Tivoli Application Dependency Discovery Manager
Versión 7.3

Guía del desarrollador del SDK

Tivoli Application Dependency Discovery Manager
Versión 7.3

Guía del desarrollador del SDK

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información que se incluye en el apartado "Avisos" en la página 187.

Nota de edición

Esta edición es aplicable a la versión 7, release 3 de IBM Tivoli Application Dependency Discovery Manager (número de producto 5724-N55) y todos los releases y modificaciones siguientes hasta que se indique de otro modo en las ediciones nuevas.

Contenido

Tablas

Acerca de esta información

El objetivo de esta versión del documento PDF es proporcionar los temas relacionados del Information Center en formato imprimible.

Convenciones utilizadas en este Information Center

En la documentación de IBM® Tivoli Application Dependency Discovery Manager (TADDM) se utilizan ciertas convenciones. Estas se emplean para hacer referencia a variables y vías de acceso dependientes del sistema operativo, el directorio `COLLATION_HOME` y la ubicación del archivo `collation.properties`, al que se hace referencia en toda la documentación de TADDM, incluso en los mensajes.

Variables y vías de acceso dependientes del sistema operativo

En este Information Center, se utilizan los convenios de UNIX para especificar variables de entorno y para la notación de directorio.

Cuando utilice la línea de mandatos de Windows, sustituya $*variable* por %*variable*% para variables de entorno, y sustituya cada barra inclinada (/) por una barra inclinada invertida (\) en las vías de acceso de directorio.

Si utiliza el shell bash en un sistema Windows, puede utilizar los convenios de UNIX.

Directorio `COLLATION_HOME`

El directorio raíz de TADDM se denomina también directorio `COLLATION_HOME`.

En sistemas operativos tales como AIX o Linux, la ubicación predeterminada para instalar TADDM es el directorio `/opt/IBM/taddm`. Por tanto, en este caso, el directorio `$COLLATION_HOME` es `/opt/IBM/taddm/dist`.

En sistemas operativos de Windows, la ubicación predeterminada para instalar TADDM es el directorio `c:\IBM\taddm`. Por lo tanto, en este caso, el directorio `%COLLATION_HOME%` es `c:\IBM\taddm\dist`.

Ubicación del archivo `collation.properties`

El archivo `collation.properties` contiene propiedades de servidor de TADDM e incluye comentarios de cada una de las propiedades. Está ubicado en el directorio `$COLLATION_HOME/etc`.

Términos y definiciones

Consulte la siguiente lista de términos y definiciones para obtener información sobre conceptos importantes de IBM Tivoli Application Dependency Discovery Manager (TADDM).

aplicación empresarial
Una colección de componentes que proporciona una aplicación empresarial que puede utilizar de forma interna o externa o con otras aplicaciones industriales.

Base de datos TADDM
En TADDM, la base de datos donde se almacenan los datos de configuración, las dependencias y el historial de cambios.

Cada servidor de TADDM, excepto los servidores de descubrimiento y los servidores de almacenamiento secundario, tiene su propia base de datos. Los servidores de descubrimiento no tienen base de datos. Los servidores de almacenamiento comparten la base de datos del servidor de almacenamiento primario.

CI Consulte *elemento de configuración*.

colección de accesos
Una colección que se utiliza para controlar el acceso a elementos de configuración y a permisos para modificar elementos de configuración. Únicamente puede crear una colección de accesos cuando se ha habilitado la seguridad a nivel de datos.

Consola de Gestión de descubrimiento
Interfaz de usuario del cliente de TADDM para gestionar descubrimientos. Esta consola también se conoce con el nombre de consola del producto. Es aplicable a un despliegue de servidor de dominio y a los Discovery Server en un despliegue de servidor continuo. La función de la consola es la misma para estos dos despliegues.

consola del producto
Consulte *Consola de gestión de descubrimiento*.

contexto de inicio
El concepto de cambiar sin problemas de una interfaz de usuario de productos Tivoli a otra interfaz de usuarios de productos Tivoli (en una consola diferente o en la misma interfaz de consola o de portal) con un inicio de sesión único y con la interfaz de usuario de destino en el punto adecuado para que los usuarios puedan continuar con sus tareas.

Data Management Portal
Interfaz de usuario basada en web de TADDM para visualizar y manipular los datos en la base de datos de TADDM. Se puede aplicar a un despliegue de servidor de dominio a un despliegue de servidor de sincronización y a un servidor de almacenamiento en un despliegue de servidor de modalidad continua. La interfaz de usuario es muy similar en todos los despliegues, aunque en un despliegue de servidor de sincronización tiene menos funciones adicionales para añadir y sincronizar dominios.

Descubrimiento asíncrono
En TADDM, la ejecución de un script de descubrimiento en un sistema de destino para descubrir sistemas a los que no se puede acceder directamente mediante el servidor de TADDM. Como el descubrimiento se realiza manualmente, y de forma independiente al descubrimiento con credenciales típico, éste se denomina “asíncrono”.

descubrimiento basado en un script
En TADDM, la utilización, en un descubrimiento credencial, de los mismos scripts de sensor proporciona apoyo al descubrimiento asíncrono.

Descubrimiento credencial
Exploración del sensor de TADDM que descubre información detallada sobre los siguientes elementos:

- Cada sistema operativo en el entorno de ejecución. Esta exploración también se conoce como descubrimiento de nivel 2 y necesita credenciales de sistema operativo.

- Infraestructura de aplicación, componentes de software desplegado, servidores físicos, dispositivos de red, sistemas virtuales y datos de host que se utilizan en el entorno de ejecución. Esta exploración también se conoce como descubrimiento de nivel 3 y necesita tanto las credenciales del sistema operativo como las credenciales de aplicación.

Descubrimiento de credenciales menores
Exploración del sensor TADDM que descubre información básica acerca de los sistemas informáticos activos en el entorno de ejecución. Esta exploración también se conoce como descubrimiento de nivel 1 y no necesita credenciales.

Descubrimiento de Nivel 1
Exploración del sensor TADDM que descubre información básica acerca de los sistemas informáticos activos en el entorno de ejecución. Esta exploración también se conoce como descubrimiento sin credenciales y, como su propio nombre indica, no necesita credenciales. Utiliza el sensor Stack Scan y el sensor IBM® Tivoli® Monitoring Scope. El descubrimiento de nivel 1 es muy superficial. Recoge solo el nombre de host, el nombre del sistema operativo, la dirección IP, el nombre de dominio completo y la dirección Media Access Control (MAC) de cada interfaz descubierta. Además, el descubrimiento de dirección MAC se limita a Linux en los sistemas System z® y Windows. El descubrimiento de nivel 1 no descubre subredes. Para cada interfaz IP descubierta que no pertenezca a ninguna subred existente hallada durante el descubrimiento de nivel 2 o de nivel 3, se crean subredes nuevas basadas en el valor de la propiedad `com.collation.IpNetworkAssignmentAgent.defaultNetmask` del archivo `collation.properties`.

Descubrimiento de Nivel 2
Exploración del sensor de TADDM que descubre información detallada sobre cada sistema operativo del entorno de ejecución. Esta exploración también se conoce como descubrimiento con credenciales y necesita credenciales de sistema operativo. El descubrimiento de nivel 2 recoge los nombres de aplicaciones y los nombres de los sistemas operativos y números de puerto asociados con cada aplicación que se esté ejecutando. Si una aplicación ha establecido una conexión TCP/IP con otra aplicación, esta información se recoge como dependencia.

Descubrimiento de Nivel 3
Exploración del sensor de TADDM que descubre información detallada sobre la infraestructura de aplicación, los componentes de software desplegado, servidores físicos, dispositivos de red, sistemas virtuales y datos de host que se utilizan en el entorno de ejecución. Esta exploración también se conoce como descubrimiento con credenciales y necesita tanto las credenciales del sistema operativo como las credenciales de aplicación.

Descubrimiento de utilización
Exploración del sensor de TADDM que descubre información de utilización sobre los siguientes elementos. Un descubrimiento de utilización requiere credenciales del sistema operativo.

despliegue del servidor de sincronización
Un despliegue de TADDM con un servidor de sincronización y dos o más despliegues de servidor de dominio, cada uno de los cuales tiene su propia base de datos local.

En este tipo de despliegue, el servidor de sincronización copia los datos de descubrimiento desde servidores de dominio múltiples, un dominio cada vez en procesos de sincronización de lotes.

En un despliegue de servidor de sincronización, la siguiente propiedad del servidor de TADDM debe definirse al siguiente valor:

```
com.collation.cmdbmode=enterprise
```

Este tipo de despliegue está obsoleto. Por tanto, en un nuevo despliegue de TADDM donde se necesita más de un servidor, utilice el despliegue de servidor en modalidad continua. Un servidor de sincronización puede convertirse en un servidor de almacenamiento primario para el despliegue del servidor en modalidad continua.

despliegue del servidor en modalidad continua
Despliegue de TADDM con un servidor de almacenamiento primario y al menos un servidor de descubrimiento. Este tipo de despliegue también puede incluir uno o más servidores de almacenamiento secundarios opcionales. El servidor de almacenamiento primario y secundario comparten la base de datos. Los servidores de descubrimiento no tiene base de datos.

En este tipo de despliegue, los datos de descubrimiento fluyen en paralelo desde los servidores de descubrimiento múltiples a una base de datos de TADDM.

En un despliegue de servidor en modalidad continua, la propiedad del siguiente servidor de TADDM debe enviarse a uno de los siguientes valores:

- `com.collation.taddm.mode=DiscoveryServer`
- `com.collation.taddm.mode=StorageServer`

Para todos los servidores excepto el servidor de almacenamiento primario, las siguientes propiedades (para el nombre de host y el número de puerto del servidor de almacenamiento primario) también deben definirse:

- `com.collation.PrimaryStorageServer.host`
- `com.collation.PrimaryStorageServer.port`

Si la propiedad com.collation.taddm.mode está definida, la propiedad com.collation.cmdbmode debe definirse o comentarse.

dominio
En TADDM, un subconjunto lógico de infraestructura de una compañía u otra organización. Los dominios pueden dibujar límites organizativos, funcionales o geográficos.

elemento de configuración (CI)
Componente de infraestructura de TI que está bajo el control de gestión de la configuración y por lo tanto está sujeto a control de cambios formal. Cada elemento de configuración de la base de datos de TADDM tiene un objeto persistente y un historial de cambios asociado a él. Ejemplos de un elemento de configuración son un sistema operativo, una interfaz de nivel 2 o un tamaño de agrupación de almacenamiento intermedio de base de datos.

equivalente de servidor (SE)
Unidad representativa de infraestructura de TI definida como un sistema informático (con configuraciones estándar, sistemas operativos, interfaces de red e interfaces de almacenamiento) con software de servidor instalado (como una base de datos, un servidor web o un servidor de aplicaciones).

El concepto de un equivalente de servidor también incluye la red, el almacenamiento y otros subsistemas que proporcionan servicios para el funcionamiento óptimo del servidor. Un equivalente de servidor depende del sistema operativo:

Sistema operativo	Número aproximado de CI
Windows	500
AIX	1000
Linux	1000
HP-UX	500
Dispositivos de red	1000

hebra Worker de descubrimiento
En TADDM, una hebra que ejecuta sensores.

recopilación
En TADDM, un grupo de elementos de configuración.

SE Consulte *equivalente de servidor*.

servidor de almacenamiento
Servidor de TADDM que procesa los datos de descubrimiento recibidos de los Discovery Servers y los almacena en la base de datos. El servidor de almacenamiento primario coordina tanto los servidores de descubrimiento como todos los otros servidores y funciona como servidor de almacenamiento. Todos los servidores de almacenamiento que no son el servidor primario se llaman servidores de almacenamiento secundario.

Servidor de descubrimiento
Un servidor de TADDM que ejecuta sensores en un despliegue de servidor en modalidad continua pero que no tiene su propia base de datos.

servidor de dominio
Un servidor TADDM que ejecuta sensores en un despliegue de servidor de dominio tiene su propia base de datos.

servidor de sincronización
Un servidor de TADDM que sincroniza los datos de descubrimiento desde todos los servidores de dominio en la empresa tiene su propia base de datos. Este servidor no descubre los datos directamente.

servidor de TADDM
Un término genérico que puede representar cualquiera de los siguientes términos:

- servidor de dominio en un despliegue de servidor de dominio
- servidor de sincronización en un despliegue de servidor de sincronización
- servidor de descubrimiento en un despliegue de servidor en modalidad continua
- servidor de almacenamiento (incluido el servidor de almacenamiento primario) en un despliegue del servidor en modalidad continua

sistemas de destino
En el proceso de descubrimiento de TADDM, el sistema que se va a descubrir.

tenencia múltiple
En TADDM, el uso que un proveedor de servicios o proveedor de TI

realiza de una instalación de TADDM para descubrir varios entornos de cliente. Además, el proveedor de servicios o proveedor de TI puede ver los datos de todos los entornos de cliente, pero dentro de cada entorno de cliente, y sólo los datos específicos del cliente respectivo se pueden visualizar en la interfaz de usuario o en los informes de dicho entorno de cliente.

un despliegue de servidor de dominio
Despliegue de TADDM con un servidor de dominio. Un despliegue de servidor de dominio puede ser parte de un despliegue de servidor de sincronización.

En un despliegue de servidor de dominio, la siguiente propiedad de servidor de TADDM debe definirse con el siguiente valor:

```
com.collation.cmdbmode=domain
```

Guía del desarrollador SDK

Presentación del kit del desarrollador de software

En este tema, se presenta el kit del desarrollador de software (SDK) de IBM Tivoli Application Dependency Discovery Manager (TADDM) y se proporciona una breve visión general del modelo de datos común de TADDM.

La Guía del desarrollador de SDK ofrece una visibilidad precisa de las aplicaciones empresariales al proporcionar mapas de aplicaciones que resaltan la relación entre la aplicación y su infraestructura de soporte. Las correlaciones de aplicación globales incluyen los componentes de infraestructura que componen la aplicación, sus configuraciones detalladas y las dependencias e interrelaciones del tiempo de ejecución.

TADDM almacena los datos de topología internamente utilizando una jerarquía de objetos Java™ denominada Modelo de datos común (CDM).

Visión general del kit del desarrollador de software (SDK)

Esta guía del SDK utiliza la arquitectura abierta y escalable de TADDM y le proporciona un mecanismo para reutilizar de forma rápida y eficaz las correlaciones de aplicaciones globales en distintas soluciones de gestión de aplicaciones.

Esta guía del SDK ofrece acceso global al proceso de descubrimiento y las correlaciones de la aplicación de TADDM, con las cuales podrá:

- Proteger la inversión de la implementación utilizando un SDK de integración probado en el mercado, abierto y basado en estándares
- Garantizar el éxito de las iniciativas de gestión de TI al compartir y reutilizar, de manera muy rentable, las correlaciones de la aplicación de TADDM en las distintas aplicaciones de gestión
- Mejorar la precisión de las soluciones de gestión integrando correlaciones de aplicaciones precisas y en tiempo real
- Utilizar integraciones y adaptadores de TADDM para despliegues eficaces

El SDK de TADDM proporciona un conjunto de interfaces de programación de aplicaciones (API) documentadas:

- API de Java
- API de Protocolo de acceso a objectos simple (protocolo SOAP)
- API de REST (Representational State Transfer)
- API de interfaz de línea de mandatos (CLI)

Estas API proporcionan acceso global a las correlaciones de aplicaciones de TADDM, incluidas las aplicaciones descubiertas y sus componentes, configuraciones y dependencias. Las API ofrecen también control completo del proceso de descubrimiento de TADDM y su ciclo de vida, incluidos el inicio, la detención y la gestión de descubrimientos.

Presentación del modelo de datos común

TADDM almacena los datos de topología internamente mediante una jerarquía de objetos Java que se conoce como el modelo de datos común (CDM).

El modelo de datos común (CDM), que se conserva en una base de datos relacional, consta de objetos de modelo que representan elementos descubiertos en el entorno empresarial. El modelo de datos contiene objetos descubiertos de cada tipo de elemento, como sistemas o aplicaciones, con los detalles correspondientes representados como objetos contenidos, tales como sistemas operativos o valores de configuración.

Puede acceder al modelo utilizando la API de TADDM de IBM, con todos los datos de detalle visualizados en Data Management Portal al que se puede acceder con esta interfaz. SDK representa los datos utilizando un formato XML con un esquema XML publicado. La mayoría de los objetos contenidos están incluidos en el documento y los objetos a los que se hace referencia varias veces aparecen duplicados en el documento. El documento XML resultante es mayor que los datos originales, aunque resulta sencillo realizar búsquedas en él con herramientas como XQuery o Xpath.

Conceptos relacionados:

"Modelo simplificado" en la página 18
Como el modelo de datos comunes ocasiona problemas, en la versión 7.3 de TADDM se introduce un nuevo modelo simplificado para almacenar datos. Los únicos elementos que se conservan del modelo antiguo son las clases.

Instalación y configuración del kit del desarrollador de software

En este tema se describen los requisitos del sistema para utilizar el kit del desarrollador de software (SDK) de IBM Tivoli Application Dependency Discovery Manager (TADDM) y se explica cómo instalar y configurar el SDK.

Requisitos del sistema

En esta sección, se describen los requisitos del sistema para utilizar el SDK de TADDM.

Tabla 1 lista los elementos del sistema y describe los correspondientes detalles de requisitos.

Tabla 1. Requisitos del sistema para el SDK de TADDM

Elemento	Detalles
Sistema operativo	Cualquier sistema operativo compatible con el Java Runtime Environment (JRE) necesario
Memoria	2 GB
Procesadores	1
Velocidad del procesador	1 GHz
Espacio del disco	200 MB (incluida la máquina virtual Java)

Tabla 1. Requisitos del sistema para el SDK de TADDM (continuación)

Elemento	Detalles
Requisitos de software adicional	Si el SDK se ejecuta en el mismo sistema que el servidor de TADDM, utilice el IBM Java SDK versión 7.0 proporcionado con el servidor de TADDM. El IBM Java SDK se encuentra en el directorio `$COLLATION_HOME/external`. Si va a instalar el SDK en un sistema distinto del servidor de TADDM, se necesita el Java SDK versión 7.0. Si el cliente no se encuentra en la misma máquina que el servidor, los niveles del Java SDK tienen que coincidir. Por ejemplo, no intente ejecutar un cliente con la versión 5.0 del Java SDK con un servidor que funcione con la version 7.0.

Instalación del SDK de TADDM

En esta sección se describe cómo instalar el software del SDK de TADDM en el sistema.

Puede utilizar el SDK en cualquiera de las modalidades siguientes:

- Modalidad incluida: el SDK se instala al mismo tiempo que TADDM en el sistema. Consulte el tema sobre la modalidad incluida para obtener más información.
- Modalidad autónoma: utilice esta modalidad para instalar el SDK en sistemas autónomos. Consulte el tema sobre la modalidad autónoma para obtener más información.

En sistemas de varios usuarios, como Linux y AIX, si hay más de una persona utilizando el SDK, los archivos de registro entrarán en conflicto con los permisos. Para evitarlo, puede instalar el SDK en el directorio de inicio.

Modalidad incluida

Si el servidor de TADDM ya está instalado en el equipo, el SDK estará disponible como parte de la distribución del directorio `$COLLATION_HOME/sdk`, tal y como se ve en la tabla siguiente.

Tabla 2. Estructura de directorios de modalidad incluida

Directorio	Contenido
dist/	El directorio raíz de TADDM, que se denomina también directorio COLLATION_HOME. En sistemas operativos tales como AIX o Linux, la ubicación predeterminada para instalar TADDM es el directorio /opt/IBM/taddm. Por tanto, en este caso, el directorio $COLLATION_HOME es /opt/IBM/taddm/dist. En sistemas operativos de Windows, la ubicación predeterminada para instalar TADDM es el directorio c:\IBM\taddm. Por lo tanto, en este caso, el directorio %COLLATION_HOME% es c:\IBM\taddm\dist.
bin/	
deploy/	
etc/	
external/	
lib/	
log/	
dist/sdk/	Contiene el SDK de TADDM. Consulte la tabla de la estructura de directorios de modalidad autónoma para obtener más información..

Modalidad autónoma

Para instalar el SDK de TADDM de forma independiente, vaya al directorio $COLLATION_HOME/sdk y extraiga el archivo sdk.zip en cualquier directorio del sistema. La estructura de directorios del SDK extraído se muestra en la tabla siguiente:

Tabla 3. Estructura de directorios en modalidad autónoma

Directorio	Contenido
adaptor	Contiene TADDM Discovery Library Adapter 1.0.
bin	Contiene scripts de shell útiles y archivos de proceso por lotes
dla	Contiene la herramienta IBM Discovery Library IdML Certification Tool
doc	Contiene archivos PDF en inglés y otros archivos de documentación
etc	Propiedades de configuración
examples	Directorio de muestras
lib	Bibliotecas de tiempo de ejecución de servidor y de cliente
log	Registros de tiempo de ejecución
schema	Esquema XML

Configuración del SDK de TADDM

Puede configurar el SDK de TADDM especificando valores para las variables de entorno. También puede, si lo desea, configurar el funcionamiento de SDK especificando valores para los parámetros de configuración.

Definición de variables de entorno

Antes de empezar

Tiene que definir las variables de entorno antes de utilizar los programas de utilidad de interfaz de línea de mandatos (CLI) y kit del desarrollador de software, o antes de ejecutar los ejemplos proporcionados.

Procedimiento

Para definir las variables de entorno, realice este paso:

Defina la variable de entorno JAVA_HOME al directorio del entorno de tiempo de ejecución Java.
Si no se ha definido JAVA_HOME, el script ejecuta el primer archivo ejecutable de Java que se encuentre en la vía de acceso de ejecución.

Configuración de las propiedades de configuración

Los parámetros de configuración se encuentran en el archivo `$COLLATION_HOME/sdk/etc/collation.properties`. Tabla 4 describe los parámetros de configuración que puede especificar:

Tabla 4. Propiedades de configuración

Parámetro	Detalles
`com.collation.version`	Versión de la API
`com.collation.LogFile`	Ubicación en la que se han registrado los mensajes del cliente. El directorio debe existir. Si el archivo no existe, se creará. El valor predeterminado proporcionado es `../log/api-client.log`. Si esta propiedad no se ha especificado, el registro toma como valor predeterminado stdout.
`com.collation.log.level`	Nivel de registro, de entre los valores siguientes: • INFO—Default • ERROR • DEBUG
`com.collation.log.filesize`	Tamaño de archivo de registro. El valor predeterminado es 20 MB.
`com.collation.log.filecount`	Recuento de aplazamiento. El valor predeterminado es 3.

Tabla 4. Propiedades de configuración (continuación)

Parámetro	Detalles
com.ibm.cdb.service.registry.public.port	Puerto predeterminado para el registro RMI de los servicios públicos de TADDM. La SDK (API) es uno de los servicios públicos de TADDM. Este valor debe ser igual que el valor para servidor de TADDM. El valor predeterminado es 9433. Si la API se conecta a varios servidores de TADDM, tiene que configurar todos los servidores para que utilicen el mismo puerto o especificar el puerto en el momento de la conexión.

El valor de la propiedad `com.ibm.cdb.service.registry.public.port` debe coincidir con el valor del servidor de TADDM. De lo contrario, el SDK de TADDM no funcionará. Es necesario para la modalidad autónoma y la incluida.

Verificación de la instalación del SDK

Antes de empezar

Puede verificar que ha instalado y configurado correctamente el SDK de TADDM.

Procedimiento

Para verificar que la instalación sea correcta, siga estos pasos:

1. Vaya al directorio binario del SDK ejecutando un mandato parecido al siguiente:

   ```
   cd $COLLATION_HOME/sdk/bin
   ```

   Nota: Usuarios de Windows: las instrucciones para la verificación en Windows son parecidas, pero se utiliza **api.bat** en lugar de **api.sh** y el directorio bin se encuentra en `%COLLATION_HOME%\sdk\bin`.
2. Para visualizar el uso de la interfaz de línea de mandatos, ejecute el mandato siguiente:

   ```
   % ./api.sh
   ```
3. Para visualizar el estado del descubrimiento, ejecute el mandato siguiente:

   ```
   % ./api.sh -u usuario -p contraseña -H host discover status
   ```

   Este mandato consulta el estado de descubrimiento actual. Si ve un estado válido (como `Idle`), se ha comunicado correctamente con el servidor de TADDM y ha ejecutado un mandato.
4. Inicie un descubrimiento ejecutando el mandato siguiente:

   ```
   % ./api.sh -u usuario -p contraseña -H host discover start 10.10.10.12
   ```

   A continuación, compruebe el estado del descubrimiento para verificar que el descubrimiento se esté ejecutando:

   ```
   % ./api.sh -u usuario -p contraseña -H host discover status
   ```
5. Consulte los ámbitos de descubrimiento definidos ejecutando el mandato siguiente:

   ```
   % ./api.sh -u usuario -p contraseña -H host find Scope
   ```

El mandato devuelve los ámbitos definido en el servidor de TADDM en formato XML.

6. Recopile todos los sistemas del servidor de TADDM ejecutando el mandato siguiente:

   ```
   % ./api.sh -u usuario -p contraseña -H host find ComputerSystem
   ```

   El mandato devuelve todos los sistemas descubiertos en formato XML.

Utilización del SDK de TADDM como componente de software

Para integrar el SDK de TADDM en una aplicación o en un entorno de servidor de aplicaciones, debe definir las vías de acceso de clases de compilación y tiempo de ejecución, y definir el control de accesos.

Antes de empezar

Las vías de acceso de clases señalan a la biblioteca Java que proporciona la API de Java.

La distribución SDK de TADDM también empaqueta las bibliotecas saxon y xalan para el procesamiento de XSLT y XQuery. Puede utilizar estas bibliotecas, o sus propias herramientas de procesamiento XML, para el procesamiento de XSLT y XQuery.

Procedimiento

Para integrar SDK como componente, siga estos pasos:

1. Defina la siguiente vía de acceso de clases en la compilación y el tiempo de ejecución:

   ```
   CLASSPATH=$COLLATION_HOME/sdk/lib/taddm-api-client.jar:
     $COLLATION_HOME/sdk/lib/platform-model.jar
   ```

2. Configure los valores de acceso (ID de usuario y contraseña).

   Para utilizar la API de Java y de la CLI, es necesario configurar los valores de acceso mediante Data Management Portal. Puede utilizar el mismo ID de usuario y la misma contraseña para el acceso a la API y Data Management Portal.

Qué hacer a continuación

Tras actualizar desde un release anterior de TADDM, es posible que tenga que actualizar la vía de acceso de clases para que se incluyan los archivos .jar correctos.

El servidor de TADDM utiliza también los archivos .jar del directorio `$COLLATION_HOME/sdk/lib`. Por lo tanto, el archivo de SDK no se debe mover la instalación. Si necesita tener los archivos SDK en otra ubicación, puede extraerlos del archivo `sdk.zip` del DVD del producto.

Archivos .jar de Java necesarios

Los archivos `taddm-api-client.jar` y `platform-model.jar` son necesarios para utilizar la API de Java y deben estar presentes en un directorio que aparece en la variable de entorno CLASSPATH del sistema. Estos archivos se proporcionan en el subdirectorio `lib` del directorio de SDK.

Los archivos `taddm-api-client.jar` y `platform-model.jar` han sustituido todos los archivos JAR de TADDM anteriores como archivos que contienen API de modelo y definiciones de modelo.

Si va a utilizar el kit de herramientas XML de IBM Tivoli Business Service Manager (TBSM) con el tipo de conexión JDBC, necesitará también `oal-topomgr.jar`. Puede descargar este archivo JAR de la ubicación siguiente:

`http://`*`nombre.máquina.servidor.taddm`*`:`*`puerto.web.servidor.taddm`*`/GetTaddmVersion/getVersion/getoaltopomgrfile`

Para detectar cambios en la versión de los archivos JAR del servidor de TADDM, una aplicación cliente puede utilizar los URL siguientes para obtener valores de suma de comprobación para los archivos:

- `taddm-api-client.jar`:

  `http://`*`nombre.máquina.servidor.taddm`*`:`*`puerto.web.servidor.taddm`*`/GetTaddmVersion/getVersion/clientjar`

- `platform-model.jar`:

  `http://`*`nombre.máquina.servidor.taddm`*`:`*`puerto.web.servidor.taddm`*`/GetTaddmVersion/getVersion/modeljar`

- `oal-topomgr.jar`:

  `http://`*`nombre.máquina.servidor.taddm`*`:`*`puerto.web.servidor.taddm`*`/GetTaddmVersion/getVersion/oaltopomgrjar`

donde *nombre.máquina.servidor.taddm* es el nombre de dominio completo del servidor donde se está ejecutando TADDM y *puerto.web.servidor.taddm* es el puerto HTTP definido para el servidor de TADDM, cuyo valor predeterminado es 9430.

Nota: Si el servidor de TADDM se inicia como parte del proceso de instalación, la suma de comprobación de `taddm-api-client.jar` se notificará incorrectamente como 11111111 posteriormente. Si es así, reinicie el servidor; las siguientes solicitudes de cliente devolverán la suma de comprobación correcta.

Un cliente puede comprobar también la versión del servidor de TADDM utilizando el URL siguiente:

`http://`*`nombre.máquina.servidor.taddm`*`:`*`puerto.web.servidor.taddm`*`/GetTaddmVersion/getVersion/taddmversion`

Este URL devuelve la versión del producto (por ejemplo, `7.2.1`).

Instalación y configuración de la API de SOAP

La API de SOAP se ha instalado con el SDK de TADDM. Sin embargo, es necesario que complete el procedimiento descrito en esta sección antes de utilizar la API.

Procedimiento

Para completar la instalación y configuración de la API de SOAP, siga estos pasos:

1. Descargue el paquete de Axis de Internet.
2. Descomprima el paquete en el directorio `$COLLATION_HOME/sdk/lib`.
3. Incluya los archivos JAR del paquete de Axis en la variable CLASSPATH.

Instalación y configuración de la API de REST

La API de REST de TADDM no necesita los archivos .jar proporcionados con TADDM. Sin embargo, es posible que necesite algunos archivos .jar si quiere trabajar con objetos de modelo de TADDM.

Acerca de esta tarea

Puede utilizar los archivos .jar de TADDM para acceder a las clases de objeto de modelo de TADDM y la clase ModelObjectFactory, que convierte los objetos de

modelo en representaciones XML, o a partir de ellos.

Procedimiento

Para acceder a las clases, complete esta tarea:

Incluya los archivos .jar, adecuados para el Java SDK en un directorio de su variable de entorno CLASSPATH:

- `$COLLATION_HOME/sdk/lib/taddm-api-client.jar`
- `$COLLATION_HOME/sdk/lib/platform-model.jar`

Modelo de datos común

El modelo de datos común (CDM) es el lenguaje de definición utilizado para integrar la comprensión y el intercambio de datos entre los productos de gestión de Tivoli relacionados con recursos y componentes de la empresa de un cliente. El CDM es el modelo utilizado para comunicar información sobre instancias de recursos con la base de datos de IBM Tivoli Application Dependenc Discovery Manager (TADDM).

El CDM se compone por completo de definiciones de datos. Estas definiciones son características que identifican recursos, sus significados y las restricciones de su longitud o sus valores. El contenido del CDM se obtiene mediante la fusión de información del sector y estándares de modelos de datos relevantes con los modelos de datos utilizados por nuestros productos actuales en un único modelo convergente. Incorpora los estándares siguientes:

- Estándar Common Information Model (CIM) de Distributed Management Task Force (DMTF)
- Los siguientes estándares de procesos empresariales:
  - Business Process Execution Language (BPEL),
  - Especificación IT Infrastructure Library (ITIL)
  - Esquema de directorio LDAP
- Los siguientes estándares específicos del dominio:
  - TeleManagement Forum (TMf),
  - Storage Networking Industry Association (SNIA) y más.

Varias aplicaciones utilizan el modelo de datos común, incluido TADDM. Las aplicaciones que utilizan el CDM pueden compartir definiciones y terminología para datos de la instancia de recursos que sean comunes a las dos, lo que permite la construcción de aplicaciones de nivel superior que comprenden el entorno de gestión global y comparten información entre dichos sistemas. El CDM describe el contenido de entrada y salida de la API de TADDM, los sensores, las aplicaciones de utilidad y la Consola de gestión de descubrimiento.

El CDM es diferente de un esquema. Un esquema suele estar asociado a una base de datos, incluye la organización de datos en un modelo lógico y la especificación de cómo se almacenan los datos en columnas específicas de tablas específicas (también se denomina modelo físico de la base de datos). El CDM representa un modelo lógico compuesto de definiciones que habilita la identificación coherente de instancias de recursos, además de información sobre ellas y relaciones entre ellas. El modelo de datos enlaza los procesos empresariales y de TI con los sistemas que los proporcionan, los usuarios que los invocan, las políticas que los controlan, los recursos que utilizan los procesos, etc. El CDM clasifica y organiza

las características gestionadas más habitualmente de usuarios, recursos y procesos e información de TI empresarial, además de presentarlas de tal manera que todas las aplicaciones las puedan utilizar.

Para obtener más información sobre el CDM, consulte la información siguiente:

- Sitio web de Tivoli Common Data Model en el directorio `$COLLATION_HOME/sdk/doc/model`.
- *IBM Tivoli Common Data Model: Guide to Best Practices* en http://www.redbooks.ibm.com/abstracts/redp4389.html.

El modelo de datos común tiene las características siguientes:

- No define el esquema físico, ni cómo funciona un sistema de gestión.
- Define los recursos y las características de un entorno de gestión que supervisa, analiza y controla el sistema de gestión.
- También se utiliza cuando las aplicaciones de gestión intercambian información sobre instancias de recursos y sus relaciones con otros recursos.
- Estandariza las características, los conceptos de clases, atributos, interfaces, reglas de denominación, políticas de denominación y los tipos de datos que se utilizan.
- Proporciona definiciones coherentes de elementos, mejores prácticas para el contenido y directrices para correlacionar los datos de la instancia de recursos con el CDM.

El modelo de datos común incluye los objetos siguientes:

Clases Tienen las siguientes características o reglas:

- Una clase es un constructo utilizado para agrupar atributos relacionados.
- Las clases son la representación de un tipo de instancia de recursos (por ejemplo, OperatingSystem como tipo de instancia de recursos).
- Igual que la estructura básica del modelo, las clases contienen atributos, implementan interfaces y, opcionalmente, se pueden implicar en relaciones.
- Las clases son jerárquicas y heredan las propiedades de las clases padre.
- Las clases también pueden incluir explícitamente propiedades que pertenecen a un nivel de detalle determinado.
- Las instancias de clases representan las instancias de recursos actuales, los *nombres* que representan los recursos físicos o lógicos del entorno.
- Las instancias tienen atributos y pueden tomar parte en las relaciones. Por ejemplo, en un entorno de gestión de bases de datos, los elementos como el servidor de bases de datos, las tablas y las conexiones son instancias.

  Nota: Las instancias incluyen también elementos que no solo son gestionados, sino que toman parte en el proceso de gestión, como usuarios o sistemas empresariales.
- Aparte de los diversos objetos del CDM, las clases son las únicas que se utilizan para representar instancias de recursos. Hay clases concretas que se mencionan en la documentación de TADDM y que tienen un significado concreto:
  - **ModelObject**: esta clase representa la clase raíz o base del CDM. Todas las clases derivan de alguna manera de ModelObject. El

término ModelObject se utiliza en la documentación para representar cualquier clase definida en el CDM.

  - **ManagedElement**: se trata de otra representación de una clase raíz o base del CDM, y se corresponde directamente con la representación del Common Information Model de DMTF con el mismo nombre. El término ManagedElement se utiliza también en la documentación para representar cualquier clase que se haya definido en el CDM. Las clases ModelObject y ManagedElement se pueden intercambiar.
  - **ManagementSoftwareSystem**: también denominada MSS, esta clase representa los productos de gestión que proporcionan datos a TADDM mediante algún mecanismo. Cada proveedor de datos (incluidos los sensores de TADDM) se representan como una instancia de recursos de tipo ManagementSoftwareSystem.

- El CDM admite especialización mediante la herencia única, aunque el uso de interfaces da al modelo algunos aspectos de herencia múltiple. Todas las clases se organizan en una jerarquía de herencia y raíz únicas con la clase **ModelObject** como raíz. Todas las clases, con la excepción de ModelObject, especifican exactamente un padre y la clase de nivel inferior hereda todas las características de la clase padre.
- El CDM incluye también reglas de denominación para objetos de modelo que especifican los atributos necesarios para denominar objetos de forma exclusiva en TADDM. Consulte la sección sobre **Instancias con nombre** para obtener más información sobre las reglas de denominación de los objetos de modelo.
- Clases persistentes frente a no persistentes:
  - Una clase persistente es una clase cuyas instancias se pueden almacenar en una base de datos, mientras que las instancias de una clase no persistente no se pueden almacenar en una base de datos.
  - Si utiliza MQL (Model Query Language), solo puede consultar los objetos de clases persistentes. La única excepción pasa por consultar el atributo "guid" de ModelObject (clase no persistente), como en el ejemplo siguiente:
    - El atributo "source" es un ModelObject y las consultas siguientes devuelven los mismos resultados:

      ```
      SELECT * FROM TransactionalDependency WHERE source.guid ==
      'E72B13789C9039BFB32E3822FE50C197'
      ```

      ```
      SELECT * FROM TransactionalDependency WHERE source ==
      'E72B13789C9039BFB32E3822FE50C197'
      ```

  - En el modelo Javadoc (Javadoc para CommonDataModel de TADDM), si la etiqueta '**Persistable**' se define en `true` para una clase determinada, será una clase persistente. Si no hay etiqueta para una clase dada, se trata de una clase no persistente.
    - Ejemplos de clases persistentes: ComputerSystem, SoftwareModule, AppServer
    - Ejemplos de clases no persistentes: ModelObject, Database, LogicalElement

Atributos

Tienen las siguientes características o reglas:

- Un atributo define una propiedad concreta válida para una clase.
- Todos los atributos tienen un significado o una semántica concretos en lo referido al contenido esperado.

- Los atributos se especifican en las clases del CDM, al igual que las interfaces.
- Las instancias de los atributos son los *adjetivos* que describen las características de instancias y se utilizan para diferenciar instancias de la misma clase, como el diferente **Fabricante** de instancias de clase **ComputerSystem**.
- Al crear una instancia de recursos, se pueden almacenar datos para cualquier atributo válido para una instancia de recursos.
- No es necesario que todos los atributos contengan un valor, pero hay algunos atributos que se utilizan para representar una identidad única para una instancia de recursos. Estos atributos suelen denominarse *atributos de identidad*.

Interfaces

Habilite la reutilización conveniente de un conjunto de atributos y proporcione una mayor flexibilidad en la definición de relaciones. Por ejemplo, el atributo **VersionString** es un atributo válido para distintos tipos (clases) de instancias de recursos. En lugar de duplicar el atributo en varias clases en el CDM, se crea una interfaz para representar el conjunto de atributos que pertenecen a los datos de la versión.

Las instancias de recursos no se pueden basar en una interfaz. Cualquier clase que implemente una interfaz automáticamente recibe el conjunto de atributos y relaciones de la interfaz, como si existiesen en la clase. Las interfaces son jerárquicas y pueden derivar de la herencia sus atributos y relaciones en la interfaz padre.

Hay una interfaz concreta mencionada en la documentación de TADDM con un significado concreto. Esta interfaz se denomina *elemento de configuración*. El elemento de configuración de la interfaz se utiliza para denotar clases particulares en el CDM cuyas instancias actúan como un elemento de configuración definido por el término de ITIL correspondiente. Determinadas clases del CDM, como los datos financieros, no se han definido para ser elementos de configuración, ya que el CDM representa aspectos de varios entornos.

Relaciones

Tienen las siguientes características o reglas:

- Asociaciones entre dos instancias de recursos, que muestran cómo se relacionan entre ellas las instancias de recursos.
- Las relaciones solo se pueden dar entre clases y se dan entre clases del mismo tipo o de tipos distintos.
- Cada relación tiene una definición o un tipo concretos. Estos distintos tipos de relación conllevan una cierta semántica que pertenece al tipo de asociación entre las instancias de recursos.

  Por ejemplo, uno de los tipos de relación del CDM es **manages**, que representa que la instancia de origen participa en una función de control sobre la instancia del recurso de destino en la relación. Otro tipo de relación es **installedOn**, que representa la instancia de origen como un objeto instalado en la instancia del recurso de destino. Ambas relaciones pueden ser válidas en las instancias de recursos en las que el origen es una instancia de clase **Agent** y el destino es una instancia de clase **OperatingSystem**. Sin embargo, las dos relaciones tienen significados distintos. Pueden existir varias relaciones entre las dos mismas clases (y las dos mismas instancias). Cada relación forma una asociación entre dos instancias.

En el CDM, cada instancia de relación tiene un origen y un destino, que son las funciones de la relación. El número de instancias que pueden participar en cada función es importante. Algunas relaciones permiten solo que una instancia participe. Otras, permiten un número ilimitado. La cantidad de instancias que pueden participar en cada función se conoce como *cardinalidad de la relación*.

Tipos de datos

La información contenida en los atributos y las medidas se debe presentar en una sintaxis conocida y, por ello, el CDM define un conjunto de tipos de datos que se deben utilizar para representar la información de la entidad.

Los tipos de datos definidos en el modelo no especifican una representación física de los datos, si no que especifican las longitudes de datos y, en ocasiones, la codificación o las mejores prácticas del contenido de los datos.

El modelo incluye también tipos de datos enumerados que permiten que los productos comprendan el significado común de determinados valores

Instancias con nombre

Los nombres (o atributos de nombre) forman la base de la identificación de recursos y conciliación entre instancias de recursos que representan el mismo objeto en el centro de datos.

La denominación se basa en la generación, el uso y la compartición de atributos legibles para identificar instancias de recursos. Al agrupar el contenido de atributos concretos para una instancia de recursos, se crea un nombre exclusivo para la instancia de recursos. Dado el tamaño del modelo de datos, hay muchas formas posibles de denominar una instancia de recursos. Para organizar el método de generación de un nombre exclusivo, el Modelo de datos común utiliza el concepto de reglas de denominación para agrupar un conjunto de atributos que constituyen una identidad única.

Reglas de denominación

Una regla de denominación es una especificación sobre cómo denominar a las instancias de una clase concreta, como recursos, personas y sistemas.

Las reglas de denominación contienen un conjunto de atributos que son necesarios para denominar un recurso dado. El caso habitual para una regla de denominación pasa por agrupar atributos con el fin de formar una identidad única. Si el nombre de las dos instancias es el mismo, se presupone que se refieren a la misma entidad. Por ejemplo, las distintas identidades de una red de capa 2 se suelen identificar de la misma manera, utilizando una dirección, aunque las entidades sean instancias de clases distintas y posiblemente no relacionadas. Las direcciones MAC, con su estructura, forman un espacio desde el que se pueden asignar todos los nombres válidos para una estación de la red de capa 2.

Nota: Esto es independiente del tipo de red implicado, que podría ser 10-BaseT, 1000-BaseT o Token Ring.

Hay dos casos especiales en los que las reglas de denominación contendrán algo más que atributos.

1. Contexto de denominación:

   En ocasiones, en la denominación de una instancia de recursos, existe una pequeña cantidad de información disponible para denominar de forma

exclusiva la instancia según los atributos disponibles en la clase. En estos casos, hay determinadas reglas de denominación que especifican una relación, además de un conjunto de atributos, tal y como requiere la regla de denominación. Estas relaciones crean lo que se conoce como *Contexto de denominación* en la instancia de recursos y requieren que haya un segundo recurso en uso para identificar en contexto otra instancia de recursos.

Por ejemplo:

- Todo lo que se sabe sobre una instancia concreta de un sistema operativo es el tipo de sistema operativo.
- El atributo que representa el tipo de sistema operativo no es lo bastante exclusivo como para crear una instancia de recursos exclusiva que represente al sistema operativo.
- Para utilizar este atributo, la regla de denominación especifica una relación **installedOn** necesaria de la instancia del sistema operativo a una instancia de un sistema informático (existe el requisito implícito de crear también una instancia válida de un sistema informático para crear la relación).

2. NOT:

   Determinadas reglas de denominación contienen un conjunto definido de atributos que resultan aceptables para denominar de manera exclusiva una instancia de recursos en la mayoría de las circunstancias. Sin embargo, hay casos en el modelo de datos común en los que se necesita otra regla de denominación para refinar la identidad de un recurso, utilizando el mismo conjunto de atributos que utiliza otra regla de denominación al añadir atributos adicionales.

   Dado que el método para crear una instancia exclusiva se basa en el cumplimiento de las reglas de denominación, no se debería permitir que reglas de denominación con requisitos menos específicos generasen una identidad si se han proporcionado más atributos específicos. Para evitar que se utilice la regla de denominación menos específica, determinadas reglas de denominación utilizan una sentencia OmittedIdentifier en el atributo. Esto se conoce también como "NOT" en la sección del sitio web del modelo de datos común sobre reglas de denominación.

   Nota: Encontrará el sitio web del modelo de datos común en el directorio `$COLLATION_HOME/sdk/doc/model`.
   Si se menciona esta operación NOT, la operación muestra que el atributo debe ser nulo. Si existe algún contexto en el atributo mencionado en la operación OmmittedIdentifier (NOT), la regla de denominación no se utilizará para identificar un recurso de manera exclusiva. Por ejemplo:

   - Existe una regla de denominación en la actividad de clase denominada `ActivityName`.
   - Esta regla de denominación requiere que el atributo `ActivityName` contenga un valor.
     - La presunción con esta regla de denominación concreta es que el nombre de la actividad es exclusivo de forma global en el entorno del cliente.
   - En circunstancias en las que los nombres de actividades no son exclusivos, existe una segunda regla de denominación, denominada `QualifiedActivity`.
     - Esta regla requiere el atributo `ActivityName` y la relación `owns` de una instancia de la clase `OrganizationalEntity` con la instancia de la clase `Activity`

- Dado que las reglas de denominación utilizan un atributo común, `ActivityName`, y una regla de denominación es una mejora de otra regla de denominación, solo se debe utilizar una regla de denominación para nombrar la instancia `Activity`.
- Por lo tanto, la regla de denominación `ActivityName` especificó la operación `NOT` en la relación `owns`. Esto significa que la relación `owns` no se debe llenar para utilizar la regla de denominación `ActivityName`.

