información pueden ser necesarias para lograr una tarea dada, WMI incluye el Proveedor de Servicios de Directorio para acceder a información guardada en el Directorio Activo.

Se puede usar la arquitectura de interfase de usuario común MMC para presentar esta información a los usuarios. La información de WMI, Directorio Activo y otras fuentes pueden ser combinadas con MMC para presentar una vista cohesiva de gestión. Se

recomienda que se desarrolle una base embebida para usar WMI como su fuente de información.

- **Proveedores WMI**

Los proveedores de WMI actúan como intermediarios entre WMI y uno o mas objetos gestionados. Cuando WMI recibe una petición de una aplicación de gestión para datos que no están disponibles en el repositorio CIM o para notificaciones de eventos que WMI no soporta, WMI remite la demanda a un proveedor. Los proveedores proporcionan datos y notificaciones de evento para objetos gestionados que son específicos para un dominio en particular.

Los proveedores son servidores COM o DCOM estándares que pueden ser implementados utilizando los tipos de servidores soportados:

- Librerías dinámicas en proceso (DLLs)
- Servicios locales o remotos de Microsoft® Windows NT®/Microsoft® Windows® 2000
- Archivos ejecutables estándares, ya sea locales o remotos (.exe files)

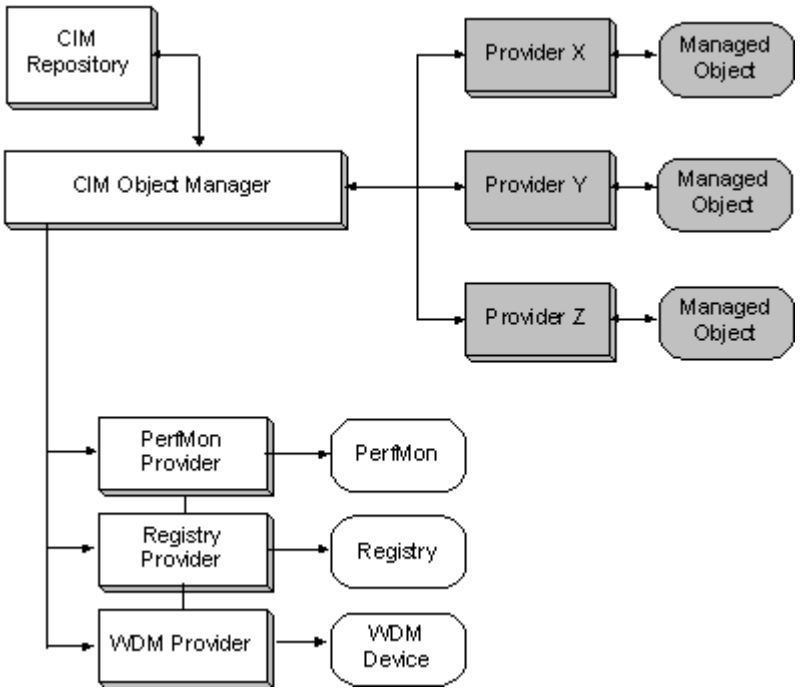
Nota: los proveedores en proceso defectuosos pueden impactar negativamente a la fiabilidad del sistema operativo. Los proveedores deben ser implementados como servicios o archivos ejecutables siempre que sea posible.

El WMI SDK incluye algunos proveedores incorporados. Los proveedores incorporados, también llamados proveedores normales utilizan fuentes de suministro de datos muy conocidas tales como el registro del sistema y el subsistema Microsoft® Win32®.

La tecnología WMI también soporta la creación de proveedores personalizados por diseñadores de terceras partes. Estos proveedores sirven peticiones relacionadas con objetos gestionados que están en ambientes específicos. Los proveedores usan típicamente el

lenguaje MOF para definir y crear clases, además usan el API de COM para acceder al repositorio CIM y responden a las demandas WMI hechas inicialmente por las aplicaciones.

El siguiente diagrama ilustra la arquitectura de proveedores. El Proveedor PerfMon, Proveedor del Registro y Proveedor WDM son ejemplos de proveedores normales. Proveedor X, Proveedor Y y Proveedor Z son ejemplos de proveedores personalizados. Los proveedores normales proporcionan WMI y aplicaciones de gestión con datos obtenidos de objetos gestionados estandares como la aplicación de monitoreo de desempeño, el registro del sistema y el modelo de Drivers de dispositivos Windows (WDM). Los proveedores personalizados proporcionan un objeto gestor CIM de WMI y aplicaciones con datos tomados desde objetos personalizados. Los proveedores normales y los personalizdos pueden acceder datos del repositorio CIM a través del API de COM.




Hay dos tipos de proveedores básicamente: aquellos que soportan recuperación de datos y modificación, y aquellos que soportan notificación de eventos. Los proveedores pueden categorizarse en la siguiente tabla:

| Tipo de proveedor | Descripción |
|---|---|
| Class | Recupera, modifica, anula, y/o enumera clases específicas de proveedor. También puede soportar procesamientos de consultas. |
| Instance | Recupera, modifica, anula, y/o enumera las instancias de clases específicas del proveedor. También puede soportar procesamiento de consultas. |
| Property | Recupera y/o modifica valores de propiedad individuales. |
| Meted | Invoca métodos para una clase específica de proveedor. |
| Event | Genera notificaciones de eventos. |
| Event consumer | Soporta notificación de evento para mapear un consumidor físico con un consumidor lógico. |

**Repositorio CIM**

Es un área central de almacenamiento gestionada por el gestor de objetos CIM. Las clases estáticas con instancias estáticas almacenan las definiciones de clases e instancias en el repositorio CIM. Las clases estáticas con instancias dinámicas almacenan solamente las definiciones. Las clases dinámicas almacenan las definiciones de las clases o las definiciones de las instancias. La información dinámica siempre es dada por la demanda del proveedor.

- **Modelo de Información común (CIM)**

El Modelo de Información Común (CIM-Common Information Model) presenta una vista consistente y unificada de todos los tipos de objetos lógicos y físicos en un ambiente gestionado. Los objetos gestionados se representan utilizando estructuras orientadas a objetos

como clases. Las clases incluyen propiedades que describen los datos y métodos que definen la conducta. El CIM está diseñado por la DMTF como parte de la iniciativa WBEM a demás CIM no depende de una plataforma particular. WMI incluye una extensión del CIM para plataformas con sistema operativo Microsoft® Windows® .

El CIM define tres niveles de clases:

- Clases que representan objetos gestionados que se aplican a todas las áreas de gestión. Estas clases mantienen un vocabulario básico para analizar y describir sistemas gestionados. Ellos son parte de lo que se llama el modelo "core".

- Clases que representan objetos gestionados que se aplican a áreas de gestión específicas pero son independientes de una aplicación o tecnología en particular. Estas clases son parte de lo que se llama el modelo común que es una extensión del modelo "core".

- Clases que representan objetos gestionados que son adiciones específicas de la tecnología para el modelo común. Estas clases se aplican típicamente a las plataformas específicas como UNIX o el ambiente Microsoft® Win32® .

Pueden derivarse clases de otras clases. Una clase derivada representa un caso especial de su clase padre y por consiguiente incluye todos las propiedades y métodos del padre. Las relaciones de herencia no están típicamente disponibles para la aplicación de gestión que los usa; la jerarquía de herencia no es importante para estas aplicaciones. Pueden obtenerse jerarquías de la clase utilizando aplicaciones que son incluidas en el WMI SDK.

WMI también soporta asociaciones. Las asociaciones son instancias de clases de asociación y son usadas para representar relaciones entre otros objetos de WMI. Las relaciones de asociación son visibles a las aplicaciones de gestión. WMI define las clases de asociación para soportar clases del sistema. Los diseñadores de terceras partes también pueden definir clases de asociación para su ambiente de gestión.

WMI soporta el concepto de esquemas para agrupar las clases e instancias que se usan dentro de un ambiente de gestión particular. El WMI SDK incluye dos esquemas: el esquema CIM y el esquema Win32. El esquema CIM contiene las definiciones de las clases para los primeros dos niveles del CIM. Estas clases representan objetos gestionados que son parte de cada ambiente de gestión sin tener en cuenta la plataforma. El esquema Win32 contiene definiciones de las clases para objetos gestionados que son parte de un ambiente típico Win32.

Los diseñadores de terceras partes pueden crear sus propios esquemas personalizados para describir ambientes específicos del vendedor. Porque se quieren esquemas infinitamente extensibles, los diseñadores siempre pueden agregar nuevas clases para representar nuevos objetos gestionados en un ambiente existente. Los esquemas personalizados son extensiones del esquema CIM o Win32.

- **WinMgmt.exe**

WinMgmt.exe es el componente primario en la infraestructura de gestión de "Windows Management Instrumentation" (WMI). En computadores corriendo Microsoft® Windows® 98, WinMgmt.exe corre como un archivo ejecutable normal. En computadores Microsoft® Windows NT®/Microsoft® Windows® 2000, WinMgmt.exe corre como un servicio. En ambos casos, WinMgmt.exe empieza cuando la primera aplicación de gestión hace una llamada para conectarse. WinMgmt.exe se activa cuando la primera aplicación del cliente se conecta con éxito y corre continuamente cuando las aplicaciones de gestión buscan sus servicios activamente.

En computadores corriendo Windows NT versión 4.0, WinMgmt.exe corre en uno de dos modos, como un servicio manual o como un servicio automático. Como un servicio automático, WinMgmt.exe entra en un estado de perfil de baja memoria cuando todos los clientes empiezan. Como un servicio manual, WinMgmt.exe espera 30 segundos y baja el servicio cuando todos los clientes hayan finalizado.

WMI soporta dos interfases de programación conocidas como el COM API para WMI y el Scripting API para WMI. El COM API es usado para aplicaciones de gestión, para proveedores, y para extensiones del esquema. El COM API, habilita estos componentes para interactuar con WMI utilizando C y C++ y esta basado en el Component Object Model (COM). El Scripting API puede ser usado para crear aplicaciones cliente con WMI utilizando lenguajes de scipting y Microsoft® Visual Basic®.

Cuando una aplicación hace una petición llamando a uno de los dos métodos ya sea el COM API o el SCRIPTING API, WMI determina si la petición involucra datos estáticos almacenados en el repositorio CIM, o datos dinámicos dados por un proveedor. Los datos estáticos pueden ser manejados por WMI, mientras que los datos dinámicos siempre involucran al proveedor. Los proveedores registran su localización y soporte para operaciones paticulares como adquisición de datos, modificación, eliminación, enumeración y procesos de consulta. WMI usa esta información registrada para hacer coincidir los requirimientos de la aplicación con el proveedor adecuado y encontrar y leer dicho proveedor cuando es necesario. Cuando el proveedor ha finalizado el procesamiento de una petición, este retorna los resultados a WMI, el cual reenvia los resultados a la aplicación.

Adicionalmente para implementar cada una de las operaciones asociadas con las peticiones de las aplicaciones, WMI también proporciona lo siguiente:

- Soporte para notificación de evento
- Soporte para el lenguaje de petición
- Soporte de seguridad
- Soporte de localización

- **Soporte para la Notificación de Evento**

WMI soporta el registro para la distribución de notificaciones de evento. Los consumidores de evento se registran para recibir tipos particulares de notificaciones. Los proveedores de evento se registran para proporcionar tipos particulares de notificaciones.

Para permitir a los consumidores de evento operar independientemente de los proveedores de evento WMI actúa como el intermediario y empareja a los consumidores registrados con los proveedores correspondientes y envía los eventos apropiados.

WMI soporta consumidores de evento temporales y consumidores de evento permanentes. Los consumidores de evento temporales sólo reciben notificaciones mientras que estén activos, cuando ellos terminan, su registro es removido. Los consumidores de evento permanentes reciben notificaciones siempre y cuando ellas ocurran. Cuando un consumidor de evento permanente es registrado, WMI no termina automáticamente con la última aplicación; se reinicia automáticamente después de que el sistema se reinicia además WMI debe estar en todo momento disponible para entregar éstas notificaciones de evento.

Además de los eventos generados por proveedores de evento, el propio WMI produce dos tipos de eventos: eventos cronometrados y los eventos intrínsecos. Los eventos cronometrados ocurren periódicamente según un intervalo de tiempo especificado, o una vez en un determinado tiempo. Los eventos intrínsecos corresponden a los cambios en datos guardados en el repositorio CIM.

Siempre que sea posible, los proveedores deben publicar los eventos intrínsecos pertinentes. Esto previene la pérdida de la ejecución que sería el resultado de registrar por defecto este evento.

Además de publicar eventos intrínsecos que corresponden a los cambios en los objetos gestionados controlados por el proveedor, los proveedores pueden necesitar a veces publicar

eventos extrínsecos, es decir, eventos que no corresponden claramente a cualquier cambio en el estado de un objeto gestionado.

WMI remite las notificaciones de todos los tipos de eventos a aplicaciones que se han registrado para recibirlas.

- **Soporte para el lenguaje de consulta**

WMI actualmente soporta el "Windows Management Instrumentation Query Language" (WQL) para manejar el proceso de consulta. WQL es un subconjunto del estandar ANSI "Structured Query Language" (SQL) con extensiones WMI específicas. Algunas extensiones soportan registro y notificación de eventos, mientras que otros soportan relaciones entre las clases e instancias. WQL es un lenguaje de solo lectura el cual no puede ser usado para actualizar, insertar o anular datos. WQL sólo puede ser usado para recuperar información.

- **Soporte de seguridad**

WMI soporta una forma limitada de seguridad que positivamente identifica y valida a cada usuario antes de que al usuario se le permita conectarse a WMI. WMI no soporta protección de clases individuales o instancias de datos dinámicos. WMI soporta protección individual de las clases e instancias de datos dinámicos a través del uso de personificación.

WMI también soporta seguridad para "namespaces" individuales.

En las plataformas Windows NT/Windows 2000, no hay ninguna distinción entre acceso local y remoto, excepto que para conectarse remotamente a un "namespace" de WMI, dado que es un derecho separado que puede o no concederse. Sin embargo, con una conexión remota, los usuarios pueden especificar un nombre del usuario y contraseña y pueden reemplazar el nombre del usuario actual y contraseña. Con una conexión local, los usuarios no pueden cambiar el nombre actual y contraseña.

En platafomas Windows 95 y Windows 98 se concede a los usuarios locales todos los derechos, no hay ninguna autenticación. Sin embargo, los usuarios remotos se validan utilizando instancias del sistema de clases WMI.

Los administradores pueden usar la aplicación de control WMI (Wbemcntl.exe) para poner permisos WMI para los usuarios.

### C.1.3 Términos y Conceptos

- **Clases e Instancias**

Una clase de WMI define una plantilla (template) para describir un tipo de objeto gestionado. Una instancia de una clase de WMI representa un objeto gestionado específico del tipo descrito por la clase. Donde la clase generalmente modela el dispositivo del mundo real o el componente en el sentido general, cada instancia representa una ocurrencia específica del dispositivo o componente. Por ejemplo, una clase general llamada FloppyDisk podría existir con las instancias, en un host particular de una computadora que representa los drives A y B.

Algunas clases de WMI son clases de base abstractas. Las clases abstractas no tienen ningúna instancia propia; sólo clases derivadas de las clases abstractas que tienen instancias. El CIM_ManagedSystemElement system class, por ejemplo, es una clase abstracta. Las instancias son creadas de las subclasses de CIM_ManagedSystemElement. Otro tipo de clase soporta sólo una instancia. Tales clases son conocidas como clases de semifallo (singleton).

Hay algunos funcionamientos que no pueden realizarse en instancias. Los siguientes funcionamientos son inválidos:

Cambiar la clase de una instancia.
Cambiar un tipo de propiedad.
Agregar o quitar una propiedad.

Agregar o quitar una Llave, Indexed, Singleton, calificador Dinámico, o Abstracto.

Todas las instancias de una clase existen dentro del "namespace" al que la clase pertenece. Dentro de un "namespace", las instancias pueden ser estáticas o dinámicas. Se guardan instancias estáticas persistentemente y permanecen válidas, aun después de reiniciar el sistema operativo, hasta que ellas se anulen explícitamente. Se debe poner instancias estáticas en el almacén de CIM (CIM Repository) utilizando el IWbemServices::PutInstance o método de IWbemServices::PutInstanceAsync, o sometiendo un archivo de MOF al recopilador de MOF.

Las instancias dinámicas son proporcionadas por un proveedor cuando surge la necesidad y no se guarda en el almacén de CIM. Los requerimientos de instancias dinámicas son enviadas a través de WMI al proveedor. El soporte dinámico de instancias de una clase permite a un proveedor siempre proporcionar valores actuales de propiedad.

Las clases estáticas pueden soportar instancias estáticas o dinámicas. Dentro de una instancia estática, pueden existir propiedades estáticas o dinámicas. Una clase estática que soporta instancias sólo estáticas que contienen propiedades estáticas siempre está disponible a través de WMI; el proveedor es innecesario en estos casos.

Las clases estáticas con instancias dinámicas guardan sólo la definición de la clase en el almacén de CIM. Cuando se piden las instancias, un proveedor debe recuperar la definición de la clase y debe construir la instancia requerida o el set de instancias.

Las clases estáticas con instancias estáticas que contienen propiedades estáticas y dinámicas que pueden guardar placeholders para las propiedades dinámicas en el almacén de CIM. Cuando un requerimiento es hecho por una instancia, WMI llama al proveedor propietario designado para proporcionar los datos actuales para cada propiedad dinámica.

Las clases dinámicas soportan sólo instancias dinámicas. Con una clase dinámica, no se guardan ni la definición de la clase ni las instancias en el almacén de CIM. El proveedor es

responsable por una información de almacenes de clases dinámicas para la definición y locaciones de las instancias. Cuando una aplicación pide una clase dinámica, WMI localiza al proveedor y continúa la demanda.

- **Clases de Sistema**

Las clases de sistemas WMI son una colección de clases predefinidas incluidas en WMI que soportan las actividades de WMI. Las clases de sistemas WMI se definen automáticamente en cada "namespace". Por ejemplo, el sistema clasifica proveedores de registro y describe eventos que relacionan los dos para cambios en el reporitorio CIM y a los componentes específicos del proveedor.

Las clases del sistema se identifican por llevar doble subrayado en su nombre. Todas las clases del sistema se derivan de la clase baja abstracta, __SystemClass. Debido a que las clases del sistema son predefinidas por WMI, los diseñadores no pueden crear su propio sistema de clases empezando un nombre de la clase con un doble subrayado. Sin embargo, los diseñadores pueden derivar clases de las clases del sistema.

- **Identidad y localización**

  - **Propiedades llave**

WMI identifica instancias de las clases a través del uso de llaves (Keys), propiedades que proporcionan un único valor. Las propiedades importantes distinguen instancias de una clase de la misma manera que las columnas importantes distinguen filas de una tabla en una base de datos.

Las propiedades importantes son designadas por el calificador Key, un calificador normal definido por WMI. Se puede atar calificadores importantes a más de una propiedad en una clase para crear una llave compuesta para la clase. Las propiedades escogidas como propiedades key (importantes) para una clase, cuando se toman juntas, deben contener valores que pueden identificar cada instancia de la clase. Cualquier tipo de propiedad puede usarse como una llave con las excepciones siguientes:

- Series (Arrays)
- Números Reales y punto-flotante
- Objetos empotrados (Embedded)
- Los carácteres inferiores (lower) de ASCII 32 (es decir, los carácteres espaciados blancos)
- Cadenas de caracteres que definen cómo las llaves deben tener rangos de Unicode de U+0020 o más altos. Esta restricción existe porque se usan valores Key en caminos del objeto (Object paths), y no se pueden usar carácteres del nonprinting en object paths.

La siguiente sintaxis Managed Object Format (MOF) ilustra cómo definir una clase con una sola propiedad Key:

```
class MyClass
{
    [key] string MyKey;
    string RegularProp;
};
```

Cuando una superclase especifica una llave, todas las clases derivadas de la superclase heredan esa llave. Las clases derivadas no pueden alterar la llave heredada o definir cualquier nueva propiedad importante. Sin embargo, cuando se deriva una subclase de una clase abstracta sin una llave, se puede introducir una llave en la subclase.

Todas las clases que definen más de una instancia deben especificar una llave. Porque las clases abstractas no definen ningúna instancia, ellas no necesitan especificar llaves. Porque las clases singleton definen sólo una instancia, ellas no pueden especificar llaves.

Las cadenas de caracter de tipo Char16 que son definidas como llaves deben contener valores mayores a U+0020.

- **"namespaces"**

Un "namespace" es una unidad lógica para agrupar clases y clases de instancias, y para controlar su alcance y visibilidad. Un "namespace" está representado en su "namespace" padre por una instancia del __namespace system class o una clase que se deriva de __namespace class. El __namespace class tiene una sola propiedad: Nombre. Es el nombre del "namespace" que lo distingue de todos los otros "namespaces". Los nombres de "namespace" no pueden llevar subrayado.

Los "namespaces" pueden anidarse dentro de otro para formar una jerarquía arbitraria de clases e instancias. El modelo anidado se parece al que es usado por un sistema de archivo jerárquico. Cada "namespace" debe tener un único nombre dentro del alcance de su objeto padre inmediato.

Para indicar una jerarquía del "namespace", se separa individualmente los nombres de los "namespace" con cuchilladas dirigidas hacia atrás, como se muestra luego:

Namespace1\Namespace2\Namespace3..... \LastNamespace

Los "namespaces" pueden colocarse en un formato jerárquico, como se muestra en el ejemplo al final de este tema, o en un formato lateral. De cualquier modo, la relación entre los "namespaces" no tiene nada que ver con la herencia de la clase. Es decir, un "namespace" niño no hereda las clases de su "namespace" padre automáticamente.

WMI proporciona apoyo limitado para las relaciones cross-namespace. Una asociación puede contener referencias para clasificar instancias en dos "namespaces" diferentes con tal de que las instancias se definan estáticamente dentro del Repositorio del Modelo de Información Común (CIM repository) y no son proporcionadas por proveedores dinámicos. Para más información sobre la estática y la definición dinámica de instancias de la clase, ver Clases e Instancias.

Típicamente, un "namespace" contiene un conjunto de clases e instancias que representan objetos gestionados en un ambiente particular. Por ejemplo, se crean las clases e instancias para gestionar objetos en el ambiente Microsoft® Win32® en su propio "namespace" único. Debido a que todos los funcionamientos en la versión actual de WMI ocurren en un "namespace" particular, los conflictos no ocurren entre clases que son nombradas igual y residen en "namespaces" diferentes. Sin embargo, cuando se crean un nombre de clase y sus propiedades asociadas y métodos, se recomienda que se evite dar el mismo nombre a la clase que el de otra clase en otro "namespace". Esto elimina posibles problemas para los clientes cuando descargan WMI.

Todas las instalaciones de WMI tienen estos "namespaces" predefinidos:

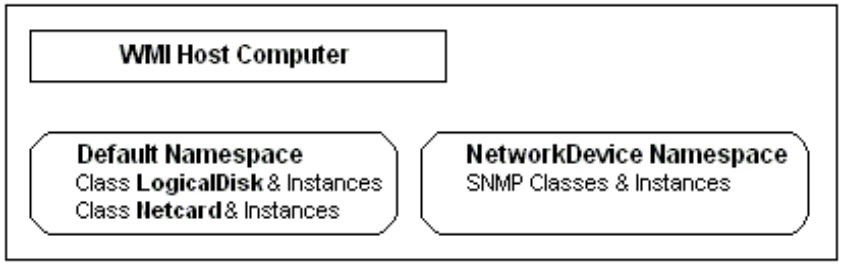
root
root\default
root\cimv2

El "namespace" de la raíz es diseñado principalmente para contener otros "namespaces". La instalación de WMI pone los otros tres "namespaces" predefinidos bajo el "namespace" de la raíz. El "namespace" del root\default sostiene la mayoría de las clases del sistema. El "namespace" root\cimv2 contiene las clases e instancias que representan un ambiente Win32, como Win32_LogicalDisk y Win32_OperatingSystem.

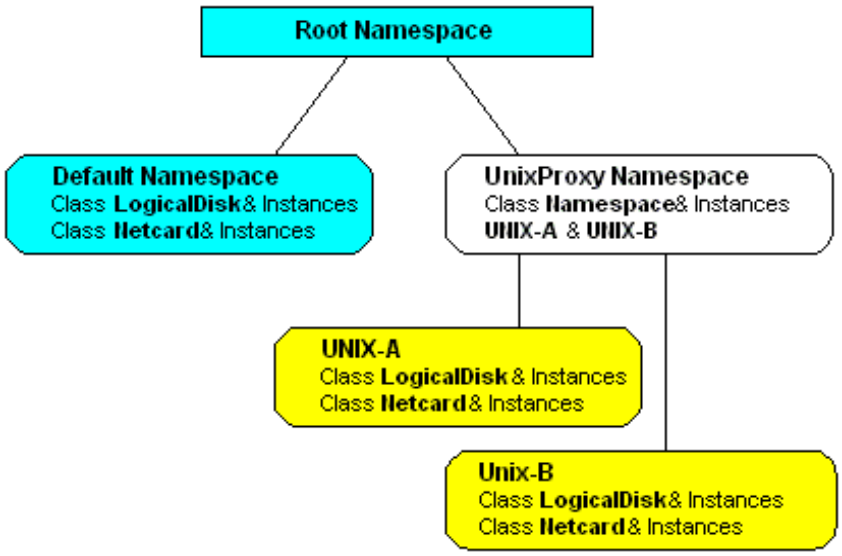
La mayoría de las operaciones ocurren dentro del "namespace" root\cimv2. Sin embargo, si la instalación de WMI también está actuando como un apoderado para un dispositivo lógico o

de red, se deben localizar las clases e instancias asociados con el dispositivo lógico o de red en otro "namespace" que no sea root\cimv2, como se muestra en la ilustración siguiente. Haciendo que sea fácil de distinguir entre las clases e instancias para la computadora local y para el dispositivo lógico o de red.

Fuente: WMI SDK




En el ejemplo siguiente, se anidan dos "namespaces" bajo el "namespace" de raíz: cimv2 y UnixProxy. El "namespace" cimv2 modela a la computadora local, y el "namespace" UnixProxy sostiene dos otros "namespaces", Unix-A y Unix-B. Los "namespaces" Unix-A y Unix-B modelan cada uno una computadora UNIX completa.




Fuente: WMI SDK

Debido a que se representan "namespaces" como instancias del __namespace class, un nuevo "namespace" se crea cuando una instancia de __namespace class se crea y un "namespace" se anula cuando una instancia de __namespace class se anula. Después de que un "namespace" es marcado para borrarlo, todos los clientes actualmente conectados a ese "namespace" reciben el código de error WBEM_E_INVALID_namespace en cualquier llamada subsecuente al "namespace".

- **"Object Paths"(rutas del objeto)**

Se usa WMI Object Paths para identificar individualmente "namespaces" en un servidor, clases dentro de un "namespace", o instancias de una clase. Los Object Paths son conceptualmente similares al Recurso Universal Locators (URLs). Los dos son jerárquicos en formato y contienen uno o más elementos. El número de elementos depende del tipo de Object Paths, y si es un camino (path) lleno o un camino que es relativo a una situación particular.

Los Object Paths pueden contener lo siguiente:

- Cadenas contenidas solo en comillas.
- Forward slashes como separadores.
- Backslashes como separador.
- Constantes Hexadecimal para los enteros.
- Constantes Boleanas, como VERDADERO y FALSO, para las clases con llaves que toman valores Booleanos.

Un servidor local supuesto con un camino del "namespace" parcial. Así, si se especifica el "namespace" root\default implica el "namespace" root\default en el servidor local.

Los Object paths no pueden contener espacio en blanco dentro de un elemento o entre los elementos. Se puede usar marcas de comillas incluídas en Object paths pero se deben delimitar con carácteres escape, como en una aplicación de C o C++. La anotación de URL puede usarse para representar caracteres non-printing (no-imprimiendo), como %20 para un espacio en blanco. Se reconocen valores sólo decimales como porciones numéricas de llaves.

Se usan Object paths en las propiedades de referencia de clases de asociación para identificar objetos relacionados, y como parámetros de entrada y salida para varios métodos. Object paths se tratan como cadenas con el propósito de lookup y comparación, el valor de Object path cuando es usado como una propiedad de referencia siempre es la propia cadena y no el objeto de referencia. Las comparaciones de cadena que se tratan con Object paths siempre son case-insensitive (caso-insensibles).

La tabla siguiente lista los métodos que requieren Object paths para "namespaces", clases, o instancias.

| **Tipo de object path** | **Metodo** |
|---|---|
| "namespace" object path | **IWbemServices::OpenNamespace** |
| Clase object path | **IWbemServices::ExecMethod**<br>**IWbemServices::ExecMethodAsync** |
| Clase o clase de instancia object path | **IWbemServices::GetObject** |
| Clase o instancia object path | **IWbemServices::GetObjectAsync.** |
| Instancia object path | **IWbemServices::DeleteInstance**<br>**IWbemServices::DeleteInstanceAsync**<br>**IWbemCallResult::GetResultString** |

Para más información sobre los tipos específicos de object paths, ver:

"namespace" Object Paths.

Class Object Paths.

Instance Object Paths.

- **“namespace Object Paths” (Rutas de objetos namespace)**

Un “namespace” object path se usa para localizar un “namespace” particular en un servidor. Un “namespace” object path lleno tiene dos elementos, servidor y “namespace”, y se estructura utilizando slashes delanteras (forward slashes) o dirigidas hacia atrás, como se muestra abajo:

```
\ \Server\”namespace”
```
ó
```
 / /Server/”namespace”
```

El elemento del servidor especifica el nombre de la red de la computadora que esta organizando el “namespace”. Si el servidor es la computadora local, puede usarse un solo punto en lugar del nombre del servidor, como se muestra luego:

```
\ \. \”namespace”
```

El elemento del “namespace” especifica cualquier “namespace” válido. Un “namespace” se representa con un cordón jerárquico que contiene elementos delimitados por backslashes simples, como root\cimv2. No se puede usar slashes delanteras (forward slashes) dentro de los nombres del “namespace”.

- **Class Object Paths (Rutas de objetos clase)**

Un Class Object Paths se usa para localizar una clase dentro de un “namespace”. Un Object Path lleno para una clase añade el nombre de la clase a un path del “namespace” como se mustra luego:

```
\ \Server\”namespace”:Class
```

Un Object Path relativo puede usarse para representar una clase que reside en el "namespace" actual. Un Object Path relativo a una clase contiene sólo el nombre de la clase:

**Clase**

Por ejemplo, el Object Path lleno siguiente puede usarse para localizar la clase Win32_LogicalDisk dentro del "namespace" root\cimv2 en un servidor llamado Admin:

```
\ \Admin\Root\CimV2:Win32_LogicalDisk
```

La misma clase puede localizarse en el servidor actual en el "namespace" actual con el Object Path relativo siguiente:

```
Win32_LogicalDisk
```

El fragmento del código siguiente muestra cómo usar el Object Path relativo para recuperar la definición de la clase para Win32_LogicalDisk:

```
BSTR Path = SysAllocString(L"Win32_LogicalDisk");
IWbemClassObject *pDiskClass = 0;
pHmmSvcs->GetObject(Path, 0, 0, &pDiskClass, 0);
```

- **Instance Object Paths (Rutas de objetos instancia)**

Un Instance Object Paths se usa para localizar una instancia de una clase particular dentro de un "namespace". Una instancia llena de object path añade el nombre y valor de la propiedad importante de la clase, como se muestra luego:

```
\ \Server\"namespace":Class.KeyName = "KeyValue"
```

Un object path relativo puede usarse para referirse a una instancia localizada en el "namespace" actual en el servidor actual. El path relativo consiste en el nombre de la clase seguido por los nombres y valores de las propiedades importantes de esta instancia:

```
MyClass.MyProp = "e":
```

Para una clase con llaves múltiples:

```
MyOtherClass.FirstKey=1,SecondKey=2
```

Para las clases con sólo una propiedad designadas como la llave, una anotación de taquigrafía puede usarse y puede omitirse el nombre de la propiedad importante:

```
MyClass = "e":
```

Para las clases del semifallo (singleton), el object path relativo consiste en el nombre de la clase seguido por " = @" como se muestra a continuación:

```
MySingletonClass = @
```

Para recuperar una instancia particular, se usa código similar al que se mustra a continuación. La llamada a la función SysAllocString asigna e inicializa a la instancia la cadena de caracteres para el Object Path. La llamada al método de IWbemServices::GetObject recupera la instancia y lugares en los volúmenes del parámetro del pComPort.

```
BSTR Path = SysAllocString(L"ComPort=2");
IWbemClassObject *pComPort = 0;
pWbemSvcs->GetObject(Path, 0, 0, &pComPort, 0);
```

Para las instancias de clases que especifican propiedades múltiples como la llave, WMI no requiere ninguna clasificación específica de propiedades importantes en object paths. Sólo se

necesita especificar el valor de cada una de las propiedades en el object path. Por ejemplo, asuma que la clase MyClass2 tiene una llave compuesta que consiste en una propiedad del entero IntVal y una propiedad del cordón StrVal. Los object paths siguientes muestran dos maneras diferentes de identificar la misma instancia de MyClass2:

```
MyClass.IntVal=33,StrVal = "AAA"

MyClass.StrVal = "AAA",IntVal=33"
```

### C.1.4 Localización de WMI

La Instrumentación de Gestión Windows (WMI) implementa una técnica que permite múltiples versiones localizadas de la misma clase que está guardada en el almacén. Para lograr esto, la definición de la clase está separada en lo siguiente:

Una versión neutral del lenguaje que contiene sólo una definición de la clase básica.

Una versión específica del lenguaje que contiene información localizada, como descripciones de propiedad que son específicas a un sitio.

Las definiciones específicas de una clase de lenguaje se guardan en un "namespace" niño bajo el "namespace" que contiene una definición básica de una clase neutral del idioma.

Cuando se pide una definición de la clase localizada para un sitio específico, WMI combina la definición de la clase básica y la información de la clase localizada para formar una clase localizada completa. Se puede conseguir una versión localizada de una clase de WMI especificando un sitio cuando se conecta a WMI y poniendo una bandera que indica que se necesita información localizada. WMI une entonces la información neutral del idioma y las versiones específicas del idioma de la definición de la clase para formar una clase localizada. Las clases de WMI que contienen información localizada son marcadas con el calificador de Enmendadura (Amendment qualifier) y se llaman clases enmendadas; una clase soporta

información localizada si tiene este calificador. Se puede determinar qué sitio de la clase se ha localizado por otro calificador llamado Sitio. El calificador sitio contiene un identificador de localización (Win32 LCID) que identifica un sitio. Por ejemplo, el sitio para inglés americano es 0x409. Si un calificador en una clase enmendada contiene información localizada, contiene entonces el calificador enmendado.

### C.1.5 Seguridad y autenticación

Windows Management Instrumentation (WMI) soporta seguridad total para sistemas operativos Microsoft® Windows NT®/Microsoft® Windows® 2000, y seguridad limitada para Windows 95/98. La seguridad WMI valida la entrada de los usuarios a la información en el computador local y para el acceso remoto. WMI garantiza a un usuario válido alguna forma de acceso controlado al repositorio CIM.

WMI no da seguridad para los recursos del sistema tales como clases o instancias individuales. Sin embargo, los administradores pueden usar WMI para controlar los permisos globales en esquemas de operación, tales como limitaciones para algunos usuarios en el acceso en operaciones de solo lectura. WMI también soporta seguridad para el "namespace".

El acercamiento a la seguridad WMI es usar un descriptor de seguridad(SD) para controlar el acceso a los servicios WMI. El SD es un descriptor estándar de Windows NT, y contiene una lista de control de acceso (ACL), una lista de control de entradas (ACEs). Cada ACE garantiza los permisos para ejecutar operaciones restringidas, tales como escribir un archivo o formatear un disco. ACEs particulares particular para permitir accesos WMI, accesos remotos, ejecución de métodos, y escritura sobre le repositorio CIMOM.

Los SDs WMI son almacenados en el repositorio CIMOM. Para Windows NT, los ACEs son almacenados. Para Windows 95/98, son almacenados unos pseudo ACEs.

Se puede usar el control de aplicaciones WMI (Wbemcntl.exe) para fijar los permisos para los usuarios WMI.

**Para ejecutar el control de la aplicación WMI**

Desde el menú Inicio, de un clic sobre ejecutar y escriba **wbemcntl** (en Windows 95/98 y Windows NT versión 4.0). En Windows 2000, escriba **wmimgmt.msc**.

WMI desarrolla chequeos de seguridad para cada llamada a WMI. Las aplicaciones pueden especificar configuraciones de seguridad en el nivel de procesamiento y en el nivel de interfase.

- **Seguridad por "namespace"**

Cada "namespace" tiene su propio descriptor de seguridad descriptor (SD), que permite al "namespace" tener sus propias configuraciones de seguridad. Como en los archivos de sistemas Microsoft® Windows NT® (NTFS), la herencia puede ser utilizada para simplificar la administración de la seguridad. Cada control de acceso de entradas (ACE) en el SD del "namespace" tiene un campo de banderas que indica que herencia, si la hay, se debe desempeñar. Por ejemplo, si el contenedor de herencia es permitido, entonces el ACE del SD del "namespace" es heredado por los "namespaces" hijos.

El SD es construido cuando una conexión al "namespace" es hecha. El SD es construido a partir de la lista de control de acceso (ACLs) del "namespaces" en la cadena de herencias, y modificado por cada herencia de cada "namespace". Por defecto, el propietario tendrá la cuenta de administrador, y el grupo de administradores locales tienen derecho a todas las operaciones, incluyendo los accesos remotos.

- **Autenticación**

Autenticación es el proceso por el cual un cliente procesa su identificación en un servidor. La Impersonalización es el proceso por el cual un cliente permite a un servidor llamar a otros servidores de su interés.

La identificación de usuario involucra la verificación del nombre de usuario y su contraseña, a través de paquetes de seguridad de sistema como NTLM. Los paquetes de seguridad no tienen información acerca de los derechos de acceso; su única responsabilidad es validar positivamente la identidad de los usuarios.

Dependiendo del sistema operativo, WMI soporta varios niveles de autenticación. En plataformas Microsoft® Windows® 95/98, todos los usuarios locales son asumidos para ser autenticados, indiferentemente del usuario y su contraseña. Aquí no hay autenticación local, y la verdadera autenticación solo ocurre sobre conexiones remotas. En sistemas Microsoft® Windows NT®/Windows 2000, todos los usuarios son autenticados. Después de la autenticación, el usuario es aun sujeto de las configuraciones de permisos que son estrictamente ejecutadas.

La asignación de permisos envuelve el acceso a los usuarios válidos y es la responsabilidad de WMI. Ya que los sistemas Windows 95 y Windows 98 no son sistemas operativos seguros, la garantización o denegación tiene menos efecto en estos sistemas operativos que en otros sistemas. Por ejemplo; en sistemas Windows NT/Windows 2000, las operaciones sobre archivos pueden negársele a un usuario así WMI considere que el usuario tiene acceso total.

Sobre Windows NT versión 4.0, el único servicio de autenticación es NTLM. Bajo Windows 2000, dos servicios de autenticación adicionales están disponibles: Kerberos y **Negotiate**. **Negotiate** es recomendado para un usuario con códigos que necesitan trabajar en dominios Kerberos y no-Kerberos.

Si el cliente y el servidor están en diferentes maquinas Windows 2000, el servicio de autenticación por defecto será **Negotiate**. En todos los otros casos, el servicio de autenticación por defecto es NTLM.

- ***Autenticación NTLM.***

NTLM esta disponible en todas las plataformas Windows. En Windows 95/98, éste esta disponible como una adición. Sobre NTLM, un servidor solo puede impersonalizar al cliente a través de un salto a una máquina simple. Un servidor no puede pasar (o delegar) la identidad del cliente a otra máquina.

- ***Autenticación Kerberos***

Kerberos solo puede ser utilizado en escenarios remotos. Si se intenta fijar a Kerberos como el paquete de autenticación para proxys locales, este fallará. Kerberos requiere autenticación mutua. En otras palabras, el cliente debe estar habilitado para identificar al servidor y el servidor debe estar habilitado para identificar al cliente. Uno de los mayores rasgos de Kerberos, aplicable a WMI, es que este permite delegación de la identidad del cliente a través de múltiples saltos de máquina. En otras palabras, una petición del cliente puede ser pasada por un servidor a otra máquina y aun mantener la identidad del cliente. Kerberos esta solo disponible sobre máquinas Windows 2000 en redes Windows 2000. Si no hay un centro de distribución de llaves (KDC) en una red, fallará el intento de usar Kerberos entre dos máquinas Windows 2000.

- ***Autenticación Negotiate***

Cuando se conectan dos máquinas Windows 2000 a WMI, la seguridad del proxy automáticamente toma el valor por defecto de **Negotiate**. El servicio de autenticación

**Negotiate** escoge el mejor paquete de autenticación para usarse. Este es el servicio recomendado para el uso en aplicaciones que deben funcionar en dominios Kerberos y no Kerberos.

- **Accesos por defecto y permisos.**

DCOM asume al usuario actual si no están especificados un nombre de usuario y contraseña. Para accesos remotos, pueden ser especificados un nombre de usuario y una contraseña sobrescribiendo al usuario actual. Para acceso local, no es posible sobrescribir el nombre de usuario puesto que DCOM no lo permite.

Por defecto, cualquier usuario que sea un miembro del grupo de administradores locales tiene garantizado el acceso local total y el acceso remoto a WMI. Este grupo típicamente incluye al administrador local y el dominio del administrador.

Los permisos de registro solo pertenecen a los sistemas Microsoft® Windows NT®/Microsoft® Windows® 2000. En estos sistemas, los permisos de registro son fijados automáticamente cuando WMI es instalado. Los permisos de lectura/escritura son asignados a los miembros del grupo administradores, y los permisos de solo lectura a cualquiera.

Nota: El intento al acceso remoto en sistemas Windows NT/Windows 2000 falla si el usuario no es un administrador para sistemas remotos y si el Servicio de Gestión de Windows no esta corriendo en el sistema remoto. Si el usuario no tiene derechos administrativos en el sistema remoto, DCOM inicia el Servicio de Gestión de Windows en el sistema remoto y permite a los usuarios conectarse, por lo que todas las credenciales de conexión son aceptadas.

## C.2 Programación de aplicaciones WMI

### C.2.1 Scripting API

Se puede usar el Scripting API del Windows Management Instrumentation (WMI) para desarrollar script y aplicaciones Microsoft® Visual Basic® que pueden ser usadas para ver información de objetos gestionados o para controlarlos. WMI provee soporte scripting para los siguientes lenguajes:

- Microsoft® Visual Basic®
- Microsoft® Visual Basic® para aplicaciones
- Microsoft® Visual Basic® Scripting Edition (VBScript)
- Microsoft® Jscript®

- **Acerca del Scripting API**

El Scripting API del Windows Management Instrumentation (WMI) es un conjunto de objetos con propiedades, métodos y eventos que pueden ser usados para desarrollar scrip y aplicaciones Microsoft® Visual Basic® que ven o controlan información de los objetos gestionados.

  - **Convenciones del documento.**

La referencia Scripting API para WMI usa las siguientes convenciones:

- Los tipos de los parámetros son definidos utilizando el prefijo: b (Booleano), str (Cadena), i (Entero), obj (Objeto de automatización), var (Variable).
- Los parámetros opcionales son puestos entre paréntesis cuadrados ([ ]) con sus valores por defecto.

- En el caso de los parámetros de objetos, los caracteres después del prefijo "obj." indican el tipo de objeto esperado:
    - *WbemObject* para un objeto SWbemObject.
    - *WbemObjectSet* para un objeto SWbemObjectSet.
    - *WbemObjectPath* para un objeto SWbemObjectPath.
    - *WbemNamedValueSet* para un objeto SWbemNamedValueSet.
    - *WbemServices* para un objeto SWbemServices.
    - *WbemLocator* para un objeto SWbemLocator.
    - *WbemMethod* para un objeto SWbemMethod (por ejemplo, **SWbemMethodSet.Item**).
    - *WbemProperty* para un objeto SWbemProperty (por ejemplo, SWbemPropertySet.Item).
    - *WbemQualifier* para un objeto SWbemQualifier (por ejemplo, **SWbemQualifierSet.Item**).
    - *WbemPropertySet* para un objeto SWbemPropertySet (por ejemplo, **SWbemObject.Properties_**).
    - *WbemQualifierSet* para un objeto SWbemQualifierSet (por ejemplo, **SWbemObject.Qualifiers_**).
    - *WbemMethodSet* para un objeto SWbemMethodSet (por ejemplo, **SWbemObject.Methods_**).
    - *WbemEventSource* para un objeto SWbemEventSource (por ejemplo, **SWbemServices.ExecNotificationQuery**).
    - *WbemPrivilegeSet* para un objeto SWbemPrivilegeSet.

- *WbemPrivilege* para un objeto SWbemPrivilege.

Por ejemplo, el siguiente codigo describe un método **Something** de un objeto SWbemServices:

```
objWbemObjectSet = objWbemServices.Something(
  strParam,
  [iFlags=0],
  [objWbemNamedValueSet=null]
);
```

Este método retorna un objetoSWbemObjectSet y toma tres parámetros, de los cuales solo el primero es esencial. El primer parámetro es una cadena, el Segundo es un entero opcional (can valor por defecto igual a cero), y el tercero es un objeto SWbemNamedValueSet opcional (con valor por defecto igual a nulo).

- **Utilizando el Sripting API**

  - **Colecciones**

Una colección es un concepto de automatización estándar que provee una interfaz uniforme para un conjunto de objetos sobre los que se pueden desempeñar una iteración. La manipulación de una colección utilizando un lenguaje de programación específico usualmente es llevada a cabo utilizando rasgos nativos para este lenguaje.

Esto simplifica las tareas de repartición con objetos que contienen otros objetos. Se puede usar los paradigmas para acceder a las colecciones a menos que se utilicen objetos soportando una interfaz diferente.

El WMI API para scripting provee un número de interfaces que conforman la colección. En cada caso, los elementos de la colección pueden ser identificados utilizando una cadena conveniente.

| Colección | Elemento | Item() Aceptado: |
|---|---|---|
| SwbemObjectSet | SWbemObject | Object path |
| SwbemPropertySet | SWbemProperty | Property name |
| SwbemMethodSet | SWbemMethod | Method name |
| SwbemNamedValueSet | SWbemNamedValue | Value name |
| SwbemPrivilegeSet | SWbemPrivilege | Privilege name |

- **Soporte de Visual Basic y VBScript para colecciones**

En Visual Basic o VBScript, las colecciones son accesadas utilizando el mecanismo "For Each *member* in *collection*". Por ejemplo:

```
'Aquí MyDiskProperties es un objeto SWbemPropertySet,
'y cada Property es una SWbemProperty
For Each Property in MyDiskProperties
    WScript.Echo Property.Name
Next
```

- **Creación de Objetos y Monikers**

En el Modelo del Objeto Común (COM), un moniker es un mecanismo normal por encapsular la localidad y se liga con otro objeto COM. La representación textual de un moniker es llamada " nombre del despliegue" (display name).

Se puede usar el WMI Scripting API para crear objetos SWbemObject y SWbemServices de una manera concisa que usa moniker display names. Esto es proporcionado como una alternativa más simple para el mecanismo de crear un objeto SWbemLocator primero y conectar explícitamente a WMI utilizando llamadas de método adicionales para obtener la interface que se necesita.

El moniker display name simplemente es una cadena leíble que actúa como un nombre para un "namespace" de WMI, o una clase de WMI o instancia. El formato de esta cadena es muy similar a un object path WMI estandar (para más información, ver Object Paths). Un moniker tiene las siguientes partes:

- El prefijo WinMgmts: (obligatorio)
- Un componente security settings (Opcional)
- Un componente WMI object path (Opcional)

La muestra de BNF siguiente describe la sintaxis del moniker para la descarga actual de WMI para Microsoft® Windows® 2000.

```
wmiMoniker : "winmgmts:" securitySetting ["[" localeSetting
"]"] ["!" objectPath]
| "winmgmts:" "[" localeSetting "]" ["!" objectPath]
| "winmgmts:" [objectPath]

localeSetting : "locale" <ows> "=" <ows> localeID

localeID : a value of the form "ms_xxxx" where xxxx is a hex
LCID value e.g. "ms_409".

objectPath : a valid WMI Object Path
```

```
securitySetting : "{" <ows> authAndImpersonSettings [<ows> ","
<ows> privilegeOverrides]<ows> "}" <ows>
| "{" <ows> privilegeOverrides <ows> "}"

authAndImpersonSettings :
authenticationLevel
| impersonationLevel
| authority
| authenticationLevel <ows> "," <ows> impersonationLevel
[<ows> "," <ows> authority]
| authenticationLevel <ows> "," <ows> authority [<ows> ","
<ows> impersonationLevel]
| impersonationLevel <ows> "," <ows> authenticationLevel
[<ows> "," <ows> authority]
| impersonationLevel <ows> "," <ows> authority [<ows> ","
<ows> authenticationLevel]
| authority <ows> "," <ows> impersonationLevel [<ows> ","
<ows> authenticationLevel]
| authority <ows> "," <ows> authenticationLevel [<ows> ","
<ows> impersonationLevel]
authority : "authority" <ows> "=" <ows> authorityValue
authorityValue : Any valid WMI authority string e.g.
"kerberos:mydomain\server" or "ntlmdomain:mydomain". Note that
backslashes need to be escaped in JScript.
authenticationLevel : "authenticationLevel" <ows> "=" <ows>
authenticationValue

authenticationValue : "default" | "none" | "connect" | "call"
| "pkt" | "pktIntegrity" | "pktPrivacy"

impersonationLevel : "impersonationLevel" <ows> "=" <ows>
impersonationValue
```

```
impersonationValue : "anonymous" | "identify" | "impersonate"
| "delegate"

privilegeOverrides : "(" <ows> privileges <ows> ")"

privileges : privilege [<ows> "," <ows> privileges <ows>]*

privilege : ["!"] privilegeName

privilegeName : "CreateToken" | "PrimaryToken" | "LockMemory"
| "IncreaseQuota"
| "MachineAccount" | "Tcb" | "Security" | "TakeOwnership"
| "LoadDriver" | "SystemProfile" | "SystemTime"
| "ProfileSingleProcess" | "IncreaseBasePriority"
| "CreatePagefile" | "CreatePermanent" | "Backup" | "Restore"
| "Shutdown" | "Debug" | "Audit" | "SystemEnvironment" |
"ChangeNotify"
| "RemoteShutdown" | "Udock" | "SyncAgent" |
"EnableDelegation"
```

Nota: Todos los literales de cadena son caso-insensibles.

El prefijo "!" es un privilegio que indica que el privilegio será desactivado; la omisión de este prefijo implica que el privilegio será habilitado.

El moniker siguiente especifica el objeto SWbemServices que representa el "namespace" root\default, con personificación y el wbemPrivilegeDebug (SeDebugPrivilege) privilegio habilitado, y el wbemPrivilegeSecurity (SeSecurityPrivilege) privilegio desactivado.

```
winmgmts:{impersonationLevel=impersonate,(debug,    !security)}
!root\default
```

**Componente Object Path**

Las asignaciones predefinidas siguientes permiten especificar el Object Path:

El servidor puede omitirse del Object Path en el caso que el servidor local sea supuesto.

Los "namespace" pueden omitirse del Object Path en el caso que el "namespace" predefinido sea supuesto. Esto esta determinado por el valor del registro HKEY_LOCAL_MACHINE\Software\Microsoft\WBEM\Scripting\Default "namespace", el valor predefinido de root\cimv2.

Una clase o instancia también pueden especificarse en el caso en el cual el object retornado es un objeto de WMI en lugar de un objeto de servicios.

Nota: Si una clase o instancia se especifica, no se puede omitir el "namespace" al especificar el servidor.

Aquí están algunos ejemplos de valid moniker display names.

**WinMgmts:**
Esto identifica el "namespace" predefinido en la computadora local. Un objeto SwbemServices es retornado.

**WinMgmts://myServer**
Esto identifica el "namespace" predefinido en el servidor. Un objeto SwbemServices es retornado.

**WinMgmts://myServer/root/cimv2**

Esto identifica el "namespace" root/cimv2 en el servidor del myServer. Un objeto SwbemServices es retornado.

**WinMgmts:root/cimv2**

Esto identifica el "namespace" root/cimv2 en el servidor local. Un objeto SwbemServices es retornado.

**WinMgmts:{impersonationLevel=impersonate}! /**
**/myServer/root/cimv2:Win32_LogicalDisk**

Esto identifica la clase Win32_LogicalDisk en el "namespace" root/cimv2 en el servidor myServer. Un objeto SWbemObject es retornado.

**WinMgmts:{impersonationLevel=impersonate} !root/cimv2:Win32_LogicalDisk**

Esto identifica la clase Win32_LogicalDisk en el "namespace" root/ en el servidor local. Un objeto SwbemObject es etornado.

**WinMgmts:{impersonationLevel=impersonate} !Win32_LogicalDisk**

Esto identifica la clase Win32_LogicalDisk en el "namespace" predefinido en el servidor local. Un objeto SWbemObject es retornado.

**WinMgmts::Win32_LogicalDisk = ' C: '**

Esto identifica la instancia de Win32_LogicalDisk que corresponde al drive C: en el "namespace" del scripting predefinido en el servidor local. Un objeto SWbemObject es retornado. El "namespace" predefinido para el scripting es determinado por el setting de configuración del "namespace" predefinido (por defecto).

**WinMgmts:{impersonationLevel=impersonate}! /**
**/myServer/root/cimv2:Win32_LogicalDisk = "C":**

Esto identifica la instancia de Win32_LogicalDisk que corresponde al drive C: en el "namespace" root/cimv2 en el servidor del myServer. Un objeto SWbemObject es retornado.

**WinMgmts:{impersonationLevel=impersonate} !root/cimv2:Win32_LogicalDisk = "C":**
Esto identifica la instancia de Win32_LogicalDisk que corresponde al drive C: en el "namespace" root/cimv2 en el servidor local. Un objeto SwbemObject es retornado.

**WinMgmts:{impersonationLevel=impersonate} !Win32_LogicalDisk = "C":**
Esto identificala instancia de Win32_LogicalDisk que corresponde al drive C: en el "namespace" predefinido en el servidor local. Un objeto SwbemObject es retornado.

**WinMgmts:{impersonationLevel=impersonate, (Ponga a punto)}**
Este moniker pone el nivel de personificación en personificar y pone el privilegio Microsoft® Windows NT®/Microsoft® Windows® 2000 SE_DEBUG.

**WinMgmts:{impersonate,(Debug, !RemoteShutdown)}**
Este moniker pone el nivel de personificación en personificar y pone el privilegio Windows NT/Windows 2000 SE_DEBUG. También revoca el Windows NT/Windows 2000 SE_SHUTDOWN privilege.

**WinMgmts:[locale=ms_409] !root/wmi:myclass**
Este moniker recupera descripciones de inglés americano localizadas para la clase myclass del "namespace" del root/wmi.

**Winmgmts:{impersonationLevel=delegate,authority=kerberos:mydomain\server}! / /myserver/root/default:__cimomidentification = @**
Este moniker pide autenticación de Kerberos utilizando el mydomain\server principal.

**Winmgmts:{impersonationLevel=impersonate,authority=ntlmdomain:mydomain}! / /myserver/root/default:__cimomidentification = @**
Este moniker pide autenticación de NTLM utilizando el dominio mydomain.

Visual Basic y VBScript tienen soporte para los Nombres del Despliegue (Display Names).

En Visual Basic o VBScript, pueden crearse objetos utilizando nombres moniker desplay que usan el mecanismo de GetObject.

Nota: Aunque los monikers proporcionan acceso más directo a los objetos, en ciertas circunstancias, el uso repetido de monikers puede ser menos eficaz que el código equivalente que explícitamente conecta a WMI. Si el desempeño de la aplicación es una consideración, se podría considerar utilizar los mecanismos alternativos.

No es posible usar la función de GetObject proporcionado por VBScript y JScript cuando los scripts incluídos estan corriendo dentro de una página HTML, ya que Microsoft® Internet Explorer desaprueba el uso de esta llamada por razones de seguridad.

- **Utilizando la librería de WMI.**

  - **Utilizando la librería WMI Scripting con visual basic y Visual InterDev**

Para usar el Scripting API para WMI con Visual Basic y Visual InterDev, se debe adicionar la librería WMI scripting a la lista de referencias disponibles para los proyectos.

Se puede usar el siguiente procedimiento para hacer ambientes de desarrollo integrado (IDE) utilizando la librería WbemScripting.

**Para adicionar la librería WMI Scripting a las referencias de los proyectos:**

1. Seleccionar **Referencias** desde el menú **Proyectos**.
2. En el cuadro de diálogo **Referencias Disponibles**, seleccionar Microsoft WBEM Scripting V1.0 Library o Microsoft WMI Scripting V1.1 Library (siempre que este aparezca).

3. Si esta opción no está disponible en la lista de **Referencias disponibles**, se adiciona utilizando el botón **Examinar** en el cuadro de diálogo **Referencias**. El botón **Explorar** abre el cuadro de diálogo **Adicionar Referencias**, lo que habilita localizar la librería WbemScripting. La librería WbemScripting reside en el archivo Wbemdisp.tlb en el directorio de instalación de WBEM. En computadores con windows98, esta librería es localizada en el directorio \Windows\System\Wbem. En computadores con sistemas operativos Microsoft® Windows NT®/Windows 2000, esta librería está localizada en el directorio \winnt\system32\wbem.

Seleccionar el archivo y hacer clic en **abrir**. La librería Microsoft WBEM Scripting V1.0 o la librería Microsoft WMI Scripting V1.1 aparecerá en la lista de referencias.

- **Haciendo una llamada asíncrona utilizando el Scripting API**

Muchos de los métodos en el Scripting API para WMI están disponibles tanto como métodos sincrónicos y asincrónicos. Los programas que llaman a los métodos sincrónos generalmente no pueden continuar o responder a las entradas de los usuarios hasta que dicha llamada sincróna este completa. Para prevenir que esto pase, la mayoría de los métodos síncronos permiten especificar el parámetro wbemFlagReturnImmediately de *iFlags* que instruye a la llamada para que retorne inmediatamente aun si los resultados no están disponibles.

Los métodos asíncronos retornan inmediatamente después de iniciada la llamada. Esto hace que los programas puedan continuar haciendo otras cosas. Cuando la llamada original está completa, esta llama al evento OnCompleted de un objeto sink en el programa y ejecuta el código que se haya puesto allí para procesar el resultado de la llamada. Se usan los métodos asíncronos para cualquier llamada que normalmente toma un largo periodo en completarse, tales como una consulta compleja o una notificación de eventos, con lo que el programa puede responder de una mejor manera.

Nota: Se puede usar las llamadas asíncronas en páginas HTML pero no desde ASPs

Los siguientes pasos describen como hacer una llamada asincrónica utilizando Visual Basic.

**Para hacer una llamada utilizando Visual Basic.**

1. En la sección de declaraciones generales del código, se declara un objeto sink del tipo **SWbemSink** utilizando la palabra clave **WithEvents**, esto se muestra a continuación:

   ```
   Dim WithEvents sink As SWbemSink
   ```

2. En una subrutina en la forma, se declara un objeto tipo SWbemLocator y se crea un objeto SWbemSink como se muestra a continuación:

   ```
   Dim loc As New SWbemLocator
   Set sink = New SWbemSink
   ```

3. Entrar a WMI y fijar el nivel de impersonalización, como se muestra a continuación:

   ```
   Set services = loc.ConnectServer ' conectarse
   services.Security_.ImpersonationLevel =
   wbemImpersonationLevelImpersonate
   ```

4. Si se hacen múltiples llamadas asíncronas utilizando el mismo objeto sink, hay que asegurarse de proveer un objeto del tipo SWbemNamedValueSet para el parámetro *objWbemAsyncContext* que pueda identificar la fuente de la llamada, como se muestra a continuación:

   ```
   Dim cntxt As Object
   Set cntxt = New SwbemNamedValueSet   ' crea un nuevo
   named value set
   cntxt.Add "AsyncId", "0"              'Adiciona un valor
   de nombre para identificar la llamada.
   ```

5. Llamar al método asíncrono y proveer un objeto sink como el primer parámetro, tal como se muestra a continuación:

```
services.ExecQueryAsync sink, "SELECT * FROM
win32_process",,,,cntxt
```

6. Poner el código que procesa la información retornada desde la llamada asíncrona en el evento apropiado del objeto **SWbemSink**. Hay cuatro eventos definidos para el objeto **SWbemSink**. **OnObjectReady** es disparado cuando la llamada asíncrona retorna un objeto. Se puede poner código para procesar las instancias o las clases retornadas desde la llamada asincrónica como ExecQueryAsync para procesar los objetos tal como son enviados. Por ejemplo, si se necesita poner el resultado de una llamada asíncrona **ExecQueryAsync** en una lista, poner el siguiente código en el evento **OnObjectReady**:

```
Private Sub sink_OnObjectReady(ByVal objWbemObject As
WbemScripting.ISWbemObject, ByVal objWbemAsyncContext As
WbemScripting.ISWbemNamedValueSet)

    Set cntxt_item = objWbemAsyncContext.Item("AsyncId")
    If cntxt_item.Value = 0 Then          'Chequee para
ver que evento es enganchado
        List1.AddItem (objWbemObject.Name)
     End If
End Sub
```

7. Poner el código que finalizará una operación asíncrona en el evento **OnCompleted**, como se muestra a continuación:

```
Private Sub sink_OnCompleted(ByVal iHResult As
WbemScripting.WbemErrorEnum, ByVal objWbemErrorObject As
WbemScripting.ISWbemObject, ByVal objWbemAsyncContext As
WbemScripting.ISWbemNamedValueSet)

    MsgBox ("Asynchronous operation complete")
End Sub
```

Se puede examinar el parámetro *iHresult* retornado en el evento OnCompleted. Este parámetro indica si la llamada fue exitosa, o si un error ocurrió. En caso de que sea exitosa, el valor que está en el parámetro *iHresult* debería ser igual a cero. Cualquier otro valor indica un error y se deben chequear los valores de error que son retornados en el parámetro *objWbemErrorObject*.