La identificación se basa en la generación, el uso y la compartición de un valor exclusivo, conciso y legible por máquina con fines de identificación de procesamiento. Las instancias de recursos que el modelo de datos común representa tienen nombres e identificadores:

- Los nombres series más largas, sobre todo alfabéticas, que se utilizan para referirse a las entidades.
- Los identificadores son valores más breves, sobre todo numéricos, que el sistema de gestión utiliza para identificar las entidades de manera exclusiva.

Identificadores exclusivos globales de TADDM

A los valores de identificación se les suele denominar identificadores exclusivos globales (GUID). Los identificadores exclusivos globales de TADDM se han creado de acuerdo con la especificación de la versión 3 de UUID (IETF Standards Track RFC 4122) y se utilizan como identificadores de los elementos de configuración (CI).

Los GUID de la versión 3 se generan procesando una serie con un algoritmo criptográfico de tipo MD5. TADDM pasa una serie creada a partir de los valores de los atributos que se utilizan en las reglas de denominación del componente de generación del GUID. La mayoría de los elementos de configuración tienen varias reglas de denominación, por lo que pueden generar varios GUID. Los valores de atributo que están disponibles cuando se crea el CI determinan los GUID que se generan. Normalmente, el primer GUID generado para un objeto se considera el GUID maestro o el identificador primario de dicho objeto. Otros GUID generados son alias del identificador exclusivo global maestro.

Si los CI se descubren con los mismos atributos y valores, siempre tienen el mismo conjunto de GUID. Sin embargo, el primer GUID, que se convierte luego en un GUID maestro, se genera aleatoriamente. Por ello, es posible que un elemento de configuración concreto no tenga el mismo GUID maestro en distintas instalaciones de TADDM. Del mismo modo, es posible que no se vuelva a elegir si el elemento se suprime o si se vuelve a crear la base de datos. Los mismos tipos de CI, como ComputerSystems, pueden utilizar también GUID que se calculan a partir de una regla de denominación distinta a la de sus GUID maestros.

Normalmente, las interfaces de programación de aplicaciones (API) de TADDM identifican los CI por sus GUID maestros, pero también los pueden identificar por sus alias. Por ello, si quiere encontrar un CI concreto, puede buscarlo utilizando el GUID de su alias.

Erosión del GUID

Los GUID que son alias del GUID maestro se pueden erosionar durante el ciclo de vida de un elemento de configuración. La erosión se produce si un atributo que define una única regla de denominación, como una firma, cambia. Tras este cambio, se genera un nuevo conjunto de GUID, que sustituye los valores antiguos.

Si los atributos de un GUID maestro cambian, este GUID sigue igual y se añade un alias nuevo.

Cambios del GUID maestro

El GUID maestro de un elemento de configuración concreto puede cambiar debido a cualquiera de estas condiciones:

Supresión y redescubrimiento de un elemento de configuración
Cuando se suprime un elemento de configuración de la base de datos de TADDM, se puede elegir un GUID distinto como GUID maestro durante el siguiente almacenamiento de este CI.

Caso de ejemplo de fusión de elementos de configuración
Si hay nuevos datos disponibles en TADDM, se pueden identificar dos elementos de configuración distintos como la misma instancia. Un usuario puede iniciar también la fusión manualmente. En este caso de ejemplo, se pueden fusionar los atributos de un elemento de configuración transitorio y otro duradero. Como resultado, el GUID maestro de un CI transitorio se convierte en un nuevo alias del duradero, y el GUID maestro del CI duradero representa un CI creado tras la fusión.

Actualización de TADDM
Si actualiza a una nueva versión de TADDM, los atributos que forman parte de las reglas de denominación pueden cambiar. Esta situación puede afectar también al proceso de migración de datos que debería garantizar que el GUID maestro siga siendo el mismo tras la actualización. Una nueva versión de sensores o de adaptadores de biblioteca de descubrimiento puede cambiar también la manera en la que se almacenan los valores de atributo.

Nombres de clase

Se puede hacer referencia a los nombres del objeto de clase del modelo de datos común de TADDM por su nombre largo o su nombre abreviado. Se puede hacer referencia a la mayoría de los nombres de objeto por su nombre abreviado.

En un sistema informático, el nombre abreviado es ComputerSystem y el nombre largo es com.collation.platform.model.topology.sys.ComputerSystem.

La excepción al uso de los nombres abreviados se da en el caso de los duplicados. Por ejemplo, se debe hacer referencia a SSLSettings por su nombre largo porque hay dos instancias de SSLSettings:

- `com.collation.platform.model.topology.app.lotus.SSLSettings`
- `com.collation.platform.model.topology.app.SSLSettings`.

El código de ejemplo siguiente muestra todos los nombres largos y abreviados de las clases del modelo de datos común. Después de ejecutar este mandato, se mostrará al final de los resultados una lista con los nombres de clases duplicadas a las que se debe hacer referencia por su nombre largo.

```
DisplayClassNames sample
import com.collation.proxy.api.client.*;
import com.collation.proxy.api.util.*;
import com.ibm.cdb.api.ApiFactory;
import java.util.*;

class DisplayClassNames {

    public static void main(String[] args) {
```

```
        CMDBApi api = null;

        try {
            System.out.println("--- Displaying Model Object Names ----" );

            ApiConnection conn = ApiFactory.getInstance().
                 getApiConnection("localhost", -1, null, false);
            ApiSession sess1 = ApiFactory.getInstance().getSession(conn,
              "administrator",
 "collation", ApiSession.DEFAULT_VERSION);
            api = sess1.createCMDBApi();

            String[] classNameArray = api.getClassNames();

            ArrayList shortNames = new ArrayList(classNameArray.length);
            ArrayList dups = new ArrayList(10);
            for (int i = 0; i < classNameArray.length; i++) {
               // imprimir los nombres de clase largos y abreviados
               System.out.println ("\nShort Name = " + classNameArray[i]);
               System.out.println ("Long  Name = " + classNameArray[i+1]);
               // Ver si el nombre abreviado es un duplicado
               if (shortNames.contains(classNameArray[i])) {
                  dups.add(classNameArray[i]);
               } else {
                  shortNames.add(classNameArray[i]);
               }
               i++;
            }
            System.out.println("\nThe following classes must be specified using
              the long name: ");
            System.out.println(dups);
            sess1.close();
        } catch(Exception ex) {
            ex.printStackTrace();
        } finally {
            if (api != null) {
              try {
                 api.close();
              } catch (Exception e) { }
            }
        }

    }

}
```

Dependencias entre recursos

TADDM descubre y categoriza varios tipos de dependencias entre niveles, que se reflejan en el CDM. Las dependencias modelan las relaciones de tiempo de ejecución entre los distintos componentes del CDM.

Hay varios tipos de dependencias, como:

- Dependencias transaccionales

  Las dependencias transaccionales se dan entre componentes de la aplicación, como servidores web, servidores de aplicaciones y bases de datos. El componente dependiente emite solicitudes para el componente del proveedor a fin de realizar determinadas funciones. Por ejemplo, una conexión con Java Database Connectivity (JDBC) desde un servidor de Java 2 Platform, Enterprise Edition (Java EE) a una base de datos es una dependencia transaccional. En este caso, el proveedor suele denominarse servidor y al dependiente se le denomina consumidor o cliente.
- Dependencias de servicio

Las dependencias de servicio se dan entre componentes de aplicaciones y servicios de infraestructura, como el sistema de nombres de dominio (DNS), el protocolo LDAP (Lightweight Directory Access Protocol) y el NFS (Network File System). El proveedor es el servicio de infraestructura y el componente dependiente solicita servicios de sistema del proveedor. Por ejemplo, solicita correlacionar un nombre de DNS con una dirección IP.

- Dependencias de IP

  Las dependencias de IP se dan entre dos sistemas informáticos o entre un servidor de aplicaciones y un sistema informático. TADDM crea este tipo de relación cuando descubre una relación entre dos sistemas informáticos, pero no puede descubrir exactamente qué servidor de aplicaciones está implicado.
- Dependencias del sistema

  Las dependencias del sistema se dan entre un servidor de aplicaciones y su sistema informático principal.
- Dependencias de aplicación a aplicación

  Las dependencias de aplicación a aplicación se dan de una aplicación empresarial a otra.

Ejemplo de dependencias

Cuando se crean dependencias de transacción para dos servidores de aplicaciones, no se crean dependencias de IP entre ellos. Tampoco se crean dependencias de IP entre el servidor de aplicaciones y sus hosts. Sin embargo, puede existir otra conexión lógica, por ejemplo entre dos procesos, y, en función de estas conexiones, podrían crearse dependencias de IP entre sistemas informáticos. Por ejemplo, tenga en cuenta el caso de ejemplo siguiente:

- El sistema informático (CS1) aloja un servidor de aplicaciones (AP1) y un proceso (P1)
- El sistema informático (CS2) aloja un servidor de aplicaciones (AP2) y un proceso (P2)

  Hay dos conexiones lógicas creadas por TADDM: AP1 <-> AP2 y P1<->P2

En este caso de ejemplo, se crea una dependencia transaccional entre AP1 y AP2 (basada en la conexión lógica AP1 <-> AP2). Se crea una dependencia de IP entre CS1 y CS2 (basada en la conexión lógica entre P1<->P2).

Modelo simplificado

Como el modelo de datos comunes ocasiona problemas, en la versión 7.3 de TADDM se introduce un nuevo modelo simplificado para almacenar datos. Los únicos elementos que se conservan del modelo antiguo son las clases.

Con el nuevo modelo simplificado puede crear sensores personalizados basados en scripts de un modo más fácil y ampliar así el alcance del descubrimiento. Puede personalizar sensores nuevos y editar los existentes.

Nota: El modelo simplificado no está soportado para otros productos que integra con TADDM.

En la tabla siguiente se enumeran las características más importantes que se han introducido con el modelo simplificado y se proporciona la ubicación en la que puede encontrar más información sobre ellas.

Tabla 5. Características del modelo simplificado

Característica	Ubicación
Clases genéricas de nivel superior en la jerarquía que representan elementos de configuración cruciales.	"Paquete, clases y jerarquías"
Nuevos objetos de nivel medio que representan componentes desplegables.	"Paquete, clases y jerarquías"
Un mecanismo para ampliar los nuevos niveles superiores con un número ilimitado de atributos (atributos ampliados).	"Atributos ampliados" en la página 23
Un mecanismo para almacenar bloques enteros de datos en bruto (por ejemplo XML o salidas de mandatos) y adjuntarlos a los elementos de configuración de nivel superior (instancias ampliadas).	"Instancias ampliadas" en la página 30
Nuevo operador `eval` utilizado en consultas MQL para atributos e instancias ampliados.	"Visión general de Model Query Language" en la página 42
Nuevo atributo de regla de denominación genérico `openId`.	"Atributo de regla de denominación genérico `OpenId`" en la página 21

Paquete, clases y jerarquías

Paquete y clases

Todas las clases nuevas se almacenan en el paquete `com.collation.platform.model.topology.simple`. Los nombres de las clases empiezan por la letra mayúscula "S" para evitar conflictos, porque TADDM diferencia tipos de datos por sus nombres abreviados.

Atributos de jerarquía

Cada objeto del modelo de datos contiene dos atributos de jerarquía, `hierarchyDomain` y `hierarchyType`, que definen la información que en el modelo antiguo estaba presente en el paquete y el tipo de cada clase. Por ejemplo, el tipo `ComputerSystem` contiene muchos sistemas específicos que representan distintos sistemas operativos, como `sys.linux.LinuxUnitaryComputerSystem` o `sys.windows.WindowsComputerSystem`. Estos objetos se tenían que almacenar por separado. El nuevo modelo simplificado permite almacenar objetos de ambos tipos como el tipo `SComputerSystem`, cuando estos dos atributos se establecen del siguiente modo:

- Para `sys.linux.LinuxUnitaryComputerSystem`:

  ```
  hierarchyDomain="sys.unix.linux"
  hierarchyType="RedHat"
  ```

- Para `sys.windows.WindowsComputerSystem`:

  ```
  hierarchyDomain="sys.windows"
  hierarchyType="Windows7"
  ```

Los valores del atributo `hierarchyDomain` especifican niveles de dominio, empezando por el nivel más general y acabando por el más específico. Por ejemplo, en el valor `"app.db.mongodb"`, `app` es un servidor de aplicaciones, `db` es un servidor de base de datos y `mongodb` es una base de datos específica, en este caso de MongoDB.

Los atributos de jerarquía se utilizan para gestionar la IU por completo. También se utilizan para consultas.

Nuevos tipos de jerarquía

La lista siguiente muestra los nuevos tipos de jerarquía:

- `SComputerSystem`: sustituye a la jerarquía `ComputerSystem`.
- `SSoftwareServer`: sustituye a la jerarquía `AppServer`.
- `SLogicalGroup`: sustituye a la jerarquía `AppServerCluster` y representa cualquier tipo de colección que almacene un sensor; por ejemplo, clústeres o dominios.
- `SFunction`: sustituye la jerarquía `Function` y representa reglas y funciones adicionales.
- `SSoftwareInstallation`: representa los paquetes de software físico que estén presentes en un sistema que constituya un servidor de software (bibliotecas creadas por proveedores).
- `SPysicalFile`: sustituye a las jerarquías `AppConfig` y `LogicalContent` y representa a un archivo de configuración. Todo el contenido de este archivo lo captura un descubrimiento.
- `SDeployableComponent`: un tipo nuevo que representa los tipos que se consideren orígenes o destinos de relaciones que estén desplegadas en sistemas, servidores de software o grupos lógicos. Son objetos de nivel medio. Algunos de los tipos CDM antiguos todavía están conectados a la jerarquía `SDeployableComponent` para garantizar la compatibilidad con versiones anteriores. La lista siguiente especifica dichos tipos:
  - `BiztalkApplication`
  - `Database`
  - `DominoDatabase`
  - `ExchangeLink`, `ExchangeStorageGroup`
  - `FileSystem`
  - `IIsWebServer`, `IIsWebVirtualDir`
  - `MBExecutionGroup`, `MBMessageFlow`, `MessageBox`
  - `MQChannel`, `MQQueue`
  - `OracleSchema`
  - `SharePointWebApplication`
  - `SoftwareModule`: el único componente del tipo `SoftwareModule` que queda es `J2EEApplication`, ya que es el único componente desplegable de este tipo.
  - `WebVirtualHost`

Todos estos tipos nuevos heredan de los siguientes tipos abstractos, que se han introducido en el modelo de datos para fines de modelado: `SStandaloneObject`, `SContextualObject`, `SGroup`.

Atributo `lastStoreTime`

El atributo `lastModifiedTime` está en desuso debido a que su nombre puede generar confusiones. Se ha sustituido por el nuevo atributo `lastStoreTime`.

Migración

La migración desde el modelo antiguo al nuevo es automática. No es necesario llevar a cabo ninguna tarea adicional. Las consultas MQL y las vistas SQL son compatibles con versiones anteriores.

Cambios en Data Management Portal

En el panel Resumen de inventario del separador **Inventario** hay un tipo de componente nuevo llamado `Generics`. Allí encontrará objetos de clases del paquete `simple`. También puede examinar carpetas genéricas en el panel Componentes descubiertos y visualizar los detalles de objetos descubiertos de nuevos tipos de clase. Para obtener información sobre la visualización de atributos e instancias ampliados, consulte "Atributos ampliados" en la página 23 y "Instancias ampliadas" en la página 30.

Aplicaciones empresariales

El motor de aplicaciones empresariales admite el modelo simplificado. Puede utilizar clases nuevas, atributos y el operador `eval` en consultas mientras trabaja con aplicaciones empresariales. También puede crear una aplicación empresarial a partir de los objetos que almacena mediante el modelo nuevo.

Atributo de regla de denominación genérico `OpenId`

Para cada objeto debe especificar atributos y definir una regla de denominación que indica cuál de estos atributos proporcionan un valor exclusivo. Las reglas de denominación varían en función del tipo de jerarquía. Los nuevos tipos genéricos que se almacena en el paquete `simple` se almacena con el uso de las mismas reglas de denominación que en el CDM antiguo. Los objetos que heredan del tipo `SStandaloneObject` se almacenan con el atributo `openId`. Los objetos que heredan del tipo `SContextualObject` se almacenan con los atributos `context` y `scopedId`.

Los atributos `openId` y `scopedId` son del tipo serie en una base de datos, pero en el nivel de la API son de un nuevo tipo Java, `OpenId`. Por lo tanto, dichos atributos tienen un formato específico, de acuerdo con el esquema de OpenId. El tipo `OpenId` contiene métodos para construir fácilmente cualquier serie con significado que represente un valor de una regla de denominación. Los valores especificados en el atributo `openId` son el origen del cálculo del GUID.

Ejemplos

Ejemplo 1

Para los servidores, la regla de denominación suele basarse en dos atributos. Son el punto de acceso a servicio primario que se crea a partir de la dirección IP primaria del servidor y un puerto que escucha este servicio. El atributo `openId` se especifica del siguiente modo:

```
id = OpenId().addId('IP' , seed.getPrimaryIpAddress().getStringNotation())
.addId('port' , str(seed.getPort()))
```

Ejemplo 2

En el modelo de datos comunes, el tipo antiguo `ComputerSystem` tiene una regla de denominación basada en los atributos `manufacturer`, `model` y `serialNumber`. Estos tres atributos se definen en la clase de forma explícita, y cuando se definen valores para ellos, puede almacenarse un objeto `ComputerSystem`.

```
LinuxUnitaryComputerSystem cs = ModelFactory.newInstance
(LinuxUnitaryComputerSystem.class);
cs.setManufacturer("RedHat");
cs.setModel("Linux");
cs.setSerialNumber("as00123012");
```

Luego un objeto con la correlación de atributos siguiente se almacena en la capa de persistencia:

```
manufacturer -> RedHat
serialNumber -> as00123012
isPlaceholder -> false
model -> Linux
```

En el modelo simplificado nuevo, los objetos almacenados del tipo simplificado `SComputerSystem` con los mismos valores parecen iguales. Sin embargo, si hay otro atributo de regla de denominación disponible para el sistema concreto (por ejemplo, un ID que un clúster asigne a cada sistema físico que gestiona), y este atributo de regla de denominación no está definido en el modelo, se puede ampliar con el uso del tipo `OpenId`. Si el atributo se amplía, no solo se almacena, sino que también representa el sistema informático de forma exclusiva. El tipo `OpenId` se utiliza de la siguiente manera:

```
SComputerSystem scs = ModelFactory.newInstance(SComputerSystem.class);
scs.setHierarchyDomain("sys.unix.linux");
scs.setHierarchyType("RedHat");
scs.setManufacturer("RedHat");
scs.setModel("Linux");
scs.setSerialNumber("as00123012");
OpenId id = new OpenId();
id.addId("clusterInternalId", "66");
scs.setOpenId(id);
```

Ejemplo 3

El tipo `OpenId` también se puede establecer de la siguiente manera:

```
scs.setOpenId(new OpenId().addId("clusterInternalId", "66"));
```

Se almacena la siguiente correlación de atributos:

```
manufacturer -> RedHat
serialNumber -> as00123012
model -> LinuxhierarchyType -> RedHat
isPlaceholder -> false
hierarchyDomain -> sys.unix.linux
openId -> <openId><id><name>clusterinternalid</name><value>66</value></id>
</openId>
```

Ejemplo 4

Añadir una función a este sistema simplificado se realiza de forma muy parecida. El ejemplo siguiente muestra cómo crear el atributo `OpenId` a partir de valores que ya se han establecido para los atributos de clase simplificada:

```
SFunction sf = ModelFactory.newInstance(SFunction.class);
sf.setName("Cisco Firewall");
sf.setHierarchyDomain("function.net.firewall");
sf.setHierarchyType("Cisco");

OpenId fid = new OpenId(sf);
fid.addId("name", null);
fid.addId("type", "firewall");
sf.setScopedId(fid);
sf.setProvider(scs);
```

Se almacena la siguiente correlación de atributos:

```
hierarchyType -> Cisco
isPlaceholder -> false
provider -> {hierarchyType=RedHat;hierarchyDomain=sys.unix.linux;
isPlaceholder=false;openId=<openId><id><name>clusterinternalid</name>
<value>66</value></id></openId>;}
hierarchyDomain -> function.net.firewall
```

```
name -> Cisco Firewall
scopedId -> <openId><id><name>name</name><value>Cisco Firewall</value>
</id><id><name>type</name><value>firewall</value></id>
</openId>
```

Ejemplo 5

Para crear fácilmente un id simple sin distinción alguna para atributos concretos dentro de él, establezca el atributo de la siguiente manera:

```
OpenId fid = new OpenId(sf);
sf.setProvider(scs);
sf.setScopedId(new OpenId().addId("id19921"));
```

Se almacena la siguiente correlación de atributos:

```
hierarchyType -> Cisco
isPlaceholder -> false
provider -> {hierarchyType=RedHat;hierarchyDomain=sys.unix.linux;
isPlaceholder=false;openId=<openId><id><name>clusterinternalid</name>
<value>66</value></id></openId>;}
hierarchyDomain -> function.net.firewall
name -> Cisco Firewall
scopedId -> <openId><id><name>id</name><value>id19921</value></id></openId>
```

Conceptos relacionados:

"Atributos ampliados"
En el modelo de datos comunes antiguo, los atributos ampliados se utilizaban para definir manualmente hasta 100 atributos por tipo de CDM para almacenar los datos. Se utilizaban para ampliar los tipos de CDM con atributos fuera del dominio, como por ejemplo el número de sala del servidor, y para ampliar el ámbito del descubrimiento de CI. En el modelo simplificado nuevo, los atributos ampliados se utilizan para almacenar todos los atributos simples de las clases de CDM que estaban antes en los tipos de la jerarquía.

Atributos ampliados

En el modelo de datos comunes antiguo, los atributos ampliados se utilizaban para definir manualmente hasta 100 atributos por tipo de CDM para almacenar los datos. Se utilizaban para ampliar los tipos de CDM con atributos fuera del dominio, como por ejemplo el número de sala del servidor, y para ampliar el ámbito del descubrimiento de CI. En el modelo simplificado nuevo, los atributos ampliados se utilizan para almacenar todos los atributos simples de las clases de CDM que estaban antes en los tipos de la jerarquía.

Por ejemplo, en el modelo antiguo, el tipo `ExchangeServer` tenía el atributo `productID` del tipo de serie definido. En el nuevo modelo, el tipo `ExchangeServer` se almacena de la siguiente manera:

```
SSoftwareServer sr = ModelFactory.newInstance(SSoftwareServer.class);
sr.setHierarchyDomain("app.messaging.exchange");
sr.setHierarchyType("Exchange");
sr.setOpenId(new OpenId().addId("serverName", "Exchange1122"));
```

El atributo `productID` no se puede almacenar porque el tipo `SSoftwareServer` solo almacena atributos esenciales y no se puede ampliar. Los atributos ampliados permiten dichos atributos específicos para el almacenaje.

Tipo de modelo de datos

En el nuevo modelo de datos se han introducido los siguientes cambios:

- Los atributos ampliados se almacenan junto con los elementos de configuración en el atributo `XA` de un nuevo tipo personalizado `ExtendedAttributesData`. Los datos que se conservan en objetos separados del tipo `UserData` se migran al atributo `XA`.
- Se ha eliminado la limitación para almacenar hasta 100 atributos por tipo de CDM. El número de atributos que se pueden almacenar en un solo objeto depende de la capacidad del tipo de columna XML de la base de datos. Además, se puede utilizar la opción `-g` del programa de carga masiva para almacenar atributos ampliados.
- Los atributos ampliados tienen categorías. Si no se selecciona ninguna categoría, se almacena un atributo en la categoría predeterminada `General`. Todos los atributos ampliados presentes en el modelo de datos antiguo se han movido a la categoría predeterminada `General`.
- El atributo antiguo `byte[] extendedAttributes` se conserva únicamente para garantizar la compatibilidad con las versiones anteriores y ha quedado en desuso. Los métodos de API públicos `setExtendedAttributes` y `getExtendedAttributes` también han quedado en desuso.

Visualización de atributos ampliados

Después de ejecutar un descubrimiento de un sensor, para el que ha especificado el atributo `XA`, puede ver atributos ampliados en Data Management Portal. Abra el panel Detalles de un objeto descubierto. El separador **Atributos ampliados** ya no está disponible; la información de los atributos ampliados de la categoría predeterminada se visualiza en el separador **General**. Los atributos ampliados de una categoría personalizada se visualizan en un separador personalizado. Por ejemplo, los atributos ampliados de la categoría Marcadores se visualizan en el separador **Marcadores**.

Ejemplo de uso en un sensor

En el ejemplo siguiente se muestra el tipo `ExtendedAttributesData`, que se utiliza para almacenar atributos ampliados con un CI. El atributo `productID` se conserva en una categoría predeterminada. Se crea una categoría nueva para la ubicación física de un servidor de software.

```
SSoftwareServer sr = ModelFactory.newInstance(SSoftwareServer.class);
sr.setHierarchyDomain("app.messaging.exchange");
sr.setHierarchyType("Exchange");
sr.setOpenId(new OpenId().addId("serverName", "Exchange1122"));

ExtendedAttributesData xa = new ExtendedAttributesData();
xa.addAttribute("productID", "ID12021");
xa.addAttribute("Location", "buildingNo", "North23");
xa.addAttribute("Location", "floorNo", "3");
xa.addAttribute("Location", "roomNo", "15");
xa.attachTo(sr);
```

Notas:

- En TADDM 7.3.0 y 7.3.0.1, puede utilizar dos métodos para almacenar los atributos ampliados. Uno de ellos es `attachTo` y se utiliza en el ejemplo anterior. Este método se debe especificar después de todas las entradas de `addAttribute()`. El modo alternativo es utilizar el método `setXA`, por ejemplo:

  ```
  xa.addAttribute("Location", "roomNo", "15");
  xa.toXML();
  sr.setXA(xa);
  ```

Si utiliza el método setXA, también debe especificar el método toXML para convertir los atributos ampliados en un diseño que se pueda almacenar.

- Fix Pack 2 En TADDM 7.3.0.2, no existe ninguna diferencia entre los métodos attachTo y setXA. Ninguno de ellos requiere que se utilice el método toXML.

Se almacena la siguiente correlación de atributos:

```
hierarchyType -> Exchange
isPlaceholder -> false
openId -> <openId><id><name>servername</name><value>Exchange1122</value>
</id></openId>
hierarchyDomain -> app.messaging.exchange
XA -> <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xml>
  <attribute name="productID" category="taddm_global">ID12021</attribute>
  <attribute name="buildingNo" category="Location">North23</attribute>
  <attribute name="floorNo" category="Location">3</attribute>
  <attribute name="roomNo" category="Location">15</attribute>
</xml>
```

Actualización parcial

Un CI se puede almacenar con los mismos valores de reglas de denominación de distintos orígenes de datos; como resultado, los valores se fusionan en un objeto. Para evitarlo, se utiliza el mecanismo de actualización parcial para fusionar dos atributos XA con formato XML diferentes. Por ejemplo, si un origen se almacena con el objeto 1 y otro origen se almacena con el objeto 2, se crea un CI que retiene el atributo fusionado.

Objeto 1

```
SSoftwareServer sr = ModelFactory.newInstance(SSoftwareServer.class);
sr.setHierarchyDomain("app.messaging.exchange");
sr.setHierarchyType("Exchange");
sr.setOpenId(new OpenId().addId("serverName", "Exchange1122"));

ExtendedAttributesData xa = new ExtendedAttributesData();
xa.addAttribute("productID", "ID12021");
xa.addAttribute("internal", "ID1233");
xa.addAttribute("Location", "buildingNo", "North23");
xa.attachTo(sr);
```

Objeto 2

```
SSoftwareServer sr = ModelFactory.newInstance(SSoftwareServer.class);
sr.setHierarchyDomain("app.messaging.exchange");
sr.setHierarchyType("Exchange");
sr.setOpenId(new OpenId().addId("serverName", "Exchange1122"));

ExtendedAttributesData xa = new ExtendedAttributesData();
xa.addAttribute("productID", "ID12024");
xa.addAttribute("customID", "ID12333");
xa.addAttribute("Location", "floorNo", "3");
xa.addAttribute("Location", "roomNo", "15");
xa.attachTo(sr);
```

Atributo fusionado

```
hierarchyType -> Exchange
isPlaceholder -> false
openId -> <openId><id><name>servername</name><value>Exchange1122</value>
</id></openId>
hierarchyDomain -> app.messaging.exchange
XA -> <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xml>
  <attribute name="customID" category="taddm_global">ID12333</attribute>
  <attribute name="internal" category="taddm_global">ID1233</attribute>
  <attribute name="productID" category="taddm_global">ID12024</attribute>
```

```
  <attribute name="buildingNo" category="Location">North23</attribute>
  <attribute name="floorNo" category="Location">3</attribute>
  <attribute name="roomNo" category="Location">15</attribute>
</xml>
```

Todos los atributos que no se solapan del primer y segundo objeto están presentes con sus respectivos valores en el atributo fusionado, y el atributo `productId` tiene un valor del último objeto almacenado. Cuando el atributo `XA` de cualquier objeto de modelo se actualiza parcialmente, tanto el tipo de categoría como el nombre de atributo se incluyen en el proceso.

Caché para los metadatos y la validación

Los atributos ampliados pocas veces se definen. Por lo tanto, se crea una caché para mantener los datos. Cada vez que se coloca un objeto de modelo en una herramienta de persistencia para su almacenaje, la herramienta valida el atributo `XA` en las metadefiniciones que se toman de esta caché. Cada tipo de categoría y par de nombre de atributo presente en el atributo `XA`, pero que no se define en la metacaché, se elimina del atributo `XA` antes de persistir el objeto de modelo. Cuando se inhabilita el proceso de renovación de la caché se toma la metainformación de la capa de persistencia. Al habilitarse el proceso, la metainformación se toma de la memoria Java.

Puede controlar la frecuencia del proceso de renovación de la memoria caché mediante la propiedad `com.ibm.cdb.ea.metaRefreshFrequency` en el archivo `collation.properties`. El valor predeterminado es `20` y se expresa en segundos. Si desea inhabilitar el proceso de renovación de la memoria caché, establezca el valor de esta propiedad en `0`.

Nota: Al definir los atributos ampliados, inhabilite el proceso de renovación de la memoria caché y habilítelo cuando acabe.

Consultas MQL con el operador EVAL (solo atributos `XA` y `XD`)

Muchos atributos del CDM se han movido al contenido XML de los atributos `XA` o `XD`. Por lo tanto, la sintaxis MQL admite un nuevo operador, `eval`, que se puede añadir a la cláusula where. El operador `eval` permite consultar los CI por los valores de atributos ampliados o de instancias ampliado.

Nota: Todos los ejemplos de consultas de MQL contienen comillas de escape (\"*value*\") porque se supone que las consultas se ejecutan de la siguiente forma:

```
./api.sh -u username -p password find "MQL query"
```

Por ejemplo, la siguiente consulta se ha ejecutado para buscar un sistema con el atributo `productID` definido como `'prod1'`:

```
SELECT * FROM ComputerSystem WHERE productID == 'prod1'
```

La siguiente consulta equivalente utiliza el operador `eval`:

```
SELECT * FROM ComputerSystem WHERE XA eval '/xml
[attribute[@category=\"taddm_global\" and @name=\"productID\"]=\"prod1\"]'
```

El operador `eval` puede ir seguido de cualquier expresión XPath válida que devuelva el valor booleano true o false para permitir que la capa de persistencia realice correctamente el filtrado del rendimiento de SQL.

Más ejemplos

- Busque todos los ComputerSystems que tengan cualquier atributo ampliado con el valor `val`:
  - MQL:

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml[attribute=\"val\"]'
    ```

  - SQL:

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml[attribute="val1"]'
    passing compsys.xa_x as "c")
    ```

- Busque un ComputerSystem con el atributo ampliado `attr2` que tenga la categoría establecida en `Other` y el valor en `two`:
  - MQL:

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml/attribute
    [@name=\"attr2\" and @category=\"Other\" and text()=\"two\"]'
    ```

  - SQL:

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml/attribute[@name="attr2"
     and and @category="Other" and text()="two"]' passing compsys.xa_x as
     "c")
    ```

Creación de atributos ampliados en el Portal de gestión de dominios

Puede definir atributos ampliados para un componente en particular en el Portal de gestión de dominios.

Procedimiento

Para crear un atributo ampliado debe llevar a cabo los siguientes pasos en el Portal de gestión de dominios.

Nota: También puede definir atributos ampliados ejecutando el programa de carga masiva.

1. En la barra de menús, pulse **Editar** > **Atributos ampliados**. Aparece la ventana Definir atributos ampliados.
2. En el menú **Tipo de componente**, seleccione el tipo de componente para el que desea crear un atributo ampliado.
3. Pulse **Nuevo**. Aparece la ventana Crear nuevo atributo ampliado.
4. En el campo **Nombre de atributo ampliado** escriba el nombre del atributo ampliado.
5. En el menú **Tipo de atributo ampliado**, seleccione el tipo de atributo ampliado que desea crear.
6. En el menú **Categoría de atributo ampliado**, especifique la categoría del atributo ampliado.
7. Pulse **Aceptar**.
8. En la ventana Definir atributos ampliados, pulse **Aceptar**.
   Consulte también el tema *Ventana Definir atributos ampliados* en el *Manual del usuario* de TADDM.

Creación de atributos ampliados en los archivos

Puede definir atributos ampliados para un componente en particular en los archivos en el directorio `$COLLATION_HOME/dist/etc/templates`.

Procedimiento

1. Asegúrese de que los archivos de plantilla del directorio `$COLLATION_HOME/dist/etc/templates/commands` solo contienen `SCRIPT:etc/templates/commands/extension-scripts/ea_sensor_1.0.py`.

2. En el servidor que desea descubrir, cree un archivo de configuración /tmp/*file_name*.conf que contenga pares de atributos que desee definir, así como sus valores. Por ejemplo:

   ```
   PrimerAtributo = Primer valor
   SegundoAtributo = Segundo valor
   ```

   Liste estos atributos en el archivo extended_attributes.properties. Consulte el paso siguiente.
3. En el directorio $COLLATION_HOME/etc/templates, cree el archivo extended_attributes.properties. Debe contener los atributos que desea recopilar. En el archivo extended_attributes.properties.example puede encontrar la estructura de este archivo. Ejemplo de una entrada:

   ```
   Linux.FileGEN.1.Key="/tmp/attribute.conf"
   Linux.FileGEN.1.Attributes="myAttribute"
   ```

   donde:

   - /tmp/attribute.conf es la ubicación del archivo, donde especifica el atributo que desea definir (/tmp/*file_name*.conf).
   - myAttribute es el nombre del atributo que desea definir, que está especificado en el archivo /tmp/attribute.conf.
4. En la consola de Discovery Management, en la ventana Sistemas, establezca el valor **Enable** en *true*.

Qué hacer a continuación

Antes de ejecutar un descubrimiento, todos los atributos que desee recopilar también deben crearse en el Portal de gestión de dominios. Si no lo están se pasarán por alto durante el descubrimiento.

Importante: Durante el descubrimiento, los nombres de los atributos que defina en los archivos se modifican. Se añade el prefijo FileGEN al inicio de cada nombre. Por ejemplo, si el nombre del atributo es myAttribute, cambiará a FileGEN_myAttribute. Por lo tanto, al crear atributos en el Portal de gestión de dominios debe proporcionar nombres con el prefijo FileGEN. Los atributos que se crean en los archivos y en el Portal de gestión de dominios deben tener los mismos nombres. Al proporcionar nombres sin el prefijo se omiten los atributos.

Supresión de atributos ampliados

Puede eliminar atributos ampliados existentes.

Procedimiento

Para suprimir un atributo ampliado, complete los siguientes pasos en el Portal de gestión de dominios (Domain Management Portal):

1. En la barra de menús, pulse **Editar** > **Atributos ampliados**. Aparece la ventana Definir atributos ampliados.
2. En el menú **Tipo de componente**, seleccione el tipo de componente para el que desea suprimir un atributo ampliado.
3. Seleccione un atributo ampliado existente.
4. Pulse **Suprimir**.
5. Pulse **Aceptar**. Se suprime el atributo ampliado.

Atributos ampliados de definición automática para la API Java pública y para el programa de carga masiva

Dado que la definición manual de atributos ampliados consume mucho tiempo, puede habilitar la definición automática de atributos ampliados. Puede trabajar con atributos ampliados en Data Management Portal únicamente para corregir y ajustar las definiciones de atributos ampliados.

La API de Java externa y el programa de carga masiva permiten al gestor de almacenamiento de TADDM comprobar si se detecta algún atributo ampliado adjunto a un CI. Si se detectan dichos atributos, los que no estén definidos como tipo `String` se pueden definir de forma automática. Con este mecanismo se recomienda utilizar el libro o script Jython para almacenar los datos junto con la definición automática forzada y, a continuación, ir a Data Management Portal para comprobar las definiciones y ajustar los tipos y categorías de atributos.

Para habilitar la definición automática de atributos ampliados con el uso de la API de Java externa, utilice el método `setAutoDefineEAs` del objeto `ApiSession`, que está establecido en `true`, como en el siguiente ejemplo:

```
conn = ApiFactory.getInstance().getApiConnection("localhost", -1, None,
Boolean(0))
sess = ApiFactory.getInstance().getSession(conn, "administrator", "collation",
ApiSession.DEFAULT_VERSION)
api = sess.createCMDBApi()

  print "Turning on auto-defining API feature for Extended Attributes"
sess.setAutoDefineEAs(Boolean(1))
```

Para habilitar la definición automática de atributos ampliados con el uso del programa de carga masiva, utilice el atributo `-loadEAMeta` o establezca la propiedad `com.ibm.cdb.bulk.loadeametaflag=true` en el archivo `collation.properties`. En esta modalidad, el programa de carga masiva no almacena ningún CI. En su lugar, extrae la información de todos los atributos ampliados que se han cargado y utiliza dichos atributos ampliados para crear definiciones correctas. Como resultado, en TADDM se crean los objetos `UserDataMeta` correctos.

Nota: Puede utilizar la opción `-g` al ejecutar el programa de carga masiva, que acorta el proceso de carga.

Atributos ampliados de definición automática para sensores

Fix Pack 2

Puede definir atributos ampliados para cualquier sensor y utilizarlos para almacenar los datos de los sensores de forma automática.

Para definir atributos ampliados para los sensores, cree un libro IdML con las definiciones de los atributos ampliados. El nombre de este libro debe ser `xa.xml`. Coloque el archivo en el directorio del paquete OSGi concreto, por ejemplo, `$COLLATION_HOME/osgi/plugins/com.ibm.cdb.discover.sensor.sys.examplesensor_1.0.0/xa.xml`.

Cada vez que se inicia el servicio TopologyBuilder, se cargan automáticamente los libros nuevos o modificados. Durante el proceso de carga, el programa de carga masiva se ejecuta con la opción `-loadEAMeta` habilitada, y se definen automáticamente los atributos ampliados. Para obtener más detalles acerca del programa de carga masiva, consulte “Atributos ampliados de definición automática para la API Java pública y para el programa de carga masiva”.

Para ver los atributos ampliados en la IU de TADDM, debe reiniciar el servidor de TADDM.

Si desea configurar el proceso de carga masiva de libros que contienen atributos ampliados, puede utilizar el archivo `$COLLATION_HOME/etc/bulkload.properties`, el cual almacena la configuración del programa de carga masiva.

Ejemplo de uso

El archivo `com.ibm.cdb.discover.sensor.sys.foobarsensor_1.0.0/xa.xml` contiene definiciones de atributos ampliados que utiliza el ejemplo FoobarSensor. Durante el primer inicio del servidor, una vez inicializado el servicio TopologyBuilder, TADDM carga el libro `com.ibm.cdb.discover.sensor.sys.foobarsensor_1.0.0/xa.xml`. A continuación, se calcula la suma CRC32 del archivo `xa.xml` y se almacena en el archivo `com.ibm.cdb.discover.sensor.sys.foobarsensor_1.0.0/xa.xml.crc`. La suma CRC32 se calcula cada vez que se modifica el archivo `xa.xml`. El servicio TopologyBuilder únicamente carga los libros cuando se modifica la suma de comprobación.

Estructura de ejemplo del archivo `xa.xml`

```
<idml:idml xmlns:idml="http://www.ibm.com/xmlns/swg/idml" xmlns:cdm=
"http://www.ibm.com/xmlns/swg/cdm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/xmlns/swg/idml idml.xsd">
 <idml:source IdMLSchemaVersion="0.8">
  <cdm:process.ManagementSoftwareSystem CDMSchemaVersion="2.8">
   <cdm:MSSName>your data</cdm:MSSName>
   <cdm:Hostname>your data</cdm:Hostname>
   <cdm:ManufacturerName>your data</cdm:ManufacturerName>
   <cdm:ProductName>your data</cdm:ProductName>
   <cdm:ProductVersion>your data</cdm:ProductVersion>
   <cdm:Label>your data</cdm:Label>
  </cdm:process.ManagementSoftwareSystem>
 </idml:source>

 <!-- Operation... -->
 <idml:operationSet opid="1">
  <idml:create timestamp="2012-07-12T05:10:58Z">
  <cdm:CDM-ER-Specification>
   <cdm:sys.ComputerSystem id="CS" sourceToken="CS">
   <cdm:extension>
       <cdm:extattr name="sensor_foobar_xa1" category="_internal"></cdm:extattr>
   </cdm:extension>
   </cdm:sys.ComputerSystem>
  </cdm:CDM-ER-Specification>
  </idml:create>
 </idml:operationSet>
</idml:idml>
```

Instancias ampliadas

En el modelo de datos comunes antiguo se tuvieron que dividir los fragmentos grandes de datos en los pares de clave-valor y almacenarse como atributos ampliados. Ahora puede utilizar las instancias ampliadas para almacenar una gran cantidad de contenido, gracias a que no necesita crear una gran cantidad de atributos ampliados. Puede almacenar objetos de nivel inferior como instancias ampliadas y adjuntarlos a objetos de nivel superior, que son CI, como datos sin formato o datos preprocesados.

Tipo de modelo de datos

La estructura del atributo `XD` es una secuencia simple de elementos XML. Cada tipo de objeto de modelo tiene el atributo `XD` de un nuevo tipo `ExtendedInstanceData` Java. Este tipo permite almacenar cualquier tipo de contenido, como por ejemplo texto sin formato (salida del mandato, XML, CSV o CDATA).

Puede agrupar instancias ampliadas por tipo; el tipo predeterminado es `General`.

Visualización de instancias ampliadas

Después de ejecutar un descubrimiento de un sensor, para el que ha especificado el atributo `XD`, puede ver instancias ampliadas en Data Management Portal. Abra el panel Detalles de un objeto descubierto. Cada tipo de instancia ampliada se visualiza en un separador aparte.

Ejemplo de uso en un sensor

En el ejemplo siguiente se muestra el tipo `ExtendedInstanceData` que se utiliza para almacenar contenido sin procesar. Las variables `lsofOutput` y `ifconfigOutput` son las variables de cadena que almacenan las salidas de mandato. En este ejemplo, las salidas no se procesan, pero pueden formatearse como JSON o XML.

```
        SComputerSystem scs = ModelFactory.newInstance(SComputerSystem.class);
        scs.setHierarchyDomain("sys.unix.linux");
        scs.setHierarchyType("RedHat");
        scs.setManufacturer("RedHat");
        scs.setModel("Linux");
        scs.setSerialNumber("as00123012");

        ExtendedInstanceData xd = new ExtendedInstanceData();
        xd.addInstance("lsof", lsofOutput);
        xd.addInstance("ipInterfaces", ifconfigOutput);
        xd.addInstance(null, "Linux vmw009128109120 2.6.32-220.el6.x86_64 #1
SMP Wed Nov 9 08:03:13 EST 2011 x86_64 x86_64 x86_64
GNU/Linux");
        scs.setXD(xd);
```

Como resultado, se almacena la siguiente correlación de atributos con el atributo `XD` rellenado con el XML correcto. El XML crea elementos para el tipo de instancia con los elementos de datos sin formato del mismo tipo, entre `<instance>`. La instancia creada con el tipo nulo se coloca en el tipo predeterminado `<general>`.

```
manufacturer -> RedHat
hierarchyType -> RedHat
XD -> <xml>
 <general>
  <instance>Linux vmw009128109120 2.6.32-220.el6.x86_64 #1 SMP Wed Nov 9 08:03:13
EST 2011 x86_64 x86_64 x86_64 GNU/Linux</instance>
 </general>
 <ipInterfaces>
  <instance>eth4      Link encap:Ethernet  HWaddr 00:50:56:00:72:92
          inet addr:9.128.109.120  Bcast:9.128.109.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe00:7292/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:805298636 errors:0 dropped:0 overruns:0 frame:0
          TX packets:665767802 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:590819760949 (550.2 GiB)  TX bytes:272393336858 (253.6 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:833223633 errors:0 dropped:0 overruns:0 frame:0
          TX packets:833223633 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:594042898379 (553.2 GiB)  TX bytes:594042898379 (553.2 GiB)

  </instance>
 </ipInterfaces>
```

```
<lsof>
 <instance>pickup    31017   postfix  mem       REG                8,2  1580928
   1066619 /usr/lib64/mysql/libmysqlclient.so.16.0.0
             pickup    31017   postfix  mem       REG                8,2   184088
    139800 /lib64/libpcre.so.0.0.1
             pickup    31017   postfix  mem       REG                8,2    63304
    139295 /lib64/liblber-2.4.so.2.5.6
             pickup    31017   postfix  mem       REG                8,2   308912
    130952 /lib64/libldap-2.4.so.2.5.6
             pickup    31017   postfix  mem       REG                8,2   156872
    130819 /lib64/ld-2.12.so
             pickup    31017   postfix    0u      CHR                1,3      0t0
      3640 /dev/null
             pickup    31017   postfix    1u      CHR                1,3      0t0
      3640 /dev/null
             pickup    31017   postfix    2u      CHR                1,3      0t0
      3640 /dev/null
             pickup    31017   postfix    3r     FIFO                0,8      0t0
     10751 pipe
             pickup    31017   postfix    5u     unix 0xffff8802350d9c80      0t0
     10659 socket
             pickup    31017   postfix    6u     FIFO                8,2      0t0
    392456 /var/spool/postfix/public/pickup
             pickup    31017   postfix    7u     unix 0xffff8802378e8680      0t0
  49305400 socket
             pickup    31017   postfix    8u      REG                0,9        0
      3638 anon_inode
             loop0     31473      root  cwd       DIR                8,2     4096
         2 /
             loop0     31473      root  rtd       DIR                8,2     4096
         2 /
             loop0     31473      root  txt   unknown
          /proc/31473/exe
 </instance>
</lsof>
</xml>
serialNumber -> as00123012
isPlaceholder -> false
hierarchyDomain -> sys.unix.linux
model -> Linux
```

En el ejemplo anterior se utilizan los métodos `addInstance` más simples. Se recomienda utilizar el método `addInstance(String type, INSTANCE_FORMAT format, boolean isVisible, String content)`, que permite controlar dos atributos XML del elemento `<instance>`.

El primer atributo XML es `format`, que se utiliza para interpretar el contenido. `INSTANCE_FORMAT` es una enumeración Java con los siguientes valores posibles: `plain`, `XML`, `CSV`, `JSON` y `CDATA`.

El segundo atributo XML es `visible`, que se utiliza cuando no se visualizan algunas instancias en la interfaz de usuario.

Ejemplo:

```
<general><instance format="plain" visible="false">Linux vmw009128109120
2.6.32-220.el6.x86_64 #1 SMP Wed Nov 9 08:03:13 EST 2011 x86_64 x86_64 x86_64
GNU/Linux</instance></general>
```

Actualización parcial

Un CI se puede almacenar con los mismos valores de reglas de denominación de distintos orígenes de datos; como resultado, los valores se fusionan en un objeto. Para evitarlo, se utiliza el mecanismo de actualización parcial para fusionar dos

atributos XD con formato XML diferentes. Por ejemplo, si un origen se almacena con el objeto 1 y otro origen se almacena con el objeto 2, se crea un CI que retiene el atributo fusionado.

Objeto 1

```
        SComputerSystem xd1 = ModelFactory.newInstance
(SComputerSystem.class);
        xd1 = ModelFactory.newInstance(SComputerSystem.class);
        xd1.setHierarchyDomain("sys.unix.linux");
        xd1.setHierarchyType("RedHat");
        xd1.setManufacturer("RedHat");
        xd1.setModel("Linux");
        xd1.setSerialNumber("as00123012xd");

        ExtendedInstanceData xdone = new ExtendedInstanceData();
        xdone.addInstance("lsof", "content from CI1");
        xdone.addInstance(null, "content from CI1");
        xd1.setXD(xdone);
```

Objeto 2

```
        SComputerSystem xd2 = ModelFactory.newInstance
(SComputerSystem.class);
        xd2 = ModelFactory.newInstance(SComputerSystem.class);
        xd2.setHierarchyDomain("sys.unix.linux");
        xd2.setHierarchyType("RedHat");
        xd2.setManufacturer("RedHat");
        xd2.setModel("Linux");
        xd2.setSerialNumber("as00123012xd");

        ExtendedInstanceData xdtwo = new ExtendedInstanceData();
        xdtwo.addInstance("ips", "content from CI2");
        xdtwo.addInstance(null, "content from CI2");
        xd2.setXD(xdtwo);
```

Atributo fusionado

```
manufacturer -> RedHat
hierarchyType -> RedHat
XD -> <xml>
 <general>
  <instance>content from CI2</instance>
 </general>
 <lsof>
  <instance>content from CI1</instance>
 </lsof>
 <ips>
  <instance>content from CI2</instance>
 </ips>
</xml>
serialNumber -> as00123012xd
isPlaceholder -> false
hierarchyDomain -> sys.unix.linux
model -> Linux
```

Todos los atributos que no se solapan del primer y segundo objeto están presentes con sus respectivos valores en el atributo fusionado, y el tipo general tiene las instancias del último objeto almacenado. La actualización parcial del atributo XD de cada objeto de modelo se efectúa con respecto a un tipo de instancia.

Consultas MQL con el operador EVAL (solo atributos XA y XD)

Muchos atributos del CDM se han movido al contenido XML de los atributos XA o XD. Por lo tanto, la sintaxis MQL admite un nuevo operador, eval, que se puede añadir a la cláusula where. El operador eval permite consultar los CI por los valores de atributos ampliados o de instancias ampliado.

Nota: Todos los ejemplos de consultas de MQL contienen comillas de escape (\"*value*\") porque se supone que las consultas se ejecutan de la siguiente forma:

```
./api.sh -u username -p password find "MQL query"
```

Por ejemplo, la siguiente consulta se ha ejecutado para buscar un sistema con el atributo `productID` definido como `'prod1'`:

```
SELECT * FROM ComputerSystem WHERE productID == 'prod1'
```

La siguiente consulta equivalente utiliza el operador `eval`:

```
SELECT * FROM ComputerSystem WHERE XA eval '/xml
[attribute[@category=\"taddm_global\" and @name=\"productID\"]=\"prod1\"]'
```

El operador `eval` puede ir seguido de cualquier expresión XPath válida que devuelva el valor booleano true o false para permitir que la capa de persistencia realice correctamente el filtrado del rendimiento de SQL.

Más ejemplos

- Busque todos los ComputerSystems que tengan cualquier atributo ampliado con el valor `val`:
  - MQL:

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml[attribute=\"val\"]'
    ```

  - SQL:

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml[attribute="val1"]'
    passing compsys.xa_x as "c")
    ```

- Busque un ComputerSystem con el atributo ampliado `attr2` que tenga la categoría establecida en `Other` y el valor en `two`:
  - MQL:

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml/attribute
    [@name=\"attr2\" and @category=\"Other\" and text()=\"two\"]'
    ```

  - SQL:

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml/attribute[@name="attr2"
     and and @category="Other" and text()="two"]' passing compsys.xa_x as
     "c")
    ```

Ampliación del ámbito de descubrimiento del sensor con el modelo simplificado

Puede ampliar fácilmente el ámbito de descubrimiento de un sensor utilizando las nuevas funciones del modelo simplificado. En el siguiente procedimiento se muestra una configuración de ejemplo en la que puede basar la personalización de sus sensores.

Acerca de esta tarea

En el siguiente procedimiento se muestra la configuración de un sensor simple basado en script para la base de datos de MongoDB al instalar el servidor de MongoDB.

Requisitos previos

Si desea que el sensor almacene atributos ampliados, primero debe definirlos para un tipo de modelo de datos determinado. Para obtener instrucciones detalladas, consulte "Creación de atributos ampliados en el Portal de gestión de dominios" en la página 27.

Para este procedimiento se han creado dos atributos ampliados con los parámetros siguientes:

- Tipo SDeployableComponent:
  - Nombre de atributo ampliado: hasDatabaseIndexDef
  - Tipo de atributo ampliado: Boolean
  - Categoría de atributo ampliado: General
- Tipo SSoftwareServer:
  - Nombre de atributo ampliado: DatabaseCnt
  - Tipo de atributo ampliado: Integer
  - Categoría de atributo ampliado: Markers

Nota: En el siguiente procedimiento no se proporcionan detalles de las nuevas funciones del modelo simplificado. En su lugar, muestra cómo utilizar dichas funciones. Para obtener más información, consulte los temas adecuados en la sección "Modelo simplificado" en la página 18 de la documentación de TADDM.

Procedimiento

1. Crear una plantilla de servidor personalizado mediante el panel Servidores personalizados de la consola de Discovery Management. El panel Servidores personalizados contiene plantillas de los sensores que se pueden ejecutar mediante un sensor de servidores genéricos o como sensor de servidor de aplicaciones personalizado.
   Abra los detalles del servidor de MongoDB. Este se habilita. Almacena objetos del tipo AppServer. En la sección Criterios de identificación debe especificar un nombre de archivo. Cuando el sensor de servidor genérico encuentra un proceso que contiene este nombre de archivo, ejecuta el sensor personalizado. En este caso, el nombre de archivo es mongod.exe.

   Ahora puede ejecutar un descubrimiento, pero solo se descubrirán los objetos RuntimeProcess.
2. Configurar la ampliación para el sensor de plantilla.
   a. Abra el directorio de inicio de TADDM y vaya al directorio dist/etc/templates/commands. Cree un archivo de mandatos con el mismo nombre que la plantilla definida en Servidores personalizados, en este caso MondoDB. El contenido de este archivo se ejecuta al ejecutarse el sensor personalizado.
      El archivo MongoDB contiene la siguiente línea:

      ```
      SCRIPT[com.ibm.cdb.core.jython253_2.5.3]:etc/templates/commands/
      extension-scripts/mongodb.py
      ```

      Significa que el script mongodb.py se ejecuta al ejecutarse el sensor.
   b. Configurar el script mongodb.py. En las siguientes secciones se muestra una descripción breve de los elementos principales de este script.

      El inicio predeterminado del sensor

      ```
      (os_handle, result, appserver, seed, log, env) = sensorhelper.init
      (targets)
      ```

      En esta sección se especifican los parámetros que pasa la plataforma targets:

      - os_handle: un objeto de sistema operativo.
      - result: el resultado donde se adjuntan los objetos.
      - appserver: el objeto de servidor de aplicaciones que representa los procesos de tiempo de ejecución que coinciden con la condición de la plantilla.
      - seed: el objeto semilla que desencadena el sensor.

- `log`: registrador.
- `env`: valores de entorno.

Creación del objeto principal: definición de clase y atributos de jerarquía

```
mdbserver = ModelFactory.newInstance(Class.forName('com.collation.
platform.model.topology.simple.SSoftwareServer'))
mdbserver.setHierarchyDomain('app.db.mongodb')
mdbserver.setHierarchyType('MongoDBServer')
```

En esta sección se utiliza un tipo nuevo de clase: `SSoftwareServer`. También hay atributos nuevos, que posicionan un CI que se representa mediante este objeto en las correlaciones de los dominios. Estos atributos proporcionan los detalles necesarios del sensor.

- `HierarchyDomain`: especifica niveles de dominios. En este caso, en el valor `app.db.mongodb`, `app` es un servidor de aplicaciones, `db` es un servidor de base de datos y `mongodb` es una base de datos específica.
- `HierarchyType`: especifica el tipo del objeto. En este caso, `mongodb`, que se utiliza en el atributo `HierarchyDomain`, representa un servidor de bases de datos, por lo que el atributo `HierarchyType` se establece en `MongoDBServer`.

`OpenId` - a new generic naming rule attribute

```
print 'Primary IP : ' + seed.getPrimaryIpAddress().
getStringNotation()
print 'Port       : ' + str(seed.getPort())
id = OpenId().addId('IP' , seed.getPrimaryIpAddress().
getStringNotation()).addId('port' , str(seed.getPort()))
ID = OpenId.generateDisplayName(id)
mdbserver.setInstanceID(ID)
mdbserver.setOpenId(id)
mdbserver.setHostComputerSystem(os_handle.getComputerSystem())
```

Debe dar un nombre al objeto que ha creado. Para hacerlo, debe definir una regla de denominación y especificar atributos que proporcionen valores exclusivos. Para los servidores, la regla de denominación suele basarse en dos atributos. Son el punto de acceso a servicio primario que se crea a partir de la dirección IP primaria del servidor y un puerto que escucha este servicio.

El nuevo atributo `OpenId` es un derivador de un atributo de serie que recopila todos los valores utilizados para el cálculo del GUID. Este nuevo atributo organiza los valores en el mismo orden cada vez. El método `addId` añade objetos y los establece como el valor del atributo `OpenId`.

También hay otros atributos predefinidos establecidos para este sensor, como por ejemplo el ID de instancia o el sistema principal para conectar el servidor a un sistema pasado por una plataforma como atributo de objeto de sistema operativo.

Definición del objeto de resultado

```
print 'Setting mongodb server'
result.setServer(mdbserver)
```

Defina el objeto de resultado com el servidor de MongoDB. Si el sensor falla durante el descubrimiento, como mínimo se almacenará esta información.

Extracción de la función de base de datos

```
def extractDatabases(mgbserver, dbListLines):
  databases = list([])
  dbDir = None
  currentDatabase = None
  XD = None
  XA = None

  for dbLine in dbListLines:
    print 'Processing line : ' + dbLine
    if "DATABASE" in dbLine :
      print 'Found a database ' + dbLine
      #Create physical files
      print 'Attach as files'
      dbLineTokens = dbLine.split(" ")
      dbName = dbLineTokens[2].strip()
      dbDir = dbLineTokens[4].strip()
      print "Database name " + dbName
      print "Database directory " + dbDir

      #Create deployable components
      print "Create deployable component for database"
      database = ModelFactory.newInstance(Class.forName
('com.collation.platform.model.topology.simple.
SDeployableComponent'))
      database.setHierarchyDomain('app.db.mongodb')
      database.setHierarchyType('Database')
      database.setOpenId(OpenId().addId("databaseName", dbName))
      database.setName(dbName)
      database.setSoftwareServer(mgbserver)
      databases.append(database)
      currentDatabase = database
      XA = ExtendedAttributesData()
      XA.attachTo(currentDatabase)
      XD = ExtendedInstanceData()
      currentDatabase.setXD(currentXD)

    elif dbDir is not None and dbDir in dbLine and "Metadata for"
in dbLine:
      print "Found file " + dbLine + " to read for XD"
      databaseConfFileLineTokens = dbLine.split(" ")
      fileType = databaseConfFileLineTokens[3]
      fileName = databaseConfFileLineTokens[5]
      print "Instance type " + fileType
      print "Files content to read " + fileName

      if not fileName.endswith("bson"):
        cnfFileContent = readFileContent(fileName)
        print "Read content : " + cnfFileContent
        XD.addInstance(fileType,
ExtendedInstanceData.INSTANCE_FORMAT.json, 1, cnfFileContent)
        print "Added instance"

        if "index" in cnfFileContent :
          XA.addAttribute("hasDatabaseIndexDef", "true")
          xml = XA.toXML()
          print "Add XA marker for database " + dbName +
" with XML " + xml

  print "Attach files and deployable components"
  mgbserver.setDeployedComponents(tuple(databases))
  XA = ExtendedAttributesData()
  XA.addAttribute("Markers", DatabasesCnt", str(len(databases)))
  XA.attachTo(mgbserver)
  print "Add XA marker for server with xml " + xml
```

Esta función utiliza dos contenedores para los datos. Uno contiene atributos ampliados y el otro, instancias ampliadas.

En la sección `#Creación de componentes desplegables`, el sensor crea un objeto para la base de datos descubierta mediante el tipo de clase `SDeployableComponent`. `HierarchyDomain` se establece en `app.db.mongodb` porque se trata del mismo dominio, pero el atributo `HierarchyType` se define en `Database`. La regla de denominación se basa en el nombre de la base de datos, por lo que el atributo `openId` se define en `databaseName`. Un nombre de atributo predefinido se define (`database.setName(dbName)`) y el componente se conecta (`databases.append(database)`) a un servidor con el atributo de servidor de software (`database.setSoftwareServer(mgbserver)`).

Lo más importante de esta sección son los dos contenedores de datos. Estos contenedores se crean para la base de datos recién creada. El atributo `XA` contiene el objeto `ExtendedAttributesData` para los atributos ampliados, y el atributo `XD` contiene el objeto `ExtendedInstanceData` para grandes cantidades de datos.

`ExtendedAttributesData`

Mediante el atributo `XA = ExtendedAttributesData()` el sensor crea un objeto `ExtendedAttributesData` y lo añade a un CI. Mediante el método `XA.addAttribute`, añade los atributos a este objeto, que conforman el nombre y el valor. Por ejemplo, en el atributo ampliado del objeto de servidor, el sensor crea el objeto `ExtendedAttributesData` y añade los atributos, que son `DatabaseCnt` y la categoría `Markers`. Para convertir el atributo en un diseño que se pueda almacenar, el sensor llama al método `toXML`.

`ExtendedInstanceData`

Mediante el atributo `XD = ExtendedInstanceData()`, el sensor crea un objeto `ExtendedInstanceData`. En este sensor, el atributo `XD` se utiliza para almacenar el contenido de los archivos que representan la estructura interna de la base de datos. El contenido de la base de datos se lee mediante la función `readFileContent` que se define del siguiente modo:

```
def readFileContent(fileName):
  print "Open file to read : " + fileName
  f = open(fileName, 'r')
  content = ""
  for line in f:
    content = content + line + "\n"
  return content
```

El siguiente método se utiliza para almacenar los datos:

```
XD.addInstance(fileType, ExtendedInstanceData.
INSTANCE_FORMAT.json, 1, cnfFileContent)
```

Este método `addInstance` determinado contiene el siguiente parámetro:

- `fileType`: especifica el tipo de instancia.
- `ExtendedInstanceData.INSTANCE_FORMAT.json`: especifica el formato de instancia.
- 1: significa "true" y especifica que el contenido es visible en el panel de detalles.
- `cnfFileContent`: especifica el contenido de la instancia.

Qué hacer a continuación

- Puede ejecutar el descubrimiento de MongoDB con el ámbito actualizado.
- Puede ver los resultados del descubrimiento en Data Management Portal, en el portal Resumen de inventario. También puede examinar carpetas genéricas en el panel Componentes descubiertos y visualizar los detalles de objetos descubiertos.
- Puede crear aplicaciones empresariales para el objeto que almacena y visualizar la topología.
- Puede ver más ejemplos del script Jython.

Más ejemplos del script Jython

Si desea descubrir los atributos ampliados que ha definido para la clase ComputerSystem (SComputerSystem) o AppServer (SSoftwareServer), consulte los ejemplos siguientes del script Jython.

Nota: Si está modificando el script existente, por ejemplo, el script de ejemplo `ea_sensor_1.0.py`, sobrescriba la parte principal del archivo. Asimismo, la sección Standard Library Imports debe contener la línea `from com.collation.platform.model.util.ea import ExtendedAttributesData`, como en el ejemplo siguiente:

```
import sys
import java

from java.lang import System
from com.collation.platform.model.util.ea import ExtendedAttributesData
```

Script Jython que se utiliza para ampliar una plantilla del sistema: la sección principal

```
try:
 (os_handle, result, computerSystem, seed, log) = sensorhelper.
init(targets)

 patchVersionOut = sensorhelper.executeCommand("tail -1 /etc/
patch_status | cut –f2 –d,")
 patchDateOut = sensorhelper.executeCommand("tail -1 /etc/patch_
status | cut –f3 –d,")

 if patchVersionOut != None:
  XA = ExtendedAttributesData()
  XA.addAttribute("patchVersion", patchVersionOut)
  XA.addAttribute("patchDate", patchDateOut)
                XA.attachTo(computerSystem)
 else:
  log.info("patchVersion command return no output")
except:
 LogError("unexpected exception getting patchVersion
information")
```

Script Jython que se utiliza para ampliar una plantilla del servidor personalizado: la sección principal

```
try:
 (os_handle, result, appServer, seed, log, env) = sensorhelper.
init(targets)

 patchVersionOut = sensorhelper.executeCommand("tail -1 /etc/
patch_status | cut –f2 –d,")
 patchDateOut = sensorhelper.executeCommand("tail -1 /etc/patch_
status | cut –f3 –d,")

 if patchVersionOut != None:
  XA = ExtendedAttributesData()
  XA.addAttribute("patchVersion", patchVersionOut)
```

```
  XA.addAttribute("patchDate", patchDateOut)
  XA.attachTo(appServer)
 else:
  log.info("patchVersion command return no output")
except:
 LogError("unexpected exception getting patchVersion
information")
```

Visión general de la API de TADDM

Las interfaces de programación de aplicaciones (API) de TADDM permiten acceder a todos los datos de descubrimiento que se muestran en Data Management Portal. En este tema se describen las principales API de TADDM: la API de Java, la API de SOAP, la API de REST y la API de la interfaz de línea de mandatos.

Visión general de la interfaz de programación de aplicaciones

Para acceder a datos de descubrimiento, utilice tipos específicos de la interfaz de programación de aplicaciones (API).

Se puede acceder a todos los datos de descubrimiento mediante los siguientes tipos de API:

API de Java
La API completa de TADDM, que habilita la integración y el desarrollo de aplicaciones Java.

API de SOAP
Expone los elementos de la API de TADDM como servicio web del Protocolo de acceso a objectos simple (protocolo SOAP).

API de REST
Expone los elementos de la API de TADDM como servicio web RESTful.

API de interfaz de línea de mandatos
Proporciona un derivador alrededor de la API de Java que habilita el acceso desde la línea de mandatos para la creación de scripts, la personalización simple y la planificación.

Visión general del esquema XML

El esquema XML de TADDM aplana la estructura jerárquica del modelo de datos común en un documento XML, con la mayoría de los objetos contenidos incluidos en el documento.

Los métodos de la API de TADDM devuelven un documento XML que contiene una lista de objetos especificados por la consulta de Model Query Language (MQL), si corresponde. Este documento es más grande que los datos originales, pero resulta más fácil realizar búsquedas utilizando herramientas como XQuery o XPath.

El SDK de TADDM representa las dependencias entre objetos utilizando objetos de dependencia independientes, lo que conecta a los proveedores con servicios dependientes mediante ID de objetos.

Al formatear los documentos XML para utilizarlos con el SDK de TADDM, tenga en cuenta lo siguiente:

- XML es un modelo jerárquico y no permite ciclos.
- property_nameX es un ModelObject.

- abbreviated_searched_class_name es el nombre de clase buscado, no el nombre de clase real.
- xsi:type y GUID son atributos XML y no se representan como elementos independientes.
- Matriz: si el elemento forma parte de una matriz, N es su índice.

En la tabla siguiente, se describe la estructura del documento XML:

Tabla 6. Estructura del documento XML

XML	Descripción
`<?xml version="1.0" encoding="UTF-8"?>`	Cabecera

`<results`
`xmlns="urn:www-collation-com:1.0"`
`xmlns:coll="urn:www-collation-com:1.0"`
`xmlns:xsi="http://www.w3.org/2001/`
`XMLSchemainstance"`
`xsi:schemaLocation=`
`"urn:www-collation-com:1.0`
`urn:www-collation-com:1.0/results.xsd">`
	Nodo de resultados de nivel superior, incluida la especificación del espacio de nombres XML

`<abbreviated_searched_class_name`
`xsi:type="full class name"`
`array="N"`
`LINK="guid"`
`lastModified="Last Modified Time in ms"`
`guid="unique model object id">`
	Atributos de clase, como la hora de la última modificación y el ID de objeto exclusivo

`<property_name1>"text value"`
`</property_name1>`
`...`
`<property_nameX`
`xsi:type="full class name"`
`guid="unique model object id">`
`<property_name1>"text`
`value"</property_name1>`
`...`
`<property_nameN>"text`
`value"</property_nameN>`
`<property_nameX/>`
`...`
`<property_nameN>"text value"`
`</property_nameN>`
`<property_nameQ/>`
	Valores de atributo y objetos incluidos

`</abbreviated_searched_class_name>`	Fin de los valores
`</results>`	Fin de los resultados

Visión general del formato JSON

La API de REST utiliza el formato JSON para devolver los datos que representan objetos de modelo; puede solicitar la salida de JSON especificando el parámetro `feed=json` en una llamada a la API de REST.

En la tabla siguiente, se describe la estructura del formato JSON utilizado para representar objetos de modelo.

Elemento JSON	Descripción
[	Empieza una matriz de objetos modelados.
{	Empieza un objeto de modelo, que puede contener de cero a varios objetos de modelo.
"*nombre*":*valor*, "*nombre*":*valor*	Uno o varios pares nombre-valor, separados por comas.
"_class":"*nombre_clase*"	Par nombre-valor que contiene el nombre del objeto de modelo. El nombre de clase del objeto de modelo puede presentar cualquiera de estos formatos: • Nombre abreviado (por ejemplo, ComputerSystem) • Nombre completo (por ejemplo, *com.collation.platform.model.topology.sys.ComputerSystem* Si especifica `longClassName=true` en una consulta de REST, todos los valores devueltos para `_class` contendrán el nombre de modelo completo. En caso contrario, el nombre abreviado se devolverá, a menos que no sea exclusivo (en cuyo caso se devolverá el nombre completo).
}	Finaliza un objeto de modelo.
]	Finaliza una matriz de objetos de modelo.

En el ejemplo siguiente se muestra la salida de JSON desde una consulta ComputerSystem con profundidad de 1:


```
[{
"displayName":"esx3-vm16-rhes4",
"devices":[{"_class":"DiskDrive","guid":"2A2827686EB03465A955DE54BD3F6AB5"},
{"_class":"DiskDrive","guid":"D7DAF9DCD1E7347684A0D02E36E212DC"}],
"lastModifiedBy":"system",
"l2Interfaces": [{"_class":"L2Interface","guid":"FA048919AA953BA5A09580496017A776"},
{"_class":"L2Interface","guid":"297B125690B33B778C347E12CFC62689"}],
"createdBy":"system",
"_class":"LinuxUnitaryComputerSystem",
"controllers":[{"_class":"Controller","guid":"7B72D3B5448D30388F9D9497EA8F970D"},
{"_class":"Controller","guid":"B619ABB8B8343C1FAB5BF87AD425559E"}],
"guid":"C2D379A936433258BABBF682A8E71A82",
"CPUSpeed":3191000000,
"fqdn":"esx3-vm16-rhes4",
"contextIp":"9.43.73.87",
"OSInstalled":[{"_class":"Linux","guid":"04BFCBCD2A1733258F5C95CD281D91AF"}],
"memorySize":3988783104,
"ipInterfaces":[{"_class":"IpInterface","guid":"9CAA8E0197333BAD924EA3CCB1860920"},
{"_class":"IpInterface","guid":"C2E6D21CF24435EABCB8AA8136BB9F1B"}],
"signature":"9.43.73.87(000C29A467A9)",
"systemId":"2b095749",
"bidiFlag":3,
"name":"esx3-vm16-rhes4",
"OSRunning":{"_class":"Linux","guid":"04BFCBCD2A1733258F5C95CD281D91AF"},
"CPUType":"Intel(R) Xeon(TM) MV",
"type":"ComputerSystem",
"numCPUs":1,
"architecture":"i686",
"fileSystems":[{"_class":"UnixFileSystem","guid":"CDA94FB8C84B300ABA2A42E1EFEE6234"},
{"_class":"NFSFileSystem","guid":"6300742848BA39478EAEE4FB4709DF7A"}],
"lastModifiedTime":1225806427541
}]
```


Visión general de Model Query Language

El mandato **`find()`** de la API de TADDM acepta una cadena de consulta, especificada mediante Model Query Language (MQL). MQL actúa como filtro para limitar los objetos seleccionados.

MQL utiliza una sintaxis parecida a la de SQL para especificar la clase de objeto de modelo u otros orígenes de datos, sus atributos, junto con una expresión de filtro.

La sintaxis de una consulta MQL es la siguiente:

```
SELECT attribute-list FROM data-sources [ WHERE expression ]
```

Tabla 7 describe los elementos de una consulta MQL. MQL no distingue entre mayúsculas y minúsculas.

Tabla 7. Elementos de consulta MQL

Elemento	Descripción
attribute-list	El valor *, o una lista separada por comas de atributos de la clase ModelObject de origen. También se pueden especificar atributos incluidos. Un nombre de atributo empieza siempre con minúscula, a menos que la primera y la segunda letras sean mayúsculas. Las letras siguientes del nombre de atributo pueden ser mayúsculas o minúsculas. Entre los ejemplos de nombres de atributo, se incluyen los siguientes: • displayName • fqdn • OSRunning
data-sources	Lista separada por comas de nombres de clase de objeto de modelo. Estas clases deben ser persistentes, dado que no puede consultar objetos no persistentes.

expression	La expresión de filtro, expresada con el formato siguiente:
	`member-name OP expression [ ... ]`
	
	donde:
	• Member-name es un atributo del origen de datos seleccionado y puede incluir miembros separados por puntos (las clases de miembro especificadas en la expresión de consulta deben ser persistentes. Puede consultar los atributos de miembro para obtener los valores que coincidan con la expresión de búsqueda).
	• OP es un operador.
	• Expression es una sentencia que devuelve un valor.
	
	Por ejemplo:
	`SELECT *`
	`FROM ComputerSystem`
	`WHERE OSRunning.OSName == 'Linux'`
	
	En este caso, OSRunning es un objeto OperatingSystem al que hace referencia ComputerSystem (que es un objeto persistente) y OSName es un miembro primitivo.
	
	Consulte Tabla 8 en la página 44 para obtener una descripción de los operadores y la prioridad asociada.

Tabla 8 en la página 44 describe la prioridad del operador MQL, con valores mayores para representar una mayor prioridad.

Tabla 8. Prioridad de operador MQL

Señal	Operador	Prioridad
or	OR lógico	1
and	AND lógico	2
instanceof	es una instancia de	3
==	es igual a	3
!=	no es igual a	3
>	mayor que	3
>=	igual o mayor que	3
<	menor que	3
<=	igual o menor que	3
empieza por	empieza por	4
acaba en	acaba en	4
es igual a	es igual a	4
not-equals	no es igual a	4
is-null	es nulo	4
is-not-null	no es nulo	4
en	en	5
()	paréntesis	5
exists	matriz contiene	5
upper()	función	5
lower()	función	5
!	no unario	5
.	selección de punto	6
contiene	contiene	3
eval	eval	3

MQL no admite los siguientes operadores o características de SQL SELECT:
- GROUP BY
- HAVING
- DISTINCT
- nested SELECTs
- BETWEEN
- Aggregates

Puede especificar el operador lógico AND como and, AND o &&, y el operador lógico OR como or, OR o ||. Además, tiene que colocar todas las series entre comillas simples, por ejemplo, 'IBM'.

Uniones

MQL admite las uniones internas izquierdas en objetos de modelo, como muestra el ejemplo siguiente:

```
SELECT Db2Server.* FROM Db2Server, OracleInstance WHERE Db2Server.port == OracleInstance.port
```

Esta unión devuelve todos los objetos de modelo Db2Server en aquellos casos en los que el número de puerto de Db2Server y OracleInstance son iguales. MQL no

admite las combinaciones de uniones exteriores derechas, exteriores izquierdas, completas o cruzadas

Limitaciones

En DB2 versión 9.5, los operadores `equals` y `not-equals` fallan cuando se ejecutan en atributos del tipo de datos CLOB en la base de datos. Se genera la excepción siguiente:

```
com.ibm.db2.jcc.am.SqlSyntaxErrorException: DB2 SQL Error: SQLCODE=-418,
SQLSTATE=42610
```

Gramática de la sentencia SELECT

En el ejemplo siguiente, se muestra la gramática de la sentencia SELECT. Consulte Javadoc para obtener más información y las últimas actualizaciones.

```
statement := SELECT attribute-list [EXCLUDING attribute-list]
   FROM [ONLY] class_list { WHERE [expression |
   exists_expr] }
[ FETCH FIRST n { ROW | ROWS } [ ONLY ]] [ ORDER BY order_list ]
attribute_list := attrib {, attrib}* | *
class_list := domain_class {, domain_class }*
class := <a model object class>
exists_expr := exists( array_attrib op value {logical_op array_attrib op value}* )
expression := [ attrib op value | attrib post-op | pre-op ( attrib )
   |[ NOT ] IN ( expression [, ...] ){logical_op expression}*
value := <data value>
in_expr := [ NOT ] IN ( expression [, expression ... ])
array_attrib := <series of attributes where at least the second to last
   attribute is an array >
op := != | == | > | < | >= | <= | contains | starts-with | ends-with |
   equals | not-equals | instanceof | eval
logical_op := AND | OR | && | ||
post_op := is-null | is-not-null
pre_op := lower | upper
attrib := {class .} [<an attribute of a class>{.embedded_attribute} | * ]
embedded_attribute := [<embedded attribute>{.embedded_attribute} | *]
domain_class := {domain_list} class
domain_list := domain {, domain}* :
domain := <the server from which to pull data from, default: local database>
order_list := attrib [ ASC | ASCENDING | DESC | DESCENDING ] [ , order_list ]
```

Los atributos pueden contener caracteres comodín. Además, todas las palabras clave, como SELECT, FROM y WHERE no distinguen entre mayúsculas y minúsculas.

Nota: El `valor` puede ser del tipo `attrib` si es el operador básico `op` (!=|==|>|<|>=|<=) el que se utiliza.

Ejemplos

En el ejemplo siguiente se muestra una consulta MQL que sirve de filtro para los sistemas informáticos que ejecutan el sistema operativo Linux:

```
SELECT *
   FROM ComputerSystem
   WHERE OSRunning.OSName == 'Linux'
```

La consulta siguiente utiliza el operador EXISTS para consultar la pertenencia a una matriz, que dará como coincidencia todos los sistemas con una interfaz que escuche en ibm.com o cuya máscara de red esté definida en 255.255.255.0:

```
SELECT *
 FROM ComputerSystem
 WHERE EXISTS (ipInterfaces.ipNetwork.name ends-with '.ibm.com'
  OR ipInterfaces.ipNetwork.netmask == '255.255.255.0')
```

La consulta siguiente selecciona todos los sistemas informáticos cuyo atributo virtual se ha definido en true:

```
SELECT *
   FROM ComputerSystem
   WHERE virtual
```

La consulta siguiente selecciona todos los sistemas informáticos cuyo atributo virtual esté definido en false:

```
SELECT *
   FROM ComputerSystem
   WHERE not virtual
```

La consulta siguiente selecciona todos los sistemas operativos que tenga el atributo de servicio instalado (installed service) cuyo nombre contenga "Wireless". Dado que el atributo installed service está disponible solo en el sistema operativo Windows, debe utilizar una unión.

```
SELECT OSInstalled
 FROM ComputerSystem,WindowsOperatingSystem
 WHERE ComputerSystem.guid==WindowsOperatingSystem.parent.guid
 AND
 EXISTS(WindowsOperatingSystem.installedServices.displayName contains 'Wireless')
```

La siguiente consulta selecciona todos los AppServers con el atributo `primarySAP` que tengan un puerto especificado como su valor:

```
SELECT primarySAP.portNumber,displayName
    FROM AppServer
    WHERE primarySAP.portNumber==9084
```

La siguiente consulta selecciona todos los RuntimeProcesses que entre sus puertos tengan el puerto 1415. La consulta debe utilizar el operador EXISTS porque el atributo `ports` para RuntimeProcess es un atributo de matriz.

```
SELECT ports.portNumber,displayName
   FROM RuntimeProcess
   WHERE EXISTS (ports.portNumber==1415)
```

Consultas MQL con el operador NOT EQUAL TO (!=)

Las consultas con el operador NOT EQUAL TO no devuelven resultados que contengan un atributo que esté definido como NULL si dicho resultado se evalúa como "UNKNOWN".

Ejemplo

A menudo se presupone que el número de resultados que se devuelven desde el siguiente mandato find de la API:

```
./api.sh -u <admin> -p <pass> find --count "select * from ComputerSystem"
```

equivale a los resultados que se devuelven desde los dos siguientes mandatos find de la API:

```
./api.sh -u <admin> -p <pass> find --count "select * from
ComputerSystem where manufacturer == 'IBM'"
```

```
./api.sh -u <admin> -p <pass> find --count "select * from
ComputerSystem where manufacturer != 'IBM'"
```

Sin embargo, el atributo manufacturer se establece en NULL, por lo que se excluye de los resultados que devuelve la consulta que contiene el operador NOT EQUAL TO.

Las consultas que contienen el operador NOT EQUAL TO pueden tener las siguientes formas:

```
select * from ComputerSystem where manufacturer != 'IBM'
select * from ComputerSystem where not(manufacturer == 'IBM')
```

Si desea seleccionar todos los ComputerSystems cuyo fabricante (manufacturer) no sea IBM, utilice la siguiente consulta:

```
select * from ComputerSystem where manufacturer != 'IBM'
or manufacturer is-null
```

Consultas MQL con el operador EVAL (solo atributos XA y XD)

Muchos atributos del CDM se han movido al contenido XML de los atributos XA o XD. Por lo tanto, la sintaxis MQL admite un nuevo operador, eval, que se puede añadir a la cláusula where. El operador eval permite consultar los CI por los valores de atributos ampliados o de instancias ampliado.

Nota: Todos los ejemplos de consultas de MQL contienen comillas de escape (\"*value*\") porque se supone que las consultas se ejecutan de la siguiente forma:

```
./api.sh -u username -p password find "MQL query"
```

Por ejemplo, la siguiente consulta se ha ejecutado para buscar un sistema con el atributo productID definido como 'prod1':

```
SELECT * FROM ComputerSystem WHERE productID == 'prod1'
```

La siguiente consulta equivalente utiliza el operador eval:

```
SELECT * FROM ComputerSystem WHERE XA eval '/xml
[attribute[@category=\"taddm_global\" and @name=\"productID\"]=\"prod1\"]'
```

El operador eval puede ir seguido de cualquier expresión XPath válida que devuelva el valor booleano true o false para permitir que la capa de persistencia realice correctamente el filtrado del rendimiento de SQL.

Más ejemplos

- Busque todos los ComputerSystems que tengan cualquier atributo ampliado con el valor val:
  - MQL:

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml[attribute=\"val\"]'
    ```

  - SQL:

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml[attribute="val1"]'
    passing compsys.xa_x as "c")
    ```

- Busque un ComputerSystem con el atributo ampliado attr2 que tenga la categoría establecida en Other y el valor en two:
  - MQL:

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml/attribute
    [@name=\"attr2\" and @category=\"Other\" and text()=\"two\"]'
    ```

  - SQL:

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml/attribute[@name="attr2"
     and and @category="Other" and text()="two"]' passing compsys.xa_x as
     "c")
    ```

Utilización de la API de Java

La API de Java API le permite controlar el proceso de descubrimiento y aspectos del Modelo de datos común, incluido el acceso a los datos del modelo resultante.

Con la API de Java, puede crear aplicaciones para añadir, actualizar y suprimir objetos de modelo. Puede consultar objetos de modelo por nombre de clase o número de ID de objeto. También puede utilizar la interfaz para gestionar las relaciones entre objetos, realizar comparaciones y examinar el historial de cambios.

Los datos de modelo se pueden filtrar en el servidor y se devuelven al cliente en formato XML. A continuación, puede realizar transformaciones y otras consultas en el cliente, según sea necesario. La API de Java ofrece también métodos que se pueden utilizar para gestionar sesiones e implementar operaciones relacionadas con la seguridad.

La API de Java se encuentra en la clase **com.collation.proxy.api.client.CMDBApi**, que se comunica con el servidor de API RMI en el servidor de TADDM.

Defina la propiedad com.collation.home como `$COLLATION_HOME/sdk` en el directorio raíz de distribución del kit de desarrollo de software. Con la línea de mandatos de Java, puede definir la propiedad como se indica a continuación:

```
java -Dcom.collation.home=$COLLATION_HOME/sdk main_classname
```

Antes de empezar a utilizar la API de Java

Antes de empezar a crear aplicaciones Java, tiene que verificar que el entorno de desarrollo se haya configurado correctamente.

Procedimiento

Para verificar el entorno, siga estos pasos:

1. Compruebe que las variables de entorno se hayan definido correctamente y configure la propiedad `com.ibm.cdb.service.registry.public.port`.

   Consulte la sección sobre la configuración del TADDM SDK para obtener más información sobre cómo definir las variables de entorno y parámetros de configuración específicos.
2. Verifique que el servidor de TADDM se esté ejecutando.

   Consulte la sección sobre cómo verificar la instalación del SDK para obtener más información.

Exploración de una aplicación Java de muestra

En esta sección, se describe cómo crear, compilar y ejecutar una aplicación Java simple.

Procedimiento

Para crear la aplicación Java de ejemplo, siga estos pasos:

1. Copie el siguiente código Java en un archivo denominado `FindXmlExample.java`.

   El código fuente también está disponible en el directorio `$COLLATION_HOME/sdk/examples/java`.

   ```
   package com.collation.proxy.api.examples.java;
   // package com.collation.proxy.api.examples.java;

   import com.collation.proxy.api.client.ApiConnection;
   ```

```
import com.collation.proxy.api.client.ApiException;
import com.collation.proxy.api.client.ApiSession;
import com.collation.proxy.api.client.CMDBApi;
import com.collation.proxy.api.client.DataResultSet;
import com.ibm.cdb.api.ApiFactory;

/**
 * Simple CMDB API findXML() example:
 * <p>  get connection and log into api server
 * <p>  find all machines which have more than 1 CPU
 * <p>  find all Oracle instances
 */

public class FindXmlExample {

  public static void main(String[] args) {

    CMDBApi api = null;
    ApiSession sess = null;
    try {
      /*
       * Establish connection to api server
       * <p>    ApiConnection.getConnection(host, port,
                                             trustoreLocation, useSSL)
       */
      ApiConnection conn =
     ApiFactory.getInstance().getApiConnection("localhost", -1,
                                                null, false);
      /*
       * Get a session:
       * <p>     ApiSession.getSession(connection, username,
                                       password, version)
       */
      sess = ApiFactory.getInstance().getSession(conn, "smartoperator",
                                                 "foobar",
                                                 ApiSession.DEFAULT_VERSION);
      /*
       * Get an CMDBApi instance
       */
      api = sess.createCMDBApi();

      System.out.println("all machines which have more than 1 CPU:");
      String query = "select * from ComputerSystem where numCPUs>1";
      /*
       * Find all of the ComputerSystem have more than 1 CPU.
       * The method: findXml(query, depth, indent, mssGuid, permissions)
       * is deprecated, as the result set may be too large to fit into
       * memory.  Instead, using cursors is encouraged:
       */
      DataResultSet data = api.executeQuery(query, null, null);
      while (data.next()) {
          System.out.println(data.getXML(4));
      }
      data.close();
      System.out.println("\nall Oracle instances:");
      query = "select * from OracleInstance";
      data = api.executeQuery(query, null, null);
      while (data.next()) {
          System.out.println(data.getXML(4));
      }
      data.close();

    } catch (ApiException ae) {
      System.err.println("api exception:" + ae);
      ae.printStackTrace();
    } catch (Exception ex) {
      System.err.println("exception:" + ex);
```

```
      ex.printStackTrace();
    } finally {
      try {
        if (api != null) {
            api.close();
        }
        if (sess != null) {
            sess.close();
        }
      } catch (Exception ex) {
        System.err.println("exception:" + ex);
        ex.printStackTrace();
      }
    }

  }
}
```

2. De forma predeterminada, el programa de ejemplo se conecta a un servidor de TADDM en el sistema principal local. Si se quiere conectar a un servidor remoto, cambie la línea siguiente:

   ```
   ApiConnection conn = ApiFactory.getInstance().getApiConnection("localhost", -1,
     null, false);
   ```

   Por ejemplo, para conectarse a un servidor denominado taddmhost.ibm.com, utilice los puertos predeterminados:

   ```
   ApiConnection conn = ApiFactory.getInstance().
        getApiConnection("taddmhost.ibm.com",-1, null, false);
   ```

   De forma predeterminada, el programa de ejemplo crea una sesión con el ID de usuario `smartoperator` y la contraseña `foobar`. Cámbielos para que coincidan con un ID de usuario y una contraseña definidos en su servidor de TADDM. Por ejemplo:

   ```
   sess = ApiFactory.getInstance().
         getSession(conn, "administrator", "collation",ApiSession.DEFAULT_VERSION);
   ```

3. Para compilar el programa de ejemplo, junto con otros programas Java de ejemplo, siga estos pasos:
   a. En sistemas UNIX:
      1) Vaya al directorio `$COLLATION_HOME/sdk/examples/java`.
      2) Cree un ejecutable build.sh:
         ```
         chmod +x build.sh
         ```
      3) Ejecute el mandato de construcción:
         ```
         ./build.sh
         ```
   b. En Windows:
      1) Vaya al directorio `%COLLATION_HOME%\sdk\examples\java`.
      2) Ejecute el mandato de construcción:
         ```
         build.bat
         ```

   Si el SDK se instala separado del servidor de TADDM, asegúrese de que **javac** se encuentre ya en la vía de acceso y esté disponible.

4. Ejecute la aplicación Java mediante un mandato parecido al siguiente mandato de ejemplo:

   ```
   % java -Dcom.collation.home=$COLLATION_HOME/sdk FindXmlExample
   ```

   Este mandato ejecuta el programa de ejemplo y recupera los datos XML del servidor de TADDM.

Información sobre la aplicación Java de ejemplo

En esta sección se describe el funcionamiento del ejemplo `FindXmlExample.java`.

```
/*
 * Establish connection to API server
 */
  ApiConnection conn = ApiFactory.getInstance().
     getApiConnection("localhost", -1,null, false);
```

Este segmento crea un nuevo objeto ApiConnection con el servidor de la API, que se utiliza como descriptor de contexto para gestionar la sesión de la API entre el programa cliente y el servidor de TADDM. Los argumentos se representan en la lista siguiente:

- El argumento `host` es el sistema en el que se ejecuta el servidor de TADDM.
- El segundo argumento es el puerto del servidor. El valor -1 especifica el puerto predeterminado configurado por `com.ibm.cdb.service.registry.public.port` en el archivo de propiedades `$COLLATION_HOME/sdk/etc/collation.properties`.
- El tercer y el cuarto argumento controlan el acceso SSL.

```
/*
 * Get a session
 *<p> ApiSession.getSession(connection, username, password,
    versión)
 */
 sess = ApiFactory.getInstance().
    getSession( conn, smartoperator, foobar,ApiSession.DEFAULT_VERSION);
/*
 * Get an CMDBApi instance
 */
 api = sess.createCMDBApi();
```

Este segmento conecta el objeto CMDBApi con el servidor de TADDM. Si tiene que conectarse a varios servidores, por ejemplo en un caso de ejemplo de centro de datos distribuido a gran escala, puede utilizar varios objetos de CMDBApi con cada contexto de mantenimiento para una instancia de servidor de TADDM específica.

```
System.out.println("all machines which have more than 1 CPU:");
  String query = "select * from ComputerSystem where numCPUs>1";
  /*
  * Find all of the ComputerSystem have more than 1 CPU.
  * The method: findXml(query, depth, indent, mssGuid, permissions)
  * is deprecated, as the result set may be too large to fit into
  * memory.  Instead, using cursors is encouraged:
  */
           DataResultSet data = api.executeQuery(query, null, null);
```

Este segmento utiliza un objeto de CMDBApi inicializado para recuperar datos del servidor de TADDM utilizando el método executeQuery. A continuación, se describen los argumentos:

- El primer argumento especifica la consulta. Vea los elementos de consulta MQL de la sección Presentación de Model Query Language para obtener más información.
- El segundo argumento es el identificador exclusivo global de los sistemas de software gestionados (MSS). Un valor nulo indica que se devuelven los resultados de la consulta de todos los registros, independientemente del MSS al que estén asociados.
- El tercer argumento es la matriz de permisos. Los permisos se proporcionan durante la configuración del control de acceso a nivel de datos. Los permisos proporcionados deben coincidir con los utilizados para configurar el control de

accesos a nivel de datos. Por ejemplo, si proporciona un permiso "Update", se limitarían los objetos devueltos a aquellos que el interlocutor puede actualizar según la configuración del control de acceso a nivel de datos. Un valor nulo indica que se devolverán todos los objetos a los que el usuario ha tenido acceso.