Nota: No se necesita crear un objeto de error WMI cuando hay un error como se hace con los métodos síncronos. El error se devolverá en el parámetro *objWbemErrorObject*.

Los siguientes pasos describen como hacer una llamada asíncrona utilizando VBSctipt.

**Para hacer una llamada asíncrona utilizando VBScript**

1. Declarar un servicio y un objeto sink, como se muestra a continuación:

```
Dim service
Dim sink
```

2. Crear UNA subrutina para cada evento asíncrono que se dispare. Se debe poner código en estas subrutinas para manejar cada evento que sea recibido, como se muestra a continuación:

```
Sub SINK_OnCompleted(iHResult, objErrorObject,
objAsyncContext)
WScript.Echo "Asynchronous operation is done."
End Sub

Sub SINK_OnObjectReady(objObject, objAsyncContext)
    WScript.Echo (objObject.Name)
End Sub
```

   Se puede examinar el parámetro *iHresult* retornado por el evento OnCompleted. Este parámetro dirá si la llamada asíncrona fue exitosa, o si un error ocurrió. Si es exitosa,

el valor retornado en el parámetro *iHresult* debería ser igual a cero. Cualquier otro valor puede indicar que ocurrió un error y se debe chequear el valor del error que es retornado en el parámetro *objErrorObject*.

3. Conectarse con WMI y conseguir un objeto services, como se muestra a continuación:

```
Set service = GetObject("winmgmts:")
```

4. Crear el objeto sink, utilizando **WScript.CreateObject** (for Windows Scripting Host 2.0 only)

```
Set sink =
WScript.CreateObject("WbemScripting.SWbemSink","SINK_")
o
<OBJECT progid="WbemScripting.SWbemSink" id="SINK"
events="true"/>
```

5. Fijar el nivel de impersonalización tal como se muestra a continuación.

```
service.Security_.ImpersonationLevel = 3
```

6. Hacer la llamada asíncrona y pasar el objeto sink como un parámetro como se muestra a continuación:

```
service.InstancesOfAsync sink, "Win32_process"
```

7. Hacer una llamada que prevenga al script de finalizar antes que todos los eventos sean recibidos. Una forma simple es usar el comando Windows Scripting Host **Echo** así:

```
WScript.Echo "Waiting for instances."
```

Cuando se ejecute este script se debe ver la primera instancia retornada antes del mensaje **Waiting for events**, o si se ve después. Esta es la naturaleza del procesamiento de las llamadas asíncronas. Si se cierra el cuadro de dialogo **Waiting for events** muy pronto, no se podrá ver ninguna instancia.

Los siguientes pasos describen como cancelar una llamada asíncrona.

**Para cancelar una llamada asíncrona**

1. Se puede cancelar una llamada asíncrona utilizando el método **Cancel**, dicho método hace que todas las llamadas asíncronas asociadas con un objeto sink sean canceladas. Además, se debe usar sink separados para las operaciones asíncronas que necesitan ser independientes.

2. Se debería liberar el objeto sink asignándole un valor **Nothing**.

El siguiente ejemplo demuestra como cancelar una llamada asíncrona.

```
objwbemsink.Cancel()
set objwbemsink= Nothing
```

### C.3 Referencia WMI

#### C.3.1 Clases del sistema WMI

- **_”namespace”**

El __namespace system class representa un “namespace” de WMI.

**class __namespace : __SystemClass { [key] string** *Name***; };**

**Propiedades**

La tabla siguiente lista las propiedades para __namespace.

| Propiedad | Tipo de Dato | Descripción |
|---|---|---|
| **Name** | **string** | Key property. “namespace” name. |

**Comentarios**

El __namespace class se deriva de __SystemClass.

Se puede usar __namespace para identificar, crear, y anular “namespaces” niño dentro del “namespace” del funcionamiento actual para el que se tiene un objeto IWbemServices. Cuando un nueva instancia de __namespace esta dentro de cualquier “namespace” de funcionamiento ( working “namespace”) crea un “namespace” niño dentro del “namespace” del funcionamiento. Recíprocamente, si se anula una instancia de __namespace se quita el “namespace” niño del “namespace” del funcionamiento. Notese que también anulando un “namespace” niño se anulan todas sus clases instancias.

Las instancias enumeradas de esta clase dentro de cualquier “namespace” de funcionamiento dan el “namespaces” niño disponible.

Por ejemplo, dentro del “namespace” Raíz están dos casos de __namespace. Uno tiene puesto su Nombre propio " por defecto," el otro tiene puesto el Nombre para "Cimv2". Estas instancias representan el Root\Default, y Root\Cimv2 “namespaces”, respectivamente.

- **_System Class**

El __SystemClass system class es la clase baja abstracta de la que la mayoría de las clases del sistema se derivan. No pueden crearse instancias de esta clase.

```
[abstract]
```

```
 class __SystemClass
{
 };
```

**Propiedades**

Esta clase no tiene parámetros.

### C.4 Scripting API for WMI

La referencia de scripting Windows Management Instrumentation (WMI) contiene la definición para el WMI Scripting API. Esta referencia provee información para desarrolladores quienes están escribiendo aplicaciones con Microsoft® Visual Basic®, Microsoft® Visual Basic® para aplicacionesApplications, Microsoft® Visual Basic® Scripting Edition (VBScript), o Microsoft® JScript®.

La siguiente tabla describe los objetos WMI scripting y como ellos son utilizados.

| **Objeto** | **Descripción** |
|---|---|
| SwbemEventSource | Recibe eventos junto con **SWbemServices.ExecNotificationQuery**. |
| SwbemLocator | Obtiene un objeto **SWbemServices** que puedo conseguir el acceso a WMI en un computador en particular. |
| SWbemMethod | Contine una definición simple de un método WMI. |
| SWbemMethodSet | Contine una colección de objetos **SWbemMethod**. |
| SWbemNamedValue | Contiene un valor simple de nombre. |
| SWbemNamedValueSet | Consigue acceso a una colección de objetos **SWbemNamedValue**. |
| SWbemObject | Contine y manipula una clase o una instancia de un objeto WMI. |

| | |
|---|---|
| SWbemObjectPath | Genera y valida una ruta de un objeto. |
| SWbemObjectSet | Consigue una colección de objetos **SWbemObject**. |
| SWbemPrivilege | Fija y clarea un privilegio. |
| SWbemPrivilegeSet | Consigue acceso a una colección de objetos **SWbemPrivilege**. |
| SWbemProperty | Contiene una propiedad WMI simple. |
| SWbemPropertySet | Consigue el acceso a una colección de objetos **SWbemProperty**. |
| SWbemQualifier | Contains a single qualifier. |
| SWbemQualifierSet | Gets access to a collection of **SWbemQualifier** objects. |
| SWbemSecurity | Maneja configuraciones de seguridad como el modelo Distributed Component Object Model (DCOM) **Privileges**, **AuthenticationLevel**, y **ImpersonationLevel**. |
| SwbemServices | Crea, actualiza, y recibe instancias o clases. |
| SWbemSink | Recibe el resultado de operaciones asincrónicas y notificaciones de eventos,que son usados por aplicaciones clientes. |

### C.4.1 SwbemEventSource

El objeto **SWbemEventSource** devuelve eventos desde una consulta de eventos en conjunción con **SWbemServices.ExecNotificationQuery**. Se puede obtener un objeto **SWbemEventSource** si se hace una llamada a **WbemServices.ExecNotificationQuery** para hacer una consulta de eventos. Se puede entonces usar el método **NextEvent** para recibir los eventos cada vez que estos lleguen.

#### Métodos

La siguiente tabla muestra la lista de eventos para el objeto **SwbemEventSource.**

| **Método** | **Descripción** |
|---|---|
| NextEvent | Usado para recibir un evento en conjunción con |

| | |
|---|---|
| | SWbemServices.ExecNotificationQuery. |

**Propiedades**

La siguiente tabla lista las propiedades del objeto **SWbemEventSource**.

| **Propiedades** | **Descripción** |
|---|---|
| Security_ | Usado para leer o cambiar las configuraciones de seguridad.. |

- **SWbemEventSource.NextEvent**

Si un evento está disponible, el método **NextEvent** del objeto **SWbemEventSource** recibe el evento desde una consulta de eventos.

La siguiente sintaxis está en un lenguaje neutral. Para más explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemObject = objwbemEventsource.NextEvent(
[iTimeoutMs = wbemTimeoutInfinite]
);
```

**Parámetros**

**iTimeoutMs**

Opcional. Número de milisegundos que la llamada espera un evento antes de retornar un error time-out. El valor por defecto para este parámetro es **wbemTimeoutInfinite** (-1), el cual hace que la llamada espere indefinidamente.

**Valores retornados**

Si el método **NextEvent** es exitoso, este retorna un objeto SWbemObject que contiene el evento requerido. Si la llamada sobrepasa el tiempo de espera, el objeto retornado es NULO y un error es llamado.

**Códigos de error**

El método **NextEvent** puede retornar los códigos de error listados a continuación.

| Error | Significado |
|---|---|
| **wbemErrTimedOut** 0x80043001 | El evento requerido no llegó en el tiempo especificado en *iTimeoutMs*. |

- **SWbemEventSource.Security_**

La propiedad **Security_** del objeto **SWbemEventSource** lee o fija las configuraciones de seguridad de un objeto**SWbemEventSource**. Esta propiedad es un objeto SWbemSecurity.

La siguiente sintaxis está en un lenguaje neutral. Para más explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemSecurity = objwbemEventSource.Security_
```

**C.4.2 SWbemLocator**

Se puede usar los métodos del objeto **SWbemLocator** para obtener un objeto **SWbemServices** que representa una conexión a un "namespace" en un computador local o remoto. Se puede usar entonces los métodos del objeto **SWbemServices** para acceder WMI.

**Métodos**

La siguiente tabla lista el método para el objeto **SWbemLocator**.

| Método | Descripción |
|---|---|
| **ConnectServer** | Conecta a WMI en el computador especificado |

**Propiedades**

La siguiente tabla lista la propiedad para el objeto **SWbemLocator**:

| Propiedad | Descripción |
|---|---|
| **Security_** | Usada para leer o cambiar las configuraciones de seguridad. |

- **SWbemLocator.ConnectServer**

El método **ConnectServer** del objeto **SWbemLocator** conecta al "namespace" con el computador especificado en el parámetro "strServer". Éste puede ser un computador designado local o remoto, típicamente con un sistema operativo Microsoft® Windows NT®/Windows® 2000 o Microsoft® Windows® 95/98. El computador designado debe tener WMI instalado. Si el método tiene éxito, un objeto **SWbemServices** es regresado para que sea unido al "namespace" en el computador "host".

La siguiete sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemServices = objwbemLocator.ConnectServer(
[strServer=""],
[strNameSpace=""],
[strUtilicer=""],
[strPassword=""],
```

```
[strLocale=""],
[strAuthority=""],
[iSecurityFlags=0],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strServer**

Opcional. Para el acceso a un computador remoto utilizando DCOM, este parámetro especifica el nombre del computador. Un ejemplo es "myserver", si no se proporciona este parámetro, la llamada tiene como valor predefinido el computador local.

**strNameSpace**

Opcional. Cadena que especifica el "namespace" al que se conecta. Por ejemplo, para conectarse al "namespace" del root\default, utilice "root\default". Si no se especifica este parámetro, tiene como valor predefinido el "namespace" que se ha configurado como el "namespace" predefinido para el scripting. Para más información sobre el valor predefinido para este parámetro, ver Creación del Objeto y Monikers.

**strUtilicer**

Opcional. Cadena que especifica el nombre del usuario para usar mientras se intenta conectar. Esto puede estar en la forma de un nombre de usuario o un nombre de ususario de Dominio. Se deja en blanco para usar el nombre del usuario actual. El parámetro del strUtilicer sólo debe ser usado con conexiones a los servidores de WMI remotos. Si se intenta especificar strUtilicer para una conexión de WMI local, los intentos de conexión fallan.

**strPassword**

Opcional. Cadena que especifica la contraseña para usar al intentar conectarse. Se deja en blanco para usar la contraseña del usuarioactual. El parámetro strPassword sólo debe ser usado con conexiones a los servidores de WMI remotos. Si se intenta especificar strPassword para una conexión de WMI local, los intentos de conexión fallan.

**strLocale**

Opcional. Cadena que especifica el código de la localidad. Si se quiere usar el sitio actual, se deja en blanco. Si no está el espacio en blanco, este parámetro debe ser una cadena que indica el sitio deseado en el que la información debe recuperarse. Para los identificadores locales Microsoft, el formato de la cadena es "MS_xxxx", donde el xxxx es una cadena en forma de hexadecimal que indica el LCID.

Por ejemplo, en inglés americano aparecería como "MS_409"

**strAuthority**

Opcional. Si el parámetro **strAuthority** empieza con la cadena "kerberos":, se usa la autenticación de Kerberos y este parámetro debe contener un nombre principal Kerberos. Si

el parámetro del strAuthority contiene algún otro valor, se usa la autenticación de NTLM y este parámetro debe contener un nombre del dominio NTLM. Si se deja este parámetro en blanco el sistema operativo negocia con DCOM para determinar si se usa la autenticación NTLM o Kerberos. Este parámetro sólo debe ser usado con conexiones a los servidores de WMI remotos. Si se intenta poner la autoridad para una conexión de WMI local, los intentos de conexión fallarán.

**iSecurityFlags**

Opcional. Reservado y si se especificó debe ser cero.

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si es exitoso, un objeto **SWbemServices** es unido al "namespace" especificado.

**Códigos del error**

El método de **ConnectServer** puede devolver los códigos de error listados en la siguiente tabla.

| **Error** | **Descripción** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El nombre del usuario y contraseña actual especificados no son válidos o no están autorizados para hacer la conexión. |

| | |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **WbemErrInvalidNamespace** 0x8004100E | Los "namespace" especificados no existen en el servidor. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado, o el "namespace" no pudo ser compilado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |
| **wbemErrTransportFailure** 0x80041015 | Ocurió un error de gestión de redes y previene el funcionamiento normal. |

**Comentarios**

Un código de error Hex 0x80070005 indica un error de acceso denegado DCOM.

Al conectar a WMI en un sistema remoto Windows 95 desde un sistema Windows NT/Windows 2000 hay dos problemas para considerar.

Si el sistema local está proporcionando credenciales (su nombre del usuario y contraseña) de un dominio que no es un miembro de las enumeraciones síncronas, las consultas pueden tomar más tiempo para procesarse.

Si se proporciona credenciales en el proceso de conectar a WMI, la conexión falla y despliega los siguientes errores:

El Servidor de RPC esta indisponible.

- **SWbemLocator.Security_**

La propiedad **Security_** del objeto **SWbemLocator** se usa para leer o poner las configuraciones de seguridad para un objeto **SWbemLocator**. Esta propiedad es un objeto de SWbemSecurity.
La siguiente sintaxis está en un lenguje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemSecurity = objwbemLocator.Security_
```

**Comentarios**

Las propiedades de un objeto **SWbemLocator.Security_** no tienen valores predefinidos. Si se intenta conseguir el valor de **SWbemSecurity.AuthenticationLevel** o **SWbemSecurity.ImpersonationLevel** en un objeto **SWbemLocator** antes de que se lo haya puesto, resulta un error wbemErrFailed.

### C.4.3 SWbemMethod

Se pueden usar las propiedades del objeto **SWbemMethod** para inspeccionar una sola definición del método de un objeto WMI.

Este objeto puede usarse para inspeccionar las definiciones de los métodos. Para invocar los métodos, se debe usar cualquier acceso directo (ver Acceso Directo) en un objeto SWbemObject (que es el mecanismo recomendado), o la llamada de SWbemServices.ExecMethod.

Nota: En esta versión del API, se escribe access para la información del método que no está soportada. Si se quiere definir métodos o modificarlos existiendo definiciones del método, se puede definir los cambios del método en un archivo MOF y puede someter los cambios utilizando al Compilador MOF. Alternativamente, se puede usar el COM API para WMI.

**Métodos**

Este objeto no tiene ningún método.

**Propiedades**

La tabla siguiente lista las propiedades para el objeto **SWbemMethod**.

| Propiedad | Descripción |
|---|---|
| InParameters | Un objeto SWbemObject cuyas propiedades definen los parámetros de la entrada para este método. |
| Name | Nombre del método. |
| Origin | Clase originaria del método. |
| OutParameters | Un objeto de SWbemObject cuyas propiedades definen el fuera los parámetros y tipo del retorno de este método. |
| Qualifiers | Un objeto SWbemQualifierSet que contiene los calificadores para este método. |

- **SWbemMethod.In Parameters**

La propiedad InParameters del objeto **SWbemMethod** es un objeto SWbemObject cuyas propiedades definen los parámetros de entrada para este método. Esta propiedad es sólo de lectura. Notese que cualquier cambio hecho a este objeto no se refleja en la definición del método subyacente.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objInParameters = objwbemMethod.InParameters
```

- **SWbemMethod.Name**

La propiedad del **Nombre** del objeto de **SWbemMethod** es un cordón que describe el nombre de este método. Esta propiedad es solo de lectura.

```
strMethodName = objwbemMethod.Name
```

- **SWbemMethod.Origin**

La propiedad **Origin** del objeto **SWbemMethod** retorna el nombre de la clase en la que este método fue introducido. Para las clases con jerarquías de herencia profundas, es a menudo deseable saber qué métodos fueron declarados en qué clases. Esta propiedad es sólo de lectura.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
strClassOriginName = objwbemMethod.Origin
```

- **SWbemMethod.OutParameters**

La propiedad **OutParameters** del objeto **SWbemMethod** es un objeto SWbemObject cuyas propiedades definen los parámetros de salida y tipo de retorno de este método. Esta propiedad es sólo de lectura. Note que cualquier cambio hecho a este objeto no se refleja en la definición del método subyacente.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objOutParameters = objwbemMethod.OutParameters
```

- **SWbemMethod.Qualifiers_**

La propiedad **Qualifiers_** del objeto **SWbemMethod** retorna un objeto SWbemQualifierSet que es la colección de calificadores para este método. Esta propiedad es sólo de lectura.

```
objQualifierSet = objwbemMethod.Qualifiers_
```

### C.4.4 SWbemMethodSet

Un objeto **SWbemMethodSet** es una colección de objetos **SWbemMethod**. Los items son recibidos utilizando el método **Item**. Para mas información sobre el uso de colecciones, ver Colecciones.

Nota: En esta versión del API, el acceso escrito a métodos de información no es soportada. Si se quiere definir métodos o modificar los métodos existentes, se deben definir el cambio de método en los archivos MOF y enviar los cambios utilizando el compilador MOF. Alternativamente, se puede usar el WMI COM API.

**Métodos**

La siguiente tabla lista los métodos del objeto **SWbemMethodSet**.

| Método | Descripción |
|---|---|
| Item | Recibe un objeto **SWbemMethod** desde una colección. Este es el método de automatización estándar de este objeto. |

**Propiedades**

La siguiente tabla lista las propiedades del objeto **SWbemMethodSet**.

| Propiedades | Descripción |
|---|---|
| Count | El número de items en la colección. |

- **SWbemMethodSet.Item**

El método **Item** del objeto **SWbemMethodSet** retorna un objeto **SWbemMethod** desde una colección.

La siguiente sintaxis está en un lenguaje neutral. Para más explicación de esta sintaxis, ver Convenciones del Documento.

```
objMethod= objwbemMethodSet.Item(
strName,
[iFlags=0]
);
```

**Parámetros**

**strName**

Requerido. Nombre del método a ser recibido.

**iFlags**

Opcional. Reservado y con valor por defecto 0.

**Valores retornados**

Si es exitoso, el objeto **SWbemMethod** requerido retorna.

**Códigos de error**

El método **Item** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **WbemErrInvalidParameter** 0x80041008 | El parámetro *iFlags* no fue válido. |
| **WbemErrFailed** 0x80041001 | Error no específico. |
| **WbemErrNotFound** 0x80041002 | El método específico no existe. |

- **SWbemMethodSet.Count**

La propiedad **Count** del objeto **SWbemMethodSet** define el número de items en la colección. Esta propiedad es de solo lectura.

La siguiente sintaxis está en un lenguaje neutral. Para más explicación de esta sintaxis, ver Convenciones del Documento.

```
iCount = objwbemMethodSet.Count
```

### C.4.5. SwbemNamedValue

El objeto **SWbemNamedValue** representa un valor simple de nombre que pertenece a una colección de objetos SWbemNamedValueSet.

#### Métodos

Este objeto no tiene métodos.

#### Propiedades

La siguiente tabla lista las propiedades del objeto **SWbemNamedValue**.

| Propiedad | Descripción |
|---|---|
| Name | Nombre del item **SwbemNamedValue**. |
| Value | Valor del item **SwbemNamedValue.** Esta es la propiedad por defecto para este objeto. |

- **SWbemNamedValue.Name**

La propiedad **Name** del objeto **SWbemNamedValue** es una cadena única que es usada cuando se recibe o se fija un item **SWbemNamedValue**. Esta propiedad es de solo lectura.

La siguiente sintaxis está en un lenguaje neutral. Para explicaciones de esta sintaxis, ver Convenciones del Documento.

```
strName = objwbemNamedValue.Name
```

- **SWbemNamedValue.Value**

La propiedad **Value** del objeto **SWbemNamedValue** retorna un valor variant de un item **SwbemNamedValue.** Esta es la propiedad por defecto para los objetos **SWbemNamedValue**. Los cambios que se hagan al valor de esta propiedad son reflejados automáticamente en la colección SWbemNamedValueSet de la que el objeto **SWbemNamedValue** proviene.

La siguiente sintaxis está en un lenguaje neutral. Para explicaciones de esta sintaxis, ver Convenciones del Documento.

```
varValue = objwbemNamedValue.Value
```

### C.4.6 SWbemNamedValueSet

Un objeto SWbemNamedValueSet es una colección de objetos SWbemNamedValue. Se usan los métodos y propiedades de SWbemNamedValueSet principalmente para enviar información adicional a los proveedores al someter ciertas llamadas a WMI. Todas las llamadas en SWbemServices, y algunas llamadas en SWbemObject, toman un parámetro Opcional que es un objeto de este tipo. Un cliente puede agregar información a un objeto SWbemNamedValueSet, y envía el objeto SWbemNamedValueSet con la llamada como uno de los parámetros.

No se debe usar este mecanismo si es posible, porque puede romper el modelo de acceso uniforme que es la base de WMI. Si un proveedor usa este mecanismo, es importante usarlo tan económicamente como sea posible. Si un proveedor exige una cantidad grande de información del contexto muy específica para responder a una demanda, todos los clientes deben codificarse para proporcionar esta información. Este mecanismo le permite acceder tales proveedores, si es necesario.

Un objeto SwbemNamedValueSet es una colección de elementos de SWbemNamedValue. Estos artículos se agregan a la colección que usa el método SWbemNamedValueSet.Add. Ellos son removidos utilizando el método SWbemNamedValueSet.Remove y son recuperados utilizando el método SWbemNamedValueSet.Item. Se puede acceder los métodos para rellenar cualquier información del contexto requerido por un proveedor dinámico. Después de que se llama a uno de los métodos de SWbemServices, se puede reusar el objeto SWbemNamedValueSet para otra llamada.

El proveedor subyacente determina la información contenida en un objeto SWbemNamedValueSet. WMI no hace uso de la información, pero simplemente se adelanta al proveedor. Los proveedores deben publicar la información de contexto que ellos exigen para reparar demandas.

**Métodos**

La tabla siguiente lista los métodos para el objeto **SWbemNamedValueSet**.

| **Método** | **Descripción** |
|---|---|
| Add | Adiciona un objeto **SWbemNamedValue** para la colección. |
| Clone | Hace una copia de esta colección **SWbemNamedValueSet**. |
| DeleteAll | Remueve todos los artículos de esta colección haciendo el objeto vacío **SWbemNamedValueSet**. |
| Item | Recupera un objeto SWbemNamedValue de la colección. Este es el método por defecto del objeto. |

| | |
|---|---|
| Remove | Remueve un objeto **SWbemNamedValue** de la colección. |

**Propiedades**

La tabla siguiente lista la propiedad para el objeto **SWbemNamedValueSet**.

| **Propiedad** | **Descripción** |
|---|---|
| Count | El número de artículos en la colección. |

- **SWbemNamedValueSet.Add**

El método **Add** del objeto **SWbemNamedValueSet** agrega un objeto **SWbemNamedValue** a la colección. Si un elemento ya existe en la colección con el mismo nombre, el nuevo elemento lo reemplaza.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objNamedValue = objwbemNamedValueSet.Add (
strName,
varVal,
[iFlags=0]
);
```

**Parámetros**

**strName**

Requerido. Nombre del nuevo valor.

**varVal**

Requerido. Variante que representa el nuevo valor.

**iFlags**

Opcional. Reservado y debe ponerse en cero si se especificó.

**Valores de retorno**

Si es exitoso, este método devuelve el objeto **SWbemNamedValue** recientemente modificado o creado.

**Comentarios**

Para los ejemplos de agregar y recuperar valores nombrados, ver **SWbemNamedValue.Value**.

- **SWbemNamedValueSet.Clone**

El método **Clone** del objeto **SWbemNamedValueSet** retorna un nuevo objeto que es un clone de la colección actual.

La siguiente sintaxis está en un lenguaje neutral, para mas información ver Convenciones del Documento.

```
objwbemNamedValueSet = objwbemNamedValueSet.Clone()
```

**Valores retornados**

Si es exitoso, un nuevo objeto **SWbemNamedValueSet** es retornado.

**Códigos de Error**

El método **Clone** puede retornar los errores listados en la siguiente tabla.

| Error | Significado |
|---|---|
| wbemErrFailed<br>0x80041001 | Error no especificado. |
| wbemErrOutOfMemory<br>0x80041006 | No se encontró memoria para clonar el objeto. |

**Comentarios**

Se usa este método para duplicar una colección **SWbemNamedValueSet**.

- **SWbemNamedValueSet.DeleteAll**

El método **DeleteAll** del objeto **SWbemNamedValueSet** remueve todos los valores nombrados de la colección y así se vacía.

```
objwbemNamedValueSet.DeleteAll
```

**Valores de retorno**

Este método no tiene valores de retorno.

**Códigos del error**

El método **DeleteAll** puede devolver el código de error listado en la tabla siguiente.

| Error | Significado |
|---|---|
| **WbemErrFailed**<br>0x80041001 | Error no especificado. |

- **SWbemNamedValueSet.Item**

El método **item** del objeto **SWbemNamedValueSet** recibe un objeto **SWbemNamedValue** de la colección.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objNamedValue = objwbemNamedValueSet.Item(
strName,
[iFlags=0]
);
```

**Parámetros**

**strName**

Requerido. El nombre del valor a recuperar.

**iFlags**

Opcional. Reservado y debe ser cero si se especificó.

**Valores del retorno**

Si es exitoso, los objetos requeridos **SWbemNamedValue** retornan.

**Códigos del error**

El método del Artículo puede devolver los códigos de error listados en la tabla siguiente

| Error | Significado |
|---|---|
| **WbemErrFailed** 0x80041001 | Error no especificado. |
| **WbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado, o los "namespaces" no pudieron ser compilados. |
| **WbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |
| **WbemErrNotFound** 0x80041002 | El artículo pedido no fue encontrado. |

**Comentarios**

Para los ejemplos de agregar y recuperar valores nombrados, ver **SWbemNamedValue.Value**.

- **SWbemNamedValueSet.Remove**

El método **Remove** del objeto **SWbemNamedValueSet** anula un valor nombrado del contexto.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemNamedValueSet.Remove(
strName,
[iFlags=0]
```

```
);
```

**Parámetros**

**strName**

Requerido. El nombre del valor a quitar.

**iFlags**

Opcional. Reservado y debe ser cero si se especificó.

**Valores del retorno**

Este método no tiene valores del retorno.

**Códigos de error**

El método Remove puede devolver los códigos de error listados en la tabla siguiente.

| **Error** | **Meaning** |
|---|---|
| **WbemErrFailed**<br>0x80041001 | Error no especificado. |
| **WbemErrInvalidParameter**<br>0x80041008 | Un parámetro inválido fue especificado, o el "namespace" no pudo ser parsed. |
| **WbemErrNotFound**<br>0x80041002 | El artículo pedido no fue encontrado. |

- **SWbemNamedValueSet.Count**

Se usa la propiedad **Count** del objeto **SWbemNamedValueSet** para determinar cuantos items hay en la colección. Esta propiedad es de solo lectura

La siguiente sintaxis está en un lenguaje neutral, para más información de esta sintaxis ver convenciones del Documento.