Referencia relacionada:

"Sistemas de software gestionado" en la página 60
Los métodos de MSS gestionan los sistemas de software gestionado del modelo de datos común. Puede utilizar los métodos del sistema de software gestionado para registrar y suprimir un MSS del modelo de datos común. También puede recuperar información sobre los sistemas de software gestionado que se han registrado en la base de datos de TADDM.

Mejores prácticas

En esta sección, se describen las siguientes mejores prácticas de uso de la API de Java:

- Optimización del acceso a datos
- Seguimiento de enlaces entre objetos de modelo

Optimización del acceso a datos

Al igual que otras API de acceso a datos, la API de TADDM puede devolver una gran cantidad de datos, lo que podría saturar los recursos del sistema. Así pues, evite recuperar todos los datos en entornos grandes. Este método requiere una frecuente sincronización con la base de datos de TADDM para garantizar que se capturen todos los cambios.

Las opciones siguientes le ofrecen distintos métodos para recuperar sus datos:

- Un patrón de uso sugerido pasa por recuperar solo los elementos e identidades y no necesariamente los datos de configuración detallados. Esto limita la cantidad de datos que se van a transferir. Si necesita la configuración detallada de un elemento, puede realizar una llamada a findChanges() utilizando el ID de objeto como parámetro.
- Realización de un acceso a datos de cambio incremental. Este método requiere que utilice el tipo siguiente de llamada al método findChanges():

  ```
  findChanges (java.lang.String root, java.lang.String query, int depth,
  long start, long end,int changeType)
  ```

  Los parámetros **start** y **end** especifican un intervalo de tiempo, mientras que el parámetro **changeType** especifica Creado, Suprimido o Actualizado. Esta llamada a findChanges() devuelve solo los objetos del tipo especificado utilizando el parámetro **changeType** en el intervalo de tiempo dado. Utilice este método al realizar la sincronización incremental de datos de topología tras una transferencia de datos de línea base completa.
- El método find devuelve todos los datos a la vez, lo que puede provocar problemas de uso de memoria. Para evitar este problema, utilice el método executeQuery para desplazarse por los datos utilizando los cursores:

  ```
  DataResultSet rs = api_.executeQuery("select * from ComputerSystem", null, null);
       while (rs.next()) {
                  ...
       }
       rs.close();
  ```
- Utilice el método findCount para contar de manera eficaz el número de objetos que coinciden con una consulta:

  ```
  long count = api.findCount("select * from ComputerSystem", null);
  ```

Mayores valores de memoria para el gestor de topologías

Si ejecuta consultas no específicas en bases de datos grandes, se puede encontrar con problemas de memoria en el servidor de TADDM y el cliente de la API. Utilice las consultas específicas para identificar el tamaño de los conjuntos de resultados y los requisitos de memoria.

En el caso de consultas más genéricas, que generan un gran conjunto de resultados, se debe asignar más memoria. Si recibe un mensaje de error de falta de memoria, aumente la mensaje disponible para la máquina virtual Java.

Para incrementar la memoria disponible, utilice los valores siguientes:

- En el servidor de TADDM se puede aumentar la memoria disponible mediante la edición del archivo de configuración del despliegue del servidor:
  - Servidor de dominio: `cmdb-context.xml`
  - Servidor de sincronización: `ecmdb-context.xml`
  - servidor de almacenamiento: `storage-server-context.xml`
  - Servidor de descubrimiento: `discovery-server-context.xml`

  En el archivo correspondiente, localice la propiedad `jvmArgs` de la máquina JVM afectada por el error de memoria y aumente la memoria modificando la propiedad `DTaddm.xmx64`.
- En el cliente de la API, aumente la memoria especificada para la aplicación cliente, por ejemplo, en los archivos `api.sh` o `api.bat`.

Seguimiento de enlaces entre objetos de modelo

La mayoría de los elementos de datos del Modelo de datos común son autónomos. En muchos casos, los enlaces entre objetos de modelo, como LogicalDependency, se representan almacenando ID de objeto, que la API de TADDM no sigue automáticamente. En estos casos, debe aplicar una lógica adicional a la aplicación cliente.

Resumen del método de la API de Java

Con la API de Java, puede crear aplicaciones para añadir, actualizar y suprimir objetos de modelo. Puede consultar objetos de modelo por nombre de clase o número de ID de objeto. También puede utilizar la interfaz para gestionar relaciones entre objetos, realizar comparaciones y examinar el historial de cambios en la base de datos de TADDM.

La API de Java se puede resumir utilizando las categorías siguientes, analizadas en detalle en los apartados correspondientes:

- Gestión de sesiones
- Gestión de descubrimiento
- Gestión del modelo
- Buscar, actualizar y suprimir operaciones
- Gestión de recopilaciones
- Gestión de relaciones
- Metadatos
- Sistemas de software gestionado
- MSSObjectLink
- Historial de cambios

- Presentación
- Gestión de versiones
- Seguridad
- Creación y gestión de listas de acceso
- Gestión de plantillas de aplicación

Historial de cambios:

Los métodos del historial de cambios determinan el historial de cambios del modelo de datos común. Puede utilizar los métodos del historial de cambios para recuperar el historial de cambios de los elementos gestionados en el modelo de datos común. También puede desencadenar la propagación y el cálculo del historial de cambios, si lo necesita.

Tabla 9 describe los métodos del historial de cambios que puede utilizar.

Tabla 9. Métodos del historial de cambios

Método	Descripción
getChangeHistory(Guid[] Guids, long start, long end)	Devuelve el historial de cambios de la matriz de GUID especificada.
getChangeHistory(Guid[] Guids, long start, long end, int filterType)	Obtenga el historial de cambios de varios GUID, filtrados por el parámetro filterType. Este método es igual a getChangeHistory(), con el añadido de filterType, que se puede definir en los valores siguientes: • DataApi.CREATED • DataApi.DELETED • DataApi.UPDATED • DataApi.UPDATECREATE • DataApi.ANYCHANGE
getChangeHistory(Guid Guid, long start, long end)	Devuelva el historial de cambios del GUID especificado.
getChangeHistory(String[] classNames, long start, long end)	Devuelva el historial de cambios de las clases especificadas.
getChangeHistoryByPersistTime(String[] classNames, long start, long end)	Devuelva el historial de cambios de las clases especificadas según el tiempo de persistencia.
getChangeHistoryInXML(Guid[] Guids, long start, long end)	Devuelva el historial de cambios de la matriz especificada de ID de objeto.
getChangeHistoryInXML(Guid[] Guids, long start, long end, int filterType)	Obtenga el historial de cambios de varios GUID, filtrados por filterType. Este método es igual a getChangeHistory(), con el añadido de filterType, que se puede definir en los valores siguientes: • DataApi.CREATED • DataApi.DELETED • DataApi.UPDATED • DataApi.UPDATECREATE • DataApi.ANYCHANGE
getChangeHistoryInXML(Guid Guid, long start, long end)	Devuelva el historial de cambios del GUID especificado.

Tabla 9. Métodos del historial de cambios (continuación)

Método	Descripción
getPropagatedChanges(long primaryKey) **Nota:** Este método está en desuso. Utilice el método getChangeHistory.	Obtenga las principales causas de un historial de cambios determinado, devuelto como representación XML de los objetos ChangeHistory que han provocado el cambio actual. Por ejemplo, si un módulo de Apache Server cambia, estos cambios se propagarán al servidor Apache de nivel superior. Puede utilizar este método para determinar la causa que ha desencadenado el cambio del servidor Apache.
getPropagatedChangesInXML(long primaryKey) **Nota:** Este método está en desuso. Utilice el método getChangeHistory.	Obtenga las principales causas de un historial de cambios determinado, devuelto como representación XML de los objetos ChangeHistory que han provocado el cambio actual. Por ejemplo, si un módulo de Apache Server cambia, estos cambios se propagarán al servidor Apache de nivel superior. Puede utilizar este método para determinar la causa que ha desencadenado el cambio del servidor Apache.
processChanges()	Desencadene la propagación y el cálculo del historial de cambios desde el último descubrimiento o desde la última vez que se llamó a processChanges(). De lo contrario, los cambios no se calcularán hasta la próxima vez que se ejecute un descubrimiento. Este método debe utilizarse en cambios aislados. Para actualizaciones múltiples, como operaciones de carga en bloque, utilice los métodos startBulkload() y endBulkload().
getChangedClasses(long start, long end, int changeType)	Devuelva una matriz de tipos de clases que hayan cambiado entre la fecha inicial y la final. El tipo de cambio especificado puede ser cualquiera de los siguientes: • DataApi.CREATED • DataApi.DELETED • DataApi.UPDATED • DataApi.UPDATECREATE • DataApi.ANYCHANGE
getChangedClassesForDeltaSynching(long start, long end, int changeType)	Devuelve una matriz de tipos de clase que han cambiado entre la fecha inicial y la final, según el tiempo de persistencia. El tipo de cambio especificado puede ser cualquiera de los siguientes: • DataApi.CREATED • DataApi.DELETED • DataApi.UPDATED • DataApi.UPDATECREATE • DataApi.ANYCHANGE

Tabla 9. Métodos del historial de cambios (continuación)

Método	Descripción
getChangeHistory(Guid[] Guids, long start, long end, int offset, int nextBatch)	Devuelve el historial de cambios de la lista de GUID especificada. El parámetro nextBatch especifica el tamaño del lote de registros que se va a devolver, empezando por el desplazamiento especificado. Esto permite un método escalable para devolver la información del historial de cambios.

Gestión de descubrimiento:

Los métodos de descubrimiento gestionan las ejecuciones del descubrimiento. Puede utilizar los métodos de descubrimiento para iniciar y detener descubrimientos, y habilitar o inhabilitar sucesos de actualización. También puede utilizar los métodos para conseguir el estado del descubrimiento y borrar todos los elementos de descubrimiento de la topología.

Tabla 10 describe los métodos de descubrimiento que puede utilizar.

Muchos de los siguientes métodos hacen referencia al descubrimiento con equilibrio de carga. Para obtener más información, consulte "Mandato de descubrimiento con equilibrio de carga" en la página 119.

Tabla 10. Métodos de descubrimiento

Método	Descripción
abortDiscovery()	Finaliza la ejecución de descubrimiento actual.
getStatus()	Devolver el estado de la ejecución de descubrimiento actual. El estado puede ser cualquiera de estos valores: • `En ejecución` • `Desocupado`
startDiscovery(String[] scope, String runName, String locationTag)	Inicie un nuevo descubrimiento con un ámbito. El ámbito puede ser un nombre, o bien contener rangos de IP, redes y direcciones, con direcciones IP específicas incluidas y excluidas.
startDiscovery(RunDefinition runDef, String runName)	Inicie un nuevo descubrimiento basado en una definición de ejecución de descubrimiento que especifique la información del perfil, el ámbito, el nombre de la ejecución y la etiqueta de ubicación.
startDiscovery(Guid[] guidList, String runName)	Inicie un redescubrimiento de los objetos con los identificadores exclusivos globales (GUID) especificados.
abortDiscovery(long runId)	Finalice la ejecución del descubrimiento especificada.

Tabla 10. Métodos de descubrimiento (continuación)

Método	Descripción
Fix Pack 2 LoadbalancedDiscoveryStatus loadbalancedDiscoveryStatus()	Devuelve el estado de la ejecución de descubrimiento con equilibrio de carga actual. El estado puede ser cualquiera de estos valores: • `En ejecución` • `Desocupado`
Fix Pack 2 startLoadbalancedDiscovery(String scopeGroupName, String discoveryPoolName, String profileName, String locationTag)	Inicia un nuevo descubrimiento con equilibrio de carga con relación a cada ámbito incluido en el atributo `scopeGroup`, y en cada servidor que pertenece al servidor de descubrimiento especificado en el atributo `poolName`.
Fix Pack 2 startLoadbalancedDiscovery(String[] fileNames, int maxScopeSize, String profileName, String locationTag)	Inicia un nuevo descubrimiento con equilibrio de carga con relación a cada ámbito que empieza desde un archivo específico. Se analiza cada archivo y se crea un nuevo grupo con el nombre del archivo (sin la extensión). En cada grupo, los ámbitos se añaden con el tamaño máximo especificado en el atributo `maxScopeSize`. Se asume que el atributo `poolName` será el mismo que el atributo `groupName`.
Fix Pack 2 stopLoadbalancedDiscovery(String discoveryPoolName)	Detiene el descubrimiento con equilibrio de carga para el atributo `poolName` especificado.
Fix Pack 2 pauseLoadbalancedDiscovery(String discoveryPoolName)	Coloca en pausa el descubrimiento con equilibrio de carga para el atributo `poolName` especificado. Todos los ámbitos en la ejecución del descubrimiento se vuelven de nuevo al estado 'forTake'. Se cancelan los descubrimientos actuales.
Fix Pack 2 resumeLoadbalancedDiscovery(String discoveryPoolName)	Devuelve el objeto LoadBalancedDiscoveryStatus que expone todas las agrupaciones de descubrimiento que están en curso, incluidas las agrupaciones que están en curso pero en pausa. Además, devuelve información sobre qué ámbitos están en curso, y cuáles están a la espera en la cola.
Fix Pack 3 startDiscoveryRetID(String[] scope, String runName, String locationTag, String addressSpace)	Inicie un nuevo descubrimiento con un ámbito y devuelva el ID de ejecución de descubrimiento. El ámbito puede ser un nombre, o bien contener rangos de IP, redes y direcciones con direcciones IP específicas incluidas y excluidas.

Operaciones de búsqueda:

Los métodos de búsqueda acceden a los objetos del modelo de datos común. Puede utilizar los métodos de búsqueda para devolver objetos de modelo que coincidan con una consulta específica o devolver información sobre elementos gestionados específicos. Puede utilizar también los métodos para devolver objetos que han cambiado durante un período de tiempo especificado.

Tabla 11 describe las operaciones de búsqueda que puede realizar.

Nota: Muchos de los métodos find que utilizan un parámetro de profundidad han caído en desuso, porque no es fácil escalarlos cuando se consulta una gran cantidad de datos. Si necesita consultar datos con una profundidad mayor que uno, utilice un método executeQuery. Todos los métodos executeQuery devuelve un objeto DataResultSet del que puede recuperar información sobre objetos de modelo, y puede utilizar el cursor para desplazarse por los datos.

Tabla 11. Métodos de búsqueda

Método	Descripción
find(Guid guid, int depth, Guid mss, String[] permissions)	Devuelva información sobre un elemento gestionado específico.
find(Guid Guid, int depth, String[] permissions)	Igual que find(String, int, Guid), salvo que se busca una instancia de objeto específica por GUID, no por un conjunto completo de objetos mediante una consulta.
find(String query, int depth, Guid mss, String[] permissions) **Nota:** Este método está en desuso. Utilice executeQuery o un método find con un parámetro fillFlag.	Devuelva los objetos de modelo que coincidan con la consulta especificada. Tenga en cuenta la información siguiente sobre el parámetro de profundidad: • La profundidad 0 devuelve un objeto con solo su GUID definido. • La profundidad 1 devuelve todos los atributos de nivel superior, junto con los objetos contenidos que solo tienen los atributos de GUID definidos. • Si la profundidad se establece en DEPTH_INFINITE, todos los atributos se ubicarán de manera recursiva hasta que no haya objetos adicionales. Se evitan los ciclos.
find(String query, boolean fillFlag, Guid mss, String[] permissions)	Devuelva los objetos de modelo que coincidan con la consulta especificada. • query: Model Query Language para seleccionar y filtrar los resultados. • fillFlag: si se deben llenar todos los atributos. • mss: ID de sistema de software gestionado, o nulo si no es ninguno • permissions: lista opcional de permisos para restringir los resultados
findChanges(String query, int depth, long start, long end, int changeType) **Nota:** Este método está en desuso. Utilice el método executeChangesQuery.	Devuelva los objetos que han cambiado durante el período especificado para un tipo de cambio determinado.
findChanges(String query, boolean fillFlag, long start, long end, int changeType) **Nota:** Este método está en desuso. Utilice el método executeChangesQuery.	Devuelva los objetos que han cambiado durante el período especificado para un tipo de cambio determinado.
findChanges(String query, int depth, long start, long end, int changeType **Nota:** Este método está en desuso. Utilice el método executeChangesQuery.	Devuelva los objetos que han cambiado en el período especificado para un tipo de cambio dado.

Tabla 11. Métodos de búsqueda (continuación)

Método	Descripción
findChanges(String query, boolean fillFlag, long start, long end, int changeType) **Nota:** Este método está en desuso. Utilice el método executeChangesQuery.	Devuelva los objetos que han cambiado en el período especificado para un tipo de cambio dado.
findCollections(Guid guid, String[] permissions)	Consulte la sección sobre gestión de recopilaciones.
findImpactedBusinessApplications(Guid[] objects)	Consulte la sección sobre la presentación.
findImpactedBusinessServices(Guid[] objects)	Consulte la sección sobre la presentación.
findJDO(String root, String jdoQuery, String jdoVarDecl, int depth, Guid mss, String[] permissions) **Nota:** Este método está en desuso. Utilice el método executeJDOQuery.	Igual que find(String, int, Guid), con la salvedad de que la consulta debe contener una consulta Java Data Object (JDOQL), con declaraciones de variable opcionales.
findJDO(String root, String jdoQuery, String jdoVarDecl, boolean fillFlag, Guid mss, String[] permissions) **Nota:** Este método está en desuso. Utilice el método executeJDOQuery.	Igual que find(String, int, Guid), con la salvedad de que la consulta debe contener una consulta Java Data Object (JDOQL), con declaraciones de variable opcionales.
findRelationships(Guid managedElementGuid, int direction, String type, int scope, String[] permissions)	Consulte la sección sobre gestión de relaciones.
findRelationships(Guid[] sourceGuids, Guid[] targetGuids, String[] types, int comparisonFlags)	Consulte la sección sobre gestión de relaciones.
findXML(String query, int depth, int indent, Guid mss, String[] permissions) **Nota:** Este método está en desuso. Utilice el método findXML(String,boolean,int,Guid,String[].	Devuelve un documento XML que contiene una lista de todos los objetos que coincidan con la consulta especificada. Este método es parecido en su función a find(String, int, Guid), con la salvedad de que los objetos se convierten a formato XML.
findXML(Guid Guid, int indent, int depth, String[] permissions)	Igual que findXML(String, int, int, Guid), con la salvedad de que se busca una instancia de objeto específica por ID, no por un conjunto completo de objetos mediante una consulta.
findXML(String query, boolean fillFlag, int indent, Guid mss, String[] permissions)	Devuelve un documento XML que contiene una lista de todos los objetos que coincidan con la consulta dada.
findCount(String query, String[] permissions)	Devuelve un recuento de los objetos que coinciden con la serie de consulta MQL especificada.
executeQuery(String query, Guid mss, String[] permissions)	Ejecuta una consulta MQL y devuelve una interfaz del cursor de desplazamiento.
executeQuery(String query, int defaultDepth, Guid mss, String[] permissions)	Ejecuta una consulta MQL y devuelve una interfaz del cursor de desplazamiento.

Tabla 11. Métodos de búsqueda (continuación)

Método	Descripción
executeChangesQuery(String query, int defaultDepth, long start, long end, int changeType, boolean deltaSynching)	Devuelve los objetos que han cambiado en el período especificado para un tipo de cambio dado. Por lo demás, su funcionalidad es similar a la de executeQuery (String,int,Guid,String[]). Si el parámetro **changeType** se define como DELETED, la cláusula WHERE de la consulta se ignora. Se devolverán todos los objetos de la clase de objeto de modelo especificada que se hayan suprimido en el lapso de tiempo. Los objetos de modelo devueltos son los objetos de modelo de shell. No contienen ningún atributo. Solo está definido el GUID. Si el parámetro **changeType** se define como ANYCHANGE, además de ejecutar la consulta find normal, se devolverán los objetos shell suprimidos. Sin embargo, la cláusula where se ignora para los objetos suprimidos.
findChangesForDeltaSynching(String query, boolean fillFlag, long start, long end, int changeType) **Nota:** Este método está en desuso. Utilice el método executeChangesQuery.	Devuelve los objetos que se han mantenido en el almacén de datos durante el periodo específico de tiempo del tipo de cambio especificado.

Sistemas de software gestionado:

Los métodos de MSS gestionan los sistemas de software gestionado del modelo de datos común. Puede utilizar los métodos del sistema de software gestionado para registrar y suprimir un MSS del modelo de datos común. También puede recuperar información sobre los sistemas de software gestionado que se han registrado en la base de datos de TADDM.

Tabla 12 describe los métodos de MSS que puede utilizar.

Tabla 12. Métodos del sistema de software gestionado

Método	Descripción
deleteManagementSoftwareSystem(Guid guid)	Suprima un sistema de software gestionado de gestión (MSS) de la base de datos de TADDM. Se suprimen también los objetos y las relaciones que son propiedad únicamente de este MSS o que solo él ha descubierto. Sin embargo, no se suprimen los objetos y las relaciones que son propiedad de, o que han sido descubiertos por, este y otros MSS. Solo se suprimen la asociación entre este MSS y los objetos y relaciones de su propiedad.

Tabla 12. Métodos del sistema de software gestionado (continuación)

Método	Descripción
getManagementSoftwareSystemLinks(Guid guid, Guid mss, String[] permissions)	Devuelva las señales de origen para una relación o un elemento gestionado. Si no se especifica un sistema de software gestionado, el método devuelve las señales de origen de cada sistema de software gestionado que ha descubierto o que posee el objeto especificado.
getManagementSoftwareSystems(Guid guid, String[] permissions)	Obtenga una matriz de los sistemas de software gestionado que se han registrado con la base de datos de TADDM. De manera opcional, se puede proporcionar el GUID de un elemento o una relación gestionados para que se devuelvan solo los sistemas de software gestionado que poseen la relación o el elemento gestionado.
registerManagementSoftwareSystem (ManagementSoftwareSystem mss)	Registre un nuevo sistema de software gestionado con la base de datos de TADDM. Tenga en cuenta los siguientes detalles: • Este método inserta solo un objeto. El método no sustituye ni actualiza un objeto existente. • El modelo tiene que contener claves para el objeto. • Si la clave se refiere a un objeto principal, este debe existir. • Si el objeto hace referencia directa a un GUID, será responsabilidad del desarrollador asegurarse de que el GUID exista en la base de datos de TADDM.
updateManagementSoftwareSystem (ManagementSoftwareSystem mss)	Actualice o inserte un sistema de software gestionado en la base de datos. Tenga en cuenta los siguientes detalles: • Este método se puede utilizar para insertar, sustituir o actualizar un objeto existente. • El modelo tiene que contener claves para el objeto. • Si la clave se refiere a un objeto principal, este debe existir. • Si el objeto hace referencia directa a un GUID, será responsabilidad del desarrollador asegurarse de que el GUID exista en la base de datos de TADDM.

MSSObjectLink:

MSSObjectLink es una asociación entre un sistema de software gestionado y un elemento de gestión de su propiedad.

Los métodos MSSObjectLink se pueden utilizar para recuperar todos los MSSObjectLinks del MSS dado y los elementos gestionados. Estos MSSObjectLinks representan todos los elementos gestionados propiedad de un MSS dado, o bien todos los MSS que poseen un elemento gestionado dado. MSSObjectLink no se

almacena como un objeto de modelo, por lo que no se puede utilizar la API de búsqueda para obtener los mensajes de MSSObjectLink. En su lugar, utilice las API descritas en Tabla 13, junto con la API getManagementSoftwareSystemLinks.

Tabla 13. MSSObjectLink

Método	Descripción
getObjectSourceSystems(Guid[] objectId, String[] permission)	Devuelve todos los MSS que tienen en propiedad relaciones o elementos gestionados. Este método devuelve una matriz de MSSObjectLink que se corresponde con el MSS que tiene un elemento gestionado en propiedad. Puede pasar un máximo de 50 elementos gestionados a la vez en la matriz del identificador exclusivo global.
getObjectSourceSystems(Guid objectId, Guid mssGuid, String[] permission)	Devuelve una matriz de MSSObjectLinks donde cada MSSObjectLink se corresponde con la señal de origen de un objeto propiedad del MSS dado. Si solo se especifica el mssGuid, este método devolverá las señales de origen de todos los objetos que posee. Por otra parte, si especifica solo el GUID del objeto, devolverá las señales de origen de todos los MSS que posee.
getObjectSourceSystems(Guid mssGuid, String mssSourceToken)	Devuelve un MSSObjectLink que se corresponde con un elemento gestionado propiedad de un MSS dado y se identifica mediante su señal de origen (se debe especificar la señal de origen).
getObjectSourceSystems(String joinQuery)	Devuelve una matriz de MSSObjectLinks que cumplen la consulta JOIN entre MSSObjectLink y cualquier otro objeto de modelo. Dado que los MSSObjectLinks se almacenarán directamente como datos relacionales, en lugar de como objeto de modelo, no se podrá utilizar MQL para JOIN MSSObjectLink con ningún otro objeto de modelo. La joinQuery que se pasa en este método será una consulta SQL en su lugar. Los atributos especificados en la cláusula SELECT de la consulta SQL deben ser los atributos del objeto MSSObjectLink, dado que este método devuelve una matriz de objetos MSSObjectLink. Ejemplo: obtención de todos los ComputerSystems propiedad de un MSS dado. **SELECT mssobjlink_rel.obj_x,** mssobjlink_rel.msssourcetoken_x, mssobjlink_rel.guid_x, **FROM mssobjlink_rel, compsys** **WHERE mssobjlink_rel.obj_x = compsys.guid_x** **AND mssobjlink_rel.mss_x='5D65T789UK3'**

Gestión de listas de acceso:

Los métodos de listas de acceso crean y gestionan entradas de listas de acceso desde la API de Java. Las aplicaciones de los proveedores pueden gestionar identidades con estos métodos.

Se pueden completar las tareas siguientes utilizando estos métodos de la API:

Crear y suprimir una entrada de la lista de acceso
Puede crear y suprimir una entrada de la lista de acceso para mantenerlas automáticamente.

Actualizar las propiedades de la entrada de la lista de acceso
Puede actualizar las propiedades de la entrada de la lista de acceso. La API se puede utilizar para cambiar la contraseña.

Obtener las propiedades de la entrada de la lista de acceso
Obtendrá las propiedades concretas de la entrada de la lista de acceso que aparece en la ventana Lista de acceso de la consola de gestión de descubrimiento. La API no se puede utilizar para recuperar la contraseña.

Verificar el valor de propiedad de la entrada de la lista de acceso
Puede verificar si el valor de propiedad dado coincide con el valor de propiedad de la entrada existente de la lista de acceso. Esta API se puede utilizar para verificar la contraseña de la entrada de la lista de acceso.

Tabla 14 describe los métodos de la lista de acceso que se pueden utilizar.

Nota: La API proporcionada no da soporte a las entradas de la lista de acceso registradas en el perfil de descubrimiento.

Tabla 14. Métodos de la lista de acceso

Método	Descripción
deleteDiscoveryAccessEntry(String authClassName, String name)	Suprima la entrada de acceso con la clase y el nombre especificados.
getAllDiscoveryAccessEntries()	Obtenga todas las entradas de acceso de descubrimiento.
getDiscoveryAccessEntry (String authClassName, String name)	Obtenga la entrada de acceso de descubrimiento con la clase y el nombre especificados.
updateDiscoveryAccessEntry(DiscoveryAccessEntry discoveryAccess)	Actualice las propiedades de la entrada de acceso con la clase y el nombre especificados.
verifyDiscoveryAccessEntry(DiscoveryAccessEntry discoveryAccess)	Verifique si las propiedades dadas coinciden con las propiedades actuales de la entrada de acceso con la clase y el nombre especificados.

Código Java de ejemplo

1. El siguiente código Java de ejemplo muestra cómo crear la conexión, la sesión y la instancia de la API:

```
CMDBApi api;
try {
    ApiConnection conn_ = ApiFactory.getInstance().getApiConnection("localhost",
   -1,null,false);
    ApiSession session_ = ApiFactory.getInstance().getSession(conn_, user,
      password, CMDB_DEFAULT);
    api = session_.createCMDBApi();
  } catch (ApiException ex) {
    ex.printStackTrace();
  } catch (Exception ex) {
    ex.printStackTrace();
  }
```

2. El siguiente código Java de ejemplo muestra cómo crear de forma programática las entradas de acceso de descubrimiento:

```
try {
    // Entrada de acceso de WebSphere
    DiscoveryAccessEntry entry = new DiscoveryAccessEntry
     (DiscoveryAccessEntry.AUTHCLASS_WEBSPHERE, "user3-websphere");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope3");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 3);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "wasadmin");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry.setProperty(DiscoveryAccessEntry.
      PROPERTY_TRUSTSTOREFILECONTENTS, new byte[] { 0x10, 0x20, 0x30 });
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_TRUSTSTOREPASSPHRASE,
   "password");
    entry = api.updateDiscoveryAccessEntry(entry);

    // Entrada de acceso de SNMP
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_SNMP,
   "user4-snmp");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope4");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 4);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_COMMUNITYSTRING, "public");
    entry = api.updateDiscoveryAccessEntry(entry);

    // Entrada de acceso de SNMPv3
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_SNMPV3,
   "user5-snmpv3");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope5");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 5);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "snmp");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_AUTHPROTOCOL, "MD5");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PRIVPASSWORD,
   "privpassword");
    entry = api.updateDiscoveryAccessEntry(entry);

    // Entrada de acceso de Cisco
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_CISCO,
   "user6-cisco");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope6");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 6);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "cisco");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry.setProperty("enablepassword", "enablepassword");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ENABLEPASSWORD,
   "enablepassword");
    entry = api.updateDiscoveryAccessEntry(entry);

    // Entrada de acceso de ccmsserver
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_
      CCMSSERVER, "user7-sapccms");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope7");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 7);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "ccms");
```

```
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_CLIENTID, "clientid");
    entry = api.updateDiscoveryAccessEntry(entry);

    // Entrada de acceso del sistema informático
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_HOST,
   "user1-host");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope1");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 1);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "root");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry = api.updateDiscoveryAccessEntry(entry);

    // Entrada de acceso del sistema informático Windows
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_
      WINDOWSHOST, "user2-winhost");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope2");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 2);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "Administrator");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_AUTHTYPE,
      "authType_default");
    entry = api.updateDiscoveryAccessEntry(entry);
 } catch (ApiException ae) {
    System.err.println("api exception:" + ae);
 } catch (Exception ex) {
    System.err.println("exception:" + ex);
 }
```

3. El siguiente código Java de ejemplo muestra cómo obtener todas las entradas de acceso de descubrimiento:

```
try {
    DiscoveryAccessEntry entry;
    DiscoveryAccessEntry[] entries = api.getAllDiscoveryAccessEntries();
    for (int i = 0; i < entries.length; i++) {
    entry = entries[i];
    String authClassName = (String) entry.getAuthClassName();
    Integer order = (Integer) entry.getProperty(DiscoveryAccessEntry.
    PROPERTY_ORDER);
    switch (order.intValue()) {
    case 1:
      // DiscoveryAccessEntry.AUTHCLASS_HOST.equals(authClassName));
      break;
    case 2:
      // DiscoveryAccessEntry.AUTHCLASS_WINDOWSHOST.equals(authClassName));
      break;
    case 3:
      // DiscoveryAccessEntry.AUTHCLASS_WEBSPHERE.equals(authClassName));
      break;
    case 4:
      // DiscoveryAccessEntry.AUTHCLASS_SNMP.equals(authClassName));
      break;
    case 5:
      // DiscoveryAccessEntry.AUTHCLASS_SNMPV3.equals(authClassName));
      break;
    case 6:
      // DiscoveryAccessEntry.AUTHCLASS_CISCO.equals(authClassName));
      break;
    case 7:
      // DiscoveryAccessEntry.AUTHCLASS_CCMSSERVER.equals(authClassName));
      break;
    default:
      break;
     }
 } catch (ApiException ae) {
```

```
    System.err.println("api exception:" + ae);
 } catch (Exception ex) {
    System.err.println("exception:" + ex);
 }
```

4. El siguiente código Java de ejemplo muestra cómo obtener una entrada de acceso de descubrimiento específica:

```
try {
    DiscoveryAccessEntry entry = api.getDiscoveryAccessEntry
      (DiscoveryAccessEntry.AUTHCLASS_SNMP, "user4-snmp");
    String authClassName = (String) entry.getAuthClassName();
    String name = (String) entry.getName();
 } catch (ApiException ae) {
    System.err.println("api exception:" + ae);
 } catch (Exception ex) {
    System.err.println("exception:" + ex);
 }
```

5. El siguiente código Java de ejemplo muestra cómo actualizar una entrada de acceso de descubrimiento específica:

```
try {
  // Crear una entrada con el mismo nombre que la entrada existente
  DiscoveryAccessEntry entry = new DiscoveryAccessEntry
   (DiscoveryAccessEntry.AUTHCLASS_HOST, "user1-host");

  // Cambiar el ámbito
  entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope1c");

  // Cambiar el orden
  entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 2);

  // Cambiar el nombre de usuario
  entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "rootc");

  // Cambiar la contraseña
  entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "passwordc");

  // Actualizar la entrada
  entry = api.updateDiscoveryAccessEntry(entry);
} catch (ApiException ae) {
   System.err.println("api exception:" + ae);
} catch (Exception ex) {
   System.err.println("exception:" + ex);
}
```

6. El siguiente código Java de ejemplo muestra cómo verificar una entrada de acceso de descubrimiento:

```
try {
// Crear una entrada con el mismo nombre que la entrada existente para verificar
DiscoveryAccessEntry entry =
 new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_HOST, "user1-host");

// Definir la propiedad de pedido en el número equivocado
entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 1);

// Este resultado debería ser falso
boolean result = api.verifyDiscoveryAccessEntry(entry));

// Definir la propiedad de pedido en el número correcto
entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 2);

// Este resultado debería ser verdadero
result = api.verifyDiscoveryAccessEntry(entry));

// Definir la contraseña en el valor incorrecto
entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
```

```
// Este resultado debería ser falso
result = api.verifyDiscoveryAccessEntry(entry));

// Definir la propiedad de la contraseña en el valor correcto
entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "passwordc");

// Este resultado debería ser verdadero
   result = api.verifyDiscoveryAccessEntry(entry));
} catch (ApiException ae) {
   System.err.println("api exception:" + ae);
} catch (Exception ex) {
   System.err.println("exception:" + ex);
}
```

7. El siguiente código Java de ejemplo muestra cómo suprimir una entrada de acceso de descubrimiento:

```
try {
    api.deleteDiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_HOST,
   "user1-host");
} catch (ApiException ae) {
    System.err.println("api exception:" + ae);
} catch (Exception ex) {
    System.err.println("exception:" + ex);
}
```

Gestión de recopilaciones:

Los métodos de recopilación gestionan las recopilaciones del modelo de datos común. Puede utilizar los métodos de recopilación para añadir y eliminar miembros de una recopilación, así como para recuperar todas las recopilaciones a las que pertenece un elemento gestionado. Tenga en cuenta que estos métodos están en desuso. Para obtener métodos equivalentes relativos a CustomCollections, consulte la descripción de cada método en desuso.

Para obtener información relacionada, consulte “Gestión de patrones de agrupación” en la página 84.

La tabla Métodos de recopilación describe los métodos de recopilación que se pueden utilizar.

Tabla 15. Métodos de recopilación

Método	Descripción
addCollectionMembers(Guid collectionGuid, Guid[] guids)	Añada miembros a una recopilación. Este método no está disponible en la interfaz de Enterprise JavaBeans (EJB). Este método está en desuso. En su lugar, utilice el método updateGroupingPattern(GroupingPattern groupingPattern) seguido de uno de los métodos runPatternNow(...).

Tabla 15. Métodos de recopilación (continuación)

Método	Descripción
findCollections(Guid guid, String[] permissions)	Recupera todas las recopilaciones a las que pertenece el elemento gestionado especificado. Una recopilación puede contener otras, pero este método solo devuelve el primer nivel de recopilación al que pertenece el elemento gestionado especificado. Este método está en desuso. En su lugar, utilice el método findCustomCollections(Guid guid, String[] permissions).
removeCollectionMembers(Guid collectionGuid, Guid[] guids)	Elimine los miembros de una recopilación. Los miembros de una recopilación se pueden eliminar si se especifica una matriz de atributos del GUID de recopilación o una matriz de atributos del GUID de elemento gestionado. Si un miembro que se va a eliminar no es actualmente miembro de la recopilación, dicho miembro se ignora. Este método está en desuso. En su lugar, utilice el método updateGroupingPattern(GroupingPattern groupingPattern) seguido de uno de los métodos runPatternNow(...).

Gestión del modelo:

Los métodos de modelo gestionan objetos en el modelo de datos común. Puede utilizar los métodos de modelo para añadir, suprimir y actualizar objetos en el modelo de datos común. También puede utilizar los métodos para comparar objetos y para recrear los datos derivados, como dependencias, relaciones y consolidación de datos.

Tabla 16 describe los métodos del modelo que puede utilizar.

Tabla 16. Métodos de gestión de modelos

Método	Descripción
add(ModelObject[] obj, Guid mss)	Añada un nuevo objeto a la base de datos. **Nota:** Este método no se puede utilizar para añadir objetos de modelo GroupingPattern o Selector. Genera una ApiException cuando alguno de los objetos proporcionados es del tipo GroupingPattern o Selector. Utilice la API GroupingPattern para todas las operaciones en GroupingPatterns o selectores.
addArrayElements(Guid object, String attrName, Guid[] elements, Guid mss)	Añada los elementos a la matriz con nombre del objeto especificado sin captar el objeto ni la matriz.

Tabla 16. Métodos de gestión de modelos (continuación)

Método	Descripción
compare(ModelObject left, ModelObject[] right, CompareOptions opts) compare(ObjectRef obj1, ObjectRef[] objs, CompareOptions opts)	Compare un objeto de modelo (o maestro definitivo) con un conjunto de objetos.
delete(Guid[] guids, Guid mss) delete(ModelObject[] obj, Guid mss)	Suprima los objetos especificados por el GUID de la base de datos y suprima en cascada todos los objetos contenidos en los objetos en uno de estos casos: • No se proporciona ningún sistema de software gestionado (MSS). • El objeto es propiedad exclusiva del MSS especificado. El objeto no se suprime cuando se proporciona un MMS y el objeto es propiedad de otro MSS. En su lugar, la asociación entre el objeto y el MSS se suprime. Cuando se suprime un objeto de la base de datos de TADDM, todas las relaciones y las pertenencias de recopilación asociadas al objeto se suprimen también. Si el objeto especificado es un objeto de modelo de nivel superior, se eliminan también todos los objetos contenidos. Por ejemplo, cuando se elimina un sistema informático, se eliminan también el sistema operativo y las interfaces de IP contenidos en el objeto, junto con las relaciones entre el sistema informático y las interfaces de IP. **Nota:** Este método no se puede utilizar para suprimir objetos de modelo GroupingPattern o Selector. Genera una ApiException cuando alguno de los objetos proporcionados es del tipo GroupingPattern o Selector. Utilice la API GroupingPattern para todas las operaciones en GroupingPatterns o selectores.

Tabla 16. Métodos de gestión de modelos (continuación)

Método	Descripción
deleteStale(Guid mss, long date) **Nota:** Este método está en desuso.	Suprima las relaciones y los elementos gestionados que no se han tocado desde una fecha especificada. Se suprimen las relaciones y los elementos gestionados con un tiempo de actualización menor o igual a la fecha especificada. Si un elemento gestionado obsoleto es propiedad de más de un sistema de software gestionado, la relación o el elemento gestionado no se suprimen de la base de datos. Solo se suprime la asociación entre la relación o el elemento gestionado y el sistema de software gestionado especificado. Sin embargo, si una relación o un elemento gestionado obsoletos son propiedad solo del sistema de software gestionado especificado, la relación o el elemento gestionado se suprime de la base de datos. Todas las relaciones y miembros de recopilaciones asociados a un elemento gestionado suprimido se suprimen también. Si el objeto especificado es un objeto de modelo de nivel superior, se eliminan también todos los objetos contenidos. Por ejemplo, si se elimina un sistema informático, se eliminan también el sistema operativo y las interfaces IP contenidos en el objeto, así como las relaciones entre el sistema informático y las interfaces de IP.
Fix Pack 2 refresh(Guid mss, long date)	Suprima las relaciones y los elementos gestionados que no se han almacenado desde una fecha especificada. Se suprimen las relaciones y los elementos gestionados con una indicación de fecha y hora actualización anterior a la fecha especificada. Si una relación o elemento gestionado obsoleto es propiedad de más de un sistema de gestión de software, la relación o el elemento gestionado se suprimen de todos modos de la base de datos. Todas las relaciones y miembros de recopilaciones asociados a un elemento gestionado suprimido se suprimen también. Si el objeto especificado es un objeto de modelo de nivel superior, se eliminan también todos los objetos contenidos. Por ejemplo, si se elimina un sistema informático, se eliminan también el sistema operativo y las interfaces IP contenidos en el objeto, así como las relaciones entre el sistema informático y las interfaces de IP.

Tabla 16. Métodos de gestión de modelos (continuación)

Método	Descripción
endBulkload(long bulkloadId)	Marque el final de una operación de carga en bloque. Cada interlocutor que llama al método startBulkload() debe llamar a endBulkload() para liberar el bloqueo en el subsistema de almacenamiento.
exportData(File directoryToWriteTo, long maxFileSize, Guid mss)	Exporte todos los objetos de la base de datos de TADDM al directorio especificado, creando archivos para cada clase de objeto utilizando el método find() con una profundidad infinita. Si se superan los maxFileSize bytes, se crea un nuevo archivo utilizando una extensión `.N`, con N incrementado según sea necesario. El formato de la salida se adhiere al formato de esquema XML.
importData(URI source, boolean rebuildTopo, Guid mss)	Convierta los datos XML del origen especificado en objetos de modelo que se actualizan en la base de datos de TADDM, de acuerdo con las reglas siguientes: • Si el origen es un archivo, el contenido del único archivo se lee y se inserta. • Si el origen es un directorio, se importan todos los archivos del directorio. • Si el origen es un objeto remoto, como una dirección HTTP, cada actualización funciona con su propia transacción. Los errores retrotraen solo la actualización del objeto actual y luego la importación continúa.
rebuildTopology()	Recree los datos derivados de la base de datos de TADDM, como las dependencias, las relaciones y la consolidación de datos. Durante esta operación, se bloque la base de datos completa frente a las actualizaciones.
removeArrayElements(Guid object, String attrName, Guid[] elements, Guid mss)	Elimine los elementos especificados del objeto dado de la matriz con nombre de la base de datos de TADDM sin captar el objeto o la matriz del cliente.
startBulkload(long timeoutInSeconds)	Bloquee el subsistema de almacenamiento frente a otros cambios en la base de datos, incluidos los descubrimientos y las sincronizaciones. Tiene que llamar a este método antes de realizar actualizaciones importantes. El bloqueo permanece hasta que se llama al método endBulkload().

Tabla 16. Métodos de gestión de modelos (continuación)

Método	Descripción
update(ModelObject obj, mss) update(ModelObject[] obj, mss)	Actualice o inserte un objeto de modelo en la base de datos. Los atributos definidos en el objeto de origen se fusionan en el destino, mientras que los atributos que no se definen no se actualizan. Si se crea un objeto con el tipo especificado y la clave no existe, se crea un nuevo objeto en la base de datos. Tenga en cuenta los siguientes detalles: • El nuevo objeto debe tener su GUID o un conjunto de claves. Si la clave se refiere a un objeto principal, este debe existir. Un objeto vacío con solo el GUID definido es suficiente para identificar a un objeto principal. • Si el objeto de origen hace referencia directa a un GUID, será responsabilidad del desarrollador que el GUID exista en la base de datos de TADDM. • Si el GUID de mss es un valor nulo, los objetos se insertan o se actualizan en la base de datos de TADDM y no se actualiza ningún enlace de MSS-Objeto. Si el GUID de mss no es un valor nulo, se actualiza el enlace MSS-Objeto. • Puede ser necesario reconstruir la topología para inferir automáticamente dependencias y relaciones explícitas para el nuevo objeto. • Las matrices se reemplazan en su totalidad. **Nota:** Este método no se puede utilizar para actualizar objetos de modelo GroupingPattern o Selector. Genera una ApiException cuando alguno de los objetos proporcionados es del tipo GroupingPattern o Selector. Utilice la API GroupingPattern para todas las operaciones en GroupingPatterns o selectores.
updateXML(String xml, Guid mss)	Igual que update(ModelObject[] obj, mss), con la salvedad de que los objetos se representan como una serie XML. **Nota:** Este método no se puede utilizar para actualizar objetos de modelo GroupingPattern o Selector. Genera una ApiException cuando alguno de los objetos proporcionados es del tipo GroupingPattern o Selector. Utilice la API GroupingPattern para todas las operaciones en GroupingPatterns o selectores.

Ejemplo

En este ejemplo se ilustra cómo comparar objetos.

```
//En primer lugar, busque dos objetos que comparar.

ModelObject mo[] = api.find(
      "SELECT * FROM SunSPARCUnitaryComputerSystem", 3, null, null);

if (mo != null) {
   if (mo.length > 1) {
      ModelObject mo1 = mo[0];
      ModelObject mo2 = mo[1];

      try {
         System.out.println("Comparing " + mo1.getDisplayName() +
            " to " + mo2.getDisplayName());
      } catch (Exception e) {
         e.printStackTrace();
      }

      // ObjectRef es una estructura de datos simple que contiene el GUID y la
      // versión del objeto que se va a comparar.

      ObjectRef objectRef2 = new ObjectRef(mo2.getGuid(), 0);
      ComparisonResult result = api.compare(new ObjectRef(mo1.getGuid(), 0),
         new ObjectRef[]{objectRef2},
         new CompareOptions(true));
         handleModel((TreeTableModel)result);
   }
}

public void handleModel(TreeTableModel model) {
   CompareResultRow row = model.getRoot();
   handleRow(model, row, "");
}

private void handleRow(
      TreeTableModel model, CompareResultRow row, String attributeName){
   System.out.println("Handling row " + row);

   int nColumns = model.getColumnCount();

   for (int i = 0; i < nColumns; i++) {
      String columnName = model.getColumnName(i);
      Object value = model.getValueAt(row, i);
      System.out.println("Col Name " + columnName + " value " + value);

      //Primera columna, es el attributeName
      if (i == 0) {
         if (!"".equals(attributeName)) {
            attributeName = attributeName + ":" + String.valueOf(value);
         } else {
            attributeName = String.valueOf(value);
         }
      }

      // Calcule el nombre de la columna y persista en la base de datos aquí.
   }

   List children = row.getChildren();

   if (children != null) {
      Iterator it = children.iterator();
      while (it.hasNext()) {
         CompareResultRow resultRow = (CompareResultRow) it.next();  //recurse
         handleRow(model, resultRow, attributeName);
      }
   }
}
```

Gestión de relaciones:

Los métodos de relación gestionan las relaciones entre objetos del modelo de datos común. Puede utilizar los métodos de relación para añadir y suprimir relaciones entre elementos gestionados en el modelo de datos común. Puede utilizar este método para recuperar un gráfico de relación para un tipo de relación determinado.

Tabla 17 describe los métodos de relación que puede utilizar.

Tabla 17. Métodos de relación

Método	Descripción
addRelationships(Relationship[] relationships, Guid mss)	Añada una o varias relaciones a la base de datos de TADDM. Es necesario que los elementos gestionados de origen y destino de una relación existan antes de añadir una relación entre los elementos. Una instancia de relación no puede existir en la base de datos de TADDM por sí misma. Debe ser descubierta por, o ser propiedad de, uno o varios sistemas de software gestionado (MSS).
deleteRelationships(Guid[] guids, Guid mss) deleteRelationships(String type, Guid source, Guid target, Guid mss)	Elimine una o varias relaciones de la base de datos de TADDM en cualquiera de estos casos: • Cuando no se ha especificado ningún sistema de software gestionado • Cuando una relación es propiedad solo del MSS especificado La relación no se suprime cuando se especifica un MSS y una relación es propiedad de otro MSS. En este caso, solo se suprime la asociación entre la relación y el MSS.
findRelationships(Guid managedElementGuid, int direction, String type, int scope, String[] permissions)	Recupere el gráfico de relación del tipo de relación dado, empezando por el elemento gestionado especificado. Las relaciones almacenadas en la base de datos de TADDM se pueden atravesar en las direcciones siguientes: • A partir de un elemento gestionado de origen y hacia delante • A partir de un elemento gestionado de destino y hacia atrás
findRelationships(Guid[] sourceGuids, Guid[] targetGuids, String[] types, int comparisonFlags) findrelationships(managedElementGuid => findRelationships(managedElementGuid	Recupera relaciones con solo información básica devuelta. Este método se ejecuta más rápido que el método findrelationships(managedElementGuid, direction, type, scope, permissions).

Gestión de sesiones:

Los métodos de sesión establecen conexiones y sesiones con el servidor. Puede utilizar los métodos de la sesión para abrir y cerrar sesiones con el servidor de TADDM y recuperar la conexión actual.

Tabla 18 describe los métodos de sesión que puede utilizar.

Tabla 18. Métodos de sesión

Método	Descripción
close()	Cierre una sesión.
ApiFactory.getInstance().getApiConnection (String host, int port, String trustStoreLocation, boolean useSSL)	Cree una conexión utilizando el host y el puerto especificados. Si el valor de puerto especificado es `-1`, se utiliza el puerto predeterminado especificado en el archivo `collation.properties`. El parámetro trustStoreLocation especifica la ubicación del archivo de certificado que se va a utilizar con las conexiones SSL.
ApiFactory.getInstance().getSession (ApiConnection conn, String user, String password, long version)	Devuelva un objeto de sesión, que se utiliza para ejecutar métodos de la API de TADDM.
ApiFactory.getInstance().getSession (ApiConnection conn, long sessionId, long version)	Devuelva un objeto de sesión, que se utiliza para ejecutar métodos de la API de TADDM.
release()	Este método no se admite, utilice el método `close()`

Ejemplos

La conexión con el servidor de TADDM implica el establecimiento de una conexión con el servidor y el inicio de sesión con una cuenta de usuario y una contraseña para establecer una sesión. En el ejemplo siguiente, se muestra cómo establecer una conexión con el servidor:

```
private ApiConnection conn_;
   try {
      conn_ = ApiFactory.getInstance().getConnection(
         "host.abcxyz.com",            //nombre de host
        9433,                         //número de puerto
         null,  //Ubicación del archivo jssecacerts.cert para SSL
         false);                      //true para SSL, false para no SSL
   } catch (Throwable th) {
        th.printStackTrace();
}
```

También puede establecer una conexión SSL, como se muestra a continuación:

```
private ApiConnection conn_;
   try {
      conn_ = ApiFactory.getInstance().getConnection(
         "host.abcxyz.com",            //nombre de host
        9433,                         //número de puerto
         "c:\\temp\\jssecacerts.cert", //Ubicación jssecacerts.cert para SSL
         true);                       //true para SSL, false para no SSL
   } catch (Throwable th) {
        th.printStackTrace();
}
```

El servidor de TADDM utiliza el puerto 9433 de forma predeterminada, aunque se puede especificar un puerto alternativo si se especifica el valor adecuado en el archivo `$COLLATION_HOME/etc/collation.properties`.

Al establecer una conexión SSL, debe especificar la ubicación del archivo `jssecacerts.cert`. Descargue este archivo del Java Web Portal del servidor de TADDM, al que puede acceder en http://*nombre_servidor*:*puerto servidor_web*, por ejemplo, http://localhost:9430.

Una vez establecida ApiConnection, inicie la sesión en el servidor para establecer una sesión, como se muestra en el ejemplo siguiente:

```
String user = "smartoperator";     // nombre de usuario de inicio de sesión
String password = "foobar";        // contraseña de usuario
long version = 0;                  // version
ApiSession session_ = ApiFactory.getInstance().getSession(conn_, user, password,
  version);
CMDBApi api = session_.createCMDBApi();
```

CMDBApi es la referencia remota para realizar todas las operaciones relacionadas con la API en el servidor de TADDM.

Gestión de transacciones:

La API de transacciones está en desuso.

Los métodos siguientes solo registran mensajes de aviso y las transacciones no se iniciarán, confirmarán ni retrotraerán:

beginTransaction()

beginTransaction(int timout)

commitTransaction()

rollbackTransaction()

Gestión de versiones:

Los métodos de versión gestionan las versiones de datos de la base de datos de TADDM. Puede utilizar los métodos de la versión para crear instantáneas con nombre de los datos de la base de datos de TADDM actual, suprimir versiones y listar las versiones definidas disponibles.

Tabla 19 describe los métodos de versión que puede utilizar.

Tabla 19. Métodos de versión

Método	Descripción
createEmptyVersion(name, description)	Cree una versión vacía sin datos.
createVersion(name, description)	Cree una instantánea con nombre de los datos de la base de datos de TADDM actual.
deleteVersion(versionID)	Suprima la versión especificada de la base de datos de TADDM.
getAllVersions()	Devuelva los nombres de todas las versiones de datos definidas de la base de datos de TADDM.
getVersion()	Devuelva el objeto TopologyVersion de la versión actual de los datos de la base de datos de TADDM visualizados.

Metadatos:

Los métodos de metadatos gestionan los metadatos del modelo de datos común. Puede utilizar los métodos de metadatos para añadir, actualizar o eliminar los atributos ampliados, así como para definir los valores asociados. También puede utilizar los métodos para devolver información de metadatos en el modelo de datos común, incluidos el número, los tipos y los nombres de todos los atributos de un objeto de modelo.

Tabla 20 describe los métodos de metadatos que se pueden utilizar.

Tabla 20. Métodos de metadatos

Método	Descripción
defineExtendedAttributeMeta(UserDataMeta udm)	Añade o actualiza metadatos de atributos ampliados.
getAllMetaData()	Devuelve los metadatos acerca del modelo de datos comunes sin su parte de modelo simplificado. No muestra ninguna clase del modelo simplificado y ningún movimiento o cambio de atributos y relaciones.
getAllMetaData(boolean flatten, Locale locale, boolean skipSimplifiedModel)	Devuelve todos los metadatos. Este método es equivalente a find("ObjectClass", ...) donde se utiliza una memoria caché de metadatos en el servidor para acceder más rápido.
getClassNames()	Devuelve una matriz de nombres abreviados de clase de modelo, pares de nombre completos.
getExtendedAttributeMeta(String classname)	Recupera metadatos de atributos ampliados de una clase. El método recupera tanto los metadatos de `taddm_global` como metadatos de atributos ampliados de categoría personalizada. Los objetos `UserDataMeta` se recopilan de la clase especificada y de todos los elementos que están más altos en la jerarquía que la clase especificada.
getExtendedAttributes(Guid objGuid)	Recupera valores de atributos ampliados sobre un objeto. **Nota:** Este método está en desuso.
getMetaData(String className)	Devuelve el número, los tipos y los nombres de todos los atributos del objeto de modelo. Este método incluye información de tipos numerados, regla de clave/nombre, contención y relación.
removeExtendedAttributeMeta(String classname, Guid acct)	Elimina atributos ampliados de toda la clase o atributos ampliados de una cuenta y una clase especificadas. **Nota:** Este método está en desuso.
setExtendedAttributes(Guid objGuid, AttrNameValue[]	Define los valores de los atributos ampliados. **Nota:** Este método está en desuso. En su lugar, utilice el atributo `XA` de los objetos.

Presentación:

Los métodos de presentación determinan los sistemas afectados y devuelven información de topología. Puede utilizar los métodos de presentación para determinar los servicios y aplicaciones empresariales afectados en función de elementos de configuración específicos. También puede utilizar los métodos para devolver detalles de objetos de modelo, gráficos y topologías.

Tabla 21 describe los métodos de presentación que se pueden utilizar.

Tabla 21. Métodos de presentación

Método	Descripción
findImpactedBusinessApplications(Guid[] objects)	Determine las aplicaciones empresariales que se ven afectadas según la matriz especificada de elementos de configuración.
findImpactedBusinessServices(Guid[] objects)	Determine los servicios empresariales que se ven afectados según la matriz especificada de elementos de configuración.
getDetailsPanel(ObjectRef ref)	Devuelva el panel de detalles de la referencia a objeto especificada. Una referencia a objeto es la combinación de la versión y el GUID del objeto.
getGraphView(ViewDefiner graphView)	Devuelva un gráfico para el ViewDefiner especificado que describe la vista gráfica.
getGraphViewImage(ViewDefiner graphView) **Nota:** Fix Pack 3 Este método está en desuso.	Devuelve un objeto ImageStream para el ViewDefiner especificado que describe la vista gráfica.
getTreeView(ViewDefiner treeView)	Devuelva un objeto TopologyTreeModel para el ViewDefiner especificado que describe la vista de árbol.

Código Java de ejemplo

- El siguiente código Java de ejemplo ilustra cómo recuperar el panel de detalles de los objetos utilizando el GUID.

  Nota:
  - GUID es un identificador exclusivo, utilizado para identificar objetos mientras se almacenan los objetos en la base de datos.
  - ObjectRef es una estructura de datos simple que mantiene el GUID y la versión del objeto para el que quiere recuperar el panel de detalles.
  - Puede recuperar el GUID del objeto que necesita especificando un mandato parecido al siguiente. Este mandato concreto le proporciona una lista de todos los objetos asociados a ComputerSystem.

    ```
    SELECT * FROM ComputerSystem
    ```

```
ModelObject mo[] = api.find(
      "SELECT * FROM ComputerSystem",
      1,
      null,
      null);

if (mo != null) {
   for (int i=0; i<mo.length; i++){
      Guid guid = mo[i].getGuid();
      System.out.println("Getting details panel for " + guid);
      DetailPanelModel model = api.getDetailsPanel(
         new ObjectRef(guid,0));     //guid and version
      System.out.println("model is " + model);
   }
}
```

- El ejemplo siguiente muestra cómo recuperar las vistas gráficas.

  La enumeración de gráficas y árboles se define en la clase com.collation.proxy.api.presentation.common.ViewDefinerEnum.

```
ViewDefiner viewDefiner = new ViewDefiner(
      ViewDefinerEnum.GRAPH_APPLICATION_PHYSICAL_INFRASTRUCTURE,
            VersionedObject.DYNAMIC);
TopologyGraphModel gv = api.getGraphView(viewDefiner);
```

- El ejemplo siguiente muestra cómo encontrar servicios empresariales y aplicaciones que se han visto afectados:

```
// Buscar los objetos para el análisis de impacto

ModelObject mo[] = api.find("SELECT * FROM ApacheServer", 1, null, null);

if (mo != null) {
   Application[] applications = api.findImpactedBusinessApplications(
         new Guid[]{mo[0].getGuid()});

   if (applications != null) {
      for (int i=0;i<applications.length;i++) {
         System.out.println(applications[i].getDisplayName());
      }
   }

   BusinessSystem[] systems = api.findImpactedBusinessServices(
         new Guid[]{mo[0].getGuid()});
   if (systems != null) {
      for (int i=0;i<systems.length;i++) {
         System.out.println(systems[i].getDisplayName());
      }
   }
}
```

Seguridad:

Los métodos de seguridad gestionan permisos, titularidades y roles en la base de datos de TADDM. Puede utilizar los métodos de seguridad para añadir y eliminar permisos, así como para determinar los permisos y las titularidades de usuarios específicos. Los métodos también se puede utilizar para determinar los roles asignados a un usuario y para determinar si un usuario tiene acceso a una o más operaciones de tiempo de ejecución.

Nota: Los métodos de seguridad funcionan en base a objetos CustomCollection con el atributo hierarchyType definido como "AccessCollection". No admiten objetos AccessCollection antiguos.

Tabla 22 describe los métodos de seguridad que se pueden utilizar.

Tabla 22. Métodos de seguridad

Método	Descripción
addAccess(Principal user, Resource resource, String role, String[] permissions)	Añada a un rol permisos para un objeto específico.
addRuntimeAccess(Principal user, String role, String[] permissions)	Añada a un rol permisos para una o más operaciones de tiempo de ejecución .
assignPersonInRoleToAccessCollection (Person user, Role role, Guid[] guids, long[] versionId)	Cree una asignación (potencialmente en varias versiones) entre una persona de un rol y una lista de colecciones de accesos.
deleteAccess(Principal user, Resource resource, String role, String[] permissions)	Suprima de un rol un permiso para una colección específica.
deleteRuntimeAccess(Principal user, String role, String [] permissions)	Suprima de un rol el permiso para una o varias operaciones de tiempo de ejecución.
getAccessDecisions(Principal user, Resource[] resources, String[] permissions)	Determine si el interlocutor puede acceder a uno o varios objetos con el permiso especificado.
getDataPermissions(Principal user, Resource[] resources)	Determine los permisos de nivel de datos que tiene un usuario para un conjunto de objetos.
getEntitlements(Principal user, String[] permissions)	Recupere las titularidades del usuario. Las titularidades son los objetos a los que puede acceder el usuario, según las políticas de seguridad definidas.
getRoles(Principal user)	Recupere los roles asignados a un usuario.
getRuntimeAccess(Principal user)	Recupere los permisos para operaciones de tiempo de ejecución correspondientes a un usuario.
getRuntimeAccessDecisions(Principal user, String[] permissions)	Determine si un usuario tiene acceso a una o varias operaciones de tiempo de ejecución.
removePersonInRoleToAccessCollection (Person user, Role role, Guid[] guids, long[] versionId)	Elimine una asignación que podría estar en varias versiones según la persona, el rol y las colecciones de listas de accesos que se hayan especificado y a las que se haya asignado a la persona del rol.
addAccess(Principal user, AccessDefinition[] accessDefinition)	Añada uno o más objetos (cada uno con un conjunto de permisos) a un rol, según lo especificado por cada objeto AccessDefinition.
addDataPermissionToRole(String role, String permission)	Añada un permiso de datos a un rol siempre que el rol exista en las políticas almacenadas.
addRuntimePermissionToRole(String role, String permission)	Añada un permiso de tiempo de ejecución a un rol.
deletePermission(String permission)	Elimine un permiso siempre que este permiso exista en las políticas almacenadas.
deletePermissionFromRole(String role, String permission)	Elimine un permiso de un rol siempre que dicho rol exista en las políticas almacenadas.
deleteRole(String role)	Elimine un rol, siempre que dicho rol exista en las políticas almacenadas.

Tabla 22. Métodos de seguridad (continuación)

Método	Descripción
getEntitlementsForRole(Principal user, String role)	Recupere las titularidades correspondientes al usuario de un rol especificado.

Gestión de plantillas de aplicación:

Los métodos de plantilla de aplicación permiten la creación, la modificación, la supresión y la recuperación de plantillas de aplicación y reglas.

Importante: Esta API está en desuso para TADDM 7.3 y versiones posteriores. GroupingPatternAPI se puede utilizar para gestionar servicios empresariales. Esta API proporciona métodos para la creación, la modificación y la supresión de patrones de agrupación. Para obtener más información, consulte “Gestión de patrones de agrupación” en la página 84.

Las plantillas de la aplicación especifican las reglas MQL que se aplican de manera periódica a la base de datos de TADDM para definir las recopilaciones o aplicaciones empresariales. Cada plantilla especifica una o varias reglas con los atributos siguientes:

MQLRuleName
Nombre de la regla MQL. Este atributo es necesario.

FunctionalGroupName
El objeto que contiene el grupo funcional y que la consulta MQL devuelve. Este atributo es necesario para cualquier regla que defina una aplicación empresarial. No se utiliza en el caso de recopilaciones definidas por reglas.

MQLQuery
Consulta MQL que ejecutar. Los objetos que esta consulta devuelve se añaden a la recopilación o la aplicación empresarial.

Tabla 23 en la página 82 describe los métodos de la plantilla de la aplicación que se pueden utilizar.

Tabla 23. Métodos de la plantilla de aplicación

Método	Descripción
createAppTemplate(String name, String type, boolean removeNonMembers, MQLRule[] operators)	Crea una plantilla con las reglas especificadas. Están disponibles los parámetros siguientes: **name** Nombre de la plantilla de la aplicación que se va a crear. **type** Tipo de plantilla que crear (aplicación, recopilación o servicio). Los valores válidos son "Application" para una aplicación empresarial, "Collection" para recopilaciones o "Service" para un servicio empresarial. **removeNonMembers** Valor booleano que especifica si los objetos existentes en la recopilación o la aplicación empresarial se pueden eliminar si ya no coinciden con las reglas de la plantilla. **MQLRule** Matriz que contiene una o varias reglas.
updateAppTemplate(String name, String type, boolean removeNonMembers, MQLRule[] operators)	Actualiza una plantilla con las reglas especificadas. Están disponibles los parámetros siguientes: **name** Nombre de la plantilla de aplicación que actualizar. **type** Tipo de plantilla que actualizar (aplicación, recopilación o servicio). Los valores válidos son "Application" para una aplicación empresarial, "Collection" para recopilaciones o "Service" para un servicio empresarial. **removeNonMembers** Valor booleano que especifica si los objetos existentes en la recopilación o la aplicación empresarial se pueden eliminar si ya no coinciden con las reglas de la plantilla. **MQLRule** Matriz que contiene una o varias reglas.
getAllAppTemplates()	Recupera todas las plantillas de la aplicación.
getAppTemplate(String name, int type)	Recupera la plantilla de aplicación para el nombre y el tipo especificados.
removeAppTemplate(String name, int type)	Elimina la plantilla de aplicación para el nombre y el tipo especificados.
removeMQLRule(String name)	Elimina una regla MQL. Una regla solo se puede eliminar si no está asociada a ninguna plantilla de aplicación.

Creación de una plantilla de aplicación empresarial

Para crear una plantilla de aplicación para una aplicación empresarial, complete los pasos siguientes:

1. Cree una aplicación empresarial y defina el nombre de la nueva aplicación empresarial nueva con el nombre pasado en la línea de mandatos; por ejemplo, `TADDM - Production`. Haga que se devuelva el GUID.
2. Defina el nombre de la plantilla de aplicación con el siguiente formato:

   GUID_apl_empresarial:*nombre_apl_empresarial*

   Por ejemplo:

   ```
   AA0A20EE5BBD336481279CA664FB380A:TADDM - Production
   ```
3. Use el GUID de la aplicación empresarial como prefijo para los nombres de regla, pero asegúrese de no incluir el signo de dos puntos en el nombre de la regla; por ejemplo:

   ```
   AA0A20EE5BBD336481279CA664FB380Adatabase
   ```

Ejemplo: Creación de una plantilla de aplicación empresarial

El ejemplo siguiente crea una plantilla de aplicación empresarial, las reglas MQL y la aplicación empresarial asociada:

```
# crear aplicación empresarial con el nombre "TADDM - Production"
BAname = "TADDM - Production"

# obtener el GUID de la aplicación empresarial
myBA = ModelFactory.newInstance(Class.forName
("com.collation.platform.model.topology.app.Application"))
myBA.setName(BAname)
appGuid = api.update(myBA,None)
MQLRuleClass = Class.forName("com.collation.platform.model.apptemplate.MQLRule")
rules = [ModelObjectFactory.newInstance(MQLRuleClass)]

# crear el nombre requerido por la regla MQL de TADDM,
# como AA0A20EE5BBD336481279CA664FB380Adatabase
rulename = "database"
RuleName= str(appGuid) + rulename
asQuery="select * from AppServer "
rules[0].setMQLRuleName(RuleName)
rules[0].setFunctionalGroupName("App Servers")
rules[0].setMQLQuery(asQuery)

# crear el nombre requerido por la plantilla de aplicación de TADDM, como
AA0A20EE5BBD336481279CA664FB380A:TADDM - Production
appTemplateName= str(appGuid) + ":" + BAname
appTemplate = api.createAppTemplate(appTemplateName, "APPLICATION", True,
jarray.array(rules,MQLRuleClass));
```

Ejemplo: Listado de aplicaciones empresariales

El ejemplo siguiente lista aplicaciones empresariales:

```
query = "select * from Application"
data = api.executeQuery(query, None, None)
while (data.next()):
 print data.getXML(4)
```

Ejemplo: Eliminación de una plantilla de aplicación empresarial

El ejemplo siguiente elimina una plantilla de aplicación, las reglas MQL y la aplicación empresarial asociada.

```
# Obtener plantilla de aplicación
appTemplate = api.getAppTemplate(nameBA, 0)

# Obtener las reglas de la plantilla de aplicación
rules = appTemplate.getMQLRules()

# Eliminar la plantilla de aplicación -> elimina solo el objeto AppTemplate
api.removeAppTemplate(appTemplate.getAppTemplateName(), appTemplate.getAppTemplateType())

# Eliminar todas las reglas
for rule in rules:
rule = api.find(rule.getGuid(), 1, None)
api.removeMQLRule(rule.getMQLRuleName())

# Eliminar la aplicación empresarial
businessApp = api.find("Select * from Application where name =='" +
appTemplate.getAppTemplateName()+ "'", False, None, None)
api.delete(businessApp, None)
```

Gestión de patrones de agrupación:

Los métodos de patrones de agrupación permiten la creación, la modificación, la supresión y la recuperación de patrones de agrupación junto con sus selectores.

Para obtener detalles, consulte el tema *Gestión de patrones de agrupación utilizando la API de Java* de la *Guía del usuario* de TADDM.

Utilización de la API de SOAP

La API del Protocolo de acceso a objectos simple (SOAP) expone elementos de la API de TADDM como servicio web.

Con la API de SOAP, puede desarrollar aplicaciones en distintos entornos de desarrollo y sistemas operativos que admiten la integración con aplicaciones de gestión, como los gestores de procesos de ITSM.

La API de SOAP ofrece control sobre el proceso de descubrimiento y aspectos del modelo de datos común, incluido el acceso a los datos de modelo resultantes. La API de SOAP delega las solicitudes en la API de Java. La API de Java puede crear aplicaciones que añaden, actualizan y suprimen objetos de modelo. Puede consultar objetos de modelo por nombre de clase o número de ID de objeto.

Puede utilizar el control de la API de SOAP para crear aplicaciones que añaden, actualizan y suprimen objetos de modelo. SOAP puede consultar objetos de modelo por nombre de clase o por número de ID de objeto. También puede utilizar la interfaz para examinar el historial de cambios y gestionar las versiones.

Resumen de la solicitud

La API de SOAP ofrece acceso a las correlaciones de aplicaciones de TADDM, incluidas las aplicaciones de descubrimiento, sus componentes y las configuraciones.

La API de SOAP se puede resumir utilizando las categorías siguientes, que se explican con todo detalle en las secciones correspondientes:

- Solicitudes de sesión

- Solicitudes de descubrimiento
- Gestión del modelo y los metadatos
- Solicitudes de búsqueda
- Solicitudes del historial de cambios
- Gestión de versiones

Solicitudes de sesión:

Las solicitudes de sesión le permite gestionar sesiones con el servidor de TADDM.

Puede utilizar las solicitudes de sesión para iniciar y cerrar la sesión en el servidor de TADDM. Tabla 24 describe las solicitudes de sesión que puede utilizar.

Tabla 24. Solicitudes de sesión

Operación	Entrada	Salida
login (Inicie la sesión en el servidor de TADDM)	loginRequest **user** Nombre de usuario registrado con el servidor de TADDM **password** Contraseña asociada al usuario **host** Nombre del host, sea un nombre o una dirección IP (utilizando la notación con puntos) **port** Número de puerto del servidor	loginReponse
cierre de sesión (Finalizar sesión en el servidor)	logoutRequest	logoutReponse

Ejemplo

En el ejemplo siguiente, se muestra una solicitud XML de inicio de sesión:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:login soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
      <ns1:arg0 xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">smartoperator</ns1:arg0>
      <ns1:arg1 xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">foobar</ns1:arg1>
      <ns1:arg2 xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">localhost</ns1:arg2>
      </ns1:login>
   </soapenv:Body>
</soapenv:Envelope>
```

En el ejemplo siguiente, se muestra la respuesta XML de inicio de sesión:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:loginResponse soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
      <loginReturn xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">1149902064172</loginReturn>
      </ns1:loginResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Solicitudes de descubrimiento:

Los métodos de descubrimiento le permiten gestionar ejecuciones de descubrimiento.

Puede utilizar las solicitudes de descubrimiento para iniciar y detener descubrimientos, obtener el estado de un descubrimiento y borrar todos los elementos de descubrimiento de la topología. Tabla 25 describe las solicitudes de descubrimiento que puede utilizar.

Tabla 25. Solicitudes de descubrimiento

Operación	Entrada	Salida
abortDiscovery (Aborte el descubrimiento activo en este momento)	abortDiscoveryRequest	abortDiscoveryResponse
clearTopology (Borre todos los elementos de descubrimiento y las relaciones de la topología)	clearTopologyRequest	clearTopologyResponse
getStatus (Obtenga el estado de la ejecución de descubrimiento actual)	getStatusRequest	getStatusResponse getStatusReturn: representación de serie del estado de la ejecución de descubrimiento actual
rebuildTopology (Recree la topología, incluidas las dependencias y relaciones)	rebuildTopologyRequest	rebuildTopologyResponse
startDiscovery (Inicie un descubrimiento utilizando el ámbito especificado)	startDiscoveryRequest **scope** Ámbito del descubrimiento: nombre de un conjunto de ámbitos o de un grupo de ámbitos. **runName** Nombre de la ejecución de descubrimiento.	startDiscoveryResponse

Tabla 25. Solicitudes de descubrimiento (continuación)

Operación	Entrada	Salida
startDiscoveryRetID (Inicie un descubrimiento utilizando el ámbito especificado)	startDiscoveryRequest **scope** Ámbito del descubrimiento: nombre de un conjunto de ámbitos o de un grupo de ámbitos. **runName** Nombre de la ejecución de descubrimiento.	startDiscoveryResponse ID de ejecución

Gestión del modelo y los metadatos:

Las solicitudes de modelo y metadatos gestionan los objetos y los metadatos de consulta en el modelo de datos común. Puede utilizar las solicitudes de modelo para insertar, importar y exportar objetos en el modelo de datos común. Puede utilizar la solicitud de metadatos para obtener todos los nombres de clases del modelo que se pueden utilizar en el lenguaje de consulta.

Tabla 26 describe las solicitudes de metadatos y modelo que se pueden utilizar.

Tabla 26. Solicitudes de modelo y metadatos

Operación	Entrada	Salida
exportData (Exporte todos los objetos de nivel superior de la base de datos de TADDM en un directorio especificado en formato XML)	exportDataRequest **directoryToWriteTo** Nombre del directorio al que se van a exportar los datos **maxfilesize** Tamaño máximo del archivo que se va a exportar **mssGuid** GUID del sistema de software gestionado (MSS)	exportDataResponse

Tabla 26. Solicitudes de modelo y metadatos (continuación)

Operación	Entrada	Salida
exportDataUsingMssName (Exporte todos los objetos de nivel superior de la base de datos de TADDM a un directorio, en formato XML, especificando el MSS mediante un nombre)	exportDataUsingMss NameRequest **directoryToWriteTo** Nombre del directorio al que se van a exportar los datos **maxfilesize** Tamaño máximo del archivo que se va a exportar **mssGuid** GUID del sistema de software gestionado (MSS)	exportDataUsingMss NameResponse
getClassNames (Obtenga todos los nombres de clase del modelo que se pueden utilizar en el lenguaje de consulta)	getClassNamesRequest	getClassNamesResponse getClassNamesReturn: matriz de pares de nombre abreviado/nombre completo de la clase modelo
importData (Convierta los datos XML del origen especificado en objetos de modelo y actualice la base de datos de TADDM)	importDataRequest **source** Origen desde el que importar los datos **rebuildTopo** Valor booleano para recrear la topología **mssGuid** GUID del sistema de software gestionado (MSS)	importDataResponse
importDataUsingMssName (Convierta los datos XML del origen especificado en objetos de modelo y actualice la base de datos de TADDM, especificando el sistema de software gestionado por nombre)	importDataUsingMss NameRequest **source** Origen desde el que importar los datos **rebuildTopo** Valor booleano para recrear la topología **mssName** Nombre del sistema de software gestionado (MSS)	importDataUsingMss NameResponse
insert (Inserción o actualización del objeto de modelo, especificado en formato XML, en la base de datos de TADDM)	insertRequest **xml** Objeto de modelo en formato XML **mssGuid** GUID del sistema de software gestionado (MSS)	insertResponse

Tabla 26. Solicitudes de modelo y metadatos (continuación)

Operación	Entrada	Salida
insertUsingMssName (Inserte el objeto de modelo, especificado en formato XML, indicando el sistema de software gestionado por nombre)	insertUsingMss NameRequest **xml** Objeto de modelo en formato XML **mssName** Nombre del sistema de software gestionado (MSS)	insertUsingMss NameResponse

Ejemplo

En el ejemplo siguiente, se muestra una solicitud XML getClassNames:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:getClassNames soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost"/>
   </soapenv:Body>
</soapenv:Envelope>
```

En el ejemplo siguiente, se muestra la respuesta XML getClassNames:

```
GETCLASSNAME (response):
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:getClassNamesResponse soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
      <getClassNamesReturn xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">
      AbstractResource,
      com.collation.platform.model.topology.process.AbstractResource,
      Accepts,
      com.collation.platform.model.topology.relation.Accepts,
       .
       .
       .
      </ns1:getClassNamesResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Solicitudes de búsqueda:

Las solicitudes de búsqueda le permiten acceder a objetos del modelo de datos común.

Puede utilizar las solicitudes de búsqueda para devolver objetos de modelo que coinciden con criterios específicos o devolver información sobre elementos gestionados específicos. También puede utilizar las solicitudes para devolver los objetos que han cambiado durante un período específico de tiempo.Tabla 27 en la página 90 describe las operaciones de búsqueda que se pueden realizar.

Tabla 27. Solicitudes de búsqueda

Operación	Entrada	Salida
find (Ejecute una consulta para un elemento de configuración especificado utilizando Model Query Language)	findRequest **query** Serie de consulta **depth** Nivel del árbol de resultados que se va a generar **indent** Sangría que utilizar para el archivo XML resultante **mssGuid** GUID del sistema de software gestionado (MSS)	findResponse findReturn: representación XML de los resultados de la consulta
findBasedOnChange (Busque objetos que hayan cambiado en el periodo especificado, con un tipo de cambio determinado)	findBasedOnChangeRequest **root** Objeto de modelo que sirve como raíz de la salida XML resultante. **depth** Nivel del árbol de resultados que se va a generar **indent** Sangría que utilizar para el archivo XML resultante **start** Hora de inicio del período de cambio, especificado en milisegundos desde el 1 de enero de 1970, 00:00:00 GMT **end** Hora de finalización del período de cambio, especificado en milisegundos desde el 1 de enero de 1970, 00:00:00 GMT **changeType** Tipo de cambio.	findBasedOnChangeResponse findBasedOnChangeReturn: representación XML de los resultados de la consulta
findUsingGuid (Búsqueda utilizando el GUID del objeto de modelo)	findUsingGuidRequest **guid** GUID contra la que ejecutar la búsqueda **depth** Nivel del árbol de resultados que se va a generar **indent** Sangría que utilizar para el archivo XML resultante	findUsingGuidResponse findUsingGuidReturn: representación XML de los resultados de la consulta
findUsingMssName (Búsqueda utilizando Model Query Language, especificando el sistema de software gestionado por nombre)	findUsingMssNameRequest **query** Serie de consulta **depth** Nivel del árbol de resultados que se va a generar **indent** Sangría que utilizar para el archivo XML resultante **mssName** Nombre del sistema de software gestionado (MSS)	findUsingMssNameResponse findUsingMssNameReturn: representación XML de los resultados de la consulta

Ejemplo

En el ejemplo siguiente se muestra una solicitud XML de búsqueda:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:find soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
      <ns1:arg0 xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">ComputerSystem</ns1:arg0>
      <ns1:arg1 href="#id0"/>
      <ns1:arg2 href="#id1"/>
      <ns1:arg3 xsi:nil="true"/>
    </ns1:find>
    <multiRef id="id1" soapenc:root="0" soapenv:encodingStyle=
       "http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int"
       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">4</multiRef>
    <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle=
       "http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int"
       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">2</multiRef>
   </soapenv:Body>
</soapenv:Envelope>
```

En el ejemplo siguiente se muestra la respuesta XML de búsqueda:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:findResponse soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
         <findReturn xsi:type="soapenc:string" xmlns:soapenc=
            "http://schemas.xmlsoap.org/soap/encoding/">
         &lt;?xml version=&quot;1.0&quot; encoding=&quot;ISO-8859-1&quot;?&gt;
         &lt;results
         xmlns=&quot;urn:www-collation-com:1.0&quot;
          .
          .
          .
         &lt;architecture&gt;Intel&lt;/architecture&gt;
         &lt;/ComputerSystem&gt;
         &lt;/results&gt;
         </findReturn>
      </ns1:findResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Solicitudes del historial de cambios:

Las solicitudes del historial de cambios le permiten determinar el historial de cambios del modelo de datos común.

Puede utilizar las solicitudes del historial de cambios para recuperar el historial de cambios de elementos gestionados en el modelo de datos común. Tabla 28 en la página 92 describe las solicitudes del historial de cambios que se pueden utilizar.

Tabla 28. Solicitudes del historial de cambios

Operación	Entrada	Salida
getChangeHistory (Obtenga el historial de cambios del período inicial y el final utilizando el GUID especificado)	getChangeHistoryRequest **guid** GUID del objeto para el que se requiere el historial de cambios **start** Hora de inicio del período de cambio, especificado en milisegundos desde el 1 de enero de 1970, 00:00:00 GMT **end** Hora de finalización del período de cambio, especificado en milisegundos desde el 1 de enero de 1970, 00:00:00 GMT	getChangeHistoryResponse getChangeHistoryReturn: representación XML de una lista de objetos ChangeHistory
getChangeHistory (Obtenga el historial de cambios del período inicial y final de varios GUID)	getChangeHistoryRequest1 **guids** Lista de GUID separados por comas de los objetos para los que se requiere el historial de cambios **start** Hora de inicio del período de cambio, especificado en milisegundos desde el 1 de enero de 1970, 00:00:00 GMT **end** Hora de finalización del período de cambio, especificado en milisegundos desde el 1 de enero de 1970, 00:00:00 GMT	getChangeHistoryResponse1 getChangeHistoryReturn: representación XML de una lista de objetos ChangeHistory

Gestión de versiones:

Las solicitudes de versiones gestionan las versiones de datos de la base de datos de TADDM. Puede utilizar las solicitudes de versiones para crear instantáneas con nombre de los datos de la base de datos actual de TADDM, suprimir versiones y listar las versiones definidas disponibles.

Tabla 29 en la página 93 describe las solicitudes de versiones que puede utilizar.

Tabla 29. Solicitudes de versiones

Operación	Entrada	Salida
createVersion (Cree una instantánea con nombre de los datos de la base de datos actual de TADDM)	createVersionRequest **name** Nombre de la versión **description** Descripción de la nueva versión	createVersionResponse
createEmptyVersion (Cree una versión vacía, sin datos, en la base de datos de TADDM)	createEmptyVersionRequest **name** Nombre de la versión **description** Descripción de la nueva versión	createEmptyVersionResponse
deleteVersion (suprima una versión de la base de datos de TADDM)	deleteVersionRequest **versionID** Identificador de la versión	deleteVersionResponse
deleteVersionUsingName (Suprima una versión, identificada por nombre, de la base de datos de TADDM)	deleteVersionUsing NameRequest **versionName** Nombre de la versión	deleteVersionUsing NameResponse
getAllVersions (Obtenga los nombres de todas las versiones definidas de datos de la base de datos de TADDM)	getAllVersionsRequest	getAllVersionsResponse

Desarrollo de aplicaciones utilizando la API de REST

Puede utilizar la API de REST de TADDM para desarrollar aplicaciones que accedan a recursos seleccionados de TADDM utilizando los principios HTTP y REST.

Visión general de la API de REST

La API de REST expone un subconjunto de las funciones de la API de Java a los clientes y exploradores web que utilizan HTTP. Con los recursos de REST, puede desarrollar aplicaciones para los sistemas operativos y lenguajes que admiten llamadas HTTP.

Los recursos de REST exponen las funciones de TADDM que puede utilizar para consultar los objetos de modelo por nombre de clase, por identificador exclusivo global (GUID) o con consultas de Model Query Language (MQL). También puede crear, suprimir y actualizar objetos de modelo, además de gestionar el proceso de descubrimiento de TADDM. Todas estas funciones utilizan interfaces HTTP estándar y admiten el formato JSON o XML para datos de entrada y de salida.

Los componentes de servidor REST están instalados en el directorio `$COLLATION_HOME/deploy-tomcat` (TADDM 7.3.0) o `$COLLATION_HOME/apps` (TADDM 7.3.0.1 y posterior) en el servidor TADDM y se inician automáticamente cuando se inicia el servidor TADDM. Los servicios de REST están disponibles utilizando los

mismos puertos TCP/IP que utiliza la interfaz web administrativa de TADDM. (Los puertos predeterminados son 9430 para HTTP y 9431 para HTTPS.)

La API de REST utiliza la autenticación HTTP básica para transmitir el ID de usuario y la contraseña utilizando la codificación Base64 de MIME. Dado que las solicitudes no tienen estado, cada llamada a la API de REST debe incluir la cabecera de autorización HTTP. En el caso de conexiones seguras, utilice una conexión HTTPS.

Los parámetros de las llamadas a REST se especifican utilizando la notación estándar de la serie de consulta:

`http://url_recurso?parámetro=valor&parámetro=valor...`

Si especifica un valor de parámetro no válido, el servidor de TADDM lo omitirá y utilizará en su lugar el valor predeterminado, si consigue determinarlo (por ejemplo, si especifica `fetchSize=-2`, el servidor utiliza un valor **`fetchSize`** de 1). Si no se puede determinar ningún valor predeterminado, la solicitud fallará.

Realización de llamadas a REST con un explorador web

Puede realizar muchas llamadas a la API de REST si especifica los URL adecuados en un explorador web.

Antes de empezar

Para acceder a las interfaces de REST de manera segura utilizando una conexión HTTPS, antes debe configurar el explorador para que acepte conexiones de TLS 1.0 (seguridad de capa de transporte).

Acerca de esta tarea

La API de REST utiliza varios métodos HTTP para realizar distintas acciones en los recursos de REST. Cualquier llamada a la API de REST que utilice el método HTTP GET se puede enviar utilizando un explorador web como Microsoft Internet Explorer o Mozilla Firefox.

Procedimiento

Para acceder a una llamada de REST con un explorador, siga estos pasos:

Especifique el URL adecuado utilizando HTTP o HTTPS.

- En este ejemplo, se envía una consulta especificada con Model Query Language utilizando HTTP:

  ```
  http://yourhost.com:9430/rest/model/MQLQuery?query=select%20name%20from%
  20ComputerSystem%20where%20signature%20starts-with%20'M&Y'&feed=xml&fetc
  hSize=100&position=1
  ```

- En este ejemplo, se muestra una consulta ComputerSystem enviada utilizando HTTPS:

  ```
  https://yourhost.com:9431/rest/model/ComputerSystem?depth=1
  ```

La primera vez que acceda a la API de REST de TADDM con un explorador, verá una página de inicio de sesión donde se solicita un ID de usuario y una contraseña válidos de TADDM.

Realización de llamadas a REST en una aplicación Java

Puede utilizar métodos Java estándar para acceder a la API de REST de TADDM.

Antes de empezar

Para acceder a las interfaces de REST de manera segura utilizando una conexión HTTPS, antes debe copiar el certificado de seguridad jssecacerts.cert en el sistema del cliente. Este archivo se encuentra en el directorio `$COLLATION_HOME/etc` del servidor de TADDM.

Procedimiento

En el ejemplo siguiente se muestra cómo acceder a la API de REST desde un programa Java.

Para acceder a la API de REST desde un programa Java, utilice los métodos Java estándar para la comunicación HTTP. En este ejemplo, se accede a la API de REST utilizando una conexión HTTPS segura:

```
HostnameVerifier hv = new HostnameVerifier() {
             public boolean verify(String urlHostName, SSLSession session) {
                System.out.println("Warning: URL Host: "+urlHostName+" vs. "
                  +session.getPeerHost());
                return true;
             }
         };

         // defina esta propiedad en la ubicación del archivo cert
         System.setProperty("javax.net.ssl.trustStore", "jssecacerts.cert");

         HttpsURLConnection.setDefaultHostnameVerifier(hv);
         URL url = new
           URL("https://cab.tivlab.austin.ibm.com:9431/rest/model/"+
               "Repository?depth=1&feed=json");
         HttpsURLConnection urlConn = (HttpsURLConnection) url.openConnection();

         System.out.println("sending request...");
         urlConn.setRequestMethod("GET");
         urlConn.setAllowUserInteraction(false); // sin interacción del usuario
         urlConn.setDoOutput(true); // para enviar
         urlConn.setRequestProperty( "Content-type", "text/xml" );
         urlConn.setRequestProperty( "accept", "text/xml" );
         urlConn.setRequestProperty( "authorization", "Basic " +
             encode("administrator:collation"));
         Map headerFields = urlConn.getHeaderFields();
         System.out.println("header fields are: " + headerFields);

         int rspCode = urlConn.getResponseCode();
         if (rspCode == 200) {
             InputStream ist = urlConn.getInputStream();
             InputStreamReader isr = new InputStreamReader(ist);
             BufferedReader br = new BufferedReader(isr);

             String nextLine = br.readLine();
             while (nextLine != null) {
                 System.out.println(nextLine);
                 nextLine = br.readLine();
             }
         }
```

Análisis de los resultados de consulta de REST

Para analizar los resultados de consulta de REST en una aplicación Java, puede utilizar métodos XPath o JXPath estándar.

Antes de empezar

Asegúrese de tener acceso a una biblioteca de XPath para analizar datos XML o a una biblioteca JXPath para analizar datos JSON. El soporte de XPath se incluye en el IBM SDK Java Technology Edition versión 5 y posteriores. El soporte de JXPath se incluye en el TADDM SDK.

Procedimiento

El ejemplo siguiente muestra cómo utilizar estas funciones para analizar la salida de las llamadas a REST de TADDM.

A continuación, puede utilizar estas funciones para analizar la salida de las llamadas a REST de TADDM. En el ejemplo siguiente, se muestra cómo puede devolver el serviceName de cada uno de los installedServices para un sistema operativo, utilizando JXPath para analizar los datos de salida de JSON.

Nota: La clase JSONArray forma parte del paquete json-simple, que no está incluido en el SDK de TADDM. Para descargar este paquete, vaya a http://code.google.com/p/json-simple/.

```
//queryResult contains the results from a TADDM Query
JSONArray arrayObject = (JSONArray) JSONValue.parse(queryResult);
   JXPathContext context2 = JXPathContext.newContext(arrayObject);
   Iterator names = context2.iterate("//installedServices/serviceName");
   while(names.hasNext()) {
    String serviceName = (String) names.next();
    System.out.println("service name is: " + serviceName);
  }
```

Depuración de aplicaciones de REST

Si la aplicación se está encontrando errores al acceder a la API de REST, hay varias técnicas que puede utilizar para determinar la naturaleza del problema.

Acerca de esta tarea

La API de REST utiliza varios mecanismos para indicar el resultado de las llamadas a REST y los errores que se han producido durante el procesamiento. Utilice estos métodos para depurar una aplicación de REST.

Procedimiento

Utilice los métodos siguientes para depurar una aplicación de REST.