```
iCount = objwbemNamedValueSet.Count
```

### C.4.7 SwbemObject

Se pueden usar los métodos y propiedades del objeto **SWbemObject** para representar una definición de clase WMI en particular o una instancia de un objeto.

Este objeto soporta dos tipos de propiedades y métodos. Aquellos definidos en esta sección son propiedades genéricas y métodos que se aplican a todos los objetos WMI. Adicionalmente, estos objetos exponen las propiedades y métodos como una automatización dinámica de las propiedades y métodos de dichos objetos. Estos tipos de nombres y propiedades dependen del objeto WMI que se seleccione. Para mas información de cómo se exponen las propiedades y métodos dinámicos.

Desde la perspectiva del cliente WMI, este objeto esta siempre en proceso. Las operaciones de escritura solo afectan la copia local del objeto, y las operaciones de lectura siempre retornan valores desde la copia local. Las actualizaciones de WMI son desarrolladas solo cuando todos los objetos son escritos utilizando una llamada al método **SWbemObject.Put_**. Si se modifican las propiedades o métodos en un objeto SWbemObject , los cambios no pueden ser escritos a WMI mientras se llame a **SWbemObject.Put_**.

El método genérico y los nombres de las propiedades definidas en esta sección siempre terminan con un trailing underscore ("_") para diferenciarlos de los métodos WMI dinámicos y las propiedades de underlying object.

#### Métodos

La siguiente tabla muestra la lista de métodos para el objeto **SWbemObject**.

| Método | Descripción |
|---|---|
| Associators_ | Recupuera las asociaciones del objeto. |
| AssociatorsAsync_ | Recupera asincrónicamente las asociaciones del objeto. |
| Clone_ | Hace una copia del objeto actual. |
| CompareTo_ | Prueba la igualdad entre dos objetos. |
| Delete_ | Borra el objeto de WMI. |
| DeleteAsync_ | Borra asincrónicamente el objeto de WMI. |
| ExecMethod_ | Ejecuta un metodo exportado por un proveedor de métodos. |
| ExecMethodAsync_ | Ejecuta asincrónicamente un método exportado por un proveedor de métodos. |
| GetObjectText_ | Recupera la representación textual del objeto (Sintaxis MOF). |
| Instances_ | Retorna una colección de instancias del objeto (que puede ser una clase WMI). |
| InstancesAsync_ | Retorna asincrónicamente una colección de instancias del objeto (puede ser una clase WMI). |
| Put_ | Crea o actualiza un objeto en WMI. |
| PutAsync_ | Crea o actualiza asincrónicamente el objeto en WMI. |
| References_ | Retorna las referencias del objeto. |
| ReferencesAsync_ | Retorna asincrónicamente las referencias del objeto. |
| SpawnDerivedClass_ | Crea una clase derivada del objeto actual (la cual puede ser una clase WMI). |
| SpawnInstance_ | Crea una nueva instancia desde el objeto actual. |
| Subclases_ | Retorna una colección de subclases del objeto (que puede ser una clase WMI). |
| SubclassesAsync_ | Retorna asincrónicamente una colección de subclases del objeto (que puede ser una clase WMI). |

**Propiedades**

La siguiente tabla muestra una lista de las propiedades del objeto **SWbemObject**.

| Propiedad | Descripción |
|---|---|
| Derivation_ | Contiene un arreglo de caracteres describiendo la derivación jerárquica para las clases. |
| Methods_ | Un objeto SWbemMethodSet que es la colección de métodos para este objeto. |
| Path_ | Contiene un objeto SWbemObjectPath que representa la ruta del objeto de la clase o la instancia actual. |
| Properties_ | Objeto SWbemPropertySet que es la colección de propiedades para este objeto. |
| Qualifiers_ | Objeto SWbemQualifierSet que es la colección de calificadores para este objeto. |
| Security_ | Contiene un objeto SWbemSecurity usado para leer o cambiar la configuración de seguridad. |

- **SWbemObject.Associators_**

El método **Associators_** del objeto SWbemObject retorna un conjunto de objetos (clases o instancias) que son asociadas con los objetos actuales. Estos objetos retornados son llamados endpoints. Este método desarrolla la misma función que la consulta ASSOCIATORS de WQL.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemObjectSet = objwbemObject.Associators_(
[strAssocClass=""],
[strResultClass=""],
[strResultRole=""],
[strRole=""],
[bClassesOnly=FALSE],
```

```
[bSchemaOnly=FALSE],
[strRequiredAssocQualifier=""],
[strRequiredQualifier=""],
[iFlags=wbemFlagReturnImmediately],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strAssocClass**

Opcional. Carácter que contiene una clase de asociación. Si se especifica, este parámetro indica que los endpoints retornados deben ser asociados con la fuente a través de una clase de asociación específica o una clase derivada de una clase de asociación.

**strResultClass**

Opcional. Cadena que contiene el nombre de la clase. Si se especifica, este parámetro indica que el endpoint retornado debe venir o derivarse de la clase específica dada en este parámetro.

**strResultRole**

Opcional. Cadena que contiene el nombre de la propiedad. Si se especifica, este parámetro indica que el endpoint retornado debe jugar un rol particular en su asociación con el objeto fuente. El rol está definido por el nombre la propiedad específica (el cual debe ser una propiedad referencia) de una asociación.

**strRole**

Opcional. Cadena que contiene el nombre de la propiedad. Si se especifica, este parámetro indica que los endpoints retornados deben participar en una asociación con el objeto fuente en la cual el objeto fuente juega un papel en particular. El papel o rol esta definido por el nombre de la propiedad específica ( la cual debe ser una propiedad referencia) de una asociación.

**bClassesOnly**

Opcional. Valor booleano que indica si una lista de los nombres de las clases que pueden ser retornadas después que las instancias actuales de la clase. Estas son clases de las que los endpoints provienen. El valor por defecto para este parámetro es FALSE.

**bSchemaOnly**

Opcional. Valor booleano que indica si la consulta es aplicada al esquema después que los datos. El valor por defecto de este parámetro es FALSE. Este solo puede ser fijado a un valor TRUE si el objeto sobre el que se aplica este método es una clase. Cuando se fija a un valor TRUE, el conjunto de endpoints retornados representan las clases que son asociados apropiadamente con la clase fuente en el esquema.

**strRequiredAssocQualifier**

Opcional. Cadena que contiene el nombre calificativo. Este parámetro, si se especifica, indica que los endpoints retornados deben ser asociados con el objeto fuente a través de una clase de asociación que incluye el calificador específico.

**strRequiredQualifier**

Opcional. Cadena que contiene el nombre calificativo. Este parámetro, si se especifica, indica que los endpoints retornados deben incluir el calificador específico.

**iFlags**

Opcional. Entero que indica banderas adicionales para la operación. El valor defecto para este parámetro es **wbemFlagReturnImmediately**, el cual dirige todas la llamadas para retornar inmediatamente una vez que se espere a que la consulta este completa. Este parámetro acepta los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemFlagForwardOnly**<br>0x20 | Causa que un enumerador solo hacia adelante (forward-only enumerator) a ser retornado. Los enumeradores Forward-only son generalmente mucho más rápidos y |

| | |
|---|---|
| | usan menos memoria que un enumerador convencional, pero no permiten llamadas a SwbemObject.Clone_. |
| **WbemFlagBidirectional**<br>0x0 | Hace que WMI de indicadores de retención a los objetos de enumeración mientras el cliente libere el enumerador. |
| **wbemFlagReturnImmediately**<br>0x10 | Hace que la llamada retorne inmediatamente. |
| **wbemFlagReturnWhenComplete**<br>0x0 | Hace que la llamada se bloquee mientras se completa la consulta. |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores Retornados**

Si la llamada es exitosa, se retorna un objeto SWbemObjectSet .

**Códigos de Error**

EL método **Associators_** puede retornar los códigos de error que se muestran en la siguiente tabla.

| Error | Significado |
|---|---|
| **WbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más clases retornadas por la llamada. |
| **WbemErrFailed** 0x80041001 | Error no específico. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

## Comentarios

Para más información acerca de ASSOCIATORS Of asociado con una consulta WQL, instancias fuentes y endpoints.

- **SWbemObject.AssociatorsAsync_**

El método **AssociatorsAsync_** del objeto SWbemObject retorna asincrónicamente una colección de objetos (clases o instancias) que son asociadas con el objeto actual. Estos objetos retornados son llamados endpoints. Este método desarrolla la misma función que la consulta ASSOCIATORS OF WQL. Esta llamada retorna inmediatamente y el resultado es devuelto al llamante a través del objeto "sink" provisto.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemObject.AssociatorsAsync_(
objWbemSink,
[strAssocClass=""],
[strResultClass=""],
[strResultRole=""],
```

```
[strRole=""],
[bClassesOnly=FALSE],
[bSchemaOnly=FALSE],
[strRequiredAssocQualifier=""],
[strRequiredQualifier=""]
[iFlags],
[objwbemNamedValueSet=null],
[objWbemAsyncContext]
);
```

**Parámetros**

**objWbemSink**

Requerido. Objeto "sink" que recibe los objetos asincrónicamente.

**strAssocClass**

Opcional. Cadena que contiene una clase de asociación. Si se especifica, este parámetro indica que el endpiont retornado debe ser asociado con la fuente a través de la clase de asociación específica o la clase derivada de la clase de asociación.

**strResultClass**

Opcional. Cadena que contiene el nombre de la clase. Si se especifica, este parámetro indica que el endpiont retornado debe pertenecer a o ser derivado de la clase especificado en este parámetro.

**strResultRole**

Opcional. Cadena que contiene un nombre de propiedad. Si se especifica, este parámetro indica que el endpiont retornado debe jugar un rol particular en la asociación con el objeto fuente. Este rol es definido por el nombre o una propiedad específica ( la que debe ser una propiedad referencia) de una asociación.

**strRole**

Opcional. Cadena que contiene el nombre de la propiedad. Si se especifica, este parámetro indica que los endpoints retornados deben participar en una asociación con el objeto fuente en la cual el objeto fuente juega un papel en particular. El papel o rol esta definido por el nombre de la propiedad específica ( la cual debe ser una propiedad referencia) de una asociación.

**bClassesOnly**

Opcional. Valor booleano que indica si una lista de los nombres de las clases que pueden ser retornadas después que las instancias actuales de la clase. Estas son clases de las que los endpoints provienen. El valor por defecto para este parámetro es FALSE.

**bSchemaOnly**

Opcional. Valor booleano que indica si la consulta es aplicada al esquema después que los datos. El valor por defecto de este parámetro es FALSE. Este solo puede ser fijado a un valor TRUE si el objeto al que se le aplica este método es una clase. Cuando se fija a un valor TRUE, el conjunto de endpoints retornados representan las clases que son asociados apropiadamente con la clase fuente en el esquema.

**strRequiredAssocQualifier**

Opcional. Cadena que contiene el nombre calificativo. Este parámetro, si se especifica, indica que los endpoints retornados deben ser asociados con el objeto fuente a través de una clase de asociación que incluye el calificador específico.

**strRequiredQualifier**

Opcional. Cadena que contiene el nombre calificativo. Este parámetro, si se especifica, indica que los endpoints retornados deben incluir el calificador específico.

**iFlags**

Opcional. Entero especificando banderas adicionales a la operación. Este parámetro puede aceptar los siguientes valores.

| Valor | Significado |
|---|---|
| **WbemFlagSendStatus**<br>0x80 | Hace enviar a la llamada asíncrona las actualizaciones de estado al manejador de eventos SWbemSink.OnProgress del objeto "sink" |
| **WbemFlagDontSendStatus**<br>0x0 | Previene a la llamada asíncrona de enviar actualizaciones de estado al manejador de eventos OnProgress del objeto "sink" |
| **WbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Este es un objeto SWbemNamedValueSet que retorna al objeto "sink" para identificar la fuente original de la llamada asíncrona. Se usa este parámetro si se hacen múltiples llamadas asíncronas utilizando el mismo objeto "sink". Para usar este parámetro, se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para adicionar un valor que identifique la llamada asíncrona que se esté haciendo. Este objeto SWbemNamedValueSet es devuelto al objeto "sink" y la fuente de la llamada puede ser extraída utilizando el método SWbemNamedValueSet.Item. Para más información, ver Haciendo una llamada asincrónica utilizando el Scripting API

**Valores Retornados**

Este método retorna un objeto SWbemSink. Los resultados de la llamada asíncrona son devueltos al objeto "sink" a través de un objeto SWbemObjectSet.

**Códigos de error**

El método **AssociatorsAsync_** puede retornar los siguiente códigos de error.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases retornadas por la llamada. |
| **WbemErrFailed** 0x80041001 | Error no específico |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

**Comentarios**

Para más información acerca de ASSOCIATORS OF asociados con la consultaWQL, instancias fuente, y endpoints.

- **SWbemObject.Clone_**

El método **Clone_** method del objeto SWbemObject retorna un nuevo objeto que es un clon del objeto actual.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemObject = objwbemObject.Clone_();
```

**Valores retornados**

Si es exitoso, este método retorna un nuevo objeto SWbemObject .

**Códigos de error**

El método **Clone_** puede retornar los códigos que se listan en la siguiente tabla.

| Error | Significado |
|---|---|
| **WbemErrFailed** 0x80041001 | Error no específico. |
| **wbemErrInvalidParameter** 0x80041008 | **Nada** fue especificado como un parámetro, y no es aceptable en este uso. |
| **wbemErrOutOfMemory** 0x80041006 | No encontró memoria para clonar el objeto. |

**Comentarios**

Se usa el método **Clone_** para duplicar una definición de clase o instancia. Este es útil cuando se necesita una copia del objeto original para propósitos de copias de seguridad mientras se está modificando una nueva copia. Por ejemplo, se puede usar SWbemObject.SpawnInstance_ para crear una sola instancias inicial, y se usa SWbemObject.Clone_ para producir 100 copias de la instancia rápidamente. Subsecuentemente, se pueden modificar los objetos, dándoles valores específicos.

No es posible usar este método con el fin de convertir una definición de clases a una instancia, o para convertir una instancia en una definición de clases.

- **SWbemObject.CompareTo_**

El método **CompareTo_** del objeto SWbemObject compara dos objetos **SWbemObject**. Esta comparación esta sujeta a reglas específicas basadas en los valores especificados en los parámetros *iFlags*.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
bAreEqual = objwbemObject.CompareTo_(
objwbemObject,
[iFlags= wbemComparisonFlagIncludeAll]
);
```

**Parámetros**

**objwbemObject**

Requerido. Este parámetro es un objeto SWbemObject . Este es el objeto con el cual el primer objeto se comparó. El objeto debe ser una instancia válida de **SWbemObject**.

**iFlags**

Opcional. Especifica las características del objeto a considerar cuando se comparen los objetos. Se puede usar **wbemComparisonFlagIncludeAll** para considerar todas las características (este esta por defecto), o una combinación de los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemComparisonFlagIncludeAll** | Compara todas la propiedades, calificadores |

| | |
|---|---|
| 0x0 | y, aditamentos. |
| **wbemComparisonFlagIgnoreObjectSource**<br>0x2 | Hace que la fuente de los objetos, es decir el servidor y el "namespace" que vengan de é, sean ignorados en la comparación con otros objetos. |
| **wbemComparisonFlagIgnoreQualifiers**<br>0x1 | Hace que todos los calificadores (incluyendo las **Llaves** y los **Dinámicos**) sean ignorados en la comparación. |
| **wbemComparisonFlagIgnoreDefaultValues**<br>0x4 | Hace que se ignoren los valores por defecto de las propiedades. Esta bandera se comparan clases.solo tiene un significado cuando. |
| **wbemComparisonFlagIgnoreFlavor**<br>0x20 | Hace que los calificadores sean ignorados. Esta bandera toma los valores de los calificadores dentro de una cuenta, pero ignora las distinciones ** flavor distinctions como una propagación de las reglas y rechaza las restricciones. |
| **wbemComparisonFlagIgnoreCase**<br>0x10 | Compra valores de cadenas de una manera insensitiva. Esto se aplica a nombres y valores calificativos. Los nombres y propiedades de los calificadores son siempre comparados de manera insensitiva donde esta bandera este o no especificada. |
| **wbemComparisonFlagIgnoreClass**<br>0x8 | Instruye al sistema a asumir que el objeto que está siendo comparado es una instancia de la misma clase. Consecuentemente, esta bandera compara instancias de solo información relacionada. Se usa esta bandera para optimizar el desempeño. Si el objeto no es de la misma clase, el resultado es indefinido. |

**Valores Retornados**

Este método retorna un valor Booleano de VERDADERO si el objeto coincide. Retorna FALSO si el objeto no coincide.

**Códigos de error**

El método **CompareTo_** puede retornar los códigos listados a continuación.

| **Error** | **Significado** |
|---|---|
| **WbemErrFailed** 0x80041001 | Error no especificado. |
| **WbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **WbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

- **SWbemObject.Delete_**

El método **Delete_** del objeto SWbemObject borra la clase o instancia actual.

Si un proveedor dinámico da la clase o instancia, algunas veces no será posible borrar este objeto a menos que el proveedor soporte la eliminación de clases o instancias.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemObject.Delete_(
[iFlags=0],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**iFlags**

Opcional. Reservado y puede ser cero si se especifica.

**objwbemNamedValueSet**

Opcional. Este parámetro típicamente está indefinido. De otra manera, este es un objeto SWbemNamedValueSet donde sus elementos representan el contexto de la información que puede ser usada por el proveedor que esta atendiendo la petición. Un proveedor que soporta o requiere dicha información debe reconocer los valores de los nombres en los documentos, tipos de valores de datos, valores permitidos, y semántica.

**Valores retornados**

Este método no tiene valores retornados.

**Códigos de error**

El método **Delete_** puede retornar los códigos de error que se listan a continuación.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El contexto actual no tiene los derechos de seguridad adecuados para borrar el objeto. |
| **wbemErrFailed** 0x8004100 | Error no especifico. |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada no existe. |
| **wbemErrInvalidOperation** 0x80041016 | El objeto no puede ser borrado. |

| | |
|---|---|
| **wbemErrNotFound**<br>0x80041002 | El objeto no existe. |
| **wbemErrOutOfMemory**<br>0x80041006 | No se encontró memoria para completar la operación. |

- **SWbemObject.DeleteAsync_**

El método **DeleteAsync_** del objeto SWbemObject borra asincrónicamente la clase o instancia actual. Esta llamada retorna inmediatamente y el resultado de la operación es retornado al llamante a través del objeto "sink" dado para este fin.

Si un proveedor dinámico da las clases o instancias, algunas veces no es posible borrar estos objetos a menos que el proveedor soporte métodos de eliminación de clases o instancias.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemObject.DeleteAsync_(
objWbemSink
[iFlags=0],
[objwbemNamedValueSet=null],
[objWbemAsyncContext]
);
```

**Parámetros**

**objWbemSink**

Objeto "sink" al que retorna la operación de borrado saliente.

**iFlags**

Opcional. Entero que determina el comportamiento de la llamada. Este parámetro puede aceptar los siguientes valores.

| Valor | Significado |
|---|---|
| **wbemFlagSendStatus**<br>0x80 | Hace enviar a la llamada asíncrona las actualizaciones de estado al manejador de eventos SWbemSink.OnProgress del objeto "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Previene a la llamada asíncrona de enviar actualizaciones de estado al manejador de eventos **OnProgress** del objeto "sink" |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Este es un objeto SWbemNamedValueSet que devuelve al objeto "sink" para identificar la fuente original de la llamada asíncrona. Se usa este parámetro si se hacen múltiple llamadas asíncronas utilizando el mismo objeto "sink". Para usar este parámetro se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para adicionar un valor que identifique la llamada asíncrona que se está haciendo. Este objeto **SWbemNamedValueSet** se retorna al objeto "sink" y la fuente de la llamada puede ser extraída utilizando el método SWbemNamedValueSet.Item. Para mayor información, ver Haciendo una llamada asincrónica utilizando el Scripting API

**Valores Retornados**

Ninguno. Si la llamada es exitosa el resultado de la operación de borrado es dado a través del objeto síncrono dado.

**Códigos de error**

El método **DeleteAsync_** puede retornar los códigos de error que se muestran a continuación.

| Error | Significado |
|---|---|
| **WbemErrAccessDenied** 0x80041003 | El contexto actual no tiene derechos de seguridad adecuados para borrar el objeto. |
| **WbemErrFailed** 0x8004100 | Error no específico. |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada no existe. |
| **wbemErrInvalidOperation** 0x80041016 | El objeto no puede ser borrado. |
| **wbemErrNotFound** 0x80041002 | El objeto no existe. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

- **SWbemObject.Derivation_**

La propiedad **Derivation_** del objeto SWbemObject contiene un arreglo de caracteres que describe la derivación jerárquica de las clases para que una instancia que esta siendo referenciada.

El primer elemento del arreglo define la superclase y el último elemento define la clase destino. Esta propiedad es de solo lectura.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
strClassArray = objwbemObject.Derivation_
```

- **SWbemObject.ExecMethod_**

El método **ExecMethod_** del objeto SWbemObject ejecuta un método exportado por un proveedor de métodos. Este método es similar a SWbemServices.ExecMethod, pero este opera directamente sobre el objeto al cual se le ejecuta el método.

Usar **SWbemObject.ExecMethod_** es una alternativa para ejecutar un método en el caso de que no sea posible ejecutar dicho método directamente. Por ejemplo, se podría usar con un lenguaje de script como Jscript que no soporta parámetros de salida. Por otra parte, el significado recomendado de la invocación a un método es usado en un método de invocación directa. El método de invocación directa esta ejecutando un método con el uso de la sintaxis **object.method**. Para más información para más información sobre el método de invocación directa.

Este método se bloqueará mientras que se ejecuta el método que es enviado se envíe al proveedor adecuado. La información y el estado son entonces retornadas.

WMI no implementa métodos directamente. Los proveedores exportan dichos métodos.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objOutParams = objwbemObject.ExecMethod_(
strMethodName,
```

```
[objwbemInParams=null],
[iFlags=0],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strMethodName**

Requerido. Nombre del método dentro del objeto.

**objwbemInParams**

Opcional. Este es un objeto SWbemObject que contiene los parámetros de entrada para los métodos que están siendo ejecutados. Por defecto, este parámetro es indefinido. Para más información sobre la construcción de este parámetro, ver la sección de comentarios.

**iFlags**

Opcional. Reservado y debe fijarse a cero si se especificó.

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si el método es exitoso, se retorna un objeto SWbemObject . El objeto retornado contiene los parámetros de salida a retorna los valores para el método que esta siendo ejecutado.

**Códigos de error**

El método **ExecMethod_** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **WbemErrFailed** 0x80041001 | Error no específico. |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada fue inválida. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |
| **wbemErrInvalidMethod** 0x8004102E | La petición al método no esta disponible. |
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no fue autorizado para ejecutar el método. |

**Comentarios**

**Para construir los parámetros *objwbemInParams***

1. Llamar **SWbemObject.Methods_.Item**("Nombre del método") para conseguir un objeto **SWbemMethod** que contenga las propiedades que describan al método, donde "Nombre del método" es el nombre del método a ser recibido.

2. Usar la propiedad **SWbemMethod.InParameters** del objeto **SWbemMethod** para conseguir un objeto **InParameters**.

3. Llamar a **SpawnInstance_** en el objeto **InParameters para crear una instancia**.

4. Fijar las propiedades de las instancias siempre que los valores sean apropiados. Por ejemplo, si se desea fijar un parámetro de entrada llamado *myinputparam* al valor

"whatever" en una instancia de **InParameters** llamada "INST", el código podría lucir como esto: INST.**Properties_.Add** ("myinputparam").Value = "Whatever"

- **SWbemObject.ExecMethodAsync_**

El método **ExecMethodAsync_** del objeto SWbemObject ejecuta asincrónicamente un método esportado por un proveedor. Este método es similar a SWbemServices.ExecMethodAsync, pero este opera directamente sobre el objeto cuyo método esta siendo ejecutado.

Se usa **SWbemObject.ExecMethodAsync_** como una forma alternativa para ejecutar un método en casos donde no es posible ejecutar el método directamente. Por ejemplo, se podría usarlo con un lenguaje de script como Jscript que no soporta parámetros de salida, si el método no tiene parámetros de salida. De otra manera, lo recomendado para invocar a un método es usar el método de invocación directa. El método de invocación directa esta ejecutando un método a través del uso de la sintaxis **object.method**. Para más información sobre el método de invocación directa**.**

Este llamado retorna inmediatamente y el resultado de la operación SWbemObject.Put es retornado al llamante a través de los objetos "sink" provistos para este fin.

WMI no implementa los métodos directamente. El proveedor de métodos exporta dichos métodos.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objOutParams = objwbemObject.ExecMethodAsync_(
objWbemSink,
strMethodName,
[objwbemInParams=null],
[iFlags=0],
```

```
[objwbemNamedValueSet=null],
[objWbemAsyncContext]
);
```

**Parámetros**

**objWbemSink**

Requerido. Este es el objeto que recibe los resultados de la llamada al método. Los parámetros de salida son enviados al evento **SWbemSink.OnObjectReady** del objeto "sink" provisto. El resultado del mecanismo de llamado es enviado al evento **SWbemSink.OnCompleted** provisto por el objeto "sink". Note que **SWbemSink.OnCompleted** no recibe el código retornado por el método. Sin embargo, este recibe el código retornado por el mecanismo actual de devolución de llamada, y solo es útil para verificar que la llamada ocurrió o que esta falló por razones mecánicas. El código resultante devuelto por el método es retornado en el objeto de parámetro de salida dado por **SWbemSink.OnObjectReady**. Si cualquier error de código retorna, entonces del objeto **IWbemObjectSink** no es usado. Si una llamad es exitosa, entonces el usuario de la implementación de **IWbemObjectSink** es llamado para indicarle el resultado de la operación.

**strMethodName**

Requerido. Este es el nombre del método para el objeto.

**objwbemInParams**

Opcional. Este es un objeto SWbemObject que contiene los parámetros de entrada para los métodos que están siendo ejecutados. Por defecto, este parámetro esta indefinido. Para información de cómo construir este parámetro, ver la sección comentarios.

**iFlags**

Opcional. Entero que determina el valor de la llamada. Este parámetro puede aceptar los siguientes valores.

| Valor | Significado |
|---|---|
| **wbemFlagSendStatus**<br>0x80 | Hace que las llamadas asíncronas envíen actualizaciones de estado al manejador de eventos SWbemSink.OnProgress del objeto "sink". |
| **WbemFlagDontSendStatus**<br>0x0 | Previene a las llamadas asíncronas para no enviar actualizaciones de estado al manejador de eventos **OnProgress** para el objeto "sink". |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Este es un objeto **SWbemNamedValueSet** que retorna al objeto "sink" para identificar la fuente original de la llamada asíncrona. Se usa este parámetro si se hacen múltiples llamadas asíncronas utilizando el mismo objeto "sink". Para usar este parámetro, se crea un objeto **SWbemNamedValueSet** y se usa el método SWbemNamedValueSet.Add para adicionar un valor que identifique la llamada asíncrona que se esté haciendo. Este objeto **SWbemNamedValueSet** es devuelto al objeto "sink" y la fuente de la llamada, y puede ser extraído utilizando el método SWbemNamedValueSet.Item. Para más información, ver Haciendo una llamada asincrónica utilizando el Scripting API.

**Valores retornados**

Este método no retorna valores. Un objeto SWbemObject retorna al objeto "sink" provisto si la llamada es exitosa. El objeto retornado contiene los parámetros de salida y loas valores retornados para el método que esta siendo ejecutado.

**Códigos de error**

El método **ExecMethodAsync_** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no específico |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada fue inválida. |
| **wbemErrInvalidParameter** 0x80041008 | Parámetro especificado inválido. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |
| **wbemErrInvalidMethod** 0x8004102E | Método pedido no estuvo disponible. |
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no fue autorizado para ejecutar el método. |

**Comentarios**

**Para construir los parámetros de *objwbemInParams***

1. Llamar a **SWbemObject.Methods_.Item**("MethodName") para conseguir un objeto **SWbemMethod** que contenga las propiedades que describen a dicho método, donde "MethodName" es el nombre del método que se recibe.

2. Usar la propiedad **SWbemMethod.InParameters** del objeto **SWbemMethod** para conseguir un objeto **InParameters**.

3. Llamar a **SpawnInstance_** en el objeto **InParameters** para crear una instancia.

4. Fijar las propiedades de la instancia a un valor cualquiera es apropiado. Por ejemplo, Si se desea fijar un parámetro de entrada llamado *myinputparam* a un valor "whatever" en una instancia de **InParameters** llamada "INST", el código podría lucir como esto: INST.**Properties_.Add** ("myinputparam").Value = "Whatever"

- **SWbemObject.GetObjectText_**

El método **GetObjectText_** del objeto SWbemObject retorna una renderización textual del objeto. Este objeto puede ser utilizado para desplegar el contenido de un objeto. Actualmente, solo la sintaxis MOF es soportada como un formato de salida. Note que el texto MOF retornado no contiene toda la información acerca del objeto; el texto MOF contiene solo la información necesaria para que el compilador MOF permita recrear el objeto original. Para las instancias, no hay información acerca de los calificadores propagados o las propiedades de la clase padre.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
strMofText = objwbemObject.GetObjectText_(
[iFlags=0]
);
```

**Parámetros**

**iFlags**

Opcional. Reservado y debe ser cero si se especifica.

**Valores retornados**

Si es exitosa, este método retorna una cadena de caracteres que contienen el texto de salida.

**Códigos de error**

El método **GetObjectText_** puede retornar los siguientes códigos de error:

| Error | Significado |
|---|---|
| **WbemErrFailed** 0x80041001 | Error no específico. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

- **SWbemObject.Instances_**

El método **Instances_** del objeto SWbemObject crea un enumerador que retorna las instancias de la clase actual del objeto. Este método implementa una consulta simple. Para consultas más complicadas se puede requerir el uso de SWbemServices.ExecQuery.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objWbemObjectSet = objwbemObject.Instances_(
[iFlags=wbemFlagReturnImmediately],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**iFlags**

Opcional. Entero que determina el comportamiento de la llamada. Este parámetro puede aceptar los siguientes valores.

| Valor | Significado |
|---|---|
| **WbemFlagUtiliceAmendedQualifiers** 0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si el método es exitoso, un objeto SWbemObjectSet es retornado.

**Códigos de error.**

El método **Instances_** puede retornar los códigos de error listados en la siguiente tabla.

| Error | Significado |
|---|---|
| **WbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver la instancia de la clase especificada. |

| | |
|---|---|
| **wbemErrFailed**<br>0x80041001 | Un error no específico ha ocurrido. |
| **wbemErrInvalidClass**<br>0x80041010 | La clase especificada es inválida. |
| **wbemErrInvalidParameter**<br>0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory**<br>0x80041006 | No se encontró memoria para completar la operación. |

**Comentarios**

El método **Instances_** solo trabaja con objetos de clase. No es un error para la colección retornada tener elementos cero.