- Compruebe el código de respuesta HTTP. Entre los códigos de respuesta más utilizados se encuentran los siguientes:

  200 La solicitud se ha realizado correctamente.

  400 Los datos de entrada no eran válidos.

  409 El servidor ha encontrado un conflicto, como un intento de añadir un objeto que ya existe.

  500 Se ha producido un error de servidor.
- Compruebe el mensaje de respuesta de la cabecera HTTP para obtener información adicional.
- Compruebe si hay mensajes relevantes en los archivos de registro. Podrían aparecer mensajes en los archivos siguientes:
  - `$COLLATION_HOME/log/tomcat.log` (TADDM 7.3.0)

– $COLLATION_HOME/log/wlp.log (TADDM 7.3.0.1 y posterior)
– $COLLATION_HOME/log/services/ApiServer.log

Consulta de objetos de modelo utilizando la API de REST

Puede utilizar la API de REST para consultar objetos de modelo utilizando uno de estos dos métodos.

Procedimiento

Para utilizar la API de REST y consultar los objetos de modelo, siga uno de estos dos métodos:

- Para consultar los objetos de modelo especificando la clase de objeto de modelo, utilice el recurso de la clase de objeto de modelo. Puede utilizar este recurso para consultar información sobre objetos de modelo de una clase concreta, incluyendo si lo desea valores de atributo especificados. Este recurso proporciona una manera sencilla de consultar objetos de una clase concreta. Este ejemplo consulta el quinto objeto de ComputerSystem cuyo atributo OSRunning está definido en `linux`:

  ```
  http://example.com:9430/rest/model/ComputerSystem?depth=2&feed=xml&OSRu
  nning.OSName=Linux&position=5
  ```

- Para consultar los objetos de modelo utilizando Model Query Language (MQL), utilice el recurso del servicio de consulta MQL. Este recurso admite cualquier consulta que se pueda expresar utilizando MQL y es más flexible que el recurso de clase de objeto de modelo. Este ejemplo consulta los datos del objeto de modelo utilizando la consulta MQL `select displayName,OSRunning.OSName from ComputerSystem`.

  ```
  http://example.com:9430/rest/model/MQLQuery?query=select%20displayName,
  OSRunning.OSName%20from%20ComputerSystem&position=2&fetchSize=2&feed=xm
  l&depth=2&position=4
  ```

Adición de objetos de modelo mediante la API de REST

Puede utilizar la API de REST para añadir un nuevo objeto de modelo utilizando uno de los dos métodos.

Acerca de esta tarea

El recurso del servicio de actualización de objetos de modelo admite la creación de nuevos objetos de modelo utilizando el método HTTP POST o PUT, dependiendo de si quiere o no permitir la modificación de un objeto existente.

En cualquier caso, antes debe describir los nuevos datos de objeto utilizando el formato JSON o XML; el servidor detecta automáticamente el formato que se utiliza. Si tiene que especificar un objeto de modelo completo, puede resultar útil consultar primero los metadatos de clase utilizando el servicio de metadatos de la clase de objeto de modelo. Los resultados de esta consulta proporcionarán los nombres de atributo correctos para el objeto, que podrá utilizar a continuación para especificar los nuevos datos en formato XML o JSON.

En determinadas situaciones, es posible que necesite añadir un objeto de modelo que incluya otro nuevo objeto de modelo como atributo, con el atributo `parent` necesario en el objeto hijo. Esto requiere que determine primero el GUID del objeto padre, a fin de poder definir el atributo `parent` del objeto hijo. Hay dos maneras de hacerlo:

- Cree primero el objeto padre, omitiendo el objeto hijo, que le permite determinar el GUID del padre. A continuación, puede crear el objeto hijo, especificando el GUID del objeto padre.
- Cree ambos objetos con una única solicitud. Para utilizar este método, debe definir el GUID del objeto padre en un valor único en el documento JSON o XML, y especificar el mismo valor en el atributo `parent` del objeto hijo. Este ejemplo de JSON utiliza el ID `cs1` como GUID del objeto padre:

  ```
  [{"signature":"JsonRestExample1","_class":"LinuxUnitaryComputerSystem","
  numCPUs":2,"guid":"cs1","OSRunning":{"_class":"Linux","parent":"cs1","na
  me":"Linux","description":"Created by sample code"}}]
  ```

  Si desea obtener más ejemplos, consulte los programas de ejemplo en el directorio `$COLLATION_HOME/sdk/examples/rest`.

Procedimiento

Utilice uno de los dos métodos siguientes:

- Utilice el servicio de actualización del objeto de modelo y el método HTTP POST, pasando los nuevos datos de objeto al cuerpo de la solicitud. Este método solo se realizará correctamente si el objeto especificado todavía no existe. Si el objeto ya existe, la solicitud fallará. Utilice este método si quiere crear un nuevo objeto, pero no quiere realizar cambios en los objetos existentes.
- Utilice el servicio de actualización del objeto de modelo y el método HTTP PUT, pasando los nuevos datos de objeto al cuerpo de la solicitud. Este método crea el objeto especificado si no existe todavía; si el objeto ya existe, se modifica con los datos nuevos. Utilice este método si quiere asegurarse de que el objeto especificado se encuentra en la base de datos de TADDM, independientemente de si ya existía antes.

Actualización de objetos de modelo utilizando la API de REST

Puede utilizar la API de REST para actualizar un objeto de modelo existente utilizando uno de estos dos métodos.

Acerca de esta tarea

Puede utilizar un recurso del objeto de modelo o el recurso del servicio de actualización del objeto de modelo para actualizar un objeto de modelo existente, en función de si quiere permitir la creación de nuevos objetos.

En cualquier caso, antes debe describir los nuevos datos de objeto utilizando el formato JSON o XML; el servidor detecta automáticamente el formato que se utiliza. Si tiene que especificar un objeto de modelo completo, puede resultar útil consultar primero los metadatos de clase utilizando el servicio de metadatos de la clase de objeto de modelo. Los resultados de esta consulta proporcionarán los nombres de atributo correctos para el objeto, que podrá utilizar a continuación para especificar los nuevos datos en formato XML o JSON.

Procedimiento

Para utilizar la API de REST para actualizar objetos de modelo existentes, siga uno de estos dos métodos:

- Utilice el recurso del objeto de modelo que representa el objeto existente y el método HTTP PUT, pasando los nuevos datos de objeto al cuerpo de la solicitud. Este recurso solo está disponible si el objeto especificado ya existe; si el

objeto no existe, la solicitud fallará. Utilice este método si quiere modificar un objeto existente, pero no quiere añadir el objeto si no existe.

- Utilice el servicio de actualización del objeto de modelo y el método HTTP PUT, pasando los nuevos datos de objeto al cuerpo de la solicitud. Este método actualiza el objeto con los nuevos datos; si el objeto no existe ya, este método lo crea. Utilice este método si quiere asegurarse de que exista un objeto con los datos de objeto especificados en la base de datos de TADDM, independientemente de si ya existía.

Supresión de objetos de modelo utilizando la API de REST

Puede utilizar la API de REST para suprimir un objeto de modelo utilizando uno de estos dos métodos.

Acerca de esta tarea

Solo se puede suprimir un objeto en una única solicitud. Si el objeto especificado no existe, no se devolverá ningún error.

Procedimiento

Para suprimir un objeto de modelo, siga uno de estos dos métodos:

- Para suprimir un objeto especificando su GUID, utilice el recurso correspondiente del objeto de modelo, especificando el GUID del objeto que se va a suprimir como parte del URL y utilizando el método HTTP DELETE. Si utiliza este método, no se necesitarán datos en el cuerpo de la solicitud HTTP. Este ejemplo, enviado utilizando el método HTTP DELETE, suprime un objeto utilizando el recurso del objeto de modelo:

  ```
  http://example.com:9430/rest/model/ModelObject/1D646C44FDEB3857B40B98BD
  F9C0F407?mssGuid=CF5EBF574E7F382289B3F35FB5776628
  ```

- Utilice el servicio de actualización del objeto de modelo con el parámetro **`delete`** y el método HTTP POST, especificando el objeto que se va a suprimir en el cuerpo de la solicitud HTTP. En los datos de entrada, solo se necesita el GUID del objeto que se va a suprimir, aunque se puede especificar el objeto completo. Este ejemplo, enviado utilizando el método HTTP POST, suprime el objeto especificado en el cuerpo de la solicitud:

  ```
  http://example.com:9430/rest/model/ModelObject?delete=true
  ```

Mantenimiento de Patrones de agrupación mediante la API de REST

Los patrones de agrupación (GP) se pueden mantener sin necesidad de utilizar la IU, solo con el uso de la API de REST.

Puede accederse a la API de REST para patrones de agrupación mediante las credenciales de inicio de sesión estándar. Se necesita autenticación WWW básica. Si se desea acceder a la API de REST a través de un explorador web con la sesión de usuario existente, no se necesita ningún inicio de sesión adicional.

Los datos se pueden enviar y recibir en dos formatos: JSON y XML (application/json y application/xml). La vía de acceso al servicio es `/cdm/api/groupingpatterns`.

La siguiente lista incluye los métodos que se pueden utilizar y las acciones que realizan:

- `/cdm/api/groupingpatterns :: method GET` - devuelve todos los patrones de agrupación.

- `/cdm/api/groupingpatterns/{GUID} :: method GET` - devuelve los patrones de agrupación con la GUID especificada.
- `/cdm/api/groupingpatterns :: method POST :: load in JSON or XML format` - crea patrones de agrupación nuevos.
- `/cdm/api/groupingpatterns/{GUID} :: method DELETE`: suprime los patrones de agrupación con la GUID especificada.
- `/cdm/api/groupingpatterns/ :: method PUT :: load in JSON or XML format` - actualiza los patrones de agrupación.

Ejemplo de carga en formato XML:

```
<groupingPattern name="GroupingPattern1" hierarchyType="ACCESS_COLLECTION" active="true">
    <selector name="Selector1" lowerDown="include" lowerUp="include" higherDown="include"
 higherUp="include" GroupingNameExpression="GroupingNameExpression"
useTraversalTemplate="false" type="MQL">
        <query>ComputerSystem where guid == '00000000000000000000000000000000'</query>
    </selector>
    <description>GroupingPattern1Description</description>
</groupingPattern>
```

Gestión de descubrimientos utilizando la API de REST

Puede utilizar la API de REST para iniciar el descubrimiento y para gestionar descubrimientos, perfiles de descubrimiento y ámbitos de descubrimiento.

Procedimiento

Para utilizar la API de REST, siga uno o varios de estos pasos:

- Para iniciar el descubrimiento, utilice el recurso de servicio de descubrimiento con el envío de una solicitud POST y con la especificación de un nombre para la ejecución del descubrimiento. Puede utilizar este recurso para iniciar el descubrimiento con o sin un perfil. Por ejemplo, este formulario HTML inicia el descubrimiento con el nombre 'TestRun2' mediante el perfil especificado para el ámbito especificado.

  ```
  <form action="http://example.com:9430/rest/discovery/start/TestRun2" method="post">
       Profile: <input type="text" name="profile"><br>
       Scope: <input type="text" name="scope"><br>
       <input type="submit" value="Submit"></input>
  </form>
  ```

- Para comprobar el estado del descubrimiento actual, utilice el recurso de estado del descubrimiento y especifique el formato XML o JSON para los datos de la salida. En este ejemplo se comprueba el estado del descubrimiento utilizando el formato JSON:

  ```
  http://example.com/rest/discovery/status?feed=json
  ```

- Para recuperar una lista de los perfiles de descubrimiento definidos, utilice el recurso del servicio del perfil de descubrimiento y especifique el formato XML o JSON para los datos de salida. En este ejemplo se muestran los perfiles de descubrimiento que utilizan el formato XML:

  ```
  http://example.com/rest/discovery/profiles?feed=xml
  ```

- Para recuperar los detalles de un perfil de descubrimiento definido, utilice el recurso del perfil de descubrimiento y especifique el formato XML o JSON para los datos de salida. Así, se recuperan los detalles del perfil `Descubrimiento de nivel 3` utilizando el formato JSON:

  ```
  http://example.com/rest/discovery/profile/Level%203%20Discovery?feed=json
  ```

- Para recuperar una lista de ámbitos de descubrimiento definidos, utilice el recurso del servicio del ámbito de descubrimiento y especifique el formato XML o JSON para los datos de salida. En este ejemplo, se muestran los ámbitos de descubrimiento utilizando el formato XML:

```
http://example.com/rest/discovery/scopes?feed=xml
```

- Para recuperar los detalles de un ámbito de descubrimiento definido, utilice un recurso del ámbito de descubrimiento y especifique el formato XML o JSON para los datos de salida. Así, se recuperarán los detalles del ámbito `scope1` utilizando el formato JSON:

  ```
  http://example.com/rest/discovery/scope/scope1?feed=json
  ```

Referencia de recursos REST

La API de REST expone los recursos que puede utilizar para consultar, crear, actualizar y suprimir objetos de modelo, además de para gestionar descubrimientos.

Clase de objeto de modelo:

El recurso de clase de objeto de modelo representa una clase de objetos de modelo definidos por el modelo de datos común.

Descripción

Utilice este recurso para recuperar información sobre objetos de modelo especificando una clase de objeto de modelo, e incluya si lo desea valores de atributo. Este tipo de solicitud proporciona un subconjunto de la información disponible mediante las consultas MQL.

Utilice el método HTTP GET para enviar una solicitud de consulta MQL.

URL

esquema//*nombre_host*:*puerto*/rest/model/*clase_objeto_modelo*

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

clase_objeto_modelo
Nombre de clase del objeto de modelo. Especifique el nombre abreviado (como ComputerSystem) o el nombre completo (such as com.collation.platform.model.topology.sys.ComputerSystem).

Métodos HTTP

GET Objetos de modelo de consultas.

Parámetros

`cols=`*`valor`*
Lista separada por comas de los nombres de columna para los que quiere devolver datos. Con el valor predeterminado se devuelven datos de todas las columnas.

depth=*valor*
Profundidad de la consulta. El valor predeterminado es 1.

Nota: Una consulta con una profundidad mayor que 1 puede devolver un gran conjunto de resultados, lo que provocaría condiciones de memoria baja en el servidor de TADDM. Para evitar este problema, especifique `fetchSize=1` y utilice consultas consecutivas para desplazarse por los datos de uno en uno. Consulte los programas de ejemplo del directorio `$COLLATION_HOME/sdk/examples/rest` para ver ejemplos sobre cómo utilizar esta técnica.

feed={json|xml}

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **feed**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

fetchSize=*valor*
Número máximo de objetos que devolver desde el conjunto de resultados. El valor predeterminado es 1.

longClassName={true|false}
Especifica si todos los nombres de clase de objeto de modelo se deben especificar utilizando el formato completo (por ejemplo, com.collation.platform.model.topology.sys.ComputerSystem). Especifique `true` o `false`. Esta opción solo es válida para la salida JSON. El valor predeterminado es `false`.

mssGuid=*valor*
El valor del GUID del sistema de software gestionado (MSS) asociado al objeto. Este parámetro es opcional.

position=*valor*
La posición inicial del conjunto de resultados de los objetos que quiere devolver de la consulta. El valor predeterminado es 1 (el primer objeto del conjunto de resultados). Si especifica una posición mayor que el número total de objetos del conjunto de resultados, no se devolverán objetos.

nombre_atributo=valor_atributo
Nombre de atributo y valor opcional. Utilice esta opción para limitar la salida de la consulta a aquellos objetos que coinciden con el valor de atributo especificado. Si especifica más de un atributo, solo se devuelven los objetos que coincidan con todos los valores de atributo especificados.

Devoluciones

Si la consulta es correcta, el servidor devuelve el código de retorno HTTP 200, y los datos del resultado de la consulta en formato JSON o XML (según lo especificado por el parámetro **feed** o la cabecera HTTP Aceptar). Si la consulta no devuelve datos, el conjunto de resultados es una matriz JSON vacía o un documento XML, según el tipo de canal de información.

La cabecera pragma TADDMQueryComplete de los datos devueltos indica si se han devuelto todos los resultados de consulta disponibles; `true` indica que todos los resultados se han devuelto y `false` indica que hay más resultados de consulta disponibles. Puede controlar los resultados que se devuelven ajustando los valores de los parámetros opcionales `position` y `fetchSize`.

Ejemplo

Este ejemplo consulta el quinto objeto de ComputerSystem cuyo atributo OSRunning está definido en `linux`:

```
http://example.com:9430/rest/model/ComputerSystem?depth=2&feed=xml&OSRu
nning.OSName=Linux&position=5
```

Servicio de consulta MQL:

El recurso del servicio de consulta MQL recupera los datos del objeto de modelo basados en las consultas escritas en Model Query Language (MQL).

Descripción

Utilice este recurso para recuperar los datos del objeto de modelo utilizando consultas escritas en MQL. El servicio de consulta MQL puede proporcionar información más detallada que está disponible en el recurso de la clase del objeto de modelo.

URL

esquema//*nombre_host*:*puerto*/rest/model/MQLQuery

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

Métodos HTTP

GET Objetos de modelo de consultas.

Parámetros

depth=*valor*
Profundidad de la consulta. El valor predeterminado es 1.

Nota: Una consulta con una profundidad mayor que 1 puede devolver un gran conjunto de resultados, lo que provocaría condiciones de memoria baja en el servidor de TADDM. Para evitar este problema, especifique `fetchSize=1` y utilice consultas consecutivas para desplazarse por los datos de uno en uno. Consulte los programas de ejemplo del directorio `$COLLATION_HOME/sdk/examples/rest` para ver ejemplos sobre cómo utilizar esta técnica.

feed={json|xml}

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **feed**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

fetchSize=*valor*
Número máximo de objetos que devolver desde el conjunto de resultados. El valor predeterminado es 1.

longClassName={true|false}
Especifica si todos los nombres de clase de objeto de modelo se deben especificar utilizando el formato completo (por ejemplo, com.collation.platform.model.topology.sys.ComputerSystem). Especifique `true` o `false`. Esta opción solo es válida para la salida JSON. El valor predeterminado es `false`.

mssGuid=*valor*
El valor del GUID del sistema de software gestionado (MSS) asociado al objeto. Este parámetro es opcional.

position=*valor*
La posición inicial del conjunto de resultados de los objetos que quiere devolver de la consulta. El valor predeterminado es 1 (el primer objeto del conjunto de resultados). Si especifica una posición mayor que el número total de objetos del conjunto de resultados, no se devolverán objetos.

query=*valor*
Serie de consulta, escrita en notación MQL. Este parámetro es necesario.

Nota: Las consultas de objeto de modelo pueden devolver una gran cantidad de datos. Para evitar problemas de memoria y rendimiento, seleccione solo las columnas que necesite.

Devoluciones

Si la consulta es correcta, el servidor devuelve el código de retorno HTTP 200, y los datos del resultado de la consulta en formato JSON o XML (según lo especificado por el parámetro **feed** o la cabecera HTTP Aceptar). Si la consulta no devuelve datos, el conjunto de resultados es una matriz JSON vacía o un documento XML, según el tipo de canal de información.

La cabecera pragma TADDMQueryComplete de los datos devueltos indica si se han devuelto todos los resultados de consulta disponibles; `true` indica que todos los resultados se han devuelto y `false` indica que hay más resultados de consulta disponibles. Puede controlar los resultados que se devuelven ajustando los valores de los parámetros opcionales `position` y `fetchSize`.

Ejemplo

Este ejemplo consulta los datos del objeto de modelo utilizando la consulta MQL `select displayName,OSRunning.OSName from ComputerSystem`.

```
http://example.com:9430/rest/model/MQLQuery?query=select%20displayName,
OSRunning.OSName%20from%20ComputerSystem&position=2&fetchSize=2&feed=xm
l&depth=2&position=4
```

Objeto de modelo:

Un recurso de objeto de modelo representa una instancia de objeto de modelo específica que existe en la base de datos de TADDM, identificada por el GUID.

Descripción

Utilice este tipo de recurso para consultar, actualizar o suprimir una única instancia del objeto de modelo identificada por el identificador exclusivo global (GUID).

URL

esquema//*nombre_host*:*puerto*/rest/model/ModelObject/*guid*

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

guid
Identificador exclusivo global (GUID) de una instancia de objeto de modelo que existe en la base de datos de TADDM. Si va a actualizar un objeto, este GUID debe coincidir con el GUID especificado en los datos de objeto JSON o XML.

Métodos HTTP

GET Consulta un objeto de modelo.

PUT Actualiza un objeto de modelo. Los nuevos datos de objeto se deben especificar en el cuerpo de la solicitud HTTP, ya sea en formato JSON o XML (el servidor detecta automáticamente el formato de los datos de entrada).

DELETE
Suprime un objeto de modelo.

Parámetros

depth=*valor*
Profundidad de la consulta. El valor predeterminado es 1. Este parámetro no se utiliza al actualizar o suprimir objetos.

feed={json|xml}

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **feed**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

El parámetro **feed** no se utiliza al actualizar ni al suprimir un objeto.

longClassName={true|false}
Indica si todos los nombres de clase del objeto de modelo de la salida de una consulta se especifican utilizando el formato completo (por ejemplo,

com.collation.platform.model.topology.sys.ComputerSystem). Especifique `true` o `false`. Esta opción solo es válida para la salida JSON. El valor predeterminado es `false`.

mssGuid=*valor*
El valor del GUID del sistema de software gestionado (MSS) asociado al objeto. Este parámetro es opcional.

Devoluciones

Si la solicitud es correcta, el servidor devuelve el código de retorno HTTP 200. Para una consulta, el servidor devuelve también los datos de resultado en formato JSON o XML (tal y como lo especifica el parámetro **feed** o la cabecera HTTP Aceptar). Si la consulta no devuelve datos, el conjunto de resultados es una matriz JSON vacía o un documento XML, según el tipo de canal de información.

Ejemplo

Este ejemplo consulta, actualiza o suprime un objeto existente, según el método HTTP utilizado (para actualizar un objeto, el cuerpo de la solicitud debe contener los datos de objeto actualizados).

```
http://example.com:9430/rest/model/ModelObject/1D646C44FDEB3857B40B98BD
F9C0F407?mssGuid=CF5EBF574E7F382289B3F35FB5776628
```

Metadatos de clase de objeto de modelo:

El recurso de metadatos de la clase de objeto de modelo representa los metadatos que describen los atributos de una clase de objeto de modelo.

Descripción

Utilice el recurso de metadatos de la clase de objeto de modelo para consultar datos sobre los atributos de una clase de objeto de modelo especificado, incluido el número, el tipo y el nombre de cada atributo. Esta información es equivalente a la devuelta por el método getMetaData() de Java.

Los metadatos se pueden devolver en formato JSON o XML.

URL

esquema//*nombre_host*:*puerto*/rest/model/meta/*clase_objeto_modelo*

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

clase_objeto_modelo
Nombre de una clase de objeto del modelo de datos común.

Métodos HTTP

GET Consulta los metadatos de la clase de objeto.

Parámetros

feed={json|xml}

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **feed**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

Ejemplo

Este ejemplo consulta la información de metadatos para el objeto de modelo ComputerSystem:

```
http://example.com:9430/rest/model/meta/ComputerSystem?feed=json
```

En este ejemplo se muestra la salida de JSON desde una consulta de metadatos:

```
[{"type":"java.lang.String","column":"BOOTORDER_X","length":192,"name":
"bootOrder","arrayType":false,"_class":"ObjectAttribute","timestampType
":false,"displayString":"Boot Order"}, {"type":"com.collation.platform.
model.topology.sys.zOS.ZReportFile","table":"COMPUTERSYSTILES_935A6002X
","column":"PK__ZREPORTFILES_X","length":192,"name":"ZReportfiles","arr
ayType":true,"reverseRelationship":true,"_class":"ObjectAttribute","rel
ationshipType":"com.collation.platform.model.topology.relation.AppliesT
o","timestampType":false,"displayString":"z\/OS Report File"}]
```

En este ejemplo se muestra la salida XML parcial de una consulta de metadatos:

```
<ObjectAttribute array="22" xsi:type="coll:com.collation.platform.model
.topology.meta.ObjectAttribute">
  <name>OSRunning</name>
  <type>com.collation.platform.model.topology.sys.OperatingSystem</type>
  <arrayType>false</arrayType>
  <timestampType>false</timestampType>
  <length>192</length>
  <relationshipType>com.collation.platform.model.topology.relation.Runs
On</relationshipType>
  <reverseRelationship>true</reverseRelationship>
  <displayString>OS Running</displayString>
  <column>PK__OSRUNNING_X</column>
  <displayName />
</ObjectAttribute>
```

Servicio de actualización del objeto de modelo:

El recurso del servicio de actualización del objeto de modelo crea, actualiza o suprime objetos de modelo pasados al servidor en formato JSON o XML.

Descripción

Utilice el servicio de actualización del objeto de modelo para actualizar o suprimir un objeto de modelo existente, o bien para añadir un nuevo objeto de modelo. En cada caso, el destino de la operación es el objeto especificado en el cuerpo de la solicitud, en formato JSON o XML.

URL

esquema//*nombre_host*:*puerto*/rest/model/ModelObject

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

Métodos HTTP

POST Crea o suprime un objeto de modelo, según el valor del parámetro **`delete`**. El objeto que se va a crear o a suprimir se debe especificar en el cuerpo de la solicitud HTTP en formato XML o JSON (el servidor detecta automáticamente el formato de los datos de entrada). Especifique solo un objeto primario; no se admiten las matrices de objetos.

Si utiliza este método para crear un objeto nuevo, el objeto especificado no debería existir todavía en la base de datos de TADDM (el método POST no se puede utilizar para actualizar un objeto existente).

Si utiliza este método para suprimir un objeto existente, solo se necesitará el GUID en los datos de entrada. Sin embargo, también se puede especificar el objeto completo. No se devuelve ningún error si el objeto especificado no existe.

PUT Actualiza un objeto existente o crea un objeto nuevo. Los nuevos datos de objeto se deben especificar en el cuerpo de la solicitud HTTP en formato JSON o XML. Especifique solo un objeto primario; no se admiten las matrices de objetos.

Si el objeto especificado ya existe, se actualiza con los datos nuevos. Si el objeto no existe, se crea.

Si va a actualizar un objeto existente, puede mejorar el rendimiento incluyendo solo el GUID y los campos requeridos para la actualización, en lugar del objeto completo. Por ejemplo, una actualización sobre la descripción de un objeto OperatingSystem podría incluir los datos siguientes:

```
[{"description":"Validated on February 4","_class":"Linux","guid":"347EE
64E4FA93139A581757EC7F3ED2D"}]
```

Los atributos de objeto que no estén especificados en los datos de actualización se mantendrán sin cambios.

Parámetros

`delete={true|false}`
Indica si el objeto de modelo especificado se debe suprimir. Utilice el método HTTP POST y `delete=true` para suprimir un objeto.

`feed={json|xml}`

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **feed**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

mssGuid=*valor*
El valor del GUID del sistema de software gestionado (MSS) asociado al objeto. Este parámetro es opcional.

Devoluciones

Si la solicitud es correcta, el servidor devuelve el código de retorno HTTP 200.

En el siguiente ejemplo, se suprime el objeto de modelo especificado por los datos de entrada:

```
http://example.com:9430/rest/model/ModelObject?delete=true
```

Servicio de descubrimiento:

El recurso del servicio de descubrimiento inicia un descubrimiento con o sin un perfil.

Descripción

Utilice este tipo de solicitud para iniciar un descubrimiento utilizando cualquier perfil actualmente definido, o sin perfiles.

URL

esquema//*nombre_host*:*puerto*/rest/discovery/start/*nombre_ejecución*

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

nombre_ejecución
Nombre de la ejecución de descubrimiento.

Métodos HTTP

POST Inicia un descubrimiento. No debe haber ningún descubrimiento en curso. Debe enviar la solicitud del URL utilizando el operador HTTP POST.

Parámetros

feed={json|xml}

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **feed**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

guids=*valores*
Uno o varios identificadores exclusivos globales (GUID) de objetos que se han descubierto previamente. Utilice este parámetro para ejecutar un redescubrimiento en objetos existentes, si ha habilitado el redescubrimiento.

profile=*nombre_perfil*
Nombre del perfil que se va a utilizar. El perfil especificado debe existir.

scope=*valores*
Uno o varios ámbitos, separados por comas. Los valores pueden ser cualquiera de los siguientes:

- Nombre de ámbito definido
- Dirección IP específica o nombre de host (por ejemplo, `192.168.1.71` o `server.example.com`)
- Dirección IP específica que excluir, entre paréntesis (por ejemplo, `(192.168.1.71)`)
- Rango de direcciones IP (por ejemplo, `10.10.10.1-10.10.10.20`)
- Subred (por ejemplo, `10.10.20.0/255.255.255.0`

Pueden aparecer entre paréntesis una dirección IP, un rango de direcciones o una subred para indicar que se deberían excluir del ámbito. Por ejemplo, `192.168.1.1-192.168.1.72,(192.168.1.71)` incluiría todas las direcciones IP del rango especificado, excepto 192.168.1.71.

locationTag=*value*
El valor de la etiqueta de ubicación que hay que definir para los objetos creados durante el descubrimiento.

Fix Pack 3 **addressSpace=*valor***
El nombre de espacio de direcciones definido para todos los objetos IpAddress o IpNetwork creados durante un descubrimiento.

Ejemplo de entrada

Este ejemplo inicia un descubrimiento utilizando el perfil `Level 3 Discovery`, con un ámbito que incluye los hosts 192.168.100.101 y 102.168.100.102. Puede utilizar cualquier herramienta o programa de utilidad que pueda realizar una solicitud HTTP y enviarla utilizando el operador POST.

```
http://example.com:9430/rest/discovery/start/TestRun2?profile=Level%203
%20Discovery&scope=192.168.100.101,192.168.100.102
```

Devoluciones

Si la solicitud es correcta, se devuelve el código de respuesta HTTP 200, así como el mensaje `Discovery start submitted`. Si ya hay un descubrimiento en curso, la solicitud falla y se devuelve un mensaje de error. A continuación, puede utilizar el recurso del estado de descubrimiento para supervisar el progreso del descubrimiento.

Estado del descubrimiento:

El recurso de estado del descubrimiento representa el estado de descubrimiento actual en el servidor de TADDM.

Descripción

Utilice este recurso para comprobar el estado de la ejecución de descubrimiento actual. La información devuelta es equivalente a la que ha devuelto el método getStatus() de Java.

URL

esquema//*nombre_host*:*puerto*/rest/discovery/status

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

Métodos HTTP

GET Estado de descubrimiento de las consultas.

Parámetros

`feed={json|xml}`

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **`feed`**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

Devoluciones

El estado del descubrimiento actual se devuelve utilizando el formato especificado. En el ejemplo siguiente, se muestra el estado del descubrimiento en formato XML:

```
<status>Idle</status>
```

Ejemplo

Este ejemplo comprueba el estado del descubrimiento:

```
http://example.com:9430/rest/discovery/status?feed=xml
```

Servicio de perfil de descubrimiento:

El recurso de servicio de perfil de descubrimiento muestra los perfiles de descubrimiento definidos.

Descripción

Utilice este recurso para recuperar una lista de todos los perfiles de descubrimiento actualmente definidos en el servidor de TADDM.

URL

esquema//*nombre_host*:*puerto*/rest/discovery/profiles

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

Métodos HTTP

GET Lista los perfiles de descubrimiento.

Parámetros

`feed={json|xml}`

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **`feed`**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

Devoluciones

Se devuelve una lista de los perfiles de descubrimiento definidos utilizando el formato especificado. En el ejemplo siguiente, se muestra la salida en formato de JSON:

```
[{"name":"profile1"},{"name":"profile2"}]
```

Ejemplo

Este ejemplo lista los perfiles de descubrimiento:

```
http://example.com:9430/rest/discovery/profiles?feed=json
```

Perfil de descubrimiento:

El recurso del perfil de descubrimiento representa un perfil de descubrimiento definido.

Descripción

Utilice el recurso del perfil de descubrimiento para recuperar información detallada sobre un perfil de descubrimiento definido.

URL

esquema//*nombre_host*:*puerto*/rest/discovery/profile/*nombre_perfil*

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

nombre_perfil
Nombre de un perfil de descubrimiento definido.

Métodos HTTP

GET Recupera detalles de un perfil de descubrimiento.

Parámetros

feed={json|xml}

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **feed**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

Devoluciones

Los detalles del perfil de descubrimiento se devuelven utilizando el formato especificado.

Ejemplo

Este ejemplo recupera información sobre el perfil `Descubrimiento de nivel 3`:
`http://example.com:9430/rest/discovery/profile/Level%203%20Discovery?feed=xml`

Servicio del ámbito de descubrimiento:

El recurso de servicio del ámbito de descubrimiento muestra los ámbitos de descubrimiento definidos.

Descripción

Utilice este recurso para recuperar una lista de todos los ámbitos de descubrimiento actualmente definidos en el servidor de TADDM.

URL

esquema//*nombre_host*:*puerto*/rest/discovery/scopes

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

Métodos HTTP

GET Lista los ámbitos de descubrimiento.

Parámetros

feed={json|xml}

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **feed**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

Devoluciones

Se devuelve una lista de ámbitos de descubrimiento definidos utilizando el formato especificado. En el ejemplo siguiente, se muestra la salida en formato de JSON:

```
[{"name":"scope1"},{"name":"scope2"}]
```

Ejemplo

Este ejemplo muestra los ámbitos de descubrimiento:

```
http://example.com:9430/rest/discovery/scopes?feed=json
```

Ámbito del descubrimiento:

El recurso del ámbito de descubrimiento representa un ámbito de descubrimiento definido.

Descripción

Utilice el recurso del ámbito de descubrimiento para recuperar información detallada sobre un conjunto de ámbitos de descubrimiento o un grupo de ámbitos definido.

URL

esquema//*nombre_host*:*puerto*/rest/discovery/scope/*nombre_ámbito*

donde:

esquema
Esquema del URL (`HTTP:` o `HTTPS:`).

nombre_host
Nombre de host TCP/IP o dirección IP numérica del servidor de TADDM.

puerto
Puerto TCP/IP del servidor de TADDM para el tipo de conexión que está utilizando (9430 para HTTP o 9431 para HTTPS).

nombre_ámbito
Nombre de un conjunto de ámbitos de descubrimiento o grupo de ámbitos definido.

Métodos HTTP

GET Recupera detalles de un ámbito de descubrimiento.

Parámetros

feed={json|xml}

Formato que utilizar para los datos devueltos. Especifique `json` o `xml`. Este parámetro es opcional.

Si no especifica el parámetro **feed**, el servidor utiliza el formato especificado por la cabecera HTTP Aceptar (`application/json` o `application/xml`). Si no se especifica la cabecera, los resultados se devuelven en formato JSON.

Devoluciones

Los detalles del ámbito de descubrimiento se devuelven utilizando el formato especificado.

Ejemplo

En este ejemplo, se recupera información sobre el ámbito `scope1`:

```
http://example.com:9430/rest/discovery/scope/scope1?feed=xml
```

API de interfaz de línea de mandatos

Puede utilizar `api.bat` o `api.sh` para emitir varios mandatos en el servidor de TADDM mediante la interfaz de línea de mandatos (CLI). Por ejemplo, puede utilizar la CLI para iniciar una ejecución de descubrimiento.

Sintaxis del mandato y parámetros

Puede utilizar `api.sh` o `api.bat` para acceder a una parte de la funcionalidad de la API de TADDM. Las sintaxis del mandato presentan las reglas para ejecutar `api.sh` y `api.bat`.

Para UNIX:

api.sh -u|**--user** *usuario* **-p**|**--password** *contraseña* [**-H**|**--host** *host*] [**-P**|**--port** *puerto*] [**-T**|**--truststorefile**] *MANDATO PARÁMETROS_MANDATO*

Para Windows:

api.bat -u|**--user** *usuario* **-p**|**--password** *contraseña* [**-H**|**--host** *host*] [**-P**|**--port** *puerto*] [**-T**|**--truststorefile**] *MANDATO PARÁMETROS_MANDATO*

Parámetros comunes

-u|--user *usuario*
Usuario que ejecuta el mandato de la API.

-p|--password *contraseña*
Contraseña que autentica al usuario.

-H|--host *host*
Opcional: el nombre de host de servidor de TADDM es, de forma predeterminada, `localhost`. Si utiliza el parámetro **-T**, también debe especificar el parámetro **-H**.

-P|--port *puerto*
Opcional: el puerto del servidor de TADDM es, de forma predeterminada, `9433`.

-v|--version *versión*
Opcional: el nombre o el número de la versión es, de forma predeterminada, 0.

-T|--truststorefile *truststore*
Opcional: la ubicación del archivo de almacén de confianza, `jssecacerts.cert`, con un certificado para la conexión al servidor de TADDM. Este parámetro es necesario para la conexión segura a TADDM. Si utiliza este parámetro, también debe especificar el parámetro **-H**.

COMMAND COMMAND-PARAMETERS
Los parámetros son distintos para cada uno de los mandatos.

Información adicional

Para obtener ayuda sobre el mandato y los parámetros del mandato, especifique el mandato siguiente desde el directorio `$COLLATION_HOME/sdk/bin`:

En sistemas UNIX
`api.sh`

En sistemas Windows
`api.bat`

Mandato changes

El mandato **changes** recupera los cambios de un objeto.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **changes** *guid fecha inicial* [*fecha final*]

Parámetros

changes
Ejecuta el mandato **changes**.

guid
GUID del objeto para el cual desea determinar los cambios.

fecha-inicial
Fecha de inicio del mandato **changes**. Utilice el formato `mm/dd/aa hh:mm:ss AM|PM`.

fecha-final
Fecha de finalización del mandato **changes**. Utilice el formato `mm/dd/aa hh:mm:ss AM|PM`.

Nota: Si desea ejecutar una consulta avanzada de historial de cambios en el servidor de descubrimiento, el parámetro `-H` debe apuntar a un servidor de almacenamiento. De lo contrario, la consulta fallará.

Ejemplo

Este mandato encuentra todos los cambios que se han realizado en un objeto entre dos fechas específicas. Especifique el mandato en una sola línea:

```
api.sh -u usuario -p contraseña -H host -P puerto changes
 10A5794E86C53A0BBB10F262055CB3EA "06/06/05 12:00:00 AM" "06/08/05 12:00:00 AM"
```

Mandato delete

El mandato **delete** elimina los objetos de la base de datos de TADDM.

Sintaxis del mandato

api.sh|**api.bat -u**|**--user** *usuario* **-p**|**--password** *contraseña* [**-H**|**--host** *host*] [**-P**|**--port** *puerto*] [**-T**|**--truststorefile**] **delete** *guid1* [*guid2 guid3 ... guidn*]

api.sh|**api.bat -u**|**--user** *usuario* **-p**|**--password** *contraseña* [**-H**|**--host** *host*] [**-P**|**--port** *puerto*] [**-T**|**--truststorefile**] **delete -f**|**--file** *archivo-lista-guid*

api.sh|**api.bat -u**|**--user** *usuario* **-p**|**--password** *contraseña* [**-H**|**--host** *host*] [**-P**|**--port** *puerto*] [**-T**|**--truststorefile**] **delete -m**|**--mql** *consulta-mql*

Parámetros

delete
Ejecuta el mandato **delete**.

guid1 [guid2 guid3 ... guidn]
Los GUID de los objetos que se quieren suprimir.

-f|--file ***archivo-lista-guid***
Ubicación y nombre del archivo de texto que contiene la lista de GUID de los objetos que suprimir. Los GUID se pueden separar por espacio, tabulador o el carácter de nueva línea. El archivo se puede generar con el script `dbquery`.

-m|--mql ***consulta-mql***
Es la consulta MQL que selecciona los objetos que suprimir.

Ejemplo

El mandato siguiente suprime un objeto con el GUID especificado. Escriba el mandato en una sola línea.

```
api.sh -u usuario -p contraseña -H host -P puerto delete
 10A5794E86C53A0BBB10F262055CB3EA
```

El mandato siguiente suprime los objetos con los GUID especificado. Escriba el mandato en una sola línea.

```
api.sh -u usuario -p contraseña -H host -P puerto delete
 C172810FD1CF3E108B8127BC47D2667B 059E4D85B34C32D1B5A80D9E2DB09EBD
35D21D3CA08539908DC1762D26897FB6
```

El mandato siguiente suprime los objetos seleccionados por la consulta MQL especificada. Escriba el mandato en una sola línea.

```
api.sh -u usuario -p contraseña -H host -P puerto delete
 --mql "select * from AppServer where objectType == 'SAS'"
```

El mandato siguiente suprime objetos en base a la lista de GUID del archivo de texto. Se usa el script `dbquery` para generar el `archivo-lista-guid`. Escriba el mandato en una sola línea.

```
dbquery.sh -u usuario -p contraseña -q "select guid_c from BB_APPSERVER6_V
where objectType_C like '%SAS'" > /tmp/guidListToDelete.txt
api.sh -u usuario -p contraseña -H host -P puerto delete
-f /tmp/guidListToDelete.txt
```

Mandato discover

El mandato **discover** inicia o detiene una ejecución de descubrimiento.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **descubrir inicio** [**--name** *nombre-ejecución*] [**--profile** *nombre-perfil*] [**--locationTag** *etiqueta-ubicación*] [**-a** | **-addressSpace** *espacioDirecciones*] *elemento-ámbito1* | *conjunto-ámbito1* | *grupo-ámbitos1 elemento-ámbito2* | *conjunto-ámbitos2* | *grupo-ámbitos2 ... elementon-ámbito* | *conjunto-ámbitosn* | *grupo-ámbitosn*

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **discover abort** | **status**

Parámetros

discover
Ejecuta el mandato **discover**.

start *elemento1-ámbito elemento2-ámbito ... elementon-ámbito*
Inicia un descubrimiento con los elementos especificados del ámbito. El elemento del ámbito puede ser un nombre de ámbito existente, o:

- Dirección IP específica: 192.168.1.71
- Exclusión de una dirección IP específica: 192.168.1.71(exclude) o (192.168.1.71) o 192.168.1.71(exc)
- Rango o exclusión de rango: 10.10.10.1-10.10.10.20 o (10.10.10.1-10.10.10.20)
- Red (subred) o exclusión de red: 10.10.20.0/255.255.255.0 o (10.10.20.0/255.255.255.0)

start *conjunto-ámbitos1 conjunto-ámbitos2 ... conjunto-ámbitosn*
Inicia un descubrimiento con los conjuntos de ámbitos especificados.

start *grupo-ámbitos1 grupo-ámbitos2 ... grupo-ámbitosn*
Inicia un descubrimiento con el grupo de ámbitos especificado.

--name *nombre-ejecución*
Es el nombre de la ejecución de descubrimiento.

--profile *nombre-perfil*
Utiliza el perfil especificado por *nombre de perfil* para el descubrimiento.

--locationTag *etiqueta-ubicación*
Especifica la etiqueta de ubicación que se utiliza para este descubrimiento.

Fix Pack 3 **-a | -addressSpace *espacioDirecciones***
Especifica el nombre de espacio de direcciones para todos los objetos IpAddress o IpNetwork creados durante el descubrimiento iniciado con **api.sh** o **api.bat**.

abort|stop
Detiene un descubrimiento activo del host especificado.

status

Devuelve el estado de descubrimiento del host especificado, de entre los valores siguientes:

- En ejecución
- Desocupado

Ejemplos

- Este mandato descubre la subred 10.10.10.0/24 utilizando un perfil de descubrimiento de nivel 1. Especifique el mandato en una sola línea:

```
api.sh -u usuario -p contraseña -H host discover start
 --profile "Level 1 Discovery" "10.10.10.0/255.255.255.0"
```

- Este mandato descubre el conjunto de ámbitos denominado MyScope mediante un perfil de descubrimiento de nivel 2. Especifique el mandato en una sola línea:

```
api.sh -u usuario -p contraseña -H host
 discover start --profile "Level 2 Discovery" "MyScope"
```

- Este mandato descubre el conjunto de ámbitos denominado MyScope mediante un perfil de descubrimiento de nivel 3 con un host 1.2.3.4 excluido y un rango 2.3.4.5-2.3.4.7 incluido. Especifique el mandato en una sola línea:

```
api.sh -u usuario -p contraseña -H host discover start
--profile "Level 3 Discovery" "MyScope" "(1.2.3.4)" "2.3.4.5-2.3.4.7"
```

- Este mandato descubre el conjunto de ámbitos denominado MyScopeSet mediante un perfil de descubrimiento de nivel 1. Especifique el mandato en una sola línea:

```
api.sh -u usuario -p contraseña discover start
 --profile "Level 1 Discovery" "MyScopeSet"
```

- Este mandato descubre el conjunto de ámbitos denominado MyScopeGroup mediante un perfil de descubrimiento de nivel 1. Especifique el mandato en una sola línea:

```
api.sh -u usuario -p contraseña discover start
 --profile "Level 1 Discovery" "MyScopeGroup"
```

Mandato de descubrimiento con equilibrio de carga

Fix Pack 2

El mandato **discoverloadbalanced** permite ejecutar un descubrimiento en modalidad de equilibrio de carga, lo que significa que el descubrimiento se realiza de forma continuada.

Acerca del descubrimiento con equilibrio de carga

El descubrimiento continuo con equilibrio de carga se logra utilizando una agrupación de servidores de descubrimiento. Esto maximiza la utilización de servidores y protege al sistema en relación a anomalías, por lo tanto, evitando que los descubrimientos que están en curso sean interrumpidos.

Los entornos descubiertos se pueden separar en áreas, cada una a partir de su propia agrupación de servidores. Es más, cada descubrimiento se puede realizar especificando conjuntos de direcciones IP para definir el ámbito del descubrimiento.

El servidor de almacenamiento primario (PrimaryStorageServer o PSS) actúa como controlador de descubrimiento con equilibrio de carga, asignando ámbitos y recursos a cada servidor de descubrimiento (DS). Si el PSS falla, todos los descubrimientos en ejecución se completarán, sin embargo, no se asignarán nuevos descubrimientos.

El PSS controla el descubrimiento colocando trabajo en una cola, pero no pasa tareas al DS; en su lugar, cada DS participante solicita de forma activa trabajo a la cola cuando está listo y, a continuación, envía una señal de 'en curso' a medida que el descubrimiento avanza. Si no se recibe una señal de 'en curso', el PSS reubica el trabajo.

Cuando se ejecuta en modalidad continuada (con equilibrio de carga), el servidor de descubrimiento inicia otro descubrimiento a medida que las hebras DiscoveryWorker están libres para ser utilizadas, incluso antes de que finalice el descubrimiento. El mandato de descubrimiento continuado solo está disponible en el servidor de almacenamiento primario (en modalidad empresarial) y en el servidor del dominio (en modalidad de un único dominio).

Consejo: Los sensores que almacenan datos se muestran en la interfaz de usuario como 'en curso'. Esto **no** es lo mismo que el número de hebras que ejecutan un descubrimiento.

Escenarios de descubrimiento con equilibrio de carga

Descubrimiento de alta disponibilidad con equilibrio de carga basado con relación a un grupo de ámbitos

Un usuario ejecuta un descubrimiento con relación a un grupo de ámbitos definidos a través de la interfaz de línea de mandatos (CLI). La agrupación de servidores se utiliza para optimizar la carga de trabajo del descubrimiento, en base a los ámbitos (desde un ámbito individual a uno múltiple) que son parte del grupo de ámbitos.

Generación dinámica de grupo de ámbitos

Un usuario ejecuta un descubrimiento con equilibrio de carga utilizando una lista de direcciones IP que se proporciona como un archivo. Esto crea un grupo de ámbitos con ámbitos hijo generados de forma automática, con direcciones IP divididas entre ámbitos hijo.

Sintaxis del mandato

Al igual que para el descubrimiento estándar mediante el mandato **`discover`**, la función de descubrimiento con equilibrio de carga (**`discoverloadbalanced`**) se controla mediante **`api.sh`**.

Parámetros

api.sh -u <usuario> -p <contraseña> discoverloadbalanced start --poolName <nombre_agrupación> --scopeGroup <grupo_ámbitos> [--profile <nombre_perfil>]

Inicia el descubrimiento con equilibrio de carga en modalidad continuada para cada ámbito especificado en `<grupo_ámbitos>` y para cada servidor perteneciente a la agrupación de descubrimiento especificada en `<agrupación_descubrimiento>`

api.sh -u <usuario> -p <contraseña> discoverloadbalanced startFromFiles --files <filePath1,filePath2,...> --maxScopeSize <tamaño> [--profile <nombre_perfil>]

Inicia el descubrimiento con equilibrio de carga en modalidad continuada para cada ámbito que se ha pasado mediante el argumento `--files`. Se analiza cada archivo y se crea un nuevo grupo con el nombre del archivo (sin la extensión). A cada grupo se la añaden ámbitos del tamaño máximo especificado en `maxScopeSize`. Se presupone que `poolName` es igual que `groupName`

api.sh -u <usuario> -p <contraseña> discoverloadbalanced startFromDirectory --dir <vía_acceso_directorio> --maxScopeSize <tamaño> [--profile <nombre_perfil>]
Del mismo modo que en el ejemplo `--files`, pero inicia el descubrimiento para cada archivo del directorio.

api.sh -u <usuario> -p <contraseña> discoverloadbalanced status
Indica el estado actual del descubrimiento con equilibrio de carga.

api.sh -u <usuario> -p <contraseña> discoverloadbalanced abort <nombre_agrupación>
Cancela el descubrimiento para el `poolName` especificado.

api.sh -u <usuario> -p <contraseña> discoverloadbalanced pause <nombre_agrupación>
Detiene el descubrimiento para el valor especificado para `poolName` y todos los ámbitos de la ejecución del descubrimiento vuelven al estado 'forTake'. Se cancelan los descubrimientos actuales.

api.sh -u <usuario> -p <contraseña> discoverloadbalanced resume <nombre_agrupación>
Reanuda el descubrimiento para el valor especificado para `poolName` y todos los ámbitos cuyo estado es 'forTake' vuelven a estar disponibles para su proceso.

Propiedades del servidor del almacenamiento primario

Establezca las propiedades siguientes para el servidor de almacenamiento primario (o dominio):

com.ibm.cdb.internalscheduling.discoverypool.scopePrefix
Especifica el prefijo para los ámbitos creados automáticamente (para las opciones startFromFiles y startFromDirectory)

Predeterminado:
`com.ibm.cdb.internalscheduling.discoverypool.scopePrefix=_auto_`

com.ibm.cdb.internalscheduling.discoverypool.timeToNonAlive
Especifica el periodo de tiempo en segundos, esto es, durante cuánto tiempo se presupone que se ejecutará el descubrimiento sin que se reciba ninguna información de su estado desde el servidor de descubrimiento.

Una vez transcurrido el periodo el tiempo, el ámbito vuelve al estado 'forTake', de modo que el otro servidor que todavía responde pueda rehacerlo.

Predeterminado:
`com.ibm.cdb.internalscheduling.discoverypool.timeToNonAlive=180`

Propiedades del servidor de descubrimiento

Establezca las propiedades siguientes para el servidor de descubrimiento (o dominio):

com.ibm.cdb.internalscheduling.discoverypool.enabled
Permite que el servidor de descubrimiento forme parte de la agrupación de descubrimiento.

Predeterminado:
`com.ibm.cdb.internalscheduling.discoverypool.enabled=false`

com.ibm.cdb.internalscheduling.discoverypool.name
Define el nombre de agrupación de un servidor de descubrimiento

Predeterminado:
`com.ibm.cdb.internalscheduling.discoverypool.name=DEFAULT`

com.ibm.cdb.internalscheduling.discoverypool.checkinginterval
Especifica el periodo de tiempo en segundos que comprueba el servidor de descubrimiento si hay nuevos trabajos que se puedan capturar (si están preparados) y la frecuencia con la que se ha de informar a PSS acerca de un trabajo en curso.

El intervalo especificado debe ser inferior a `com.ibm.cdb.internalscheduling.discoverypool.timeToNonAlive` en un servidor de almacenamiento primario, de lo contrario es posible que se devuelvan los ámbitos para que se vuelvan a ejecutar.

Predeterminado:
`com.ibm.cdb.internalscheduling.discoverypool.checkinginterval=20`

Nota: Los registros del descubrimiento continuado se almacenan en ApiServer.log en el dominio o en el servidor de almacenamiento primario.

Mandato export

El mandato **export** exporta datos para objetos de modelo de nivel superior en la base de datos de TADDM.

Sintaxis del mandato

api.sh | **api.bat** **-u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **export** [**--mssguid** *mss-guid* | **--mssname** *mss-name*] [**--maxfilesize** *size*] *local-directory-to-write-data*

Parámetros

export
Ejecuta el mandato **export**.

--mssguid ***guid-mss*** | **--mssname** ***nombre-mss***
GUID o nombre del sistema de software gestionado (MSS). Solo se exportan los datos asociados al MSS especificado.

Encontrará el nombre del sistema de software de gestión en la IU. Vaya al panel Detalles de un objeto de modelo y abra el separador **Información de MSS**. **Nombre de instancia del subcomponente** es el nombre del sistema de software de gestión, como por ejemplo `LinuxComputerSystemSensor`. Si desea exportar todos los objetos que descubre este sensor, ejecute el mandato siguiente:

```
export --mssName LinuxComputerSystemSensor
```

Si desea buscar el nombre y el GUID del sistema de software de gestión, ejecute el mandato siguiente:

```
api.sh -u user -p password find -d 1 ManagementSoftwareSystem
```

Como resultado, obtendrá una lista de objetos de modelo con un conjunto específico de información, como por ejemplo:

```
<ManagementSoftwareSystem array="1"
    guid="MY_GUID" xsi:type="coll:com.collation.platform.model.topology.process.
ManagementSoftwareSystem">
    <manufacturerName>IBM</manufacturerName>
```

```
        <productName>TADDM</productName>
        <hostname>hostname.domain</hostname>
        <subcomponent>Discovery</subcomponent>
        <subcomponentInstanceName>IvmSensor</subcomponentInstanceName>
        <displayName>IBM:TADDM:hostname.domain:Discovery:IvmSensor</displayName>
        <bidiFlag>3</bidiFlag>
    </ManagementSoftwareSystem>
```

En este ejemplo, el GUID de MSS es `MY_GUID` y el nombre de MSS es `IvmSensor`. Si desea ejecutar el mandato **export** con estos datos, utilice uno de los mandatos siguientes:

```
export --mssguid MY_GUID
```

or

```
export --mssname IvmSensor
```

Nota: En el caso del GUID, utilice el valor correcto de la consulta de la API.

--maxfilesize ***tamaño***
Tamaño máximo de los archivos exportados, en bytes.

directorio-local-donde-escribir-datos
Nombre del directorio al que se exportarán los datos.

Ejemplo

Este mandato exporta objetos de modelo de nivel superior al directorio especificado:

```
api.sh -u usuario -p contraseña -H host export directory/
```

Mandato find

El mandato **find** busca un conjunto de objetos y devuelve una representación XML.

Sintaxis del mandato

api.sh | **api.bat** **-u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **find** [**--depth** *profundidad*] [**--indent** *núm_espacios*] [**-o** | **--outfile** *archivo-local-donde-escribir* [**-x --maxfilesize** *tamaño*]] [**-s --suppress** *lista-de-clases-que-suprimir*] *raíz*

api.sh | **api.bat** **-u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **find** [**--depth** *profundidad*] [**--indent** *núm_espacios*] **--guid** *guid-objeto*

api.sh | **api.bat** **-u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **find** [**--depth** *profundidad*] [**--indent** *núm_espacios*] [**--changetype** *tipo* **--from** *fecha-inicial* [**--end** *fecha-final*]] *raíz*

api.sh | **api.bat** **-u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **find** [**--depth** *profundidad*] [**--indent** *núm_espacios*] [**--mssguid** *guid-mss* | **--mssname** *nombre-mss*] *consulta-mql*

api.sh | **api.bat** **-u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **find** **--count** *consulta-mql*

Parámetros

find
Ejecuta el mandato **find**.

--depth ***profundidad***
Nivel del árbol de resultados que se va a construir.

Consultar una gran cantidad de datos o especificar más de un nivel puede generar mensajes de memoria insuficiente. Para evitar problemas de memoria, limite el valor de profundidad o aumente el tamaño de almacenamiento dinámico máximo de la memoria de JVM. Si es posible, no utilice un valor de profundidad mayor de 3. Puede aumentar la memoria con la opción `-Xmx` de la JVM en `api.bat` o `api.sh`.

--indent ***núm espacios***
Es la sangría que se utiliza para la salida XML resultante.

--changetype ***tipo***

Es el tipo de cambio, de entre los valores siguientes:

0 Creado

1 Actualizado

2 Suprimido

3 Creaciones y actualizaciones

4 Todos los cambios

--from ***fecha-inicial***
Fecha de inicio del parámetro change. Utilice el formato `mm/dd/aa hh:mm:ss AM|PM`.

--end ***fecha-final***
Fecha de finalización del parámetro change. Utilice el formato `mm/dd/aa hh:mm:ss AM|PM`.

-o|--outfile ***archivo-local-donde-escribir***
Nombre del archivo para redirigir la salida del mandato **find**.

-x|--maxfilesize ***tamaño***
Es el archivo de salida que se puede dividir en varios archivos más pequeños especificando el tamaño de archivo máximo en bytes. La salida se divide en varios archivos por debajo del tamaño de archivo máximo, si es posible.

-s|--suppress ***lista-de-clases-que-suprimir***
Lista de clases que se omitirán de los resultados de búsqueda. Las clases son clases de nombre de objeto de modelo, como ComputerSystem u OperatingSystem.

--guid ***guid-objeto***
GUID del objeto para el cual se está ejecutando el mandato **find**.

--mssguid ***guid-mss*****|--mssname** ***nombre-mss***
GUID o nombre del sistema de software gestionado (MSS).

--count ***consulta-mql***
Devuelve el número de objetos que satisfacen la consulta MQL.

consulta-mql
Consulta especificada utilizando MQL (Model Query Language), como por ejemplo SELECT attributes FROM object type [WHERE expression]. Puede utilizar nombres largos o abreviados para los tipos de objeto de este

argumento. Para obtener más información sobre nombres de clase y consultas MQL, consulte los conceptos relacionados.

raíz
Objeto de modelo que sirve como raíz de la salida XML resultante. Puede utilizar nombres largos o abreviados para los tipos de objeto de este argumento. Para obtener más información sobre los nombres de clase, consulte el concepto relacionado.

Ejemplos

- Este mandato busca sistemas informáticos y guarda los resultados en el archivo `cs_output.xml` con un tamaño de archivo máximo de 1000 bytes:

  ```
  api.sh -u usuario -p contraseña -H host -P puerto find -o cs_output.xml -x 1000 ComputerSystem
  ```

- Este mandato cuenta el número de objetos ComputerSystem de la base de datos:

  ```
  api.sh -u usuario -p contraseña find --count "select * from ComputerSystem"
  ```

- Este mandato limita la búsqueda a un nivel de profundidad definido. Los resultados se guardan en el archivo `cs_output.xml`:

  ```
  api.sh -u usuario -p contraseña -H host -P puerto find --depth profundidad
  -o cs_output.xml ComputerSystem
  ```

Mandato import

El mandato **import** importa datos a la base de datos de TADDM.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **import** [**--timeout** *tiempo*] [**--mssguid** *guid_mss* **--mssname** *nombre_mss*] [**--maxfilesize** *tamaño*] *local-directory-to-read-data-from*

Parámetros

import
Ejecuta el mandato **import**.

--timeout ***tiempo***
Valor de tiempo de espera, resulta útil en importaciones de archivos grandes. Especifique el valor en segundos.

--mssguid ***guid-mss*** | **--mssname** ***nombre-mss***
GUID o nombre del sistema de software gestionado al que están asociados los datos importados.

directorio-local-donde-leer-datos
Nombre del directorio del que se importan los datos.

Ejemplo

Este mandato importa datos en TADDM. El mandato trata de importar todos los archivos al directorio especificado. Si el mandato encuentra un archivo XML no válido, devuelve una excepción, pero el mandato continúa la importación.

```
api.sh -u usuario -p contraseña -H host import directory/
```

Mandato merge

El mandato **merge** agrupa el CI en la base de sus GUID.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **merge** *DurableCI_GUID TransientCI_GUID* [**type** *type*]

Parámetros

merge
Ejecuta el mandato **merge**.

DurableCI_GUID
El GUID que persiste después de fusionarse con el otro.

TransientCI_GUID
El GUID que se fusiona con el GUID duradero y se elimina de la base de datos.

type ***type***
Tipo de fusión. Hoy dos valores posibles:

- *0*, que representa un tipo superficial de la fusión, lo que significa que solo se fusionan los objetos principales. Los objetos hijo del GUID transitorio se sustituyen por los objetos hijo del GUID duradero. El tipo superficial de la fusión es el tipo predeterminado.
- *1*, que representa un tipo profundo de la fusión, lo que significa que se fusionan los objetos principales junto con sus objetos hijo.

Nota: Actualmente solo está habilitado el tipo superficial de la fusión. Al especificar *1* para el parámetro **type** se sigue utilizando el tipo superficial. El tipo profundo se habilitará en el futuro.

Importante: Al fusionar CI ambos CI deben ser de la misma clase. Por ejemplo, puede fusionar dos objetos ComputerSystem, pero no puede fusionar un objeto ComputerSystem con un objeto ApplicationServer.

Ejemplo

El siguiente mandato fusiona dos GUID especificados mediante el tipo superficial de la fusión. Escriba el mandato en una sola línea.

```
api.sh -u user -p password -H host -P port merge 10A5794E86C53A0BBB10F262055CB3EA
C172810FD1CF3E108B8127BC47D2667B type 0
```

Mandato naming

El mandato **naming** devuelve los GUID asociados a un elemento de configuración (CI). El mandato devuelve solo los GUID de nivel superior del archivo XML.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **naming -f** *model-object-xml-file*

Parámetros

naming
Ejecuta el mandato **naming**.

-f ***archivo-xml-objeto-modelo***
Ubicación y nombre del archivo XML que contiene el elemento de configuración (objeto de modelo).

Ejemplo

Este mandato muestra los GUID de los CI del archivo XML:

```
api.sh -u usuario -p contraseña -H host naming sample.xml
```

Mandato rediscover

El mandato **rediscover** vuelve a descubrir objetos.

Nota: Antes de ejecutar el redescubrimiento, asegúrese de que el parámetro **com.collation.rediscoveryEnabled** está definido como `true`. Los objetos que se deseen volver a descubrir tienen que haberse descubierto antes al menos una vez.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* **rediscover** *guid1* [*guid2 guid3 ... guidn*]

Parámetros

rediscover
Ejecuta el mandato **rediscover**.

guid1 [guid2 guid3 ... guidn]
Los GUID de los objetos que se quieren volver a descubrir.

Ejemplo

El mandato siguiente vuelve a descubrir los objetos con los GUID especificado. Escriba el mandato en una sola línea.

```
./api.sh -u administrator -p collation rediscover 4C778F231DB03FCF815E38EAD7CB1D66
B609D10039C23AE9A51E433EC311A9EE F503F3B70DF93587A635D574A78B248A
```

Mandato servers

El mandato **servers** muestra información sobre los servidores en un despliegue de servidor en modalidad continua.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **servers getservers**

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **servers getdiscoveryservers**

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **servers getdiscoveryserverstatus**

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **servers getstorageservers**

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **servers getstorageserverstatus**

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **servers getlocalserver**

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **servers getlocalserverstatus**

Parámetros

servers
Ejecuta el mandato **servers**.

getservers
Lista todos los servidores activos de descubrimiento y almacenamiento.

getdiscoveryservers
Lista todos los servidores de descubrimiento activos.

getdiscoveryserverstatus
Muestra información detallada sobre el estado y el rendimiento para todos los servidores de descubrimiento.

getstorageservers
Lista todos los servidores de almacenamiento activos.

getstorageserverstatus
Muestra información detallada sobre el estado y el rendimiento para todos los servidores de almacenamiento.

getlocalserver
Muestra información sobre el servidor local.

getlocalserverstatus
Muestra información detallada sobre el estado y el rendimiento del servidor local.

Ejemplos

- Este mandato lista todos los servidores de descubrimiento y los servidores de almacenamiento activos:

  ```
  api.sh -u usuario -p contraseña -H host -P puerto servers getservers
  ```

- Este mandato lista todos los servidores de descubrimiento activos:

  ```
  api.sh -u user -p contraseña -H host -P puerto servers getdiscoveryservers
  ```

- Este mandato muestra información detallada sobre el estado y el rendimiento para todos los servidores de almacenamiento:

  ```
  api.sh -u usuario -p contraseña -H host -P puerto servers getstorageserverstatus
  ```

Mandato sync

El mandato **sync** inicia la sincronización de un servidor de dominio.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **sync start** *dominio*

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **sync status** *dominio*

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **sync logs** *dominio*

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **sync stop** *dominio*

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* **[-H** | **--host** *host*] **[-P** | **--port** *puerto*] **[-T** | **--truststorefile] sync delete** *dominio*

Parámetros

sync
Ejecuta el mandato **sync**.

start
Inicia la sincronización de un servidor de dominio.

status
Muestra el estado de sincronización de un servidor de dominio.

logs
Muestra los archivos de registro de la sincronización de un servidor de dominio.

stop
Detiene la sincronización de un servidor de dominio.

delete
Suprime la sincronización de un servidor de dominio.

dominio
Es el nombre del dominio que se añade al servidor de sincronización, no el nombre de host del servidor de dominio.

Ejemplo

Este mandato inicia la sincronización de un servidor de dominio:

```
api.sh -u usuario -p contraseña -H host sync dominio
```

Mandato topology

El mandato **topology** muestra el estado de grupos de topología.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* **[-H** | **--host** *host*] **[-P** | **--port** *puerto*] **[-T** | **--truststorefile] topology groups**

Parámetros

topology
Ejecuta el mandato **topology**.

groups
Muestra el estado detallado de los grupos de topología.

Ejemplo

Este mandato muestra el estado detallado de los grupos de topología:

```
api.sh -u user -p password  topology groups
```

Mandato version

El mandato **version** gestiona las versiones en el TADDM.

Sintaxis del mandato

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **version** [**-c** | **--create** *nombre-versión descripción-versión*]

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **version** [**-e** | **--createempty** *nombre-versión descripción-versión*]

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **version** [**-d** | **--delete** *id-o-nombre-versión*]

api.sh | **api.bat -u** | **--user** *usuario* **-p** | **--password** *contraseña* [**-H** | **--host** *host*] [**-P** | **--port** *puerto*] [**-T** | **--truststorefile**] **version getall**

Parámetros

version
Ejecuta el mandato **version**.

-c|--create ***nombre-versión descripción-versión***
Crea una nueva versión con el nombre proporcionado.

-e|--createempty ***nombre-versión descripción-versión***
Crea una nueva versión vacía con el nombre proporcionado.

-d|--delete ***id-o-nombre-versión***
Suprime la versión especificada.

getall
Muestra todas las versiones existentes.

Ejemplos

- Este mandato crea una versión:

```
api.sh -u user -p contraseña -H host -P puerto version -create "version1.0"
 "This is the initial version"
```

- Este mandato suprime una versión:

```
api.sh -u usuario -p contraseña -H host -P puerto version -delete "version1.0"
```

Desarrollo de extensiones de servidor personalizado

Puede utilizar las extensiones de servidor personalizado para descubrir aquellos destinos para los que TADDM tiene soporte limitado o soporte integrado específico, o bien para añadir funcionalidades a TADDM.

Las extensiones de servidor personalizado proporcionan una interfaz de programación de aplicaciones (API) que se puede utilizar para crear programas que definen los atributos definidos en el modelo de datos común (CDM) o los atributos ampliados que se han añadido al CDM utilizando Data Management Portal.

Estas extensiones basadas en Jython se ejecutan dentro del motor de descubrimiento de TADDM, que proporciona a las extensiones de servidor personalizado una infraestructura para utilizar muchos de los bloques de creación del sensor en TADDM.

Las extensiones de servidor personalizado ofrecen las características siguientes:

- Puede utilizar la interfaz de usuario de TADDM y la API para ver los atributos descubiertos.
- Los mensajes de extensión de servidor personalizado se graban en el registro de Discovery Manager y en el registro correspondiente del sistema informático (o en los registros de CustomAppServerSensor si se han habilitado los registros de sensor partidos).
- No se necesita software adicional para utilizar el sistema.

Limitación: No puede utilizar sensores basados en scripts para crear extensiones de servidor personalizado.

Limitación: Si amplía el descubrimiento en los sistemas operativos Windows para ejecutar mandatos que devuelvan la salida con caracteres Unicode, dichos caracteres no se almacenan.

Fix Pack 3 En TADDM 7.3.0.3, y posterior, se admite el almacenamiento de caracteres Unicode. Sin embargo, primero debe enviar la salida a un archivo y a continuación leer el archivo. En el siguiente ejemplo se muestran mandatos que se pueden utilizar para estas dos operaciones. En el ejemplo, la salida del mandato **unicodetest.bat** se envía al archivo `c:\\r.txt`.

```
os_handle.executeCommand("c:\\unicodetest.bat | out-file c:\\r.txt")
output =  os_handle.executeCommand("cmd.exe /u /c type c:\\r.txt")
```

Visión general

Puede utilizar Data Management Portal y la API de extensiones de servidor personalizado para definir atributos ampliados o incorporados.

El desarrollo de una extensión de servidor personalizado implica la identificación de los atributos ampliados e integrados que quiere definir, así como la adición de los atributos ampliados al modelo de datos común. Una vez hecho esto, tendrá que escribir la aplicación para definir los atributos y comprobar que los atributos se están recopilando según lo esperado.

A continuación, se indica el procedimiento para desarrollar una extensión de servidor personalizado:

1. Identifique los atributos ampliados o incorporados que quiera recopilar.
2. Si ha identificado atributos ampliados, añádalos al modelo de datos común mediante Data Management Portal.

   Para obtener más información, consulte “Gestión de atributos ampliados”.
3. Desarrolle la aplicación para definir los atributos utilizando la API de extensiones de servidor personalizado.

   Para obtener más información sobre la API de extensiones de servidor personalizado, consulte “API de extensiones de servidor personalizado” en la página 132. Para ver una aplicación de ejemplo, consulte “Aplicación de ejemplo de extensiones de servidor personalizado” en la página 162.
4. Ejecute la aplicación de extensiones de servidor personalizado.
5. Mediante Data Management Portal, verifique que los atributos se estén definiendo según lo esperado.

Gestión de atributos ampliados

Es necesario definir los atributos ampliados para poder recopilar los atributos utilizando la aplicación de servidor personalizada.

Acerca de esta tarea

Puede utilizar Data Management Portal para añadir o suprimir atributos ampliados de un tipo de componente del modelo de datos común.

Tabla 30 describe los valores que puede especificar para los atributos ampliados.

Tabla 30. Atributos ampliados

Campo	Descripción
Tipo de componente	El tipo de componente.
Nombre de atributo ampliado	Nombre del atributo ampliado.
Tipo de atributo ampliado	Tipo de atributo ampliado.
Nombre de atributo heredado	Si esta clase es una subclase de otra clase del modelo de datos común, y si se han definido atributos ampliados para la clase padre, estos atributos aparecerán aquí.
Tipo de atributo heredado	Tipo de atributo heredado.

Procedimiento

Para especificar los atributos ampliados, siga estos pasos:

1. Inicie Data Management Portal.
2. Seleccione **Editar** > **Atributos ampliados** en el menú principal.
3. Elija un tipo de componente de la lista **Tipo de componente**. La ventana Definir atributos ampliados muestra los atributos ampliados definidos actualmente para el tipo de componente seleccionado.
4. Para añadir un atributo ampliado, pulse **Nuevo**. Especifique el nombre de atributo y el tipo de atributo en los campos correspondientes y pulse **Aceptar**. El sistema añade el nombre de atributo a la lista de atributos ampliados.
5. Para suprimir un atributo ampliado, siga estos pasos:
   a. Seleccione el tipo de componente correspondiente de la lista **Tipo de componente**.
   b. Seleccione el atributo que quiera eliminar de la ventana Definir atributos ampliados.
   c. Pulse **Suprimir**.
6. Pulse **Aceptar** para guardar los cambios y descartar la ventana, o bien **Cancelar** para descartar la ventana sin guardar los cambios.

API de extensiones de servidor personalizado

La API de extensiones de servidor personalizadas proporciona un conjunto de funciones que se pueden utilizar para crear aplicaciones de Jython que recuperen información sobre procesos en ejecución, ejecución de mandatos, captura de archivos, normalización de datos, creación de nuevos objetos del modelo de datos común y definición de atributos o atributos ampliados en el modelo de datos común.

En esta sección, se describen las acciones que se deben realizar antes de empezar a utilizar la API de extensiones de servidor personalizado y, a continuación, se proporciona una visión general de los tipos de funciones disponibles en la API. La

sección incluye también una descripción de cada clase de función disponible, junto con código de ejemplo que muestra los elementos más comunes que se tienen que incluir en las aplicaciones.

También puede ampliar las plantillas de servidor personalizado y sistemas informáticos si ejecuta scripts Jython. Para obtener más detalles, consulte el tema *Ampliación de plantillas de servidor personalizado y sistemas informáticos* en la *Guía del usuario* de TADDM.

Requisitos previos para utilizar la API de extensiones de servidor personalizado

Debe comprender los conceptos clave antes de utilizar la API de extensiones de servidor personalizado.

Para poder crear aplicaciones mediante la API de extensiones de servidor personalizado, debe comprender los conceptos siguientes:

- Modelo de datos común (CDM)

  Si va a utilizar las extensiones de servidor personalizado para crear nuevos ModelObjects de CDM, es necesario que defina, al menos, una regla de denominación. De lo contrario, TADDM no podrá generar un identificador exclusivo global para el objeto y se generará un error en el sensor en el que se ejecuta la extensión (se emitirá un error de almacenamiento).
- Cómo hacer que la aplicación configure el entorno de Jython para utilizarlo con TADDM y la API de extensiones de servidor personalizado.

  Puede utilizar el archivo `sensorstub.py` del directorio `$COLLATION_HOME/lib/sensor-tools` como base para la aplicación de extensiones de servidor personalizado. Este código actúa como el apéndice que configura correctamente el entorno de Jython, pero no realiza operaciones en el sistema.
- Cómo crear y gestionar servidores personalizados.

  Para obtener más información, consulte la *Guía del usuario* de TADDM.

Visión general de las funciones

La API de extensiones de servidor personalizado ofrece varias clases de funciones que le ayudarán a escribir extensiones de servidor personalizado.

La API de extensiones de servidor personalizado consta de un conjunto de funciones de Python que se pueden utilizar para ejecutar mandatos y gestionar procesos, realizar búsquedas de DNS y obtener acceso a directorios y archivos de destinos remotos. La API proporciona también funciones para manipular las direcciones IP y de control de acceso a soportes (MAC), además de utilizar los descriptores de contexto del sistema operativo para recuperar información.

La API proporciona también un conjunto de funciones de programa de utilidad que se pueden utilizar para realizar tareas útiles en las aplicaciones.

Las siguientes categorías de funciones están disponibles en la API de extensiones de servidor personalizado.

Prestación
: Utilizar prestaciones como ExecuteCapability, MibQueryCapability u OsInfoCapability.

Mandato y proceso
: Ejecutar mandatos en un destino, además de gestionar y visualizar la información relacionada con el proceso.

DNS y dominios
Realizar búsquedas de nombres de dominio y validar los nombres de dominio completos.

Acceso a archivos
Listar el contenido de directorios y capturar archivos de destinos remotos.

Dirección IP y MAC
Manipular y convertir las direcciones IP y MAC.

Sistema operativo
Crear y utilizar los descriptores de contexto del sistema operativo para recuperar información.

Vía de acceso
Convertir los caracteres de separador de vía de acceso de Windows y Unix.

Programa de utilidad
Inicializar la API del servidor personalizado y realizar distintas tareas de utilidad.

Información de la versión
Determinar los números de versión de la API.

Funciones de capacidad:

Las funciones de capacidad le permiten utilizar prestaciones como ExecuteCapability, MibQueryCapability u OsInfoCapability. Utilizar funciones de capacidad facilita la realización de una operación necesaria en un destino especificado.

Puede utilizar la función de prestaciones para recuperar la fábrica responsable de crear prestaciones para un destino especificado. Tabla 31 describe las funciones que puede utilizar.

Tabla 31. Funciones de capacidad

Función	Descripción
getSimpleCapabilitiesFactory	Devuelve SimpleCapabilitiesFactory para una dirección IP dada.

Funciones de proceso y mandatos:

Las funciones de proceso y los mandatos le permiten ejecutar mandatos en un destino, así como gestionar y visualizar información relacionada con los procesos.

Puede utilizar las funciones de proceso y mandatos para ejecutar y mandato en un destino, especificando, si lo desea, un valor de tiempo de espera que determine durante cuánto tiempo se puede ejecutar el mandato. También puede utilizar las funciones para añadir un proceso de tiempo de ejecución a la agrupación de procesos y devolver el mapa de conexiones, la lista de puertos, el mapa de procesos de tiempo de ejecución y los procesos del servidor asociados a identificadores de procesos.

Tabla 32 en la página 135 describe las funciones que puede utilizar.

Tabla 32. Funciones de proceso y mandatos

Función	Descripción
addProcessToPool	Añadir un proceso de tiempo de ejecución a una agrupación de procesos.
executeCommand	Ejecutar un mandato en el destino.
executeCommandWithTimeout	Ejecutar un mandato en el destino con un tiempo de espera que especifique durante cuánto tiempo se puede ejecutar el mandato.
getPidConnectionMap	Devuelva un diccionario de Python de listas de Python con la información siguiente: • claves: ID de proceso • listas: conexiones TCP de los ID de proceso
getPidPortList	Devuelva un diccionario de Python de listas de Python con la información siguiente: • claves: ID de proceso • listas: puertos que utiliza el proceso para escuchar o para conectarse
getPidToRuntimeProcessMap	Devolver un diccionario de Python con la información siguiente: • ID de proceso • información de procesos de tiempo de ejecución
getProcessByPid	Devolver el objeto RuntimeProcess del CDM asociado a un ID de proceso determinado.
getServerProcesses	Devuelva un diccionario de Python de listas de Python con la información siguiente: • claves: ID de proceso • listas: direcciones de enlace de los puertos de escucha

Funciones del modelo de datos común:

Las funciones del modelo de datos común le permiten gestionar el modelo de datos común.

Puede utilizar las funciones del modelo de datos común (CDM) para crear y clonar nuevos objetos del CDM y definir el valor de los atributos ampliados en objetos del CDM.

Tabla 33 describe las funciones que puede utilizar.

Tabla 33. Funciones del modelo de datos común

Función	Descripción
cloneModelObject	Crear una copia de ModelObject del CDM.
newModelObject	Crear un objeto del CDM.
setExtendedAttributes	Definir los valores de los atributos ampliados.

Funciones de dominio y DNS:

Las funciones de dominio y DNS le permiten realizar búsquedas de nombres de dominio y validar nombres de dominio completos.

Puede utilizar las funciones de dominio y DNS para realizar búsquedas de nombres del servidor de TADDM y nombres extraídos de una configuración remota. También puede utilizar estas funciones para validar un nombre de dominio completo (FQDN).

Tabla 34 describe las funciones que puede utilizar.

Tabla 34. Funciones de DNS

Función	Descripción
getLocalDNSLookup	Realizar una búsqueda por nombre en el servidor de TADDM.
getRemoteDNSLookup	Buscar un nombre extraído de una configuración remota que quizás no se resuelva en el servidor de TADDM.
validateFqdn	Comprobar un FQDN para asegurarse de que cumple con las reglas señaladas en RFC 1035.

Funciones de acceso a archivos:

Las funciones de acceso a archivos le permiten listar el contenido de directorios y capturar archivos de destinos remotos.

Puede utilizar las funciones de acceso a archivos para listar el contenido de un directorio y para capturar el contenido y los metadatos de los archivos en destinos remotos. Tabla 35 describe las funciones que puede utilizar.

Tabla 35. Funciones de acceso a archivos

Función	Descripción
getFile	Capturar un archivo de un destino remoto y devolver el contenido y los metadatos del archivo.
getFileWithLengthLimit	Capturar un archivo, hasta la longitud máxima especificada, en un destino remoto y devolver el contenido y los metadatos del archivo.
listDirectory	Devolver una lista de Python con el contenido de un directorio en un destino remoto.

Funciones de dirección IP y MAC:

Las funciones de dirección IP y MAC le permiten manipular y convertir las direcciones IP y MAC.

Puede utilizar las funciones de dirección IP y MAC para manipular las direcciones MAC, validar direcciones IP y convertir direcciones IP entre distintas representaciones. Tabla 36 en la página 137 describe las funciones que puede utilizar.

Tabla 36. Funciones de dirección IP y MAC

Función	Descripción
binToDot	Convertir una representación binaria de una dirección IP en una notación con puntos.
bitsMaskToDottedDecimalMask	Convertir la notación de máscara de bits de red en una notación decimal con puntos, por ejemplo, 24 en 255.255.255.0.
calcNetworkAddress	Calcular la dirección de red con una dirección IP y una máscara de red.
canonicalMac	Eliminar los separadores o la notación de raíz de una dirección MAC y devolver la dirección MAC hexadecimal como una serie con caracteres alfabéticos en mayúsculas.
classlessNotation	Calcular la notación sin clase de una red IP.
dotToBin	Convertir una dirección IPv4 de la notación con puntos al formato binario.
ipInSubnet	Determinar si una dirección IP es miembro de una subred dada y no la dirección de difusión.
networkToList	Devolver todas las direcciones IP que son miembros de la representación del CDM del parámetro IpNetwork.
validateIp	Validar una dirección IP en notación con puntos.

Funciones del sistema operativo:

Las funciones del sistema operativo le permiten crear y utilizar los descriptores de contexto del sistema operativo para recuperar información.

Puede utilizar las funciones del sistema operativo para crear descriptores de contexto del sistema operativo y recuperar información utilizando estos descriptores de contexto. Tabla 37 describe las funciones que puede utilizar.

Tabla 37. Funciones del sistema operativo

Función	Descripción
getAppTarget	Devuelva la información siguiente: • descriptor de contexto del sistema operativo para el destino • objeto de resultado del sensor • objeto del servidor de aplicaciones para el destino • entorno de proceso para el destino • objeto de semilla que ha provocado que el motor de descubrimiento genere el destino

Tabla 37. Funciones del sistema operativo (continuación)

Función	Descripción
getCsTarget	Devuelva la información siguiente: • descriptor de contexto del sistema operativo para el destino • objeto de resultado del sensor • objeto del sistema informático para el destino • entorno de proceso para el destino • objeto de semilla que ha provocado que el motor de descubrimiento genere el destino
getComputerSystem	Devolver un objeto al que está conectado el descriptor de contexto del sistema operativo con los atributos completados.
getNewOsHandle	Intentar crear un nuevo descriptor de contexto de sistema operativo en el destino especificado. Se puede utilizar para comunicarse con una máquina distinta a aquella para la que se inició en un principio el sensor personalizado.
getOperatingSystem	Devolver un objeto que representa el sistema operativo al que está conectado el descriptor de contexto del sistema operativo.
queryRegistry	Devolver una clave de registro como XML.

Funciones de la vía de acceso:

Las funciones de la vía de acceso le permiten convertir los caracteres de separador de vía de acceso de Windows y Unix.

Puede utilizar las funciones de la vía de acceso para sustituir los caracteres de separador de vía de acceso de Microsoft Windows y Unix. Tabla 38 describe las funciones que puede utilizar.

Tabla 38. Funciones de la vía de acceso

Función	Descripción
unixSlashes	Sustituir los caracteres de separador de vía de acceso de Windows por separadores de vía de acceso de Unix
windowsSlashes	Sustituir los caracteres de separador de vía de acceso de Unix por separadores de vía de acceso de Windows

Funciones de programa de utilidad:

Las funciones de programa de utilidad le permiten inicializar la API de extensiones de servidor personalizado y realizar distintas tareas útiles.

Pueden utilizarse las funciones de programas de utilidad para inicializar la API de extensiones de servidor personalizadas. Es posible crear una matriz en Python para utilizarla con determinadas funciones de Java y TADDM, además de dividir las líneas de mandatos en sus componentes.

La Tabla 39 describe las funciones que puede utilizar.

Tabla 39. Funciones de programa de utilidad

Función	Descripción
findElementsForXPath	Consulta objetos y devuelve una colección de objetos a partir de la consulta.
getArray	Obtener una matriz de Python para utilizarla con determinadas funciones de Java y TADDM.
init	Inicializar la API de extensión de servidor personalizado.
splitArgs	Dividir una línea de mandatos en sus componentes y devolverla como una secuencia de Python.

Funciones de información sobre la versión:

Las funciones de información sobre la versión le permiten determinar los números de versión de la API.

Puede utilizar las funciones de información sobre la versión para determinar los números de versión principal y menor de la API de extensiones de servidor personalizado, además del número de versión de TADDM. Tabla 40 describe las funciones que puede utilizar.

Tabla 40. Funciones de información sobre la versión

Función	Descripción
getApiMinorVersion	Devolver la versión menor de la API.
getApiVersion	Devolver la versión principal de la API.
getTADDMVersion	Devolver el número de versión de TADDM.

Referencia de funciones

Esta referencia describe las funciones disponibles en la API de extensiones de servidor personalizado. Las funciones aparecen listadas en orden alfabético.

Función addProcessToPool:

Añada un proceso de tiempo de ejecución a una agrupación de procesos.

Descripción

La función addProcessToPool añade un objeto RuntimeProcess del modelo de datos común (CDM) a una ProcessPool. Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

addProcessToPool (*rp*, *pool*, **os*)

Parámetros

rp Objeto RuntimeProcess del CDM

pool
Objeto ProcessPool del CDM

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

La función devuelve el objeto del descriptor de contexto del sistema operativo conectado al nuevo destino.

Excepciones

OsException.

Función binToDot:

Convierta una representación binaria de una dirección IP en una notación con puntos.

Descripción

La función binToDot convierte una representación binaria de una dirección IP en una notación con puntos.

Sintaxis de la función

binToDot (*binIp*)

Parámetros

binIp
Registro de Python que contiene la representación binaria de la dirección IP

Devoluciones

La función devuelve la representación de una dirección de red IP en notación con puntos.

Excepciones

Ninguna.

Función bitsMaskToDottedDecimalMask:

Convierta la máscara de bits de red a la notación decimal con puntos.

Descripción

La función bitsMaskToDottedDecimalMask convierte una representación de notación de la máscara de bits de red en una representación decimal con puntos, por ejemplo 24 a 255.255.255.0. Tenga en cuenta que debe omitir la / inicial (barra

inclinada) en la máscara de bits. Los recuentos de bits válidos son 8 y 16-32.

Sintaxis de la función

bitsMaskToDottedDecimalMask (*bits*)

Parámetros

bits
Representación de serie del número de bits de red

Devoluciones

La función devuelve el formato decimal con puntos de la dirección.

Excepciones

Ninguna.

Función calcNetworkAddress:

Calcule la dirección de red con una dirección IP y una máscara de red.

Descripción

La función calcNetworkAddress calcula la dirección de red utilizando la dirección IP y la máscara de red especificadas.

Sintaxis de la función

calcNetworkAddress (*ip*)

Parámetros

ip La representación de serie de la dirección IP

(*máscara*)
Representación de serie de la máscara de subred

Devoluciones

La función devuelve la representación de serie de la dirección de red IP.

Excepciones

Ninguna.

Función canonicalMac:

Elimine los separadores o la notación de raíz de una dirección MAC y devuelva la dirección MAC hexadecimal.

Descripción

La función canonicalMac elimina los separadores o la notación de raíz de una dirección MAC y devuelve la dirección MAC hexadecimal como una serie con caracteres alfanuméricos en mayúsculas.

Sintaxis de la función

canonicalMac (*mac*)

Parámetros

mac
Dirección MAC

Devoluciones

La función devuelve una representación de serie de la dirección MAC canónica.

Excepciones

Ninguna.

función classlessNotation:

Calcule la notación sin clase de una red IP.

Descripción

La función classlessNotation calcula la notación sin clase de una red IP.

Sintaxis de la función

classlessNotation (*ip*, *máscara*)

Parámetros

ip Representación de serie de la red IP

máscara
Representación de serie de la máscara de subred

Devoluciones

La función devuelve la notación sin clase en forma de serie.

Excepciones

Ninguna.

Función cloneModelObject:

Cree una copia del ModelObject de un modelo de datos común.

Descripción

La función cloneModelObject crea una copia de ModelObject del modelo de datos común (CDM), en una recurrencia indefinida por atributos secundarios de ModelObject.

Sintaxis de la función

cloneModelObject (*mo*)

Parámetros

mo ModelObject del CDM que se va a clonar

Devoluciones

La función devuelve el clon de ModelObject del CDM.

Excepciones

Ninguna.

Función dotToBin:

Convierte una dirección IPv4 de la notación con puntos al formato binario.

Descripción

La función dotToBin convierte una dirección IPv4 de la notación con puntos al formato binario.

Sintaxis de la función

dotToBin (*ip*)

Parámetros

ip Representación de serie A de una dirección IP

Devoluciones

La función devuelve la dirección IP en formato binario como tipo largo de Python.

Excepciones

NumberFormatException si la dirección IP no es válida.

Función executeCommand:

Ejecute un mandato en el destino.

Descripción

La función executeCommand ejecuta un mandato en el destino utilizando el tiempo de espera del mandato predeterminado, dos minutos. Tenga en cuenta que la función añade delante el valor PATH del tipo de destino, tal y como se especifica en el archivo collation.properties.

Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

executeCommand (*cmd*, **os*)

Parámetros

cmd
Serie de mandato que ejecutar

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

La función devuelve una serie que contiene la salida del mandato.

Excepciones

OsException.

Función executeCommandWithTimeout:

Ejecute un mandato en el destino con un tiempo de espera que especifique durante cuánto tiempo se puede ejecutar el mandato.

Descripción

La función executeCommandWithTimeout ejecuta un mandato en el destino, con un tiempo de espera que especifica durante cuánto tiempo se puede ejecutar el mandato. Tenga en cuenta que la función añade delante el valor PATH del tipo de destino, tal y como se especifica en el archivo collation.properties.

Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

executeCommandWithTimeout (*cmd*, *tiempoespera*, **os*)

Parámetros

cmd
Serie de mandato que ejecutar

timeout
Tiempo durante el que se puede ejecutar el mandato (en milisegundos)

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

La función devuelve una serie que contiene la salida del mandato.

Excepciones

OsException.

Función getApiMinorVersion:

Devuelve la versión menor de la API.

Descripción

La función getApiMinorVersion devuelve la versión menor de la API de extensiones de servidor personalizado.

Sintaxis de la función

getApMinoriVersion

Parámetros

Ninguno

Devolución

La función devuelve el número de versión menor de la API.

Excepciones

Ninguna.

Función getApiVersion:

Devuelve la versión principal de la API.

Descripción

La función getApiVersion devuelve la versión principal de la API de extensiones de servidor personalizado.

Sintaxis de la función

getApiVersion

Parámetros

Ninguno

Devolución

La función devuelve el número de versión principal de la API.

Excepciones

Ninguna.

Función getAppTarget:

Devuelve una tupla con información sobre el destino de la aplicación.

Descripción

La función getAppTarget devuelve la información siguiente:

- Descriptor de contexto del sistema operativo del destino
- Objeto de resultado del sensor

- Objeto del servidor de aplicaciones para el destino
- Entorno de proceso para el destino
- Objeto de semilla que ha provocado que el motor de descubrimiento generase el destino

Sintaxis de la función

getAppTarget (*destino*)

Parámetros

destino
Correlación de destinos

Devoluciones

Devuelve una tupla que contiene el descriptor de contexto del sistema operativo para el destino, el objeto de resultado, el servidor de aplicaciones del modelo de datos común (CDM), el entorno de proceso, si lo hay, y el objeto de semilla.

Excepciones

Ninguna.

Función getArray:

Obtenga una matriz de Python para utilizarla con determinadas funciones de Java y TADDM.

Descripción

La función getArray devuelve una matriz de Python para utilizarla con determinadas funciones de Java y TADDM. La función utiliza el módulo jarray de Jython para crear la matriz.

Sintaxis de la función

getArray (*sec*, *nombre_clase*)

Parámetros

sec
Secuencia o lista de Python

nombre_clase
Nombre de clase Java completo que coincide con el tipo de objetos especificado en el parámetro **seq**

Devoluciones

La función devuelve una matriz de Java adecuada para pasarla a los métodos de Java que requieren una matriz.

Excepciones

Ninguna.

Función getComputerSystem:

Devuelve un objeto al que está conectado el descriptor de contexto del sistema operativo con los atributos llenos.

Descripción

La función getComputerSystem devuelve el objeto ComputerSystem del modelo de datos común (CDM) al que está conectado el descriptor de contexto del sistema operativo con los atributos llenos.