- **SWbemObject.InstancesAsync_**

El método **InstancesAsync_** del objeto SWbemObject crea asincrónicamente un enumerador que retorna las instancias de la clase actual del objeto. Este método implementa una consulta simple. Para consultas más complejas puede requerirse el uso de SWbemServices.ExecQuery. Esta llamada retorna inmediatamente y los resultados son devueltos al llamante a través del objeto "sink" provisto.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objwbemObject.InstancesAsync_(
[objWbemSink ]
[iFlags],
[objwbemNamedValueSet=null],
```

```
[objWbemAsyncContext]
);
```

**Parámetros**

**objWbemSink**

Objeto "sink" que retorna las instancias.

**iFlags**

Opcional. Entero que determina el valor de la llamada. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **WbemFlagSendStatus**<br>0x80 | Hace que las llamadas asíncronas envíen actualizaciones de estado al manejador de eventos SWbemSink.OnProgress del objeto "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Previene a las llamadas asíncronas para no enviar actualizaciones de estado al manejador de eventos **OnProgress** para el objeto "sink". |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Este es un objeto SWbemNamedValueSet que retorna al objeto "sink" para identificar la fuente original de la llamada asíncrona. Se usa este parámetro si se hacen múltiples llamadas asíncronas utilizando el mismo objeto "sink". Para usar este parámetro, se crea un objeto **SWbemNamedValueSet** y se usa el método SWbemNamedValueSet.Add para adicionar un valor que identifique la llamada asíncrona que se esté haciendo. Este objeto **SWbemNamedValueSet** es devuelto al objeto "sink" y la fuente de la llamada puede ser extraída utilizando el método SWbemNamedValueSet.Item. Para más información, ver Haciendo una llamada asincrónica utilizando el Scripting API**.**

**Valores retornados**

Este método no retorna valores. Si la llamada es exitosa, el resultado de la llamada asíncrona es provisto a través de un objeto SWbemObjectSet.

**Códigos de error**

El método **InstancesAsync_** puede retornar los códigos de error mostrados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **WbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver las instancias de la clase especificada. |
| **wbemErrFailed** 0x80041001 | Ha ocurrido un error no especificado. |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada es invalida. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

**Comentarios**

El método **Instances_** solo trabaja con objetos de clase. No es un error para la colección retornada tener elementos cero.

- **SWbemObject.Methods_**

La propiedad **Methods_** del objeto SWbemObject retorna un objeto SWbemMethodSet que es una colección de los métodos para la clase actual o instancia. Esta propiedad es solo de lectura.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objMethodSet = objwbemObject.Methods_
```

- **SWbemObject.Path_**

La propiedad **Path_** del objeto SWbemObject retorna un objeto SWbemObjectPath que representa el object path de la clase actual o instancia. Esta propiedad puede pasarse como un parámetro a métodos que requieren una ruta del objeto.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objObjectPath = objwbemObject.Path_
```

**Comentarios**

Sólo la propiedad **Class** de la instancia retornada SWbemObjectPath puede modificarse. Si se intenta modificar cualquier otra propiedad, o se intenta llamar los métodos **SetAsClass** o **SetAsSingleton**, se cometerá un error del wbemErrReadOnly.

Debido a esto, no se puede modificar el objeto SWbemNamedValueSet que es el valor de la propiedad Keys de la instancia SWbemObjectPath retornada. Si se intenta llamar los métodos **Add**, **Remove**, o **DeleteAll** en este valor, se cometerá un error del wbemErrReadOnly. Además, no se puede modificar cualquier SWbemNamedValue obtenido de esta colección. Intentar modificar la propiedad Value retorna el mismo código de error.

Sin embargo, si se llama SWbemObject.Clone_ para crear una copia, la propiedad **Path_** de la copia es totalmente modificable.

- **SWbemObject.Properties_**

La propiedad **Properties_** del objeto SWbemObject retorna un objeto SWbemPropertySet que es una colección de las propiedades para la clase actual o instancia. Esta propiedad es solo de lectura.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objPropertySet = objwbemObject.Properties_
```

- **SWbemObject.Put_**

El método **Put_** del objeto SWbemObject crea o actualiza una instancia o clase de objeto para WMI. Se puede usar este método después de que se modifique cualquier propiedad o métodos en un objeto SWbemObject y sus cambios se escribirán a WMI.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objObjectPath = objwbemObject.Put_ (
[el iFlags = el wbemChangeFlagCreateOrUpdate],
[el objwbemNamedValueSet=null]
);
```

**Parámetros**

**iFlags**

Opcional. Este parámetro determina si la llamada crea o actualiza la clase o instancia y si la llamada vuelve inmediatamente. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemChangeFlagUpdateCompatible** 0x0 | Permite actualizar una clase si no hay ninguna clase derivada y no hay ningúna instancia para esa clase. También permite actualizaciones en todos los casos si el cambio es justo para los calificadores insignificantes (por ejemplo, el calificador de la Descripción). Ésta es la conducta predefinida por esta llamada y se usa para la compatibilidad con versiones anteriores de WMI. Si la clase tiene instancias las actualizaciones fallan. |
| **wbemChangeFlagUpdateSafeMode** | Permite actualizaciones de clases aun cuando hay |

| | |
|---|---|
| 0x20 | una clase niño si el cambio no causa ningún conflicto con clases niño. Se puede usar esta bandera al agregar una nueva propiedad a una clase baja que no se mencionó previamente en cualquiera de las clases niño. Si la clase tiene instancias la actualización falla. |
| **WbemChangeFlagUpdateForceMode** 0x40 | Esta bandera fuerza actualizaciones de clases cuando existen conflictos de clases niño. Por ejemplo, esta bandera forzará una actualización si un calificador de la clase fué definido en una clase niño, y la clase base intenta agregar el mismo calificador entrando en conflicto con el que existe. En modo de fuerza este conflicto sería resuelto anulando el calificador en conflicto en la clase niño. Si la clase tiene instancias la actualización fallará. |
| **wbemChangeFlagCreateOrUpdate** 0x0 | Causa la clase o instancia a ser creada si no existe o la borra si ya existe. |
| **wbemChangeFlagCreateOnly** 0x2 | Sólo es usado para la creación. La llamada falla si la clase o instancia ya existe. |
| **wbemChangeFlagUpdateOnly** 0x1 | Causa esta llamada para actualizarla. La clase o instancia debe existir para la llamada para tener éxito. |
| **wbemFlagReturnImmediately** 0x10 | Causa la llamada para volver inmediatamente. |
| **wbemFlagReturnWhenComplete** 0x0 | Causa esta llamada para bloquearla hasta que la pregunta se haya completado. |
| **wbemFlagUtiliceAmendedQualifiers** 0x20000 | Causa WMI para escribir datos de enmendadura de clase así como la definición de la clase base. Para más información sobre los calificadores enmendados ver Localización de WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si la llamada tiene éxito, un objeto SWbemObjectPath es retornado. Este objeto contiene el object path de la instancia o clase que se han comprometido con éxito en WMI.

**Códigos del error**

El método **Put_** puede devolver los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para actualizar una instancia de la clase especificada. |
| **wbemErrAlreadyExists** 0x80041019 | La bandera wbemChangeFlagCreateOnly fue especificada, pero la instancia ya existe. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrIllegalNull** 0x8004100A | Un valor de Nada se especificó para una propiedad que no puede ser Nada. Un ejemplo de semejante propiedad es marcado por un calificador Key, Indexed, or Not_Null. |
| **wbemErrInvalidObject** 0x80041014 | La instancia especificada es inválida. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrNotFound** | La bandera wbemChangeFlagUpdateOnly fue especificada, |

| | |
|---|---|
| 0x80041002 | pero la instancia o la clase no existe. |
| **wbemErrIncompleteClass**<br>0x80041020 | Las propiedades requeridas para las clases no han sido todas fijadas. |
| **wbemErrOutOfMemory**<br>0x80041006 | No hay suficiente memoria para completar la operación. |

- **SWbemObject.PutAsync_**

El método **PutAsync_** del objeto SWbemObject crea o actualiza una instancia u objeto clase para WMI. Se puede usar este método después de que se ha modificado cualquier propiedad o método en un objeto **SWbemObject** y sus cambios se escriben en WMI. Esta llamada retorna inmediatamente y se devuelven los resultados de la operación son retornados al llamante a través del object "sink" proporcionado.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemObject.PutAsync_ (
objWbemSink,
[el iFlags = el wbemChangeFlagCreateOrUpdate],
[el objwbemNamedValueSet=null],
[el objWbemAsyncContext]
);
```

**Parámetros**

**objWbemSink**

Requerido. object "sink" que asincrónicamente recibe el resultado de la operación.

**iFlags**

Opcional. Determina si la llamada crea o actualiza la clase o instancia y si la llamada retorna inmediatamente. Este parámetro puede aceptar los siguientes valores.

| **Value** | **Meaning** |
|---|---|
| **WbemChangeFlagUpdateCompatible** 0x0 | Permite actualizar una clase si no hay ninguna clase derivada y si no hay ningúna instancia para esa clase. También permite actualizaciones en todos los casos si el cambio es justo para los calificadores insignificantes (por ejemplo el calificador de la Descripción). Ésta es la conducta predefinida por esta llamada y se usa para la compatibilidad con versiones anteriores de WMI. Si la clase tiene instancias la actualización falla. |
| **wbemChangeFlagUpdateSafeMode** 0x20 | Permite actualizaciones de clases, aun cuando hay clases niño, si el cambio no causa conflictos con clases niño. Se puede usar esta bandera al agregar una nueva propiedad a una clase baja que no se mencionó previamente en cualquiera de las clases niño. Si la clase tiene instancias la actualización falla. |
| **WbemChangeFlagUpdateForceMode** 0x40 | Fuerza la actualización de clases cuando existen conflictos de clases niño. Por ejemplo, esta bandera fuerza una actualización si un calificador de la clase se definiera en una clase |

| | |
|---|---|
| | niño, y la clase baja intenta agregar el mismo calificador en conflicto con el que existe. En modo de fuerza este conflicto está resuelto anulando el calificador que está en conflicto en la clase niño. Si la clase tiene instancias la actualización falla. |
| **wbemChangeFlagCreateOrUpdate**<br>0x0 | Causa la clase o instancia para ser creada si no existe o la borra si ya existe. |
| **wbemChangeFlagCreateOnly**<br>0x2 | Sólo es usado para crear. La llamada falla si la clase o instancia ya existe. |
| **wbemChangeFlagUpdateOnly**<br>0x1 | Causa esta llamada para actualizar. La clase o instancia deben existir para que la llamada tenga éxito. |
| **wbemFlagReturnImmediately**<br>0x10 | Causa la llamada para retornar inmediatamente. |
| **wbemFlagReturnWhenComplete**<br>0x0 | Causa esta llamada para bloquear hasta que la pregunta se haya completado. |
| **wbemFlagSendStatus**<br>0x80 | Causa las llamadas asíncronas para enviarle actualizaciones de estado al manejador de SWbemSink.OnProgress para el object “sink”. |
| **wbemFlagDontSendStatus**<br>0x0 | Impide a las llamadas asíncronas enviarle actualizaciones de estado al evento manejador OnProgress para el object “sink”. |
| **WbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Causa WMI para escribir datos de enmendadura de clase junto con la definición de la clase baja. Para más información sobre los calificadores enmendados ver Localización de WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Éste es un objeto de SWbemNamedValueSet que devuelve al object "sink" para identificar la fuente de la llamada asíncrona original. Se usa este parámetro si se está haciendo llamadas asíncronas múltiples que usan el mismo object "sink". Para usar este parámetro, se crea Un objeto Swbem**NamedValueSet** y se usa el método SWbemNamedValueSet.Add para agregar un valor que identifica la llamada asíncrona que se está haciendo. Este objeto **SWbemNamedValueSet** ha devuelto al object "sink" y la fuente de la llamada puede extraerse utilizando el método SWbemNamedValueSet.Item. Para más información, ver Haciendo una Llamada Asíncrona utilizando el Scripting API.

**Valores retornados**

Este método no tiene Valores retornados. Si la llamada tiene éxito, el evento **OnObjectPut** del object "sink" proporcionado recibe un objeto SWbemObjectPath y contiene el object path de la instancia o clase que se han comprometido con éxito en WMI.

**Códigos del error**

El método de **PutAsync_** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para actualizar una instancia de la clase especificada. |
| **wbemErrAlreadyExists** | La bandera wbemChangeFlagCreateOnly fue |

| | |
|---|---|
| 0x80041019 | especificada, pero la instancia ya existe. |
| **wbemErrFailed**<br>0x80041001 | Error no especificado. |
| **wbemErrIllegalNull**<br>0x8004100A | Un valor de Nada se especificó para una propiedad que no puede ser Nada. Un ejemplo de semejante propiedad es uno que es marcado por un calificador Key, Indexed, or Not_Null. |
| **wbemErrInvalidObject**<br>0x80041014 | El caso especificado es inválido. |
| **wbemErrInvalidParameter**<br>0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrNotFound**<br>0x80041002 | La bandera wbemChangeFlagUpdateOnly fue especificada, pero la instancia o la clase no existe. |
| **wbemErrIncompleteClass**<br>0x80041020 | Required properties for classes have not all been set. |
| **wbemErrOutOfMemory**<br>0x80041006 | Not enough memory to complete the operation. |

- **SWbemObject.Qualifiers_**

La propiedad **Qualifiers_** del objeto SWbemObject retona un objeto SWbemQualifierSet que es una colección de los calificadores para la clase actual o instancia. Esta propiedad es solo de lectura.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objQualifierSet = objwbemObject.Qualifiers_
```

- **SWbemObject.References_**

El método **References_** del objeto SWbemObject retorna una colección de toda la asociación de clases o instancias que se refieren al objeto actual.

Este método realiza la misma función como REFERENCES OF WQL query.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemObjectSet = objwbemObject.References_ (
[el strResultClass =""],
[el strRole =""],
[el bClassesOnly=FALSE],
[el bSchemaOnly=FALSE],
[el strRequiredQualifier =""],
[el iFlags=wbemFlagReturnImmediately],
[el objwbemNamedValueSet=null]
);
```

**Parámetros**

**strResultClass**

Opcional. Cadena que contiene un nombre de la clase. Si se especificó, este parámetro indica que los objetos de la asociación devueltos deben pertenecer a o deben derivarse de la clase especificada en este parámetro.

**strRole**

Opcional. Cadena que contiene un nombre de propiedad. Si se especificó, este parámetro indica que los objetos de la asociación devueltos deben limitarse a aquéllos en los que el

objeto de la fuente juega un papel particular. El papel es definido por el nombre de una propiedad especificada (que debe ser una propiedad de referencia) de una asociación.

**bClassesOnly**

Opcional. Valor Boleano que indica si una lista de nombres de clase debe devolverse en lugar de las instancias actualles de las clases. Estas son las clases a las que los objetos de la asociación pertenecen. El valor predefinido para este parámetro es FALSO.

**bSchemaOnly**

Opcional. Valor Boleano que indica si la pregunta se aplica al esquema en lugar de los datos. El valor predefinido para este parámetro es FALSO. Sólo puede ponerse en TRUE si el objeto en el que este método se invoca es una clase. Cuando se pone TRUE, el set de endpoints devuelto representa clases que son adecuadamente asociadas con la clase fuente en esquema.

**strRequiredQualifier**

Opcional. Cadena que contiene un nombre del calificador. Si se especificó, este parámetro indica que los objetos de la asociación devueltos deben incluir el calificador especificado.

**iFlags**

Opcional. Entero que especifica banderas adicionales al funcionamiento. El valor por defecto para este parámetro es wbemFlagReturnImmediately que dirige la llamada para retornar inmediatamente en lugar de esperar hasta que la pregunta se haya completado. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemFlagForwardOnly**<br>0x20 | Causa un enumerador delantero-único para ser devuelto. Empadronadores Delantero-únicos generalmente son muy más rápidos y usan menos memoria que los enumeradores convencionales, pero |

| | |
|---|---|
| | ellos no le permiten llamadas a SWbemObject.Clone_. |
| **wbemFlagBidirectional**<br>0x0 | Causa WMI para retener indicados a los objetos de la enumeración hasta queel cliente suelte el empadronador. |
| **wbemFlagReturnImmediately**<br>0x10 | Causa la llamada para volver inmediatamente. |
| **wbemFlagReturnWhenComplete**<br>0x0 | Causa esta llamada para bloquear hasta que la pregunta se haya completado. |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Causa WMI para devolver datos de enmendadura de clase con la definición de la clase baja. Para más información sobre los calificadores enmendados ver Localización de WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si la llamada tiene éxito, un objeto SWbemObjectSet es devuelto.

**Códigos del error**

El método de **References_** puede devolver los códigos del error listados en la siguiente tabla.

| Error | Significado |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases deueltas por la llamada. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |

**Comentarios**

Para información sobre las REFERENCES OF asociadas al requerimiento WQL, instancias fuente, y asociación de objetos.

- **SWbemObject.ReferencesAsync_**

El método **ReferencesAsync_** del objeto SWbemObject retorna una colección de toda la asociación de clases o instancias que se refieren al objeto actual.

Este método realiza la misma función que los requerimientos WQL REFERENCES OF realizan. Esta llamada vuelve inmediatamente y se devuelven los resultados y estados al llamante a través del object "sink" proporcionado.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemObject.ReferencesAsync_ (
objWbemSink,
[el strResultClass =""],
```

```
[el strRole =""],
[el bClassesOnly=FALSE],
[el bSchemaOnly=FALSE],
[el strRequiredQualifier =""],
[el iFlags],
[el objwbemNamedValueSet=null],
[el objWbemAsyncContext]
);
```

**Parámetros**

**objWbemSink**

Requerido. Object “sink” que recibe los objetos asincrónicamente.

**strResultClass**

Opcional. Cadena que contiene un nombre de la clase. Si se especificó, este parámetro indica que los objetos de la asociación devueltos deben pertenecer a o deben derivarse de la clase especificada en este parámetro.

**strRole**

Opcional. Cadena que contiene un nombre de propiedad. Si se especificó, este parámetro indica que los objetos de la asociación devueltos deben limitarse a aquéllos en los que el objeto fuente juega un papel particular. El papel es definido por el nombre de una propiedad especificada (qué debe ser una propiedad de referencia) de una asociación.

**bClassesOnly**

Opcional. Valor Boleano que indica si una lista de nombres de la clase debe devolverse en lugar de las instancias reales de las clases. Estas son las clases a las que los objetos de la asociación pertenecen. El valor predefinido para este parámetro es FALSE.

**bSchemaOnly**

Opcional. Valor Boleano que indica si la pregunta se aplica al esquema en lugar de los datos. El valor predefinido para este parámetro es FALSE. Sólo puede ponerse enTRUE si el objeto en el que este método se invoca es una clase. Cuando se pone en TRUE, el juego de endpoints devuelto representa clases que son adecuadamente asociadas con la clase fuente en el esquema.

**strRequiredQualifier**

Opcional. Cadena que contiene un nombre del calificador. Si se especificó, este parámetro indica que los objetos de la asociación devueltos deben incluir el calificador especificado.

**iFlags**

Opcional. Entero que especifica banderas adicionales al funcionamiento. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemFlagSendStatus**<br>0x80 | Causa las llamadas asíncronas para enviarle actualizaciones de estado al manejador de eventos SWbemSink.OnProgress para el object “sink”. |
| **wbemFlagDontSendStatus**<br>0x0 | Impide a las llamadas asíncronas enviarle actualizaciones de estado al evento manejador OnProgress para el object “sink”. |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Causa WMI para devolver datos de enmendadura de clase junto con la definición de la clase baja. Para más información sobre los calificadores enmendados ver Localización de WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que

puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Éste es un objeto SWbemNamedValueSet que devuelve al object "sink" para identificar la fuente de la llamada asíncrona original. Se usa este parámetro si se está haciendo llamadas asíncronas múltiples que usan el mismo object "sink". Para usar este parámetro, se crea un objeto de SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para agregar un valor que identifica la llamada asíncrona que se está haciendo. Este objeto **SWbemNamedValueSet** es devuelto al object "sink" y la fuente de la llamada puede extraerse utilizando el método de SWbemNamedValueSet.Item. Para más información, ver Haciendo una Llamada Asíncrona utilizando el Scripting API.

**Valores retornados**

Este método no tiene valores de retorno. Si es exitoso, se proporcionan los resultados de la llamada asíncrona a través de este fregadero como un objeto SWbemObjectSet. Este objeto es una colección de toda la asociación de clases o instancias que se refieren al objeto actual.

**Códigos de error**

El método de **References_** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases devueltas por la llamada. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |

| | |
|---|---|
| **wbemErrOutOfMemory**<br>0x80041006 | No hay suficiente memoria para completar la operación. |

**Comentarios**

Para información sobre las REFERENCES OF asociadas al requerimiento WQL asociado, instancias fuente, y asociación de objetos.

- **SWbemObject.Security_**

La propiedad **Security_** del objeto SWbemObject se usa para leer o poner las escenas de seguridad para un objeto **SWbemObject**. Esta propiedad es un objeto **SWbemSecurity**.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemSecurity = objwbemObject.Security_
```

- **SWbemObject.SpawnDerivedClass_**

Utilice el método **SpawnDerivedClass_** del objeto SWbemObject para crear un objeto clase derivado del objeto actual. El objeto debe ser una definición de la clase que se vuelve la superclase del objeto desovado.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objNewClass = objwbemObject.SpawnDerivedClass_ (
```

```
[iFlags=0]
);
```

**Parámetros**

**iFlags**

Opcional. Reservado y debe ser cero si se especificó.

**Valores retornados**

Si la llamada tiene éxito, un objeto SWbemObject retorna lo que contiene el nuevo objeto definido en clase. Ningún objeto retorna cuando hay un error.

**Códigos de error**

El método **SpawnDerivedClass_** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **WbemErrFailed** 0x8004100 | Error no especificado. |
| **wbemErrIllegalOperation** 0x8004101E | El usuario pidió un funcionamiento ilegal, como desovar una clase de una instancia. |
| **wbemErrIncompleteClass** 0x80041020 | La clase fuente no fue definida completamente o registrada con WMI, de modo que una nueva clase derivada no se permite. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |

**Comentarios**

El objeto devuelto automáticamente se vuelve una subclase del objeto actual. Esta conducta no puede atropellarse. No hay ningún otro método por el que se pueda crear clases derivadas.

No se puede crear una clase derivada de una clase que es local a su propio proceso del cliente. Antes de que se pueda usar este método para crear una clase derivada, se debe crear la clase baja. Para crear la clase baja, se debe llamar a **SWbemObject.Put_**, y recuperar la clase baja que usa SWbemServices.Get.

- **SWbemObject.SpawnInstance_**

Se usa el método **SpawnInstance_** del objeto SWbemObject para crear una nueva instancia de una clase. El objeto actual debe ser una definición de la clase obtenida de WMI vía un método como SWbemServices.Get o SWbemServices.ExecQuery. Entonces, se usa esta definición de la clase para crear nuevos casos. Se debe crear cada nueva instancia localmente dentro del proceso, y entonces se llama a SWbemObject.Put_ para crear la instancia realmente dentro de WMI.

Notese que Desovar una instancia de una instancia es soportada pero la instancia devuelta está vacía.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objNewInstance = objwbemObject.SpawnInstance_ (
[iFlags=0]
);
```

**Parámetros**

**iFlags**

Opcional. Reservado y debe ser cero si se especificó.

**Valores retornados**

Si es exitoso, esta llamada devuelve un objeto SWbemObject que contiene una nueva instancia de la clase.

**Códigos del error**

El método de **SpawnInstance_** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrIncompleteClass** 0xH80041020 | El objeto actual no es una definición de clase válida, y no puede desovar nuevos casos. O está incompleto, o no ha sido registrado con WMI utilizando SWbemObject.Put_. |
| **wbemErrIllegalOperation** 0xH8004101E | Retornó si este método es usado en una instancia en lugar de una clase. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | Nohay suficiente memoria para completar la operación. |

- **SWbemObject.Subclasses_**

El método **Subclasses_** del objeto SWbemObject retorna un objeto SWbemObjectSet. Este objeto es una colección de subclasses del objeto actual que debe ser una clase. Pueden

obtenerse artículos en la colección devuelta utilizando métodos de la colección normales. Ver Colecciones para información sobre cómo usar colecciones.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemObjectSet = objwbemObject.Subclasses_ (
[el iFlags = el wbemFlagReturnImmediately+wbemQueryFlagDeep],
[el objwbemNamedValueSet=null]
);
```

**Parámetros**

**iFlags**

Opcional. Entero que determina cómo profundizar la llamada enumerada. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **WbemQueryFlagDeep**<br>0 | Fuerza la enumeración recursiva en todas las subclases derivadas de la superclase especificada. La propia superclase no se devuelve en la enumeración. |
| **wbemQueryFlagShallow**<br>0x1 | Predefinido para este parámetro. Obliga a la enumeración incluir solamente subclases inmediatas de la superclase especificada. |
| **WbemFlagReturnImmediately**<br>0x1*0* | Causa la llamada para volver inmediatamente |
| **wbemFlagReturnWhenComplete**<br>0x0 | Causa esta llamada para bloquear hasta que la llamada se haya completado. |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Causa WMI para devolver datos de enmendadura de clase junto con la definición de la clase baja. |

| | |
|---|---|
| | Para más información sobre los calificadores enmendados ver Localización de WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si la llamada tiene éxito, un objeto SWbemObjectSet es retornado.

**Códigos del error**

El método de **Subclasses_** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases devueltas por la llamada. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada no existe. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |

**Comentarios**

No es un error para la colección devuelta tener ceros elementos si no hay ningúna subclase del objeto actual. El método **Subclasses_** sólo trabaja para los objetos clase.

- **SWbemObject.SubclassesAsync_**

El método **SubclassesAsync**_ del objeto SWbemObject asincrónicamente retorna una colección de subclases del objeto actual que debe ser una clase. Esta llamada vuelve inmediatamente y se devuelven los resultados del funcionamiento a través del object "sink" proporcionado. Pueden obtenerse artículos en la colección devuelta utilizando métodos de la colección normales. Ver Colecciones para información sobre cómo usar colecciones.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemObject.SubclassesAsync_ (
objWbemSink,
[el iFlags = el wbemQueryFlagDeep],
[el objwbemNamedValueSet=null],
[el objWbemAsyncContext]
);
```

**Parámetros**

**objWbemSink**

Requerido. Object "sink" que recibe los objetos asincrónicamente.

**iFlags**

Opcional. Determina cómo profundiza la llamada enumerada. Este parámetro puede aceptar los siguientes valores.

| Valor | Significado |
|---|---|
| **wbemQueryFlagDeep**<br>0 | Fuerza la enumeración recursiva en todas las subclases derivadas de la superclase especificada. La propia superclase no se devuelve en la enumeración. |
| **wbemQueryFlagShallow**<br>0x1 | Predefinido para este parámetro. Obliga a la enumeración incluir solamente subclases inmediatas de la superclase especificada. |
| **wbemFlagSendStatus**<br>0x80 | Causa las llamadas asíncronas para enviarle actualizaciones de estado almanejador de eventos SWbemSink.OnProgress para el object "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Impide a las llamadas asíncronas enviarle actualizaciones de estado al evento manejador OnProgress para el object "sink". |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Causa WMI para devolver datos de enmendadura de clase con la definición de la clase baja. Para más información sobre los calificadores enmendados ver Localización de WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Éste es un objeto SWbemNamedValueSet que devuelve al object "sink" para identificar la fuente de la llamada asíncrona original. Se usa este parámetro si se está haciendo llamadas asíncronas múltiples que usan el mismo object "sink". Para usar este parámetro, se crea Un objeto Swbem**NamedValueSet** y se usa el método

SWbemNamedValueSet.Add para agregar un valor que identifica la llamada asíncrona que se está haciendo. Este objeto SWbemNamedValueSet es devuelto al object "sink" y la fuente de la llamada puede extraerse utilizando el método de SWbemNamedValueSet.Item. Para más información, ver Haciendo una Llamada Asíncrona utilizando el Scripting API.

**Valores retornados**

Este método no tiene valores de retorno. Si la llamada tiene éxito, un objeto SWbemObjectSet ha devuelto al object "sink" proporcionado.

**Códigos de error**

El método **SubclassesAsync_** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases devueltas por la llamada. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada no existe. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **WbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |

**Comentarios**

No es un error para la colección devuelta tener ceros elementos si no hay ningúna subclase del objeto actual. El método **SubclassesAsync_** sólo trabaja para los objetos clase.

### C.4.8 SWbemObjectPath

Se usan los métodos y propiedades del objeto **SWbemObjectPath** para construir y validar un object path.

#### Métodos

La tabla siguiente lista los métodos para el objeto **SWbemObjectPath**.

| Métodos | Descripción |
|---|---|
| **SetAsClass** | Fuerza el camino para dirigirse a una clase de WMI. |
| SetAsSingleton | Fuerza el camino para dirigirse a una sola instancia de WMI. |

#### Propiedades

La tabla siguiente lista las propiedades para el objeto **SWbemObjectPath**.

| Propiedad | Descripción |
|---|---|
| **Authority** | Cadena que define el componente Authority del **object path**. |
| Class | Nombre de la clase que es parte del **object path**. |
| DisplayName | Cadena que contiene el path en una forma que puede usarse como un nombre de despliegue de moniker. Ver Creación del Objeto y Monikers. |
| **IsClass** | Valor Boleano que indica si este path representa una clase. Esto es análogo a la propiedad de **__Genus** en el COM API. |
| IsSingleton | Valor Boleano que indica si este path representa una sola instancia. |
| Keys | Un objeto **SWbemNamedValueSet** que contiene los |

| | |
|---|---|
| | uniones de valor importantes. |
| Locale | Cadena que contiene el sitio para este object path. |
| "namespace" | Nombre del "namespace" que es parte del object path. Esto es lo mismo que la propiedad __namespace en el COM API. |
| ParentNamespace | Nombre del padre del "namespace" que es parte del object path.. |
| Path | Contiene el path absoluto. Esto es lo mismo que la propiedad del sistema __path en el COM API. Ésta es la propiedad predefinida de este objeto. |
| Relpath | Contiene el path relativo. Esto está igual que la propiedad del sistema __Relpath en el COM API. |
| Security_ | Usada para leer o cambiar las escenas de seguridad. |
| Server | Nombre del servidor. Esto es lo mismo que la propiedad del sistema __Server en el COM API. |

- **SWbemObjectPath.SetAsClass**

El método **SetAsClass** del objeto **SWbemObjectPath** fuerza el camino para dirigir una clase de WMI.

**Valores de retorno**

Este método no tiene valores de retorno.

**Códigos del error**

El método SetAsClass puede devolver el código de error siguiente.

| Error | Significado |
|---|---|
| **WbemErrFailed**<br>0x80041001 | Error no especificado. |

- **SWbemObjectPath.SetAsSingleton**

El método **SetAsSingleton** del objeto **SWbemObjectPath** fuerza el camino para dirigirse a una sola instancia de WMI de una clase. Una sola (singleton) clase es una clase que nunca puede tener más de una instancia.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

**Valores de retorno**

Este método no tiene valores de retorno.

**Códigos del error**

El método **SetAsSingleton** puede devolver el código de error siguiente.

| Error | Significado |
|---|---|
| **WbemErrFailed**<br>0x80041001 | Error no especificado. |

- **SWbemObjectPath.Authority**

La propiedad **Authority** del objeto **SWbemObjectPath** contiene una cadena que define el componente Authority del object path. Para más información, ver el parámetro **strAuthority**

del método SWbemLocator.ConnectServer y Using Kerberos o Autenticación de NTLM con el Scripting API.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
strAuthority = objwbemObjectPath.Authority
objwbemObjectPath.Authority = el strAuthority
```

- **SWbemObjectPath.Class**

La propiedad **Class** del objeto **SWbemObjectPath** es el nombre de la clase que es parte del object path.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
strClass = objwbemObjectPath.Class
objwbemObjectPath.Class = el strClass
```

- **SWbemObjectPath.DisplayName**

La propiedad **DisplayName** del objeto **SWbemObjectPath** es una cadena que contiene el camino en una forma que puede usarse como un nombre de despliegue del moniker.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
strDisplayName = objwbemObjectPath.DisplayName
```

```
objwbemObjectPath.DisplayName = el strDisplayName
```

- **SWbemObjectPath.IsClass**

La propiedad **IsClass** del objeto **SWbemObjectPath** es un valor Boleano que indica si este camino representa una clase. Esta propiedad es solo de lectura.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
bIsClass = objwbemObjectPath.IsClass
```

- **SWbemObjectPath.IsSingleton**

La propiedad **IsSingleton** del objeto **SWbemObjectPath** es un valor Boleano que indica si este camino representa una instancia singleton. Una instancia singleton es una instancia de una clase que nunca puede tener más de una instancia. Esta propiedad es solo de lectura.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
bIsSingleton = objwbemObjectPath.IsSingleton
```

- **SWbemObjectPath.Keys**

La propiedad **Keys** del objeto **SWbemObjectPath** es un objeto **SWbemNamedValueSet** que contiene los ligamentos de valor importantes.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objNamedValueSet = objwbemObjectPath.Keys
```

- **SWbemObjectPath.Locale**

La propiedad **Locale** del objeto **SWbemObjectPath** contiene el sitio del camino del objeto.

Cuando la propiedad de **SWbemObjectPath.Locale** es parte de un objeto standalone **SWbemObjectPath**, se lee y se escribe y puede usarse para poner la localización del componente del Moniker.

Cuando la propiedad **SWbemObjectPath.Locale** se accede como parte de una propiedad de SWbemObject.Path _, es solo de lectura y el valor del sitio que usaron los informes es para ligar al "namespace" desde el cual el objeto fue obtenido.

Para los identificadores de sitios de Microsoft, el formato de la cadena es "MS_xxxx", donde el xxxx es una cadena en forma de hexadecimal que indica el LCID. Por ejemplo, el identificador del sitio Ingles Americano (American English) es "MS_409"

Para más información sobre la localización de WMI ver Localización de WMI.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
strLocale = objwbemObjectPath.Locale
objwbemObjectPath.Locale = el strLocale
```

- **SWbemObjectPath.”namespace”**

La propiedad “namespace” del objeto **SWbemObjectPath** contiene el nombre del “namespace” que es parte del object path.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

strNamespace = objwbemObjectPath.”namespace”
objwbemObjectPath.”namespace” = el strNamespace

- **SWbemObjectPath.ParentNamespace**

La propiedad **ParentNamespace** del objeto **SWbemObjectPath** contiene el nombre del “namespace” del padre que es parte del object path. Esta propiedad es solo de lectura.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
strParentNamespace = objwbemObjectPath.ParentNamespace
```

- **SWbemObjectPath.Path**

La propiedad **Path** del objeto **SWbemObjectPath** contiene el camino absoluto. Esto es igual que la propiedad __Path en el COM API. Esta es la propiedad predefinida de este objeto.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
strPath = objwbemObjectPath.Path
```

```
objwbemObjectPath.Path = el strPath
```

- **SWbemObjectPath.Relpath**

La propiedad **Relpath** del objeto **SWbemObjectPath** contiene el camino relativo.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
strRelPath = objwbemObjectPath.RelPath
objwbemObjectPath.RelPath = el strRelPath
```

- **SWbemObjectPath.Security_**

La propiedad **Security** del objeto **SWbemObjectPath** se usa para leer o poner los componentes de seguridad de un object path. Note que no se debe usar para poner atributos de seguridad del **objetoSWbemObjectPath**. Sólo se usa para representar los componentes de seguridad del camino para un objeto SWbemLocator. Esta propiedad es un objeto SWbemSecurity.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemSecurity = objwbemObjectPath.Security_
```

- **SWbemObjectPath.Server**

La propiedad **Server** del objeto **SWbemObjectPath** contiene el nombre del servidor que es sea parte del camino del objeto.

La sintaxis siguiente es neutral del idioma. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
strServer = objwbemObjectPath.Server
objwbemObjectPath. Servidor = el strServer
```

### C.4.9 SWbemObjectSet

Un objeto **SWbemObjectSet** es una colección de objetos SWbemObject. Para información sobre cómo usar colecciones, ver Colecciones.

Se puede conseguir un objeto **SWbemObjectSet** llamando cualquiera de los siguientes métodos o sus equivalentes asíncronos:

- SWbemServices.InstancesOf
- SWbemServices.SubclassesOf
- SWbemObject.Instances_
- SWbemServices.ExecQuery
- SWbemServices.AssociatorsOf
- SWbemServices.ReferencesTo
- SWbemObject.Associators_
- SWbemObject.References_
- SWbemObject.Subclasses_

Nota: El objeto **SWbemObjectSet** no soporta los métodos **Add** ni **Remove** de la colección.

#### Métodos

La siguiente tabla lista el método para el objeto **SWbemObjectSet**.

| Método | Descripción |
|---|---|
| Item | Recupera un objeto de SWbemObject de la colección. Éste es el método predefinido (por defecto) del objeto. |

**Propiedades**

La siguiente tabla lista las propiedades para el objeto SWbemObjectSet.

| Propiedad | Descripción |
|---|---|
| Count | El número de items en la colección. |
| Security_ | Usado para leer o cambiar las configuraciones de seguridad. |

- **SWbemObjectSet.Item**

El método **Item** del objeto **SWbemObjectSet** consigue un SWbemObject de la colección.

La siguiente sintaxis está en un lenguje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemObject = objwbemObjectSet.Item (
strObjectPath,
[iFlags=0]
);
```

**Parámetros**

**strObjectPath**

Requerido. Ruta relativa del objeto que se quiere recuperar de la colección (por ejemplo, Win32_LogicalDisk = "C":).

**iFlags**

Opcional. Reservado y el valor por defecto es 0.

**Valores retornados**

Si es exitoso, los objetos SWbemObject pedidos retornan.

**Códigos del error**

El método **Item** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |
| **wbemErrNotFound** 0x80041002 | Item requerido no fue encontrado. |

**Comentarios**

El método Item puede ser caro (requiere mucho tiempo del procesador) porque la enumeración completa por el proveedor de los elementos del conjunto exige devolver el resultado.

- **SWbemObjectSet.Count**

Se usa la propiedad **Count** del objeto **SWbemObjectSet** para determinar cuántos items están en una colección **SWbemObjectSet**. Esta propiedad es de solo lectura.

La siguiente sintaxis está en un lenguaje nuetral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
iCount = objWbemObjectSet.Count
```

**Comentarios**

La propiedad **Count** puede ser cara (requiera mucho tiempo del procesador) para recuperar porque la enumeración completa por el proveedor de los elementos del conjunto exige devolver el resultado.

Nota: Si se intenta obtener esta propiedad de un objeto **SWbemObjectSet** que volvió de un método donde está incluida la bandera wbemFlagForwardOnly, se obtendrá un error wbemErrFailed.

- **SWbemObjectSet.Security_**

La propiedad **Security_** del objeto **SWbemObjectSet** se usa para leer o poner settings de seguridad para un objeto **SWbemObjectSet**. Esta propiedad es un objeto **SWbemSecurity**.

La siguiente sintaxis está en un lenguaje nuetral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemSecurity = SWbemObjectSet.Security_
```

### C.4.10 SWbemServices

Se puede usar los métodos de un objeto **SWbemServices** para realizar operaciones en un "namespace" en un "host" local o remoto.

#### Métodos

La siguiente tabla lista los métodos para el objeto **SwbemServices**

| **Método** | **Descripción** |
|---|---|
| AssociatorsOf | Retorna una colección de objetos (clases o instancias) que son asociadas con un objeto especifico. Este método desempeña la misma función que la consulta ASSOCIATORS OF WQL. |
| AssociatorsOfAsync | Retorna asincrónicamente una colección de objetos (clases o instancias) que son asociadas con un objeto especificado. Este método desempeña la misma función que el la consulta ASSOCIATORS OF WQL. |
| Delete | Elimina una instancia o clase. |
| DeleteAsync | Elimina asincrónicamente una instancia o clase. |
| ExecMethod | Ejecuta un objeto método. |
| ExecMethodAsync | Ejecuta asincrónicamente un objeto method. |
| ExecNotificationQuery | Ejecuta un requerimiento para recibir eventos. |
| ExecNotificationQueryAsync | Ejecuta asincrónicamente un requerimiento para recibir eventos. |

| | |
|---|---|
| ExecQuery | Ejecuta un requerimiento para recuperar una colección de objetos (clases o instancias). |
| ExecQueryAsync | Ejecuta asincrónicamente un requerimiento para recuperar una colección de objetos (clases o instancias). |
| Get | Recupera una clase o instancia. |
| GetAsync | Recupera asincrónicamente una clase o instancia |
| InstancesOf | Retorna una colección de instancias de una clase especificada. |
| InstancesOfAsync | Retorna asincrónicamente una colección de instancias de una clase especificada. |
| ReferencesTo | Retorna una colección de objetos (clases o instancias) que se refieren a un objeto simple. Este método desempeña la misma función que la consulta REFERENCES OF WQL. |
| ReferencesToAsync | Retorna asincrónicammente una colección de objetos (clases o instancias) que se refieren a un objeto simple. Este método desempeña la misma función que la consulta REFERENCES OF WQL. |
| SubclassesOf | Retorna una colección de subclases de una clase especificada. |
| SubclassesOfAsync | Retorna asincrónicamente una colección de subclases de una clase especificada. |

**Propiedades**

La siguiente tabla lista la propiedad para el objeto **SWbemServices**.

| **Propiedad** | **Descripción** |
|---|---|
| Security_ | Usada para leer o cambiar las configuraciones de seguridad. |

- **SWbemServices.AssociatorsOf**

El método **AssociatorsOf** del objeto **SWbemServices** retorna una colección de objetos (clases o instancias) que son asociados con un objeto especificado. Estos objetos retornados se llaman "endpoints". Este método realiza la misma función que la consulta ASSOCIATORS OF WQL.

La siguiente sintaxis está en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemObjectSet = objwbemServices.AssociatorsOf(
strObjectPath,
[strAssocClass=""],
[strResultClass=""],
[strResultRole=""],
[strRole=""],
[bClassesOnly=FALSE],
[bSchemaOnly=FALSE],
[strRequiredAssocQualifier=""],
[strRequiredQualifier=""],
[iFlags=wbemFlagReturnImmediately],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strObjectPath**

Requerido. Cadena que contiene la ruta de la clase o instancia fuente.

**strAssocClass**

Opcional. Cadena que contiene una clase de asociación. Si se especificó, este parámetro indica que los "endpoints" retornados deben asociarse con la fuente a través de la clase de asociación especificada o una clase derivada de esta clase de asociación.

**strResultClass**

Opcional. Cadena que contiene el nombre de una clase. Si se especificó, este parámetro Opcional indica que los "endpoints" retornados deben pertenecer o deben ser derivados de la clase especificada en este parámetro.

**strResultRole**

Opcional. Cadena que contiene un nombre de propiedad. Si se especificó, este parámetro indica que los "endpoints" retornados deben jugar un papel particular en su asociación con el objeto fuente. El papel es definido por el nombre de una propiedad especificada (que debe ser una propiedad de referencia) de una asociación.

**strRole**

Opcional. Cadena que contiene un nombre de propiedad. Si se especificó, este parámetro indica que los "endpoints" retornados deben participar en una asociación con el objeto fuente en el que el objeto fuente juega un papel particular. El papel es definido por el nombre de una propiedad especificada (que debe ser una propiedad de referencia) de una asociación.

**bClassesOnly**

Opcional. Valor Boolean que indica si una lista de nombres de clases debe volverse en lugar de las instancias reales de las clases. Estas son las clases a las que las instancias del "endpoint" pertenecen. El valor predefinido para este parámetro es FALSO.

**bSchemaOnly**

Opcional. Valor Boolean que indica si la consulta se aplica al esquema en lugar de a los datos. El valor predefinido para este parámetro es FALSO. Sólo puede ponerse VERDADERO si el parámetro del strObjectPath especifica la ruta del objeto de una clase.

Cuando se pone VERDADERO, el conjunto de "endpoints" retornado representan las clases que son adecuadamente asociadas con la clase fuente en el esquema.

**strRequiredAssocQualifier**

Opcional. Cadena que contiene un nombre del calificador. Si se especificó, este parámetro indica que los "endpoints" retornados deben asociarse con el objeto fuente a través de una clase de asociación que incluye el calificador especificado.

**strRequiredQualifier**

Opcional. Cadena que contiene un nombre del calificador. Si se especificó, este parámetro indica que los "endpoints" retornados deben incluir el calificador especificado.

**iFlags**

Opcional. Entero que especifica banderas adicionales a la operación. El valor predefinido para este parámetro es wbemFlagReturnImmediately que dirige la llamada para ser retornada inmediatamente en lugar de esperar a que la consulta se haya completado. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **WbemFlagForwardOnly** 0x20 | Causa que un enumerador delantero-único (Forward-only) sea retornado. Los enumeradores Delantero-único generalmente son muy rápidos y usan menos memoria que los emnumeradores convencionales, pero ellos no le permiten llamadas a SWbemObject.Clone. |
| **WbemFlagBidirectional** 0x0 | Causa WMI para retener indicadores para los objetos de la enumeración hasta que el cliente libere el enumerador |
| **WbemFlagReturnImmediately** 0x10 | Causa la llamada para volver inmediatamente. |
| **WbemFlagReturnWhenComplete** | Causa esta llamada para bloquear hasta que la |

| | |
|---|---|
| 0x0 | consulta se haya completado |
| **WbemFlagUtiliceAmendedQualifiers** 0x20000 | Causa WMI para devolver datos enmendados de clase (class amendment data) junto con la definición de la clase base. |

**objwbemNamedValueSet**

Opcional. Típicamente, esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el reconocimiento de los valores nombrados, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si la llamada tiene éxito, un objeto SWbemObjectSet es retornado.

**Códigos del error**

El método **AssociatorsOf** puede devolver los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases retornadas por la llamada. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar esta operación. |

| | |
|---|---|
| **wbemErrNotFound**<br>0x80041002 | El artículo requerido no fue encontrado. |

**Comentarios**

No es un error para la colección retornada tener cero elementos.

- **SWbemServices.AssociatorsOfAsync**

El método **AssociatorsOfAsync** del objeto **SWbemServices** asincrónicamente retorna una colección de objetos (clases o instancias) que son asociados con un objeto especificado. Estos objetos retornados se llaman "endpoints". Este método realiza la misma función que la consulta ASSOCIATORS OF WQL. Esta llamada vuelve inmediatamente y el resultado y el estado son retornados al llamante a través del objeto "sink" proporcionado

La siguiente sintaxis está en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemServices.AssociatorsOfAsync(
objWbemSink
strObjectPath,
[strAssocClass=""],
[strResultClass=""],
[strResultRole=""],
[strRole=""],
[bClassesOnly=FALSE],
[bSchemaOnly=FALSE],
[strRequiredAssocQualifier=""],
[strRequiredQualifier=""],
```

```
[iFlags = wbemFlagDontSendStatus],
[objWbemNamedValueSet=null],
[objWbemAsyncContext=null]
);
```

**Parámetros**

**objWbemSink**

Requerido. Objeto "sink" que recibe los objetos asincrónicamente.

**strObjectPath**

Requerido. Cadena que contiene la ruta del objeto de la clase o instancia fuente.

**strAssocClass**

Opcional. Cadena que contiene una clase de asociación. Si se especificó, este parámetro indica que los "endpoints" retornados deben asociarse con la fuente a través de la clase de asociación especificada o una clase derivada de esta clase de asociación.

**strResultClass**

Opcional. Cadena que contiene un nombre de la clase. Si se especificó, este parámetro Opcional indica que los "endpoints" retornados deben pertenecer o se deben derivar de la clase especificada en este parámetro.

**strResultRole**

Opcional. Cadena que contiene un nombre de propiedad. Si se especificó, este parámetro indica que los "endpoints" retornados deben jugar un papel particular en su asociación con el objeto fuente. El papel es definido por el nombre de una propiedad especificada (que debe ser una propiedad de referencia) de una asociación.

**strRole**

Opcional. Cadena que contiene un nombre de propiedad. Si se especificó, este parámetro indica que los "endpoints" retornados deben participar en una asociación con el objeto fuente

en el que el objeto fuente juega un papel particular. El papel es definido por el nombre de una propiedad especificada (que debe ser una propiedad de referencia) de una asociación.

**bClassesOnly**

Opcional. Valor Booleano que indica si una lista de nombres de la clase debe ser retornada en lugar de las instancias actuales de las clases. Estas son las clases a las que las instancias del "endpoint" pertenecen. El valor predefinido para este parámetro es FALSO.

**bSchemaOnly**

Opcional. Valor Booleano que indica si la pregunta se aplica al esquema en lugar de a los datos. El valor predefinido para este parámetro es FALSO. Sólo puede ponerse VERDADERO si el parámetro del strObjectPath especifica la ruta del objeto de una clase. Cuando se pone VERDADERO, el conjunto de "endpoints" retornado representan clases que son adecuadamente asociadas con la clase fuente en el esquema.

**strRequiredAssocQualifier**

Opcional. Cadena que contiene un nombre del calificador. Si se especificó, este parámetro indica que los "endpoints" retornados deben asociarse con el objeto fuente a través de una clase de asociación que incluye el calificador especificado.

**strRequiredQualifier**

Opcional. Cadena que contiene un nombre del calificador. Si se especificó, este parámetro indica que los "endpoints" retornados deben incluir el calificador especificado.

**iFlags**

Opcional. Entero que especifica banderas adicionales a la operación. El valor por defecto para este parámetro es wbemFlagDontSendStatus. Este parámetro puede aceptar los siguientes valores.

| Valor | Significado |
|---|---|
| **wbemFlagSendStatus**<br>0x80 | Causa las llamadas asíncronas para enviar actualizaciones de estado al manejador de eventos OnProgress para el objeto "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Impide a las llamadas asíncronas enviarle actualizaciones de estado al manejador de eventos OnProgress del objeto "sink". |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Causa WMI para devolver datos enmendados de clase junto con la definición de la clase base. |

**objWbemNamedValueSet**

Opcional. Típicamente, esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el reconocimiento de los valores nombrados, tipos de datos de los valores, valores permitidos, y semántica.

**objWbemAsyncContext**

Opcional. Un objeto SWbemNamedValueSet que vuelve al objeto "sink" para identificar la fuente de la llamada asíncrona original. Se usa este parámetro si se está haciendo múltiples llamadas asíncronas que usan el mismo objeto "sink". Para usar este parámetro, se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para agregar un valor que identifica la llamada asíncrona que se está haciendo. Este objeto SWbemNamedValueSet es retornado al objeto "sink" y la fuente de la llamada puede extraerse utilizando el método SWbemNamedValueSet.Item. Para más información, ver Making an Asynchronous Call Using the Scripting API.

**Valores retornados**

Este método no retorna valores. Si la llamada tiene éxito, un objeto SWbemObjectSet es devuelto al objeto "sink".

**Códigos del error**

El método **AssociatorsOfAsync** puede devolver los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases retornadas por la llamada. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidParameter** 0x80041008 | Parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para continuar la operación. |
| **wbemErrNotFound** 0x80041002 | El item requerido no fue encontrado. |

**Comentarios**

El método **AssociatorsOfAsync** devuelve una colección que tiene cero elementos si no hay ninguna clase o instancias asociadas con el objeto fuente.

- **SWbemServices.Delete**

El método **Delete** del objeto **SWbemServices** anula la clase o la instancia especificada en la ruta del objeto. Se puede anular sólo objetos en el "namespace" actual.

Si un proveedor dinámico proporcionara la clase o instancia, a veces no es posible anular este objeto a menos que el proveedor soporte la anulación de la clase o de la instancia.

La siguiente sintaxis está en un lenguaje neural. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemServices.Delete(
strObjectPath,
[iFlags=0],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strObjectPath**

Requerido. Cadena que contiene la ruta del objeto que se quiere anular. Las rutas de los objetos pueden ser construidas por un manejador o por medio del objeto SWbemObjectPath o la propiedad Path_ de una instancia de SWbemObject.

**iFlags**

Reservado. Debe ser cero.

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que

puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el reconocimiento de los valores nombrados, tipos de datos de los valores, valores permitidos, y semántica.

**Valores retornados**

Este método no tiene Valores retornados.

**Códigos del error**

El método **Delete** puede devolver los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El contexto actual no tiene los derechos de seguridad adecuados para anular el objeto. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada no existe. |
| **wbemErrInvalidOperation** 0x80041016 | El objeto no puede ser eliminado. |
| **wbemErrNotFound** 0x80041002 | El objeto no existe. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |

- **SWbemServices.DeleteAsync**

El método **DeleteAsync** del objeto **SWbemServices** anula asincrónicamente la clase o la instancia especificada en la ruta del objeto. Esta llamada vuelve inmediatamente y el estado

es devuelto al llamante a través del objeto "sink" proporcionado. Se puede anular sólo objetos en el "namespace" actual.

Si un proveedor dinámico proporcionara la clase o instancia, a veces no es posible anular este objeto a menos que el proveedor soporte la anulación de la clase o de la instancia.

La siguiente sintaxis está en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemServices.DeleteAsync(
[ObjWbemSink = null],
strObjectPath,
[iFlags=0],
[objWbemNamedValueSet=null],
[objWbemAsyncContext=null]
);
```

**Parámetros**

**ObjWbemSink**

Opcional. Objeto "sink" que recibe los resultados de la anulación.

**strObjectPath**

Requerido. Cadena que contiene la ruta del objeto que se quiere anular. Las rutas de los objetos pueden construirse con un manejador.o por medio del objeto SWbemObjectPath o la propiedad Path_ de una instancia de SWbemObject.

**iFlags**

Opcional. Determina si se devuelve la actualización de estado. Este parámetro puede aceptar los siguientes valores.

| Valor | Significado |
|---|---|
| **wbemFlagSendStatus**<br>0x80 | Motiva a las llamadas asíncronas a enviar actualizaciones de estado al manejador de eventos OnProgress para el objeto "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Impide a las llamadas asíncronas enviarle actualizaciones de estado al manejador de eventos OnProgress para el objeto "sink" |

**objWbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el reconocimiento de los valores nombrados, tipos de datos de los valores, valores permitidos, y semántica.

**objWbemAsyncContext**

Opcional. Un objeto SWbemNamedValueSet que devuelve al object "sink" para identificar la fuente de la llamada asíncrona original. Se usa este parámetro si se está haciendo llamadas asíncronas múltiples que usan el mismo object "sink". Para usar este parámetro, se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para agregar un valor que identifique la llamada asíncrona que se está haciendo. Este objeto SWbemNamedValueSet es devuelto al object "sink" y la fuente de la llamada puede extraerse utilizando el método SWbemNamedValueSet.Item. Para más información, ver Haciendo una Llamada Asíncrona utilizando el Scripting API.

**Valores retornados**

Este método no tiene Valores retornados. Si la llamada tiene éxito, el object "sink" recibirá notificación de la anulación.

**Códigos de error**

El método **DeleteAsync** puede devolver los códigos de error listados en la siguiente tabla.

| Error | Significado |
|---|---|
| **WbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |
| **wbemErrTransportFailure** 0x80041015 | Error de red ocurrido, previniendo la operación normal. |
| **wbemErrAccessDenied** 0x80041003 | La contraseña y el nombre del usuario especificado o actual no son válidos o autorizados para hacer la conexión. |
| **wbemErrNotFound** 0x80041002 | El item pedido no fue encontrado. |

- **SWbemServices.ExecMethod**

El método **ExecMethod** del objeto **SWbemServices** ejecuta un método exportado por un proveedor de métodos. Este método puede usarse como una alternativa para ejecutar un método en casos donde no es posible ejecutar un método directamente. Por ejemplo, si el método tiene los parámetros fuera, se usaría este método con un idioma del scripting como JScript que no soporta parámetros de salida (output). Por otra parte, se recomienda que se utilice la invocación directa del método la cual ejecuta un método utilizando la sintaxis de object.method. Por ejemplo, para ejecutar un método llamado "**get**" sobre un objeto llamado "myobject", debe usarse la sintaxis: "myobject.get". Para más información sobre invocación directa del método.

Este método bloquea mientras el método que se remite al proveedor apropiado se ejecuta. Se devuelven entonces, la información y los estados.

WMI no lleva a cabo los métodos directamente. Los proveedores del método exportan dichos métodos.

La siguiente sintaxis está en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objOutParams = objwbemServices.ExecMethod(
strObjectPath,
strMethodName,
[objwbemInParams=null],
[iFlags=0],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strObjectPath**

Requerido. Cadena que contiene la ruta del objeto sobre el que se va ha ejecutar el método. Las rutas de los objetos pueden contruirse con manejadores o a través del objeto SWbemObjectPath o la propiedad Path_ de una instancia de SWbemObject.

**strMethodName**

Requerido. El nombre del método para el objeto.

**objwbemInParams**

Opcional. Un objeto SWbemObject que contiene los parámetros de entrada para el método que está siendo ejecutado. Por defecto, este parámetro es indefinido. Para información de cómo construir este parámetro, debe verse la sección de los Comentarios.

**iFlags**

Opcional. Reservado; el valor debe ser cero.

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si el método tiene éxito, se retorna un objeto SWbemObject. El objeto devuelto contiene los parámetros de salida y el valor retornado para el método que está siendo ejecutado.

**Códigos del error**

El método **ExecMethod** puede devolver los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrFailed**<br>0x80041001 | Error no especificado. |
| **wbemErrInvalidClass**<br>0x80041010 | La clase especificada es inválida. |
| **wbemErrInvalidParameter**<br>0x80041008 | Un parámetro inválido fue especificado. |

| | |
|---|---|
| **wbemErrOutOfMemory**<br>0x80041006 | No hay suficiente memoria para completar la operación. |
| **wbemErrInvalidMethod**<br>0x8004102E | El método requerido no está disponible. |
| **wbemErrAccessDenied**<br>0x80041003 | El usuario actual no fue autorizado para ejecutar el método. |

**Comentarios**

Para construir el parámetro objwbemInParams

1. Se debe llamar a **SWbemObject.Item** ("MethodName") para conseguir un objeto **SWbemMethod** que contiene las propiedades que describen el método, donde "MethodName" es el nombre del método a recuperar.

2. Se usa la propiedad **SWbemMethod.InParameters** del objeto **SWbemMethod** para conseguir un objeto de **InParameters**.

3. Se debe llamar a **SpawnInstance_** en el objeto **InParameters** para crear una instancia.

4. Se pone las propiedades de la instancia a cualquiera de los valores que sean apropiados. Por ejemplo, Si se quiere poner un parámetro de entrada nombrado myinputparam al valor "whatever" en una instancia de **InParameters** llamado "INST", el código se parecería a esto: **INST.Properties_.Add** ("myinputparam") .Value = "whatever"

- **SWbemServices.ExecMethodAsync**

El método **ExecMethodAsync** del objeto **SWbemServices** ejecuta un método traído asincrónicamente por un proveedor del método. La llamada retorna inmediatamente al cliente mientras los parámetros de entrada se remiten al proveedor apropiado donde el método se ejecuta. Se devuelven información y el estado al llamante a través del objeto "sink" proporcionado.

Este método puede usarse como una alternativa para ejecutar un método cuando no es posible ejecutarlo directamente. Si su objeto tiene parámetros de salida, por ejemplo, se usaría este método en un lenguaje scripting como JScript que no soporta parámetros de rendimiento (output). Por otra parte, el medio recomendado para invocar a un método es usar la invocación directa del mismo. La invocación directa del método ejecuta un método utilizando la sintaxis de object.method. Por ejemplo, para ejecutar un método llamado "get" sore un objeto llamado "myobject" se usa la sintaxis: myobject.get. Para más información sobre invocación directa del método, se debe ver Acceso Directo.

WMI no lleva a cabo los métodos directamente. Los proveedores del método exportan los métodos.

La siguiente sintaxis está en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemServices.ExecMethodAsync(
objWbemSink,
strObjectPath,
strMethodName,
[objWbemInParams=null],
[iFlags=0],
[objWbemNamedValueSet=null],
```

```
[objWbemAsyncContext=null]
);
```

**Parámetros**

**objWbemSink**

Requerido. Object "sink" que recibe el resultado de la llamada al método. Los parámetros que salen se envían al evento **SWbemSink.OnObjectReady** del object "sink" proporcionado. El resultado del mecanismo de la llamada se envía al evento SWbemSink.OnCompleted del object "sink" proporcionado. Se debe notar que ese SWbemSink.OnCompleted no recibe el código de retorno del método de WMI. Sin embargo, recibe el código de retorno del mecanismo actual ( ó real) de la llamada-retornada y es sólo útil para verificar que la llamada ocurrió o que falló por razones mecánicas. El código de resultado devuelto desde el método de WMI es retornado en el objeto de los parámetros que sale proporcionado a **WbemSink.OnObjectReady**. Si cualquier código de error retorna, entonces el objeto IWbemObjectSink proporcionado no se usa. Si la llamada tiene éxito, entonces la aplicación IWbemObjectSink del usuario es llamada para indicar el resultado del funcionamiento.

**strObjectPath**

Requerido. Cadena que contiene la ruta del objeto sobre el cual se está ejecutando el método. Las rutas de los objeto pueden contruirse con un manejador o utilizando el objetoSWbemObjectPath o la propiedad Path_ de una instancia de SWbemObject.

**strMethodName**

Requerido. El nombre del método a ser ejecutado.

**objWbemInParams**

Opcional. Un objeto SWbemObject que contiene los parámetros de entrada para el método que está siendo ejecutado. Por defecto, este parámetro es indefinido. Para información de cómo construir este parámetro, debe verse la sección de los Comentarios.

**iFlags**

Opcional. Entero que determina la conducta de la llamada. Este parámetro puede aceptar los valores siguientes.

| Valor | Significado |
|---|---|
| **wbemFlagSendStatus** 0x80 | Causa las llamadas asíncronas para enviarle actualizaciones de estado al manejador de eventos OnProgress del object "sink". |
| **wbemFlagDontSendStatus** 0x0 | Impide a las llamadas asíncronas enviarle actualizaciones de estado al manejador de eventos OnProgress para el object "sink". |

**objWbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Un objeto SWbemNamedValueSet que se devuelve al object "sink" para identificar la fuente de la llamada asíncrona original. Se usa este parámetro si se está haciendo llamadas asíncronas múltiples que usan el mismo object "sink". Para usar este parámetro, se debe crear un objeto SWbemNamedValueSet y usar el método SWbemNamedValueSet.Add para agregar un valor que identifica la llamada asíncrona que se está haciendo. Este objeto SWbemNamedValueSet es retornado al object "sink" y la fuente de la llamada puede extraerse utilizando el método SWbemNamedValueSet.Item. Para más información, se debe ver Haciendo una Llamada Asíncrona utilizando el Scripting API.

**Valores retornados**

Este método no retorna valores. Si esta llamada tiene éxito, el object "sink" retorna un objeto SWbemObject. El objeto retornado contiene los parámetros de salida y retorna el valor para el método que está siendo ejecutado.

**Códigos de error**

El método **ExecMethodAsync** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada es inválida. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para continuar la operación. |
| **wbemErrInvalidMethod** 0x8004102E | El método requerido no está disponible. |
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no fue autorizado para ejecutar al método. |

**Comentarios**

Para construir el parámetro objwbemInParams

1. Se debe llamar a **SWbemObject.Methods_.Item** ("MethodName") para conseguir un objeto **SWbemMethod** que contiene las propiedades que describen el método, donde "MethodName" es el nombre del método para recuperar.

2. Se usa la propiedad **SWbemMethod.InParameters** del objeto **SWbemMethod** para conseguir un objeto de **InParameters**.

3. Se debe llamar a **SpawnInstance_** en el objeto **InParameters** para crear una instancia.

4. Se ponen las propiedades de la instancia a cualquiera de los valores que sean apropiados. Por ejemplo, Si se quiere poner un parámetro de entrada llamado myinputparam al valor "whatever" en una instancia de InParameters llamado "INST", el código se parecería a esto: **INST.Properties_.Add** ("myinputparam").Value = "whatever"

- **SWbemServices.ExecNotificationQuery**

El método **ExecNotificationQuery** del objeto **SWbemServices** ejecuta una consulta para recibir eventos. La llamada vuelve inmediatamente. El usuario puede registrar el enumerator retornado para cuando llegen los eventos.
La siguiente sintaxis está en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento

```
objwbemEventsource = objwbemServices.ExecNotificationQuery(
strQuery,
[strQueryLanguage="WQL"],
[iFlags=(wbemFlagForwardOnly + wbemFlagReturnImmediately)],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strQuery**

Requerido. Cadena que contiene el texto de la consulta relacionada con el evento. Este parámetro no puede estar en blanco.

**strQueryLanguage**

Opcional. Cadena que contiene el idioma de la consulta a ser usado. Si se especificó, este valor debe ser "WQL."

**iFlags**

Opcional. Éste es un entero que determina la conducta de la consulta. El valor predefinido es **wbemFlagReturnImmediately** + **wbemFlagForwardOnly**. Si se especifica este parámetro debe ponerse en **wbemFlagReturnImmediately** y **wbemFlagForwardOnly** o la llamada fallará. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemFlagForwardOnly** 0x20 | Causa un enumerador delantero-único para ser devuelto. Los enumeradores delanteros-unicamente (Forward-only) generalmente son muy más rápidos y usan menos memoria que los enumeradores convencionales, pero ellos no le permiten llamadas a SWbemObject.Clone_. |
| **wbemFlagReturnImmediately** 0x10 | Causa la llamada para retornarla inmediatamente. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si ningún error ocurre, este método devuelve un objeto SWbemEventSource. Se puede llamar el método SWbemEventSource.NextEvent para recuperar eventos cuando ellos llegen.

**Códigos del error**

El método **ExecNotificationQuery** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no está autorizado para ver el conjunto de resultados. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrInvalidQuery** 0x80041017 | La sintaxis de la pregunta no es válida. |
| **wbemErrInvalidQueryType** 0x80041018 | El idioma de la pregunta pedido no está soportado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completarla operación. |

- **SWbemServices.ExecNotificationQueryAsync**

El método **ExecNotificationQueryAsync** del objeto **SWbemServices** ejecuta una pregunta asincrónicamente para recibir eventos. Esta llamada vuelve inmediatamente. Se devuelven el estado y los eventos a la llamada a través del objeto "sink" proporcionado.

La siguiente sintaxis está en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemServices.ExecNotificationQueryAsync(
objWbemSink,
strQuery,
[strQueryLanguage="WQL"],
[iFlags=0],
[objwbemNamedValueSet=null],
[objWbemAsyncContext=null]
);
```

**Parámetros**

**objWbemSink**

Requerido. Objeto “sink” que recibe notificación de eventos asincrónicamente.

**strQuery**

Requerido. Cadena que contiene el texto de la pregunta consulta relacionada con el evento. No puede estar en blanco.

**strQueryLanguage**

Opcional. Cadena que contiene el idioma de la consulta a ser usado. Si se especificó, este valor debe ser "WQL."

**iFlags**

Opcional. Entero que determina la conducta de la consulta. Este parámetro puede ponerse a los siguientes valores.

| Valor | Significado |
|---|---|
| **wbemFlagSendStatus**<br>0x80 | Causa las llamadas asíncronas para enviarle actualizaciones de estado al manejador de eventos OnProgress del objeto "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Impide a las llamadas asíncronas enviarle actualizaciones de estado al manejador de eventos OnProgress del objeto "sink". |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Éste es un objeto SWbemNamedValueSet se que devuelve al objeto "sink" para identificar la fuente de la llamada asíncrona original. Se usa este parámetro si se está haciendo llamadas asíncronas múltiples que usan el mismo objeto "sink". Para usar este parámetro, se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para agregar un valor que identifica la llamada asíncrona que se está haciendo. Este objeto SWbemNamedValueSet es retornado al objeto "sink" y la fuente de la llamada puede extraerse utilizando el método SWbemNamedValueSet.Item. Para más información, debe verse Haciendo una Llamada Asíncrona utilizando el Scripting API.

**Valores retornados**

Este método no tiene Valores retornados. Si es exitoso, el objeto "sink" recibe objetos SWbemObject a medida que los eventos estén disponibles.

**Códigos del error**

El método de **ExecNotificationQueryAsync** puede devolver los códigos del error listados en la siguiente tabla.

| Error | Significado |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no es autorizado para ver el conjunto de resultados. |
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrInvalidQuery** 0x80041017 | La sintaxis de la consulta no es válida. |
| **wbemErrInvalidQueryType** 0x80041018 | El idioma de la consulta pedido no está soportado. |
| **wbemErrOutOfMemory** 0x80041006 | No hay suficiente memoria para completar la operación. |

- **SWbemServices.ExecQuery**

El método **ExecQuery** del objeto **SWbemServices** ejecuta una consulta para recuperar objetos. Estos objetos están disponibles a través de la colección SWbemObjectSet retornada.

La siguiente sintaxis siguiente está en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objWbemObjectSet = objwbemServices.ExecQuery(
strQuery,
[strQueryLanguage="WQL"],
```

```
[iFlags=wbemFlagReturnImmediately],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strQuery**

Requerido. Cadena que contiene el texto de la consulta. Este parámetro no puede estar en blanco.

**strQueryLanguage**

Opcional. Cadena que contiene el idioma de la consulta a ser usado. Si se especificó, este valor debe ser "WQL."

**iFlags**

Opcional. Entero que determina la conducta de la pregunta y determina si esta llamada vuelve inmediatamente. El valor predefinido para este parámetro es wbemFlagReturnImmediately. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemFlagForwardOnly** 0x20 | Causa un empadronador delantero-único a ser vuelto. Enumeadores delanteros-unicamente generalmente son más rápidos y usan menos memoria que los empadronadores convencionales, pero ellos no le permiten llamadas a SWbemObject.Clone_. |
| **wbemFlagBidirectional** 0x0 | Causa WMI para retener indicados (apuntados) a los objetos de la enumeración hasta que el cliente suelte el enumerador. |
| **wbemFlagReturnImmediately** 0x10 | Causa la llamada para devolverla inmediatamente. |

| | |
|---|---|
| **wbemFlagReturnWhenComplete**<br>0x0 | Causa esta llamada para bloquearla hasta que la pregunta se halla completado. |
| **wbemQueryFlagPrototype**<br>0x2 | Usado para el prototyping. Detiene la pregunta desde que pasa y en cambio retorna un objeto que se parece a un típico resultado de objeto. |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Causa WMI para devolver datos de enmendadura de clase con la definición de la clase base. Para más información sobre los calificadores enmendados ver Localización de WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si ningún error ocurre, este método devuelve un objeto SwbemObjectSet que es una colección de objeto que contiene el resultado de la consulta. El llamante puede examinar la colección utilizando la aplicación de colecciones para el lenguaje de programación que se está utilizando. Para más información, ver Colecciones.

**Códigos del error**

El método **ExecQuery** puede devolver los códigos del error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied**<br>0x80041003 | El usuario actual no tiene permiso para ver el conjunto de resultados. |

| | |
|---|---|
| **wbemErrFailed**<br>0x80041001 | Error no especificado. |
| **wbemErrInvalidParameter**<br>0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrInvalidQuery**<br>0x80041017 | La sintaxis de la consulta no es válida. |
| **wbemErrInvalidQueryType**<br>0x80041018 | El idioma pedido de la consulta no está soportado. |
| **wbemErrOutOfMemory**<br>0x80041006 | No hay suficiente memoria para completar la operación. |

**Comentarios**

No es un error para la consulta devolver un conjunto vacío en el resultado. El método **ExecQuery** devuelve propiedades importantes sin tener en cuenta si la propiedad importante se pide en el argumento del strQuery. Si un error ocurre al ejecutar este método y se usa la bandera wbemFlagReturnImmediately, el objeto "err" no es fijado hasta que se intente acceder el "set" del objeto devuelto. Sin embargo, si se usa la bandera wbemFlagReturnWhenComplete, el objeto "err" no se fija hasta cuando se llame al método ExecQuery.

- **SWbemServices.ExecQueryAsync**

El método **ExecQueryAsync** del objeto **SWbemServices** ejecuta asincrónicamente una consulta para devolver los objetos. Estos objetos están disponibles como una colección SWbemObjectSet que retorna al objeto "sink".

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objWbemObjectSet = objwbemServices.ExecQueryAsync(
objWbemSink,
strQuery,
[strQueryLanguage="WQL"],
[iFlags],
[objwbemNamedValueSet=null],
[objWbemAsyncContext=null]
);
```

**Parámetros**

**objWbemSink**

Opcional. Objeto “sink” que ejecuta la consulta asincrónicamente.

**strQuery**

Requerido. Cadena que contiene el texto de la consulta. Este parámetro no puede ser vació.

**strQueryLanguage**

Opcional. Cadena que contiene el lenguaje de consulta a ser usado. Si se especifica, el valor debe ser "WQL".

**iFlags**

Opcional. Entero que determina el valor de la consulta. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **WbemFlagSendStatus**<br>0x80 | Hace que las llamadas asíncronas envíen actualizaciones de estado al manejador de eventos SWbemSink.OnProgress del objeto “sink”. |
| **WbemFlagDontSendStatus**<br>0x0 | Impide que las llamadas asíncronas envien actualizaciones de estado al manejador de eventos |

| | |
|---|---|
| | OnProgress del objeto "sink". |
| **WbemQueryFlagPrototype**<br>0x2 | Usado para prototipado. Este detiene la consulta cuando está sucediendo y en su lugar retorna un objeto que luce como un tipo objeto de resultados. |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Este es un objeto SWbemNamedValueSet que retorna al objeto "sink" para identificar la fuente original de la llamada asíncrona. Se usa este parámetro si se hacen múltiples llamadas asíncronas utilizando el mismo objeto "sink". Para usar este parámetro, se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para adicionar un valor que identifique a la llamada asíncrona que se esté haciendo. Este objeto SWbemNamedValueSet es devuelto al objeto "sink" y la fuente de la llamada puede ser extraída utilizando el método SWbemNamedValueSet.Item. Para más información, ver Haciendo una llamada asincrónica utilizando el Scripting API.

**Valores retornados**

Si es exitoso, este método retorna un objeto "sink". Si no ocurren errores el objeto "sink" recibe un objeto SWbemObjectSet que es una colección de objetos que contiene el resultado de la consulta. El llamante puede examinar la colección utilizando la implementación de

colecciones para el lenguaje de programación que se esta utilizando. Para más información sobre colecciones, ver Colecciones.

**Códigos de error**

El método **ExecQueryAsync** puede retornar los siguientes códigos de error.

| **Error** | **Significado** |
|---|---|
| **WbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver el resultado. |
| **WbemErrFailed** 0x80041001 | Error no específico. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **WbemErrInvalidQuery** 0x80041017 | La sintaxis de la consulta no es válida. |
| **wbemErrInvalidQueryType** 0x80041018 | La petición del lenguaje de consulta no es soportada. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

**Comentarios**

El método **ExQueryAsync** retorna un resultado vació si no hay objetos que sean compatibles con el criterio de la consulta. Este método retorna propiedades principales a menos que la propiedad principal sea requerida en el parámetro *strQuery*.

- **SWbemServices.Get**

El método **Get** del objeto **SWbemServices** devuelve un objeto o una definición de clase o una instancia basado en la ruta de acceso al objeto. Este método solo devuelve objetos del "namespace" asociado con el objeto **SWbemServices** actual.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objWbemObject = objwbemServices.Get(
[strObjectPath=""],
[iFlags=0],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strObjectPath**

Opcional. Ruta de acceso del objeto a ser recibido. Si este es vacío, retorna un objeto vacío que puede llegar a ser una nueva clase. La ruta del objeto puede ser construida por un manejador o utilizando el objeto SWbemObjectPath o la propiedad Path_ de las instancias de SWbemObject.

**iFlags**

Opcional. Entero que determina el valor de la llamada. Este parámetro acepta los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **WbemFlagUtiliceAmendedQualifiers** 0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si es exitoso, este método retorna un objeto SWbemObject representando al objeto pedido.

**Códigos de error**

El método **Get** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para acceder al objeto. |
| **wbemErrFailed** 0x80041001 | Error no específico. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrInvalidObjectPath** 0xH8004103A | La ruta especificada fue inválida. |
| **wbemErrNotFound** 0x80041002 | El objeto requerido no fue encontrado. |
| **WbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

- **SWbemServices.GetAsync**

El método **GetAsync** del objeto **SWbemServices** devuelve asincrónicamente un objeto, una clase o una instancia, basado en la ruta del objeto. Esta llamada retorna inmediatamente el estado y el objeto son retornados al llamante a través del objeto "sink" provisto para tal fin.

Este método solo devuelve objetos del "namespace" asociado con el objeto **SWbemServices** actual.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objwbemServices.GetAsync(
objWbemSink = null,
[strObjectPath=""],
[iFlags=0],
[objwbemNamedValueSet=null],
[objWbemAsyncContext=null]
);
```

**Parámetros**

**objWbemSink**

Requerido. Objeto "sink" que obtiene objetos asincrónicamente.

**strObjectPath**

Opcional. Ruta de acceso del objeto a ser recibido. Si este es vacío, el objeto vacío retornado puede llegar a ser una nueva clase. La ruta del objeto puede ser construida por un manejador o utilizando el objeto SWbemObjectPath o la propiedad Path de las instancias de SWbemObject.

**iFlags**

Opcional. Entero que determina el valor de la llamada. Este parámetro acepta los siguientes valores.

| Valor | Significado |
|---|---|
| **WbemFlagSendStatus**<br>0x80 | Hace que las llamadas asíncronas envíen actualizaciones de estado al manejador de eventos SWbemSink.OnProgress del objeto "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Previene a las llamadas asíncronas para no enviar actualizaciones de estado al manejador de eventos OnProgress para el objeto "sink". |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Este es un objeto SWbemNamedValueSet que retorna al objeto "sink" para identificar la fuente original de la llamada asíncrona. Se usa este parámetro si se hacen múltiples llamadas asíncronas utilizando el mismo objeto "sink". Para usar este parámetro, se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para adicionar un valor que identifique la llamada asíncrona que se esté haciendo. Este objeto SWbemNamedValueSet es devuelto al objeto "sink" y la fuente de la llamada puede ser

extraída utilizando el método SWbemNamedValueSet.Item. Para más información, ver Haciendo una llamada asincrónica utilizando el Scripting API**.**

**Valores retornados.**

Este método no retorna valores. Si la llamada es exitosa, el objeto "sink" recibe un objeto SWbemObject representando el objeto pedido.

**Códigos de error**

El método **GetAsync** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para acceder al objeto. |
| **wbemErrFailed** 0x80041001 | Error no específico |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrInvalidObjectPath** 0xH8004103A | La ruta especificada fue inválida. |
| **wbemErrNotFound** 0x80041002 | El objeto requerido no fue encontrado. |
| **WbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

- **SWbemServices.InstancesOf**

El método **InstancesOf** del objeto **SWbemServices** crea un enumerador que retorna las instancias de una clase específica de acuerdo al criterio de selección dado por el usuario. Este

método implementa una consulta simple. Para consultas mas complejas se requiere el uso de SWbemServices.ExecQuery.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objWbemObjectSet = objwbemServices.InstancesOf(
strClass,
[iFlags=wbemFlagReturnImmediately],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strClass**

Requerido. Cadena que contiene el nombre de la clase para la cual las instancias son deseadas. Este parámetro no puede estar en blanco.

**iFlags**

Opcional. Este parámetro determina el valor de la llamada y si la llamada retorna inmediatamente. El valor por defecto de este parámetro es **wbemFlagReturnImmediately** y puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **WbemFlagForwardOnly**<br>0x20 | Retorna un enumerador solo hacia delante. Los enumeradores solo hacia adelante son generalmente mucho más rápidos y usan menos memoria que los enumeradores convencionales, ya que no permiten llamadas a SWbemObject.Clone_. |
| **wbemFlagBidirectional**<br>0x0 | Hace que WMI de indicadores de retención a los objetos de enumeración mientras el cliente libere el enumerador. |

| | |
|---|---|
| **wbemFlagReturnImmediately**<br>0x10 | Valor por defecto para este parámetro. Esta bandera hace que la llamada retorne inmediatamente. |
| **wbemFlagReturnWhenComplete**<br>0x0 | Hace que la llamada se bloquee mientras se completa la consulta. |
| **wbemQueryFlagShallow**<br>0x1 | Obliga a la enumeración a incluir solo las subclases inmediatas de la superclase especificada. |
| **wbemQueryFlagDeep**<br>0x0 | Valor por defecto para este parámetro. Este valor obliga a la enumeración para que incluya todas las clases en la jerarquía. |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si es exitoso, un objeto SWbemObjectSet es retornado.

**Códigos de error**

El método **InstancesOf** puede retornar los códigos listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **WbemErrAccessDenied**<br>0x80041003 | El usuario actual no tiene permiso para acceder al objeto. |

| | |
|---|---|
| **WbemErrFailed**<br>0x80041001 | Error no específico. |
| **WbemErrInvalidClass**<br>0x80041010 | La clase específica es inválida. |
| **WbemErrInvalidParameter**<br>0x80041008 | Un parámetro inválido fue especificado. |
| **WbemErrOutOfMemory**<br>0x80041006 | No se encontró memoria para completar la operación. |

- **SWbemServices.InstancesOfAsync**

El método **InstancesOfAsync** del objeto **SWbemServices** crea un enumerador que retorna las instancias de una clase específica de acuerdo al criterio de selección dado por el usuario. Esta llamada retorna inmediatamente; el resultado y el estado son retornados al llamante a través del objeto "sink" provisto para tal fin. Este método implementa una consulta simple. Para consultas más complejas se requiere el uso de SWbemServices.ExecQuery.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Localización WMI.

```
objwbemServices.InstancesOfAsync(
ObjWbemSink,
strClass,
[iFlags],
[objwbemNamedValueSet=null],
[objWbemAsyncContext=null]
);
```

**Parámetros**

**ObjWbemSink**

Requerido. Objeto "sink" que obtiene objetos asincrónicamente.

**strClass**

Requerido. Cadena que contiene el nombre de la clase para la cual las instancias son deseadas. Este parámetro no puede estar en blanco.

**iFlags**

Opcional. Determina que tan profundamente la llamada enumera y si esta llamada retorna inmediatamente. Este parámetro puede aceptar los siguientes valores.

| Valor | Significado |
|---|---|
| **WbemQueryFlagShallow**<br>0x1 | Obliga a la enumeración a incluir solo las subclases inmediatas de la superclase especificada. |
| **wbemQueryFlagDeep**<br>0x0 | Valor por defecto para este parámetro. Este valor obliga a la enumeración a incluir todas las clases en la jerarquía. |
| **wbemFlagSendStatus**<br>0x80 | Hace que las llamadas asíncronas envíen las actualizaciones de estado al manejador de eventos OnProgress del objeto "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Previene a la llamada asíncrona de enviar actualizaciones de estado al manejador de eventos **OnProgress** del objeto "sink" |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Este es un objeto SWbemNamedValueSet que retorna al objeto "sink" para identificar la fuente original de la llamada asíncrona. Se usa este parámetro si se hacen múltiples llamadas asíncronas utilizando el mismo objeto "sink". Para usar este parámetro, se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para adicionar un valor que identifique la llamada asíncrona que se esté haciendo. Este objeto SWbemNamedValueSet es devuelto al objeto "sink" y la fuente de la llamada puede ser extraída utilizando el método SWbemNamedValueSet.Item.

**Valores retornados**

Este método no retorna valores. Si es exitoso, el objeto "sink" devuelve un objeto SWbemObjectSet que representa las instancias.

**Códigos de error**

El método **InstancesOf** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para acceder al objeto. |
| **wbemErrFailed** 0x80041001 | Error no específico |
| **wbemErrInvalidClass** 0x80041010 | La clase específica es inválida. |

| | |
|---|---|
| **wbemErrInvalidParameter**<br>0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory**<br>0x80041006 | No se encontró memoria para completar la operación. |

**Comentarios**

El método **InstancesOfAsync** solo trabaja para los objetos clase. No es un error para el enumerador retornado tener elementos vacíos.

- **SWbemServices.ReferencesTo**

El método **ReferencesTo** del objeto **SWbemServices** retorna una colección de todas las asociaciones de clases o instancias que se refieren a una clase fuente o instancia.

Este método desempeña la misma función que la consulta REFERENCES de WQL.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objWbemObjectSet = objwbemServices.ReferencesTo(
strObjectPath,
[strResultClass=""],
[strRole=""],
[bClassesOnly=FALSE],
[bSchemaOnly=FALSE],
[strRequiredQualifier=""],
[iFlags=wbemFlagReturnImmediately],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strObjectPath**

Requerido. Cadena que contiene la ruta del objeto fuente para este método.

**strResultClass**

Opcional. Cadena que contiene el nombre de la clase. Si se especifica, este parámetro indica que los objetos de asociación retornados deben provenir o derivarse de la clase especificada en este parámetro.

**strRole**

Opcional. Cadena que contiene el nombre de la propiedad. Si se especifica, este parámetro indica que las asociaciones de objetos retornadas deben ser limitadas con aquellas en que el objeto fuente juega un rol particular. El papel o rol esta definido por el nombre de la propiedad específica ( la cual debe ser una propiedad referencia) de una asociación.

**bClassesOnly**

Opcional. Valor booleano que indica si una lista de los nombres de las clases que pueden ser retornadas después que las instancias actuales de la clase. Estas son clases de las que los objetos de asociación provienen. El valor por defecto para este parámetro es FALSE.

**bSchemaOnly**

Opcional. Valor booleano que indica si la consulta es aplicada al esquema después que a los datos. El valor por defecto de este parámetro es FALSO. Este solo puede ser fijado a un valor VERDADERO si el parámetro *strObjectPath* especifica la ruta del objeto en la clase. Cuando se fija a un valor TRUE, el conjunto de endpoints retornados representan las clases que son asociadas apropiadamente con la clase fuente en el esquema.

**strRequiredQualifier**

Opcional. Cadena que contiene un nombre calificativo. Si se especifica, este parámetro indica que los objetos de asociados retornados deben incluir el calificador específico.

**iFlags**

Opcional. Entero especificando banderas adicionales para la operación. El valor defecto para este parámetro es **wbemFlagReturnImmediately**, el cual le dice a la llamada para retornar inmediatamente después de esperar a que la consulta se complete. Este parámetro acepta los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemFlagForwardOnly**<br>0x20 | Causa que un enumerador solo hacia adelante (forward-only enumerator) a ser retornado. Los enumeradores Forward-only son generalmente mucho más rápidos y usan menos memoria que un enumerador convencional, pero no permiten llamadas a SwbemObject.Clone_. |
| **wbemFlagBidirectional**<br>0x0 | Hace que WMI de indicadores de retención a los objetos de enumeración mientras el cliente libere el enumerador. |
| **wbemFlagReturnImmediately**<br>0x10 | Hace que la llamada retorne inmediatamente. |
| **wbemFlagReturnWhenComplete**<br>0x0 | Hace que la llamada se bloquee mientras se completa la consulta. |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores retornados**

Si el método es exitoso, un objeto SWbemObjectSet es retornado.

**Códigos de error**

El método **ReferencesTo** puede retornar los códigos de error listados a continuación

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases retornadas por la llamada. |
| **WbemErrFailed** 0x80041001 | Error no específico |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |
| **WbemFlagUtiliceAmendedQualifiers** 0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**Comentarios**

No es un error para la colección retornada tener elementos vacíos.

Para información acerca de REFERENCES OF asociada con las consultas WQL, instancias fuente y objetos de asociación.

- **SWbemServices.ReferencesToAsync**

El método **ReferencesToAsync** del objeto **SWbemServices** retorna asincrónicamente una colección de todas las asociaciones de clases o instancias que se refieren a una clase o instancia fuente en particular.

Este método desempeña la misma función que la consulta REFERENCES de WQL . Esta llamada retorna inmediatamente y los resultados y estado son retornados al llamante a través del objeto "sink" proporcionado.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objwbemServices.ReferencesToAsync(
ObjWbemSink,
strObjectPath,
[strResultClass=""],
[strRole=""],
[bClassesOnly=FALSE],
[bSchemaOnly=FALSE],
[strRequiredQualifier=""],
[iFlags],
[objwbemNamedValueSet=null],
[objWbemAsyncContext=null]
);
```

**Parámetros**

**ObjWbemSink**

Requerido. Objeto "sink" que recibe el objeto asincrónicamente.

**strObjectPath**

Requerido. Cadena que contiene la ruta del objeto para este método.

**strResultClass**

Opcional. Cadena que contiene el nombre de la clase. Si se especifica, este parámetro indica que los objetos de asociación retornados deben provenir o derivarse de la clase especificada en el parámetro.

**strRole**

Opcional. Cadena que contiene el nombre de la propiedad. Si se especifica, este parámetro indica que los objetos de asociación retornados deben ser limitadas con aquellos en que el objeto fuente juega un rol particular. El papel o rol esta definido por el nombre de la propiedad específica (la cual debe ser una propiedad referencia) de una asociación.

**bClassesOnly**

Opcional. Valor booleano que indica si una lista de los nombres de las clases puede ser retornada después que las instancias actuales de la clase. Estas son clases de las que los objetos de asociación provienen. El valor por defecto para este parámetro es FALSE.

**bSchemaOnly**

Opcional. Valor booleano que indica si la consulta es aplicada al esquema después que los datos. El valor por defecto de este parámetro es FALSE. Este solo puede ser fijado a un valor TRUE si el parámetro *strObjectPath* especifica la ruta del objeto en la clase. Cuando se fija a un valor TRUE, el conjunto de endpoints retornados representan las clases que son asociadas apropiadamente con la clase fuente en el esquema.

**strRequiredQualifier**

Opcional. Cadena que contiene un nombre calificativo. Si se especifica, este parámetro indica que el objeto de asociación retornado debe incluir el calificador específico.

**iFlags**

Opcional. Entero que especifica banderas adicionales para la operación. El valor defecto para este parámetro es **wbemFlagBidirectional**. Este parámetro puede aceptar los siguientes valores.

| Valor | Significado |
|---|---|
| **wbemFlagSendStatus**<br>0x80 | Hace que las llamadas asíncronas envíen las actualizaciones de estado al manejador de eventos OnProgress del objeto "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Previene a la llamada asíncrona de enviar actualizaciones de estado al manejador de eventos **OnProgress** del objeto "sink" |
| **wbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Este es un objeto SWbemNamedValueSet que retorna al objeto "sink" para identificar la fuente original de la llamada asíncrona. Se usa este parámetro si se hacen múltiples llamadas asíncronas utilizando el mismo objeto "sink". Para usar este parámetro, se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para adicionar un valor que identifique la llamada asíncrona que se esté haciendo. Este objeto SWbemNamedValueSet es devuelto al objeto "sink" y la fuente de la llamada puede ser

extraída utilizando el método SWbemNamedValueSet.Item. Para más información, ver Haciendo una llamada asincrónica utilizando el Scripting API**.**

**Valores Retornados**

Este método no retorna valores. Si la llamada es exitosa, el objeto "sink" recibe un objeto SWbemObjectSet representando la asociación de clases o de instancias.

**Códigos de error**

El método **ReferencesToAsync** puede retornar los códigos de error listados es la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases retornadas por la llamada. |
| **wbemErrFailed** 0x80041001 | Error no específico |
| **WbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **WbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

**Comentarios**

No es un error para la colección retornada tener elementos vacíos.

Para información acerca de REFERENCES OF asociada con las consultas WQL, instancias fuente y objetos de asociación.

- **SWbemServices.Security_**

La propiedad **Security_** del objeto **SWbemServices** es usada para leer o fijar las configuraciones de seguridad de un objeto **SWbemServices**. Esta propiedad es un objeto **SWbemSecurity**.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objwbemSecurity = objwbemServices.Security_
```

- **SWbemServices.SubclassesOf**

El método **SubclassesOf** del objeto **SWbemServices** retorna un objeto SWbemObjectSetswobjectset. Este objeto es una colección de subclases de una clase especificada. Los items en la colección retornada pueden ser recuperados utilizando métodos estándar de colección. Para más información, ver Colecciones.

Este método solo trabaja para objetos clase.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objWbemObjectSet = objwbemServices.SubclassesOf(
[strSuperclass=""],
[iFlags=wbemFlagReturnImmediately+wbemQueryFlagDeep],
[objwbemNamedValueSet=null]
);
```

**Parámetros**

**strSuperclass**

Opcional. Especifica el nombre de la superclase. Solo las subclases de esta clase retornan en el enumerador. Si está en blanco y si *iFlags* es **wbemQueryFlagShallow**, solo las clases de alto nivel (esto es, clases que no tienen clases padres o superclases) son retornadas. Si este parámetro está en blanco y si *iFlags* es **wbemQueryFlagDeep**, todas las clases dentro del "namespace" son retornadas.

**iFlags**

Opcional. Determina la profundidad de los enumeradores de la llamada. Los valores defecto para este parámetro son **wbemFlagReturnImmediately** y **wbemQueryFlagDeep**. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemQueryFlagShallow**<br>0x1 | Obliga a la enumeración a incluir solo las subclases inmediatas de la superclase especificada. |
| **wbemQueryFlagDeep**<br>0x0 | Valor por defecto para este parámetro. Este valor obliga a una enumeración recursiva dentro de todas las clases derivadas de la superclase específica. La superclase no es retornada en la enumeración. |
| **wbemFlagReturnImmediately**<br>0x10 | Hace que la llamada retorne inmediatamente. |
| **wbemFlagReturnWhenComplete**<br>0 | Hace que la llamada se bloquee mientras se completa la consulta. |
| **WbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**Valores Retornados**

Si el método es exitoso, un objeto SWbemObjectSet es retornado.

**Códigos de error**

El método **SubclassesOf** puede retornar los códigos listados a continuación.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases retornadas por la llamada. |
| **wbemErrFailed** 0x80041001 | Error no específico |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada no existe. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |

**Comentarios**

No es un error para la colección retornada tener elementos vacíos.

- **SWbemServices.SubclassesOfAsync**

El método **SubclassesOfAsync** del objeto **SWbemServices** retorna asincrónicamente una colección de subclases para una clase específica. Esta llamada retorna inmediatamente; el resultado y el estado son retornados al llamante a través del objeto "sink". Los items en la colección retornada al objeto "sink" pueden ser obtenidos utilizando métodos de colección estándar. Para más información, ver Colecciones.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

Este método solo trabaja sobre objetos clase.

```
objwbemServices.SubclassesOfAsync(
ObjWbemSink,
[strSuperclass=""],
[iFlags=wbemQueryFlagDeep],
[objwbemNamedValueSet=null],
[objWbemAsyncContext]
);
```

**Parámetros**

**ObjWbemSink**

Requerido. Objeto "sink" que recibe las subclases asincrónicamente.

**strSuperclass**

Opcional. Especifica el nombre de la superclase. Solo las clases que subclases de esta clase retornan en el enumerador. Si está en blanco y si *iFlags* es **wbemQueryFlagShallow**, solo las clases de alto nivel (esto es, clases que no tienen clases padres o superclases) son

retornadas. Si este parámetro está en blanco y si *iFlags* es **wbemQueryFlagDeep**, todas las clases dentro del "namespace" son retornadas.

**iFlags**

Opcional. Determina la profundidad de los enumeradores de la llamada. El valor defecto para este parámetro es **wbemQueryFlagDeep**. Este parámetro puede aceptar los siguientes valores.

| **Valor** | **Significado** |
|---|---|
| **wbemQueryFlagShallow**<br>0x1 | Obliga a la enumeración a incluir solo las subclases inmediatas de la superclase especificada. |
| **wbemQueryFlagDeep**<br>0x0 | Valor por defecto para este parámetro. Este valor obliga a una enumeración recursiva dentro de todas las clases derivadas de la superclase específica. La superclase no es retornada en la enumeración. |
| **wbemFlagSendStatus**<br>0x80 | Hace que las llamadas asíncronas envíen las actualizaciones de estado al manejador de eventos OnProgress del objeto "sink". |
| **wbemFlagDontSendStatus**<br>0x0 | Previene a la llamada asíncrona de enviar actualizaciones de estado al manejador de eventos **OnProgress** del objeto "sink" |
| **WbemFlagUtiliceAmendedQualifiers**<br>0x20000 | Hace que WMI retorne la clase de datos corregida con la definición de la clase básica. Para más información acerca de los calificadores de corrección ver Localización WMI. |

**objwbemNamedValueSet**

Opcional. Típicamente esta indefinido. De otra manera, este es un objeto SWbemNamedValueSet cuyos elementos representan el contexto de la información que puede ser usada por el proveedor que atiende la petición. Un proveedor que soporta o

requiere dicha información debe documentar el nombre de los valores reconocidos, tipos de datos de los valores, valores permitidos y semántica.

**objWbemAsyncContext**

Opcional. Este es un objeto SWbemNamedValueSet que retorna al objeto "sink" para identificar la fuente original de la llamada asíncrona. Se usa este parámetro si se hacen múltiples llamadas asíncronas utilizando el mismo objeto "sink". Para usar este parámetro, se crea un objeto SWbemNamedValueSet y se usa el método SWbemNamedValueSet.Add para adicionar un valor que identifique la llamada asíncrona que se esté haciendo. Este objeto SWbemNamedValueSet es devuelto al objeto "sink" y la fuente de la llamada puede ser extraída utilizando el método SWbemNamedValueSet.Item. Para más información, ver Haciendo una llamada asincrónica utilizando el Scripting API**.**

**Valores Retornados**

Este método no retorna valores. Si el método es exitoso el objeto "sink" recibe un objeto SWbemObjectSet que representa a las subclases.

**Códigos de error**

El método **SubclassesOfAsync** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrAccessDenied** 0x80041003 | El usuario actual no tiene permiso para ver una o más de las clases retornadas por la llamada. |
| **WbemErrFailed** 0x80041001 | Error no específico |
| **wbemErrInvalidClass** 0x80041010 | La clase especificada no existe. |
| **wbemErrInvalidParameter** 0x80041008 | Un parámetro inválido fue especificado. |

| | |
|---|---|
| **wbemErrOutOfMemory**<br>0x80041006 | No se encontró memoria para completar la operación. |

**Comentarios**

No es un error para la colección retornada tener elementos cero si las subclases no son encontradas.

### C.4.11 SWbemPrivilege

El objeto **SWbemPrivilege** representa un privilegio simple.

**Métodos**

Este objeto no tiene métodos.

**Propiedades**

La siguiente tabla lista las propiedades para el objeto **SWbemPrivilege**.

| **Propiedades** | **Descripción** |
|---|---|
| Displayname | Despliega el nombre de este privilegio. |
| Identifier | Entero que representa el privilegio que esta siendo fijado o recibido. |
| IsEnabled | Booleano que indica si el privilegio está o no habilitado. |
| Name | Nombre de este privilegio. |

- **SWbemPrivilege.DisplayName**

La propiedad **DisplayName** del objeto **SWbemPrivilege** es una cadena de caracteres que despliega una descripción de un objeto **SWbemPrivilege**. Por ejemplo, el privilegio que controla si un usuario pude ejecutar el método de apagado del equipo de un objeto que tiene

la cadena "Apagar el sistema" en la propiedad **SWbemPrivilege.DisplayName**. Esta propiedad es de solo lectura.

La siguiente sintaxis está en un lenguaje neutral, para más explicación de esta sintaxis, ver Convenciones del Documento.

```
sDisplayName = objWbemPrivilege.DisplayName
```

- **SWbemPrivilege.Identifier**

La propiedad **Identifier** de un objeto **SWbemPrivilege** es un entero **WbemPrivilegeEnum** que representa el privilegio que esta siendo fijado o recibido. Esta propiedad es de solo lectura.

La siguiente sintaxis está en un lenguaje neutral, para más explicación de esta sintaxis, ver Convenciones del Documento.

```
iPrivilege = objWbemPrivilege.Identifier
objWbemPrivilege.Identifier= Identifier
```

- **SWbemPrivilege.IsEnabled**

La propiedad **IsEnabled** de un objeto **SWbemPrivilege** es un valor booleano que se usa para habilitar o deshabilitar este privilegio.
La siguiente sintaxis está en un lenguaje neutral, para más explicación de esta sintaxis, ver Convenciones del Documento.

```
bIsEnabled = objwbemPrivilege.IsEnabled
objwbemPrivilege.IsEnabled = bIsEnabled
```

- **SWbemPrivilege.Name**

La propiedad **Name** del objeto **SWbemPrivilege** es una cadena que describe únicamente un privilegio. Por ejemplo, el privilegio que controla si un usuario puede ejecutar el método de apagado de un objeto que tiene la cadena "SeShutdownPrivilege" en la propiedad **SWbemPrivilege.Name**. Esta propiedad es de solo lectura.

La siguiente sintaxis está en un lenguaje neutral, para más explicación de esta sintaxis, ver Convenciones del Documento.

```
sName = objWbemPrivilege.Name
```

### *C.4.12 SWbemPrivilegeSet*

Un objeto **SWbemPrivilegeSet** es una colección de objetos **SWbemPrivilege** en un objeto SWbemSecurity que requiere privilegios específicos para los objetos WMI. Los items son adicionados a la colección utilizando los métodos **Add** y **AddAsString**, recibidos utilizando el método **Item**, y borrados de la colección utilizando el método **Remove**. Para más información sobre el uso de colecciones, ver Colecciones.

Un objeto **SWbemPrivilegeSet** es un conjunto de privilegios que manejan peticiones para un objeto en particular. Cuando una llamada al API es hecha con este objeto, el privilegio que maneja las peticiones es intentado. Un objeto **SWbemPrivilegeSet** no define los privilegios disponibles para el proceso o el usuario actual.

#### Métodos

La siguiente tabla muestra la lista de métodos para el objeto **SWbemPrivilegeSet**.

| Métodos | Descripción |
|---|---|
| Add | Adiciona un objeto **SWbemPrivilege** a la colección **SWbemPrivilegeSet utilizando una constante WbemPrivilegeEnum**. |
| AddAsString | Adiciona un objeto **SWbemPrivilege** a la colección **SWbemPrivilegeSet** utilizando una cadena de caracteres con privilegios Windows NT/Windows 2000. |
| Item | Devuelve un objeto **SWbemPrivilege** desde una colección. Este es el método por defecto para este objeto. |
| DeleteAll | Borra todos los privilegios de una colección. |
| Remove | Borra un objeto **SWbemPrivilege** de una colección. |

**Propiedades**

La siguiente tabla lista las propiedades del objeto **SWbemPrivilegeSet**.

| Propiedades | Descripción |
|---|---|
| Count | El número de items en la colección. |

- **SWbemPrivilegeSet.Add**

El método **Add** del objeto **SWbemPrivilegeSet** adiciona un objeto SWbemPrivilege a la colección **SWbemPrivilegeSet**. Si un privilegio con el mismo nombre ya existe en la colección, este es reemplazado.

La siguiente sintaxis está en un lenguaje neutral, para más explicación de esta sintaxis, ver Convenciones del Documento.

```
objPrivilege = objwbemPrivilegeSet.Add(
iPrivilege,
```

```
[bIsEnabled = True]
);
```

**Parámetros**

**iPrivilege**

Requerido. Una de las constantes WMI del grupo **WbemPrivilegeEnum**. Estas constantes son esencialmente enteros que representan privilegios específicos. Por ejemplo, para adicionar privilegios que permitan apagar sistemas Windows NT/Windows 2000, se usa la constante **wbemPrivilegeShutdown** o el equivalente numérico de 0x17.

**bIsEnabled**

Opcional. Valor booleano que habilita o deshabilita este privilegio. El valor por defecto es TRUE.

**Valores retornados**

Si es exitoso, el método retorna un objeto **SWbemPrivilege** que representa el Nuevo privilegio. De otra manera, un objeto nulo es retornado.

**Códigos de error**

El método **Add** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no específico. |

- **SWbemPrivilegeSet.AddAsString**

Se puede usar el método **AddAsString** del objeto **SWbemPrivilegeSet** para adicionar un privilegio a una colección **SWbemPrivilegeSet** utilizando una cadena de caracteres de privilegios de Windows NT/Windows 2000. Teste método es muy útil cuando se consigue una cedena de privilegios desde una clase WMI que anuncia los privilegios requeridos como los valores calificativos de tipo **string[]**.

La siguiente sintaxis está en un lenguaje neutral, para más explicación de esta sintaxis, ver Convenciones del Documento.

```
objPrivilege = objwbemPrivilegeSet.AddAsString(
strPrivilege,
[bIsEnabled = True]
);
```

**Parámetros**

**strPrivilege**

Requerido. Una de las cadenas de privilegios de Windows NT/Windows 2000. Cada cadena de privilegios Windows NT/Windows 2000 representan un privilegio específico. Por ejemplo, para adicionar el privilegio que se pueda usar para apagar un sistema Windows NT/Windows 2000, utilice la cadena "SeShutdownPrivilege".

**bIsEnabled**

Opcional. Valor booleano que habilita o deshabilita estos privilegios. El valor por defecto es TRUE.

**Valores retornados**

Si es exitoso, este método retorna un objeto **SWbemPrivilege** que representa el Nuevo privilegio. De otra manera, un objeto nulo es devuelto.

**Códigos de error**

El método **AddAsString** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **WbemErrFailed** 0x80041001 | Error no específico. |

- **SWbemPrivilegeSet.DeleteAll**

El método **DeleteAll** del objeto **SWbemPrivilegeSet** borra todos los privilegios de una colección, vaciándola así.

```
objwbemPrivilegeSet.DeleteAll
```

**Valores retornados**

Este método no retorna valores.

**Códigos de error**

El método **DeleteAll** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no específico. |

- **SWbemPrivilegeSet.Count**

La propiedad **Count** del objeto **SWbemPrivilege** contiene el número de items en una colección **SWbemPrivilegeSet**. Esta propiedad es de solo lectura.

La siguiente sintaxis está en un lenguaje neutral, para más explicación de esta sintaxis, ver Convenciones del Documento.

```
iCount = objwbemPrivilegeSet.Count
```

- **SWbemPrivilegeSet.Item**

El método **Item** del objeto **SWbemPrivilegeSet** retorna un objeto SWbemPrivilege de una colección. El método **Item** es el método por defecto del objeto **WbemPrivilegeSet.**

La siguiente sintaxis está en un lenguaje neutral, para más explicación de esta sintaxis, ver Convenciones del Documento.

```
objPrivilege= objwbemPrivilegeSet.Item(
iPrivilege
);
```

**Parámetros**

**iPrivilege**

Requerido. Una de las constantes WMI del grupo **WbemPrivilegeEnum**. Estas constantes son esencialmente enteros que representan privilegios específicos. Por ejemplo, para adicionar privilegios que permitan apagar sistemas Windows NT/Windows 2000, se usa la constante **wbemPrivilegeShutdown** o el equivalente numérico de 0x17.

**Valores retornados**

Si es exitoso, el objeto **SWbemPrivilege** retorna.

## Códigos de error

El método **Item** puede retornar los códigos de error listados en la siguiente tabla.

| Error | Significado |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no específico. |
| **wbemErrNotFound** 0x80041002 | El privilegio específico no existe. |

- **SWbemPrivilegeSet.Remove**

El método **Remove** del objeto **SWbemPrivilegeSet** borra un privilegio de una colección.

La siguiente sintaxis está en un lenguaje neutral, para más explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemPrivilegeSet.Remove(
iPrivilege
);
```

**Parámetros**

**iPrivilege**

Requerido. Una de las constantes WMI del grupo **WbemPrivilegeEnum**. Estas constantes son esencialmente enteros que representan privilegios específicos. Por ejemplo, para

adicionar privilegios que permitan apagar sistemas Windows NT/Windows 2000, se usa la constante **wbemPrivilegeShutdown** o el equivalente numérico de 0x17.

**Valores retornados**

Este método no retorna valores.

**Códigos de error**

El método **Remove** puede retornar los códigos de error listados en la siguiente tabla.

| Error | Significado |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no específico. |
| **wbemErrNotFound** 0x80041002 | El privilegio específico no existe. |

### *C.4.13 SWbemPropertySet*

Un objeto **SWbemPropertySet** es una colección de objetos **SWbemProperty**. Se puede agregar artículos a la colección utilizando el método **Add**, recuperar artículos de la colección utilizando el método Item, y quitar artículos de la colección utilizando el método **Remove**. Para información sobre usar colecciones, ver Colecciones.

Se usan los objetos **SWbemProperty** que constituyen una colección de **SWbemPropertySet** para describir las propiedades de una sola clase o instancia de WMI.

**Métodos**

La tabla siguiente lista los métodos para el objeto **SWbemPropertySet**.

| Mét¡odo | Descripción |
|---|---|
| **Add** | Agrega un objeto **SWbemProperty** a la colección **SWbemPropertySet**. |
| **Item** | Recupera un **SWbemProperty** de la colección. Éste es el método predefinido de este objeto. |
| **Remove** | Quita un objeto **SWbemProperty** de la colección. |

**Propiedad**

| Propiedad | Descripción |
|---|---|
| **Count** | El número de artículos en la colección. |

### *C.4.14 SWbemQualifierSet*

Un objeto **SWbemQualifierSet** es una colección de objetos **SWbemQualifier**. Se agregan artículos a la colección que usa el método **Add**, se recuperan de la colección utilizando el método de **Item**, y se quitan de la colección utilizando el método remove.

Para información sobre usar colecciones, ver Colecciones.

Los objetos **SWbemQualifier** que constituyen una colección de **SWbemQualifierSet** describen los calificadores atados a una clase de WMI, instancia, método o parámetro del método.

**Métodos**

La tabla siguiente lista los métodos para el objeto **SWbemQualifierSet**.

| Método | Descripción |
|---|---|
| **Add** | Agrega un objeto **SWbemQualifier** a la colección **SWbemQualifierSet**. |
| **Item** | Recupera un objeto **SWbemQualifier** de la colección. Éste es el método predefinido de este objeto. |
| **Remove** | Quita un objeto **SWbemQualifier** de la colección. |

**Propiedades.**

La tabla siguiente lista la propiedad para el objeto **SWbemQualifierSet**.

| Propiedad | Descripción |
|---|---|
| **Count** | El número de artículos en la colección. |

### *C.4.15 SWbemSecurity*

El objeto SWbemSecurity se usa para leer o poner las escenas de seguridad como Windows NT/Windows 2000 Privilegios, Personificación de DCOM, y niveles de la Autenticación asignados a un objeto. Los SWbemLocator, SWbemServices, SWbemObject, SWbemObjectSet, SWbemObjectPath, SWbemLastError, y objetos SWbemEventSource tienen una propiedad llamada Security_ éste es un objeto SWbemSecurity. Se puede necesitar poner las propiedades Security_ al realizar tareas como recuperar una instancia o ver el log de seguridad de Windows NT/Windows 2000 que usa WMI.

**Métodos**

Este objeto no tiene métodos.

**Propiedades**

La siguiente tabla lista las propiedades para el objeto **SWbemSecurity**.

| **Propiedades** | **Descripción** |
|---|---|
| AuthenticationLevel | Valor numérico que define el nivel de autenticación DCOM asignado a este objeto. Esta configuración determina como proteger la información enviada desde WMI. |
| ImpersonationLevel | Valor numérico que define el nivel de impersonalización DCOM asignado a este objeto. Esta configuración determina si el procesamiento propio de WMI puede detectar o usar las credenciales de seguridad cuando se hacen las llamadas a otros procesos. |
| Privileges | Un objeto SWbemPrivilegeSet que define los privilegios de Windows NT/Windows 2000 para este objeto. |

Las propiedades de un objeto SWbemLocator.Security_ no tienen valores por defecto. Si se intenta obtener el valor de **SWbemSecurity.AuthenticationLevel** o de **SWbemSecurity.ImpersonationLevel** o de un objeto SWbemLocator antes de fijar dichos objetos, un error wbemErrFailed resultará.

- **SWbemSecurity.AuthenticationLevel**

La propiedad **AuthenticationLevel** es un entero que define el nivel de autenticación DCOM asignado a un objeto. Esta configuración determina como se protege la información enviada desde WMI. Generalmente hablando, no es necesario fijar el nivel de autenticación cuando se hacen llamadas al WMI API. Si no se fija o se pone un valor a esta propiedad, el nivel de autenticación DCOM por defecto del sistema es utilizado.

La siguiente sintaxis está en un lenguaje neutral. Para más explicación de esta sintaxis, ver Convenciones del Documento.

```
iAuthenticationlevel = objWbemSecurity.AuthenticationLevel
objWbemSecurity.AuthenticationLevel = iAuthenticationlevel
```

**Comentarios**

Se puede fijar el nivel de autenticación de los objetos SWbemServices, SWbemObject, SWbemObjectSet, SWbemObjectPath, y SwbemLocator fijando la propiedad **AuthenticationLevel** al valor deseado.

El siguiente ejemplo muestra como fijar el nivel de autenticación para un objeto **SwbemObject**:

objinstance.Security_.AuthenticationLevel = wbemAuthenticationLevelPkt

También se puede especificar el nivel como parte de un moniker. El siguiente ejemplo fija el nivel de autenticación y el nivel de impersonalización y devuelve una instancia de Win32_LogicalDisk 'C:'

```
Set objinst =
GetObject("WinMgmts:{impersonationLevel=impersonate,"& _
                        "authenticationLevel=pktPrivacy}" & _
                        "!root/cimv2:Win32_LogicalDisk='c:'")
```

- **SWbemSecurity.ImpersonationLevel**

La propiedad **ImpersonationLevel** es un entero que define el nivel de impersonalización del DCOM asociado a un objeto. Esta configuración determina si el procesamiento propio de WMI puede detectar o usar las credenciales cuando se hacen llamadas a otros procesos. Si específicamente no se fija el nivel de impersonalización, en o en un moniker o fijando la propiedad **SWbemSecurity.ImpersonationLevel** de un objeto seguro, WMI fija el nivel de imprsonalización por defecto a un valor especificado en el registo. Este valor registrado es fijado en W**bemImpersonationLevelImpersonate**. En las versiones previas de WMI se fijaba el valor por defecto para el nivel de impersonalización en **WbemImpersonationLevelIdentify**. Si esta configuración no era suficiente, el proveedor no podría atender a la petición y la llamada al WMI API podía fallar con un error wbemErrAccessDenied (hexadecimal 0x80041003).

La siguiente sintaxis está en un lenguaje neutral. Para más explicación de esta sintaxis, ver Convenciones del Documento.

```
iImpersonationLevel  = objWbemSecurity.ImpersonationLevel
objWbemSecurity.ImpersonationLevel = iImpersonationLevel
```

**Comentarios**

SWbemObjectSet, SWbemObjectPath, y SwbemLocator fijando la propiedad Se puede fijar el nivel de impersonalización de los objetos SWbemServices, SWbemObject, **ImpersonationLevel** al valor deseado. El siguiente ejemplo muestra como fijar el nivel de impersonalización para un objeto **SwbemObject**:

```
objinstance.Security_.ImpersonationLevel =
wbemImpersonationLevelImpersonate
```

Se puede especificar los niveles de impersonalización como parte de un moniker. El siguiente ejemplo fija el nivel de autenticación y el de impersonalización, y devuelve una instancia de Win32_Service 'Alerter'

```
Set objinst =
GetObject("WinMgmts:{impersonationLevel=impersonate,"& _
                          "authenticationLevel=pktPrivacy}"& _

"!root/cimv2:Win32_service='ALERTER'")
```

- **SWbemSecurity.Privileges**

La propiedad **Privileges** es un objeto **SWbemPrivilegeSet**. Esta propiedad es usada para habilitar o deshabilitar privilegios específicos de Windows NT/Windows 2000. Se necesita fijar una de estos privilegios para desempeñar una tarea específica utilizando el WMI API.

La siguiente sintaxis está en un lenguaje neutral. Para más explicación de esta sintaxis, ver Convenciones del Documento.

```
objPrivilegeSet = objwbemObject.Privileges
```

**Comentarios**

Se puede cambiar el privilegio definido por los objetos SWbemServices, SWbemObject, SWbemObjectSet, SWbemObjectPath, y SwbemLocator adicionando objetos **SWbemPrivilege** con la propiedad **Privileges**.

Hay diferencias fundamentales en la manera en que las diferentes versiones de Windows manejan los cambios en los privilegios. Si se esta desarrollando una aplicación que solo usa plataformas Windows 2000, se pueden fijar o revocar los privilegios en cualquier momento.

Si las aplicaciones necesitan ser portables para el uso con plataformas Windows NT versión 4.0, se deben fijar los privilegios antes de que se haga la primera llamada a SWbemLocator.ConnectServer o como parte del moniker.

El siguiente ejemplo trabaja sobre plataformas Windows NT/Windows 2000:

```
Set Service =
GetObject("winmgmts:{impersonationLevel=impersonate,
(SeDebugPrivilege)}")
```

El siguiente ejemplo no trabaja sobre plataformas Windows NT 4.0. Este intenta desempeñar la misma tarea, pero este fija el privilegio después de conectarse con WMI:

```
Set Service =
GetObject("winmgmts:{impersonationLevel=impersonate}")
Service.Security_.Privileges.AddAsString "SeDebugPrivilege",
True
```

### C.4.16 SWbemSink

El objeto **SWbemSink** es implementado por la aplicación cliente para recibir los resultados de operaciones asincrónicas y notificaciones de eventos. Para hacer una llamada asincrónica , se deberá crear una instancia de un objeto **SWbemSink** y poner el parámetro (y pasarlo como ) *ObjWbemSink*. Los eventos en la implementación del objeto **SWbemSink** son enganchados (disparados) cuando se retorna el estado o los resultados o cuando la llamada este completa.

#### Métodos

La siguiente tabla muestra los métodos para el objeto **SWbemSink**.

| Método | Descripción |
|---|---|
| Cancel | Cancela todas las operaciones asincrónicas asociadas con el objeto "sink". |

**Propiedades**

Este objeto no tiene propiedades.

**Eventos**

| Evento | Descripción |
|---|---|
| OnCompleted | Disparado cuando la operación asincrónica esta completa. |
| OnObjectPut | Disparado después de una operación **put** asíncrona. |
| OnObjectReady | Disparado cuando un objeto provisto por una llamada asíncrona esta disponible. |
| OnProgress | Disparado para proveer el estado de una operación |

- **SWbemSink.Cancel**

El método **Cancel** del objeto **SWbemSink** cancela todas las operaciones asincrónicas vigentes asociadas con el objeto "sink".

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemsink.cancel()
```

**Parámetros**

Este método no tiene parámetros.

**Valores Retornados**

Este método no retorna valores.

**Códigos de error**

El método **Cancel** puede retornar los códigos de error que se listan en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **WbemErrFailed** 0x80041001 | Error no específico. |
| **wbemErrOutOfMemory** 0x80041006 | No encontró memoria para completar la operación. |
| **wbemErrTransportFailure** 0x80041015 | Ocurrió un error en la red, previniendo la operación normal. |
| **wbemErrAccessDenied** 0x80041003 | El nombre de usuario y la contraseña actual no son válidos o autorizados para hacer la conexión. |

**Comentarios**

No es posible cancelar una llamada asíncrona en particular. Si hay múltiples llamadas asincrónicas pendientes que usan el objeto "sink", todas las llamadas que usan dicho objeto son canceladas por este método. Las llamadas asincrónicas asociadas a otros objetos "sink" no son afectadas.

Se puede asignar al objeto "sink" el valor **Nothing** para cancelar una operación asíncrona no deseada. Se debería llamar al método **Cancel** para hacer que WMI desconecte la operación y libere los recursos asociados. Estos es particularmente importante para operaciones

asíncronas extensas, como consultas, o operaciones que no se completan, como en **ExecNotificationQueryAsync**.

El siguiente ejemplo muestra como cancelar una llamada asíncrona:

```
objwbemsink.Cancel()
set objwbemsink= Nothing
```

- **SWbemSink.OnCompleted**

El evento **OnCompleted** del objeto **SWbemSink** es disparado cuando una llamada asincrónica esta completa. Este evento informa el resultado de la operación asincrónica a la aplicación cliente o da una información del error sí la llamada asíncrona falla.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemsink.OnCompleted(
iHResult,
objWbemErrorObject,
objWbemAsyncContext
);
```

**Parámetros**

**iHResult**

El HRESULT de un método completo asíncrono. El HRESULT es el mismo que el valor equivalente retornado por la llamada al método. Se chequea este valor para determinar si la llamada asíncrona fue exitosa. Si la llamada asíncrona fue exitosa, este parámetro contiene el valor WBEM_S_NO_ERROR (0). Si la llamada falla, este parámetro contiene el código de error.

**objWbemErrorObject**

Contiene el objeto **SWbemLastError** si el método asíncrono falla.

**objWbemAsyncContext**

Este es un objeto SWbemNamedValueSet que fue enviado a la llamada asincrónica original. Se usa este parámetro para identificar el origen de la llamada asíncrona que disparó este evento si múltiples llamadas asincrónicas son hechas utilizando el objeto "sink".

**Códigos de error**

El evento **OnCompleted** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **WbemErrFailed**<br>0x80041001 | Error no especificado. |
| **WbemErrOutOfMemory**<br>0x80041006 | No encontró memoria para completar la operación. |
| **wbemErrTransportFailure**<br>0x80041015 | Error de red que previene la operación normal. |

- **SWbemSink.OnObjectPut**

El evento **OnObjectPut** del objeto **SWbemSink** es disparado cuando la operación asíncrona **put** esta completa. Este evento retorna la ruta del objeto, de la instancia o clase que han sido guardados.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemsink.OnObjectPut(
objWbemObjectPath,
objWbemAsyncContext
);
```

**Parámetros**

**objWbemObjectPath**

Un objeto SWbemObjectPath , contiene la dirección del objeto de la clase o instancia que ha sido depositada exitosamente en WMI.

**objWbemAsyncContext**

El objeto SWbemNamedValueSet que fue pasado a la llamada asíncrona original. Se usa este parámetro para identificar el origen de la llamad asíncrona que dispara este evento si múltiples llamadas asíncronas fueran hechas utilizando el objeto "sink".

**Códigos de error**

El evento **OnObjectPut** puede retornar los códigos de error listados a continuación.

| **Error** | **Significado** |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no especificado. |
| **wbemErrOutOfMemory** 0x80041006 | No se encuentra memoria para completa la operación. |
| **wbemErrTransportFailure** 0x80041015 | Ha ocurrido un error en la red previniendo la operación normal. |

- **SWbemSink.OnObjectReady**

El evento **OnObjectReady** del objeto SWbemSink es disparado cuando una operación asíncrona retorna un objeto. Se usa este evento para procesar objetos a través de llamadas asíncronas como SWbemObject.InstancesAsync_ o SWbemServices.ExecQueryAsync. Las llamadas asíncronas retornan una enumeración llamando eventos repetidamente. Esto retorna un solo objeto SWbemObject cada vez que la enumeración este completa.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento**.**

```
objwbemsink.OnObjectReady (
objWbemObject,
objWbemAsyncContext
);
```

**Parámetros**

**objWbemObject**

El objeto **SWbemObject** . Este es similar al que podría ser retornado por el equivalente síncrono de la llamada asíncrona que engancha este evento. Por ejemplo, una llamada al método **SWbemServices.GetAsync** retorna un objeto **SWbemObject** en el parámetro *objWbemObject* del evento **OnObjectReady** del objeto **SWbemSink**, el cual pasa como un parámetro *objWbemSink* de la llamada original. El mismo objeto **SWbemObject** puede obtenerse con un equivalente síncrono de la llamada a **SWbemServices.Get**.

**objWbemAsyncContext**

El objeto SWbemNamedValueSet que se paso a la llamada asíncrona original. Se usa este parámetro para identificar el origen de la llamada asíncrona que disparó este evento si múltiples llamadas asíncronas fueran hechas utilizando el objeto "sink".

**Códigos de error**

El evento **OnObjectReady** puede retornar los códigos de error retornados en la siguiente tabla.

| Error | Significado |
|---|---|
| **wbemErrFailed** 0x80041001 | Error no específico |
| **wbemErrOutOfMemory** 0x80041006 | No se encontró memoria para completar la operación. |
| **wbemErrTransportFailure** 0x80041015 | Ocurrió un error en la red, previniendo la operación normal. |

- **SWbemSink.OnProgress**

El evento **OnProgress** de **SWbemSink** es disparado cuando una llamada asíncrona devuelve el estado de la llamada que esta en progreso. Si los eventos, instancias, o clases son producidos a través de un proveedor que soporta la actualización de los estados, se puede poner un código al evento para dar a los usuarios una realimentación a cerca del estado de la operación asíncrona. Se debe fijar los parámetros *iFlags* de la llamada asíncrona a wbemFlagSendStatus (0x80) si se quiere recibir actualizaciones de estados, de otra manera el evento no se engancha.

La siguiente sintaxis esta en un lenguaje neutral. Para una explicación de esta sintaxis, ver Convenciones del Documento.

```
objwbemsink.OnProgress (
iUpperBound,
iCurrent,
```

```
strMessage,
objWbemAsyncContext
);
```

**Parámetros**

**iUpperBound**

Entero que describe el numero total de tareas a completar.

**iCurrent**

Item actual que esta siendo procesado.

**strMessage**

Mensaje que describe el estado actual de la tarea.

**objWbemAsyncContext**

Un objeto SWbemNamedValueSet que paso a la llamada asíncrona original. Se usa este parámetro para identificar el origen de la llamada asíncrona que disparó este evento si múltiples llamadas asíncronas fueron hechas utilizando este objeto "sink".

**Códigos de error**

El evento **OnProgress** puede retornar los códigos de error listados en la siguiente tabla.

| **Error** | **Significado** |
|---|---|
| **wbemErrFailed**<br>0x80041001 | Error no específico |
| **wbemErrOutOfMemory**<br>0x80041006 | No se encontró memoria para completar la operación. |
| **wbemErrTransportFailure**<br>0x80041015 | Ocurrió un error en la red, previniendo la operación normal. |

**Comentarios**

El evento **OnProgress** es disparado cuando la llamada asíncrona devuelve el estado de la llamada en progreso. Si los eventos, instancias o clases son generados por un proveedor que soporta la actualización de estados, se puede poner un código en este evento que de a los usuarios la realimentación en el estado de la operación asíncrona.