Sintaxis de la función

getComputerSystem (**os*)

Parámetros

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

La función devuelve el objeto ComputerSystem del CDM.

Excepciones

OsException.

Función getCsTarget:

Devuelve una tupla con información sobre el destino.

Descripción

La función getCsTarget devuelve una tupla con la información siguiente:

- Descriptor de contexto del sistema operativo del destino
- Objeto de resultado del sensor
- Objeto del sistema informático para el destino
- Entorno de proceso para el destino
- Objeto de semilla que ha provocado que el motor de descubrimiento generase el destino

Sintaxis de la función

getCsTarget (*destino*)

Parámetros

destino
Correlación de destino pasada a la extensión de servidor personalizado.

Devoluciones

La función devuelve una tupla que contiene el descriptor de contexto del sistema operativo, el objeto de resultado, el objeto ComputerSystem del CDM y el objeto de semilla.

Excepciones

Ninguna.

Función getFile:

Capture un archivo de un destino remoto y devuelva el contenido y los metadatos del archivo.

Descripción

La función getFile captura un archivo de un destino remoto y devuelve un objeto FileSystemContent del modelo de datos común (CDM) con el contenido y los metadatos del archivo.

Sintaxis de la función

getFile (*vía-acceso*, **os*)

Parámetros

vía-acceso
Vía de acceso del archivo que se va a capturar

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

Las funciones devuelven el objeto FileSystemContent del CDM con el contenido y los metadatos del archivo.

Excepciones

OsException.

Función getFileWithLengthLimit:

Capture un archivo, con la longitud máxima especificada, en un destino remoto y devuelva el contenido y los metadatos del archivo.

Descripción

La función getFileWithLengthLimit captura un archivo de un destino remoto y devuelve un objeto FileSystemContent del modelo de datos común (CDM) con el contenido y los metadatos del archivo.

Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

getFileWithLengthLimit (*vía-acceso*, *longitud*, **os*)

Parámetros

vía-acceso
Vía de acceso del archivo que se va a capturar

longitud
Longitud máxima que capturar (en bytes)

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

La función devuelve el objeto FileSystemContent del CDM con el contenido y los metadatos del archivo.

Excepciones

OsException.

Función getLocalDNSLookup:

Realice una búsqueda por nombre en el servidor de TADDM.

Descripción

La función getLocalDNSLookup realiza una búsqueda por nombre en el servidor de TADDM y devuelve un objeto DNSLookup del modelo de datos común (CDM) con el resultado. La función acepta también un descriptor de contexto del sistema operativo opcional, pero, aparte de eso, la búsqueda es siempre local.

Sintaxis de la función

getLocalDNSLookup (*nombre*, **os*)

Parámetros

nombre
Nombre que se va a resolver

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

La función devuelve el objeto DNSLookup del CDM.

Excepciones

OsException.

Función getNewOsHandle:

Cree un nuevo descriptor de contexto del sistema operativo en el destino especificado.

Descripción

La función getNewOsHandle trata de crear un nuevo descriptor de contexto del sistema operativo en el destino especificado. Puede utilizarlo para comunicarse con una máquina distinta a aquella para la que se inició en origen la extensión de servidor personalizado. Se produce una excepción si no se puede establecer una sesión de SSH o WMI mediante las listas de acceso configuradas actualmente en el servidor de TADDM.

Sintaxis de la función

getNewOsHandle (*ip*)

Parámetros

ip Dirección IP de la máquina a la que se quiere conectar

Devoluciones

La función devuelve el objeto del descriptor de contexto del sistema operativo conectado al nuevo destino.

Excepciones

OsException.

Función getOperatingSystem:

Devuelve un objeto que representa el sistema operativo al que está conectado el descriptor de contexto del sistema operativo.

Descripción

La función getOperatingSystem devuelve el objeto OperatingSystem del modelo de datos común (CDM) que representa el sistema operativo al que está conectado el descriptor de contexto del sistema operativo.

Sintaxis de la función

getOperatingSystem (**os*)

Parámetros

**os*
Objeto de descriptor de contexto del sistema operativo

Devoluciones

La función devuelve el objeto OperatingSystem del CDM.

Excepciones

OsException.

Función getPidConnectionMap:

Devuelve un diccionario de Python que contiene los ID de proceso y las conexiones de TCP de los ID de proceso

Descripción

La función getPidConnectionMap devuelve un diccionario de Python de listas de Python que contiene los ID de proceso, además de las claves y las listas de conexiones TCP de los ID de proceso.

Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

getPidConnectionMap ( )

Parámetros

Ninguno.

Devoluciones

La función devuelve un diccionario de Python de listas de Python que contiene la información siguiente:

- ID de proceso
- Conexiones TCP de los ID de proceso

Excepciones

Ninguna.

Función getPidPortList:

Devuelve un diccionario de Python que contiene los ID de proceso y los puertos que el proceso utiliza para escuchar o para conectarse.

Descripción

La función getPidPortList devuelve un diccionario de Python de listas de Python que contiene los ID de proceso, además de las claves y listas de puertos que el proceso utiliza para escuchar o para conectarse.

Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

getPidPortList (**os*)

Parámetros

**os*
(Opcional) Objeto de descriptor de contexto del sistema operativo.

Devoluciones

La función devuelve un diccionario de Python con la información siguiente:
- ID de proceso
- Listas de Python de objetos BindAddress del CDM

Excepciones

OsException.

Función getPidToRuntimeProcessMap:

Devuelve un diccionario de Python que contiene los ID de proceso y la información de proceso del tiempo de ejecución.

Descripción

La función getPidToRuntimeProcessMap devuelve un diccionario de Python con las claves que contienen los ID de proceso y los valores que representan a los objetos RuntimeProcess del modelo de datos común (CDM). Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

getPidToRuntimeProcessMap (**os*)

Parámetros

**os*
(Opcional) Objeto de descriptor de contexto del sistema operativo.

Devoluciones

La función devuelve un diccionario de Python con la información siguiente:
- ID de proceso
- RuntimeProcesses del CDM

Excepciones

Ninguna.

Función getProcessByPid:

Devuelve el objeto RuntimeProcess del modelo de datos común asociado a un ID de proceso determinado.

Descripción

La función getProcessByPid devuelve el objeto RuntimeProcess del modelo de datos común (CDM) asociado al ID de proceso especificado. Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

getProcessByPid (*pid*, **os*)

Parámetros

pid
ID de proceso.

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

La función devuelve el objeto RuntimeProcess del CDM o el valor "Ninguno" si el ID de proceso no existe.

Excepciones

Ninguna.

Función getRemoteDNSLookup:

Busque un nombre extraído de una configuración remota que quizás no se resuelva en el servidor de TADDM.

Descripción

La función getLocalDNSLookup realiza una búsqueda en el sistema especificado en el primer parámetro. Puede utilizar esta función para resolver los nombres extraídos de las configuraciones remotas que podrían no resolverse en el servidor de TADDM.

Sintaxis de la función

getRemoteDNSLookup (*ip*, *nombre*)

Parámetros

ip Dirección IP de la máquina donde se realizará la búsqueda

nombre
Nombre que se va a resolver

Devoluciones

La función devuelve el objeto DNSLookup del CDM.

Excepciones

OsException.

Función getServerProcesses:

Devuelve un diccionario de Python de listas de Python que contiene los ID de proceso y direcciones de enlace de los puertos de escucha.

Descripción

La función getServerProcesses devuelve un diccionario de Python de listas de Python de objetos BindAddress del modelo de datos común (CDM). Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

getServerProcesses (**os*)

Parámetros

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

Devuelve un diccionario de Python con la información siguiente:

- ID de proceso
- Objetos BindAddress del CDM para los puertos de escucha de los ID de proceso

Excepciones

OsException.

Función getSimpleCapabilitiesFactory:

Devuelve SimpleCapabilitiesFactory para una dirección IP dada.

Descripción

La función getSimpleCapabilitiesFactory devuelve un valor SimpleCapabilitiesFactory para una dirección IP dada. SimpleCapabilitiesFactory se puede utilizar para recuperar las prestaciones siguientes:

ExecuteCapability
Esta capacidad permite que se ejecute un mandato en un destino dado, independientemente del protocolo de comunicación utilizado.

MibQueryCapability
Esta capacidad permite que se ejecute una consulta de MIB en un destino determinado.

OsInfoCapability
Esta capacidad permite que se recupere información del sistema operativo en un destino determinado.

Para obtener más información sobre estas prestaciones, consulte el Javadoc que se describe en “Información Javadoc sobre TADDM” en la página 184:

Sintaxis de la función

getSimpleCapabilitiesFactory (*ip*)

Parámetros

ip Objeto IpV4Address que representa la dirección IP del host de destino.

Devolución

Esta función devuelve SimpleCapabilitiesFactory.

Excepciones

IllegalArgumentException si el parámetro IP es nulo o no es una dirección IPv4 válida.

Función getTADDMVersion:

Devuelve el número de versión de TADDM.

Descripción

La función getTADDMVersion devuelve la versión de TADDM utilizada.

Sintaxis de la función

getTADDMVersion

Parámetros

Ninguno

Devoluciones

La función devuelve la versión de TADDM utilizada.

Excepciones

Ninguna.

Función init:

Inicialice la API de extensión de servidor personalizado.

Descripción

La función init inicializa las rutinas del ayudante de la API de extensión de servidor personalizado.

Sintaxis de la función

init (*destino*)

Parámetros

destino
Correlación de destinos

Devoluciones

La función devuelve una tupla que contiene lo siguiente:
- Descriptor de contexto del sistema operativo para el destino
- Objeto de resultado
- ComputerSystem o AppServer del CDM
- Objeto de semilla
- Registrador para escribir en el entorno de registro del sensor (si el destino es AppServer)

Excepciones

Ninguna.

Función ipInSubnet:

Determine si una dirección IP es miembro de una subred dada y no la dirección de difusión.

Descripción

La función ipInSubnet determina si una dirección IP es miembro de una subred dada y no la dirección de difusión.

Sintaxis de la función

ipInSubnet (*ip*, *red*, *máscara*)

Parámetros

ip Representación de serie A de una dirección IP

red
Representación de serie de una red

máscara
Máscara de subred de la red

Devoluciones

La función devuelve lo siguiente:
- Valor distinto a cero si la dirección IP es miembro de la subred
- 0 si la dirección IP no es miembro de la subred

Excepciones

Ninguna.

Función listDirectory:

Devuelve una lista de Python con el contenido de un directorio en un destino remoto.

Descripción

La función listDirectory devuelve una lista de Python del contenido de un directorio en un destino remoto.

Sintaxis de la función

listDirectory (*vía-acceso*, **os*)

Parámetros

vía-acceso
Vía del acceso del directorio

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

La función devuelve la secuencia de Python del contenido del directorio.

Excepciones

OsException.

Función networkToList:

Devuelve todas las direcciones IP que son miembros de la representación del CDM del parámetro IpNetwork.

Descripción

La función networkToList devuelve todas las direcciones IP que son miembro de la representación del modelo de datos común (CDM) del parámetro IpNetwork.

Sintaxis de la función

networkToList (*red*)

Parámetros

red
Objeto IpNetwork del CDM

Devoluciones

La función devuelve una lista de Python de representaciones de serie de direcciones IP.

Excepciones

Ninguna.

Función newModelObject:

Crea un objeto del modelo de datos común (CDM).

Descripción

La función newModelObject crea un nuevo objeto de modelo del tipo de clase de modelo de datos común (CDM) especificado.

Sintaxis de la función

newModelObject (*nombre-clase*)

Parámetros

nombre_clase
Nombre de clase completo de CDM del ModelObject de CDM que se va a crear

Devoluciones

La función devuelve el ModelObject del CDM.

Excepciones

Ninguna.

Función queryRegistry:

Devuelve una clave de registro como XML.

Descripción

La función queryRegistry devuelve la clave de registro solicitada como XML. Esta función solo funciona si el descriptor de contexto del sistema operativo está conectado a un destino de Windows; de lo contrario, la función generará una excepción.

Si lo desea, puede pasar un descriptor de contexto del sistema operativo a la función. De lo contrario, se utilizará de forma predeterminada el descriptor de contexto del sistema operativo pasado en la correlación de destinos a la función init.

Sintaxis de la función

getOperatingSystem (*clave*, **os*)

Parámetros

clave
Clave de registro que captar

**os*
(Opcional) Objeto del descriptor de contexto del sistema operativo

Devoluciones

La función devuelve la representación XML de la clave de registro

Excepciones

OsException y MethodNotImplementedException.

Función setExtendedAttributes:

Defina los valores de los atributos ampliados.

Descripción

La función setExtendedAttributes acepta un ModelObject del modelo de datos común (CDM) y un diccionario de Python de pares nombre-valor y define los pares nombre-valor como atributos ampliados de ModelObject.

Sintaxis de la función

setExtendedAttributes (*mo*, *atrsamp*)

Parámetros

mo ModelObject del modelo de datos común

atrsamp
Diccionario de Python de pares nombre-valor donde el nombre es el nombre de atributo ampliado y el valor es una serie

Excepciones

IoException.

Función splitArgs:

Divida una línea de mandatos en sus componentes y devuélvala como una secuencia de Python.

Descripción

La función splitArgs divide una línea de mandatos en sus componentes y los devuelve como una secuencia de Python.

Sintaxis de la función

argsDiv (*líneamdt*)

Parámetros

líneamdt
Línea de mandatos (el parámetro se debe entrecomillar si contiene espacios incluidos)

Devoluciones

La función devuelve una secuencia de Python que contiene las señales de la línea de mandatos.

Excepciones

Ninguna.

Función unixSlashes:

Convierte los caracteres de separador de la vía de acceso de Windows en caracteres de separador de la vía de acceso de UNIX.

Descripción

La función unixSlashes convierte los caracteres de separador de la vía de acceso de Windows en caracteres de separador de la vía de acceso de UNIX.

Sintaxis de la función

unixSlashes (*vía de acceso*)

Parámetros

vía-acceso
Vía de acceso del sistema de archivos que puede contener separadores de la vía de acceso de Windows

Devoluciones

La función devuelve una vía de acceso del sistema de archivos que contiene solo separadores de la vía de acceso de UNIX.

Excepciones

Ninguna.

Función validateFqdn:

Compruebe el nombre de dominio completo para asegurarse de que cumple las reglas señaladas en RFC 1035.

Descripción

La función validateFqdn comprueba un nombre de dominio completo (FQDN) para asegurarse de que cumple las reglas especificadas en RFC 1035. Tenga en cuenta que la consola de gestión de descubrimiento no muestra los FQDN que no las cumplen.

Sintaxis de la función

validateFqdn (*fqdn*)

Parámetros

fqdn
Nombre de dominio completo

Devoluciones

La función devuelve los valores siguientes:

- Distinto de cero si el FQDN es válido
- 0 si el FQDN no es válido

Excepciones

Ninguna.

Función validateIp:

Valide una dirección IP en notación con puntos.

Descripción

La función validateIp valida que una dirección IP en forma de notación con puntos es una dirección IP válida.

Sintaxis de la función

validateIp (*ip*)

Parámetros

ip Representación de serie de una dirección IP en notación con puntos

Devoluciones

La función devuelve los valores siguientes:

- Distinto a cero si la dirección IP es válida
- 0 si la dirección IP no es válida

Excepciones

Ninguna.

Función windowsSlashes:

Convierte los caracteres de separador de la vía de acceso de UNIX en caracteres de separador de la vía de acceso de Windows.

Descripción

La función windowsSlashes convierte los caracteres de separador de la vía de acceso de UNIX en caracteres de separador de la vía de acceso de Windows.

Sintaxis de la función

windowsSlashes (*vía-acceso*)

Parámetros

vía-acceso
Vía de acceso del sistema de archivos que puede contener separadores de vía de acceso de UNIX

Devoluciones

La función devuelve una vía de acceso del sistema de archivos que solo contiene separadores de la vía de acceso de Windows.

Excepciones

Ninguna.

Aplicación de ejemplo de extensiones de servidor personalizado

Una aplicación corriente de extensiones de servidor personalizado incluye varios segmentos de código comunes.

La siguiente aplicación de ejemplo de extensiones de servidor personalizado muestra los elementos estándar y los segmentos de código que se pueden incluir en las aplicaciones:

```
import sys
import java

from java.lang import System
coll_home = System.getProperty("com.collation.home")

System.setProperty("jython.home",coll_home +
"/osgi/plugins/com.ibm.cdb.core.jython_1.0.0/lib")
System.setProperty("python.home",coll_home +
"/osgi/plugins/com.ibm.cdb.core.jython_1.0.0/lib")

jython_home = System.getProperty("jython.home")
sys.path.append(jython_home + "/Lib")
sys.path.append(coll_home + "/lib/sensor-tools")
sys.prefix = jython_home + "/Lib"

import traceback
import string
import re
import jarray
import sensorhelper

############################
# LogError      Error logger
############################
def LogError(msg):
        log.error(msg)
        (ErrorType, ErrorValue, ErrorTB) = sys.exc_info()
        traceback.print_exc(ErrorTB)

############################
# main
############################

try:
 (os_handle, result, appserver,seed,log,env) = sensorhelper.init(targets)

 response = sensorhelper.executeCommand("ssh -V 2>&1")

 if response != None:
  match = re.search("OpenSSH_([^,]+)",response)

  if match != None:
   appserver.setProductVersion(match.group(1))
   appserver.setProductName("OpenSSH")
   appserver.setVendorName("openssh.org")
  else:
   log.info("This ssh server does not appear to be OpenSSH")
 else:
    log.info("'ssh -V' returned no output")
except:
 LogError("unexpected exception getting ssh information")
```

Explicación de los segmentos de la aplicación de ejemplo

En esta sección se describen los segmentos de la aplicación de ejemplo de extensiones de servidor personalizado.

Inicialización del entorno

Esta sección del código configura el entorno de manera que el intérprete de Jython pueda encontrar los módulos estándar de Python y el módulo Python de las herramientas del sensor de TADDM.

```
from java.lang import System
coll_home = System.getProperty("com.collation.home")

System.setProperty("jython.home",coll_home +
"/osgi/plugins/com.ibm.cdb.core.jython_1.0.0/lib")
System.setProperty("python.home",coll_home +
"/osgi/plugins/com.ibm.cdb.core.jython_1.0.0/lib")

jython_home = System.getProperty("jython.home")
sys.path.append(jython_home + "/Lib")
sys.path.append(coll_home + "/lib/sensor-tools")
sys.prefix = jython_home + "/Lib"
```

Importación de sensorhelper

Esta sección del código importa el módulo Python de las herramientas del sensor de TADDM. Este código permite a la aplicación llamar a las funciones de la extensión de servidor personalizado, por ejemplo, **sensorhelper.executeCommand** ("echo hello world").

```
import sensorhelper
```

Errores de registro

Esta sección del código registra los rastreos de la pila de excepciones utilizando el módulo de rastreo inverso de Python. Para el registro habitual, puede utilizar el objeto de registro devuelto por la llamada sensorhelper.init().

```
def LogError(msg):
        log.error(msg)
        (ErrorType, ErrorValue, ErrorTB) = sys.exc_info()
        traceback.print_exc(ErrorTB)
```

Inicialización de sensorhelper

Esta sección de código inicializa el módulo Python de las herramientas del sensor con información sobre el destino de descubrimiento que ha pasado el motor de descubrimiento de TADDM a la aplicación de extensiones de servidor personalizado.

```
(os_handle, result, appserver,seed,log,env) = sensorhelper.init(targets)
```

Ejecución del mandato

Esta sección de código llama a la función **executeCommand()** de herramientas del sensor para que ejecute "`ssh –V`" en el destino de descubrimiento. La salida de resultado del mandato se almacena en la variable de respuesta.

Normalmente, una extensión del servidor personalizado debe ejecutar uno o varios mandatos, o capturar uno o varios archivos (o una combinación de los dos) para descubrir el destino buscado.

```
response = sensorhelper.executeCommand("ssh -V 2>&1")
```

Búsqueda de la respuesta

Esta sección de código comprueba primero la variable de respuesta para garantizar que es válida (por ejemplo, que el valor no es None). El código utiliza luego el módulo de expresión regular de Python para analizar la salida desde el mandato `ssh –V` almacenado en la variable de respuesta.

```
if response != None:
   match = re.search("OpenSSH_([^,]+)",response)
```

Definición de los atributos

Esta sección de código comprueba primero si la variable de respuesta ha sido analizada correctamente por el módulo de expresión regular de Python (el valor de la variable match no es igual a None). Si la variable de respuesta se ha analizado correctamente, la versión de ssh está almacenada en el atributo productVersion del objeto AppServer del modelo de datos común (CDM). Además, se han definido los atributos productName y vendorName.

```
if match != None:
   appserver.setProductVersion(match.group(1))
   appserver.setProductName("OpenSSH")
   appserver.setVendorName("openssh.org")
```

Definición del objeto

Esta sección de código almacena el objeto AppServer del CDM en el objeto de resultado. Cuando el sensor se completa, todos los ModelObjects del CDM contenidos por el objeto de resultado del sensor se envían al motor de almacenamiento de TADDM para establecerse como permanentes en la base de datos.

```
result.setAppServer(appserver)
```

Mejores prácticas para desarrollar aplicaciones de extensiones de servidor personalizado

Puede optimizar la aplicación de extensiones de servidor personalizado siguiendo un conjunto de sencillas directrices.

Utilice las directrices siguientes cuando desarrolle aplicaciones de extensiones de servidor personalizado:

- Registre las operaciones realizadas por la aplicación.

  Puede utilizar el objeto de registro devuelto por la función `sensorhelper.init()` para realizar operaciones de registro.
- Utilice el módulo de rastreo inverso de Python para registrar el rastreo de pilas de excepciones.

  Si utiliza el archivo `sensorstub.py` del directorio `dist/lib/sensor-tools` como base de la aplicación de extensión de servidor personalizado, la función `LogError` definida en el archivo realizará esta tarea.
- Al incrementar el número de objetos de modelo del objeto de resultado, se incrementa el tiempo necesario para almacenar los objetos.

Vistas y esquema de base de datos de TADDM

La base de datos de TADDM contiene todos los elementos de configuración (CI) gestionados por un dominio de TADDM.

La base de datos se puede llenar con elementos de configuración de distintas maneras:

- Descubrimientos de TADDM
- Carga masiva de archivos de libro del adaptador de biblioteca de descubrimiento (DLA)
- Adición manual de elementos de configuración mediante la interfaz gráfica de usuario

- Adición mediante programación de elementos de configuración mediante la interfaz de programación de aplicaciones de TADDM

Los elementos de configuración de la base de datos de TADDM se organizan según su clasificación en el modelo de datos común (CDM) de IBM Tivoli. Los tipos de objeto, atributos, relaciones y reglas de denominación del CDM aparecen documentados en el archivo CDMWebsite.zip, ubicado en el directorio `$COLLATION_HOME/sdk/doc/model`. Para examinar esta documentación, extraiga el archivo .zip en un nuevo directorio y, a continuación, abra el archivo `misc/CDM.htm` en un navegador web.

Para obtener más información sobre el modelo de datos común de Tivoli, consulte el apartado "Presentación del modelo de datos común" en la página 1.

Al desarrollar informes personalizados para TADDM, puede utilizar consultas SQL para recuperar los datos almacenados en la base de datos de TADDM. Sin embargo, en lugar de consultar los datos directamente desde las tablas de base de datos de TADDM, los informes deben utilizar las vistas de base de datos de TADDM. TADDM proporciona muchas vistas de bases de datos predefinidas, con el fin de simplificar la tarea de escribir consultas SQL para extraer datos de la base de datos, además de permitirle diseñar también sus propias vistas personalizadas.

Hay cuatro categorías de vistas de base de datos de TADDM:

- Vistas de bloques de creación
- Vistas del panel Detalles
- Vistas personalizadas
- Vistas de atributos ampliados

Nota: Fix Pack 3 En TADDM 7.3.0.3, y posterior, el número máximo de caracteres que pueden tener los nombres de columnas en las vistas de base de datos es de 30.

Conceptos relacionados:

"Diccionario de datos de TADDM" en la página 182
El diccionario de datos de TADDM es una recopilación de páginas HTML generadas automáticamente que proporcionan una correlación entre la información del modelo de datos común (CDM) y la información de la base de datos de TADDM.

Vistas de bloques de creación

Puede utilizar las vistas de bloques de creación para escribir consultas basadas en el punto de vista del modelo de datos común (CDM) de Tivoli. Estas vistas resultan útiles si está familiarizado con el CDM; no es necesario saber dónde están almacenados los elementos de configuración en las tablas base de la base de datos de TADDM.

Para ver información detallada para las vistas del bloque de creación, vaya al directorio `$COLLATION_HOME/etc/views` y abra uno de los archivos siguientes:

- Para bases de datos DB2: `create_building_block_views_db2.sql`
- Para bases de datos Oracle: `create_building_block_views_oracle.sql`

En los comentarios de estos archivos se describen los tipos de objeto de modelo de datos común y las vistas de base de datos correspondientes, además de proporcionarse correlaciones entre el modelo de datos común y el esquema de base de datos:

- del tipo de objeto del CDM al nombre de vista de base de datos del bloque de creación
- del atributo al nombre de columna de vista de base de datos
- de la relación a la sintaxis JOIN

El nombre de las vistas del bloque de creación se encuentra en el formato siguiente:

```
BB_%_V
```

En cada nombre de vista, `%` es el nombre del tipo de objeto. Cada tipo de objeto se correlaciona en una vista del bloque de creación, que ofrece más de 1.000 vistas de base de datos del bloque de creación disponibles que representan tipos de objetos.

Por ejemplo, el CDM define el tipo de objeto `sys.windows.WindowsComputerSystem`, almacenado en la tabla base COMPSYS de la base de datos de TADDM. Esta tabla incluye también muchos otros tipos de objeto que amplían el tipo de objeto `sys.ComputerSystem` (por ejemplo, `sys.LinuxUnitaryComputerSystem`).

Para consultar los elementos de configuración de `sys.windows.WindowsComputerSystem` desde la base de datos utilizando una vista de bloque de creación, consulte la vista de base de datos BB_WINDOWSCOMPUTERSYSTEM20_V.

Además de las vistas que representan tipos de objetos, hay más de 800 vistas especiales que dan soporte a las relaciones "muchos a muchos" entre tipos de objetos. Cada una de estas vistas de bloques de creación de tablas de correlación tiene un nombre con el formato siguiente:

```
BB_%J
```

Para obtener más información sobre estas vistas especiales, consulte el apartado “Definiciones JOIN” en la página 168.

Vistas de definición

Para cada vista del bloque de creación, los comentarios incluyen una sección donde se describe la vista y los objetos del CDM correspondientes. Por ejemplo, la sección de la vista de definición de la vista del bloque de creación BB_WINDOWSCOMPUTERSYSTEM20_V es la siguiente:

```
-- ######## model.topology.sys.windows.WindowsComputerSystem ########
--
-- View...... BB_WINDOWSCOMPUTERSYSTEM20_V
-- Class..................... model.topology.sys.windows.WindowsComputerSystem
-- Super classes................. model.topology.sys.ComputerSystem
-- model.topology.core.ManagedElement
-- model.ModelObject
-- model.topology.process.itil.ConfigurationItem
```

En este ejemplo, se muestra que la vista BB_WINDOWSCOMPUTERSYSTEM20_V se corresponde con el tipo de objeto `model.topology.sys.WindowsComputerSystem` y muestra también varias superclases de las cuales este tipo hereda atributos y definiciones de relación.

Definiciones de columna para atributos

Para columna correspondiente a un atributo de CDM, los comentarios incluyen una sección donde se describe la columna y el atributo correspondiente. Por ejemplo, la sección de definición de columna para la columna CPUSPEED_C de la vista BB_WINDOWSCOMPUTERSYSTEM20_V es igual a esta:

```
-- Column.... CPUSPEED_C
-- Attribute..................... CPUSpeed
-- Java Type..................... long, primitive
-- Declared By................... model.topology.sys.ComputerSystem
```

Este ejemplo muestra que la columna CPUSPEED_C se corresponde con el atributo `CPUSpeed` definido por el tipo de objeto `model.topology.sys.ComputerSystem` Este atributo lo hereda el tipo `model.topology.sys.WindowsComputerSystem`. También muestra el tipo Java utilizado para representar el valor del atributo.

Nota: Los comentarios no muestran el tipo de base de datos utilizado para almacenar el atributo. Puede determinar el tipo de base de datos utilizando un mandato de SQL **`describe`**: El mandato Describe devuelve varias columnas incluyendo los nombres de vistas de base de datos, que puede utilizar para consultar datos que aparecen en el separador **Panel detalles**.

```
db2 describe table BB_WindowsComputerSystem20_V
```

Definiciones de columna para relaciones [0..1]

Por cada columna que representa una relación de "cero o uno" del CDM con otro elemento de configuración, los comentarios incluyen una sección que describe la columna utilizada para realizar la operación JOIN de SQL con el elemento de configuración del otro lado de la relación. Por ejemplo, la sección de definición de columna de la columna PK_OSRUNNING_C de la vista BB_WINDOWSCOMPUTERSYSTEM20_V es la siguiente:

```
-- Column.... PK__OSRUNNING_C
-- Attribute..................... OSRunning
-- Java Type..................... model.topology.sys.OperatingSystem, notContained
-- Declared By................... model.topology.sys.ComputerSystem
```

Este ejemplo muestra que la columna PK__OSRUNNING_C se utiliza para realizar la operación JOIN en la tabla OperatingSystem, que representa la relación `OSRunning` definida por el tipo de objeto `model.topology.sys.ComputerSystem`. También muestra que el tipo de Java del valor de atributo es, en este caso, otro tipo de CDM (`model.topology.sys.OperatingSystem`).

Nota: Todas las columnas que representan relaciones con otros elementos de configuración (y que no son, por lo tanto, tipos primitivos) tienen nombres que empiezan por `PK__`. Desde el punto de vista de una base de datos relacional, el valor de dicha columna es el GUID del elemento de configuración del otro lado de la relación.

Fix Pack 3

La columna de indicación de fecha y hora en TADDM 7.3.0.3, y posterior

En TADDM 7.3.0.3, o posterior, las vistas de bloque de creación contienen una columna adicional de indicación de fecha y hora. Por ejemplo, la sección de la vista de definición de la columna LASTSTOREDTIME_C de la vista

BB_WINDOWSCOMPUTERSYSTEM20_V contiene una serie adicional (`timestamp`):

```
-- Column.... LASTSTOREDTIME_C
-- Attribute..................... lastStoredTime
-- Java Type..................... long, primitive (timestamp)
-- Declared By................... model.ModelObject
```

Cada atributo de indicación de fecha y hora contiene las siguientes dos columnas:

- LASTSTOREDTIME_C, que se establece en el tipo `long`, por ejemplo `1445417251307`.
- LASTSTOREDTIME_T, que contiene una indicación de fecha y hora legible, por ejemplo `Oct 21, 2015 10:47:31 AM`.

Como en el caso del convenio de denominación original de la columna de indicación de fecha y hora, la columna añadida siempre tiene el mismo sufijo, que en este caso es `_T`.

Definiciones JOIN

Para las relaciones "cero a uno" y "muchos a muchos" de CDM, los comentarios proporcionan una consulta SQL de ejemplo que muestra cómo cumplir con la operación JOIN de SQL. Por ejemplo, la sección de la definición de JOIN para la relación `OSRunning` de la vista BB_WINDOWSCOMPUTERSYSTEM20_V es la siguiente:

```
-- Join......
-- Attribute..................... OSRunning
-- Java Type..................... model.topology.sys.OperatingSystem, notContained
-- Declared By................... model.topology.sys.ComputerSystem
-- Test Join..................... SELECT COUNT(1) FROM
--                                BB_WINDOWSCOMPUTERSYSTEM20_V T1,
--                                BB_OPERATINGSYSTEM62_V T2
--                                WHERE T1.PK__OSRUNNING_C = T2.PK_C
```

Algunas relaciones entre elementos de configuración de relaciones de muchos a muchos; por ejemplo, el elemento de configuración `sys.windows.WindowsComputerSystem` puede tener una relación con varios elementos de configuración `sys.FileSystem`, representados por una relación `contains` que utiliza el atributo `fileSystems` de `sys.windows.WindowsComputerSystem`:

```
-- Join......
-- Attribute..................... fileSystems
-- Java Type..................... Array of model.topology.sys.FileSystem, array
-- Declared By................... model.topology.sys.ComputerSystem
-- Test Join..................... SELECT COUNT(1) FROM
--                                BB_WINDOWSCOMPUTERSYSTEM20_V T1,
--                                BB_COMPUTERSYSTTEMS_88841D4BJ T2,
--                                BB_FILESYSTEM71_V T3
--                                WHERE T1.PK_C = T2.PK__JDOID_C
--                                AND T3.PK_C = T2.PK__FILESYSTEMS_C
```

Las relaciones de muchos a muchos se almacenan en la tabla de correlación del intermediario (en este ejemplo, la tabla de correlación BB_COMPUTERSYSTTEMS_88841D4BJ).

Vistas en desuso

Algunas vistas de bloques de creación han quedado en desuso después de los cambios producidos en Tivoli Common Data Model (CDM) y se suprimirán en el futuro. Consulte la tabla con los nuevos equivalentes de vistas en desuso para realizar los cambios necesarios.

Para hacer que los informes que ya están definidos funcionen después de la actualización, las vistas de compatibilidad se han definido y marcado como en desuso.

Si se utiliza alguna de las vistas que se indican en la tabla siguiente, corrija la definición del informe para que utilice la nueva definición de vista equivalentes.

La tabla siguiente indica vistas en desuso que admiten relaciones "varios a varios" entre tipos de objetos y sus nuevos equivalentes.

Tabla 41. Vistas en desuso y sus equivalentes nuevos.

Vista en desuso	Nuevo equivalente
BB_APPCONFIGJDONCES_FCB57E09J	BB_SPHYSICALFILNCES_ECF04BA6J
BB_APPCONFIGJDO_ROLES_J	BB_SPHYSICALFILEJDO_ROLES_J
BB_APPSERVERCLUNCES_E03E73D6J	BB_SGROUPJDO_SENCES_CF68C060J
BB_APPSERVERCLUOLES_F1CF6FA2J	BB_SGROUPJDO_ROLES_J
BB_APPSERVERJDONCES_EF1C0CA8J	BB_SSOFTWARESERNCES_19E863EFJ
BB_APPSERVERJDO_ROLES_J	BB_SSOFTWARESERVERJDO_ROLES_J
BB_COLLECTIONJDNCES_2AB18ECEJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_COLLECTIONJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_COMPOSITEJDONCES_C1981AC5J	BB_SGROUPJDO_SENCES_CF68C060J
BB_COMPOSITEJDO_MEMBERS_J	BB_COLLECTIONJDO_MEMBERS_J
BB_COMPOSITEJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_COMPUTERSYSTEMJDO_ROLES_J	BB_SCOMPUTERSYSTEMJDO_ROLES_J
BB_COMPUTERSYSTNCES_611B3FC2J	BB_SCOMPUTERSYSNCES_119A868FJ
BB_COMPUTERSYSTNCES_6A2E6F7CJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_COMPUTERSYSTOLES_6B483FBCJ	BB_SGROUPJDO_ROLES_J
BB_CONFIGURATIONCES_F7B28A51J	BB_SFUNCTIONJDONCES_F8394E61J
BB_CONFIGURATIOOLES_33A89807J	BB_SFUNCTIONJDO_ROLES_J
BB_DB2DATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_DB2DATABASEJNCES_83C5253DJ	BB_SDEPLOYABLECNCES_572F3E83J
BB_DB2SYSTEMJDONCES_6CAED009J	BB_SSOFTWAREINSNCES_549D08D8J
BB_DB2SYSTEMJDO_ROLES_J	BB_SSOFTWAREINSOLES_7C7BE7E0J
BB_DOMINOCONNECNCES_FAE09606J	BB_SPHYSICALFILNCES_ECF04BA6J
BB_DOMINOCONNECOLES_F12CCB72J	BB_SPHYSICALFILEJDO_ROLES_J
BB_DOMINODATABANCES_BE00AD89J	BB_SDEPLOYABLECNCES_572F3E83J
BB_DOMINODATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_DOMINODOMAINJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_DOMINODOMAINNCES_AC5777E0J	BB_SGROUPJDO_SENCES_CF68C060J
BB_EXCHANGEADMINCES_6B41ACDEJ	BB_SGROUPJDO_SENCES_CF68C060J

Tabla 41. Vistas en desuso y sus equivalentes nuevos. (continuación)

Vista en desuso	Nuevo equivalente
BB_EXCHANGEADMIOLES_5F958B9AJ	BB_SGROUPJDO_ROLES_J
BB_EXCHANGEFOLDNCES_EC2EEA5DJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_EXCHANGEFOLDOLES_B3A7C77BJ	BB_SGROUPJDO_ROLES_J
BB_EXCHANGEMAILNCES_2B5725CJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_EXCHANGEMAILOLES_250648DCJ	BB_SGROUPJDO_ROLES_J
BB_FABRICJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_FABRICJDO_SENCES_783F8B27J	BB_SGROUPJDO_SENCES_CF68C060J
BB_FUNCTIONJDO_NCES_C03DA9D4J	BB_SFUNCTIONJDONCES_F8394E61J
BB_FUNCTIONJDO_ROLES_J	BB_SFUNCTIONJDO_ROLES_J
BB_HACMPAPPRESONCES_B47F5A8AJ	BB_SFUNCTIONJDONCES_F8394E61J
BB_HACMPAPPRESOOLES_229BB16EJ	BB_SFUNCTIONJDO_ROLES_J
BB_HACMPLOCALREENTS_F67E36ECJ	BB_ITSYSTEMJDO_COMPONENTS_J
BB_HACMPLOCALRENCES_8ADCB6D9J	BB_SGROUPJDO_SENCES_CF68C060J
BB_HACMPLOCALREOLES_3C1CF07FJ	BB_SGROUPJDO_ROLES_J
BB_HACMPLOCALREOUPS_354DD1CCJ	BB_SERVICEGROUPOUPS_82AA2EE3J
BB_HACMPLOCALREROUP_21D3AC87J	BB_SERVICEGROUPOUPS_76D12CEDJ
BB_HACMPNODEJDONCES_F0DF78FDJ	BB_SFUNCTIONJDONCES_F8394E61J
BB_HACMPNODEJDO_ROLES_J	BB_SFUNCTIONJDO_ROLES_J
BB_HIRDBSYSTEMJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_HIRDBSYSTEMJNCES_8CDFFAEEJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_IDSDATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_IDSDATABASEJNCES_CFB10C79J	BB_SDEPLOYABLECNCES_572F3E83J
BB_IPNETWORKJDONCES_E3F0C245J	BB_SGROUPJDO_SENCES_CF68C060J
BB_IPNETWORKJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_ITSYSTEMJDO_NCES_31C6E3D2J	BB_SGROUPJDO_SENCES_CF68C060J
BB_ITSYSTEMJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_J2EEDOMAINJDNCES_111AC7A0J	BB_SGROUPJDO_SENCES_CF68C060J
BB_J2EEDOMAINJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_LINKSERVICEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_LINKSERVICEJNCES_F441B071J	BB_SDEPLOYABLECNCES_572F3E83J
BB_LOGICALCONTENCES_F200987CJ	BB_SPHYSICALFILNCES_ECF04BA6J
BB_LOGICALCONTENTJDO_ROLES_J	BB_SPHYSICALFILEJDO_ROLES_J
BB_MBEXECUTIONGNCES_A5DA5D10J	BB_SGROUPJDO_SENCES_CF68C060J
BB_MBEXECUTIONGOLES_6D4906A8J	BB_SGROUPJDO_ROLES_J
BB_MBMESSAGEFLONCES_2C1D04EAJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_MBMESSAGEFLONCES_EB919DCCJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_MBMESSAGEFLOOLES_18E0C30EJ	BB_SGROUPJDO_ROLES_J
BB_MBMESSAGEFLOWJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_MQINSTALLABLNCES_EDA0F908J	BB_SGROUPJDO_SENCES_CF68C060J
BB_MQINSTALLABLOLES_688C35B0J	BB_SGROUPJDO_ROLES_J

Tabla 41. Vistas en desuso y sus equivalentes nuevos. (continuación)

Vista en desuso	Nuevo equivalente
BB_MQQUEUEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_MQQUEUEJDO_SNCES_CAE5631FJ	BB_SDEPLOYABLECNCES_572F3E83J
BB_MSCLUSTERJDONCES_61127DF8J	BB_SGROUPJDO_SENCES_CF68C060J
BB_MSCLUSTERJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_MSCLUSTERNODEJDO_ROLES_J	BB_SFUNCTIONJDO_ROLES_J
BB_MSCLUSTERNODNCES_3899EF16J	BB_SFUNCTIONJDONCES_F8394E61J
BB_MSCLUSTERRESNCES_22175CCFJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_MSCLUSTERRESNCES_A728F28AJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_MSCLUSTERRESOLES_324E6849J	BB_SGROUPJDO_ROLES_J
BB_MSCLUSTERRESOLES_3A01196EJ	BB_SGROUPJDO_ROLES_J
BB_NETWORKSERVICEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_NETWORKSERVINCES_CDD7C865J	BB_SDEPLOYABLECNCES_572F3E83J
BB_ORACLEDATABANCES_98E1A333J	BB_SDEPLOYABLECNCES_572F3E83J
BB_ORACLEDATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_ORACLELISTENERJDO_ROLES_J	BB_SSOFTWARESERVERJDO_ROLES_J
BB_ORACLELISTENNCES_72725D9AJ	BB_SSOFTWARESERNCES_19E863EFJ
BB_ORACLESERVERJDO_ROLES_J	BB_SSOFTWAREINSOLES_7C7BE7E0J
BB_ORACLESERVERNCES_6F134AEBJ	BB_SSOFTWAREINSNCES_549D08D8J
BB_REALSERVERGRNCES_91B57D2EJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_REALSERVERGROUPJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_REALSERVERJDNCES_8B01D2CBJ	BB_SSOFTWARESERNCES_19E863EFJ
BB_REALSERVERJDO_ROLES_J	BB_SSOFTWARESERVERJDO_ROLES_J
BB_SAMETIMESERVERJDO_ROLES_J	BB_SFUNCTIONJDO_ROLES_J
BB_SAMETIMESERVNCES_7FDA5716J	BB_SFUNCTIONJDONCES_F8394E61J
BB_SEGMENTJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_SEGMENTJDO_SNCES_CBC65999J	BB_SGROUPJDO_SENCES_CF68C060J
BB_SERVICEGROUPROUP_6F30099EJ	BB_SERVICEGROUPOUPS_76D12CEDJ
BB_SERVICEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SERVICEJDO_SNCES_63DE5757J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SMSHIERARCHYJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_SMSHIERARCHYNCES_C6234B50J	BB_SGROUPJDO_SENCES_CF68C060J
BB_SMSSITECOMPONCES_FC696136J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SMSSITECOMPOOLES_37331E42J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SOFTWARECOMPNCES_5E176596J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SOFTWARECOMPOLES_AA1C15E2J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SOFTWAREINSTNCES_EECCCFCBJ	BB_SSOFTWAREINSNCES_549D08D8J
BB_SOFTWAREINSTOLES_B502FACDJ	BB_SSOFTWAREINSOLES_7C7BE7E0J
BB_SOFTWAREMODULEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SOFTWAREMODUNCES_1FEAE999J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SPECIALITYSENCES_A28738B4J	BB_SFUNCTIONJDONCES_F8394E61J

Tabla 41. Vistas en desuso y sus equivalentes nuevos. (continuación)

Vista en desuso	Nuevo equivalente
BB_SPECIALITYSEOLES_33910984J	BB_SFUNCTIONJDO_ROLES_J
BB_SQLSERVERDATNCES_4C800F20J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SQLSERVERDATOLES_E33DFE98J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_STORAGEEXTENNCES_614C0287J	BB_SDEPLOYABLECNCES_572F3E83J
BB_STORAGEEXTENTJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_STORAGEPOOLJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_STORAGEPOOLJNCES_AC507A15J	BB_SGROUPJDO_SENCES_CF68C060J
BB_SYBASEDATABANCES_DFF7B51AJ	BB_SDEPLOYABLECNCES_572F3E83J
BB_SYBASEDATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SYSPLEXJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_SYSPLEXJDO_SNCES_4B3FE470J	BB_SGROUPJDO_SENCES_CF68C060J
BB_VCSSYSTEMJDONCES_831828D7J	BB_SGROUPJDO_SENCES_CF68C060J
BB_VCSSYSTEMJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_WEBLOGICMACHINEJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_WEBLOGICMACHNCES_41F6228FJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_WEBLOGICNODENCES_1032390BJ	BB_SSOFTWARESERNCES_19E863EFJ
BB_WEBLOGICNODEOLES_14E2D98DJ	BB_SSOFTWARESERVERJDO_ROLES_J
BB_WEBSPHERENODEJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_WEBSPHERENODNCES_182E84E9J	BB_SGROUPJDO_SENCES_CF68C060J
BB_WINDOWSSERVICEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_WINDOWSSERVINCES_9F30EF3AJ	BB_SDEPLOYABLECNCES_572F3E83J
BB_ZONEJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_ZONEJDO_SERVICEINSTANCES_J	BB_SGROUPJDO_SENCES_CF68C060J

Vistas del panel Detalles

Puede utilizar el archivo `detail_panel_views.txt` para escribir consultas y recuperar información adicional. Estas vistas resultan más útiles si está familiarizado con el panel Detalles de Data Management Portal.

Para ver un listado de las vistas disponibles del panel Detalles, vaya al directorio `$COLLATION_HOME/etc/views` y abra el archivo `detail_panel_views.txt`. Este archivo muestra todos los paneles Detalles disponibles en Data Management Portal, junto con las vistas de base de datos correspondientes. El nombre de cada una de las vistas del panel Detalles tiene el formato siguiente:

```
DP_%_V
```

En todos los nombres de la vista, `%` es el nombre del panel Detalles de Data Management Portal. Hay más de 600 vistas del panel Detalles.

Encontrará información sobre estas vistas, en forma de comentarios, en los archivos siguientes del mismo directorio:

- Para bases de datos DB2: create_detail_panel_views_db2.sql
- Para bases de datos Oracle: create_detail_panel_views_oracle.sql

Vistas de definición

Las vistas del panel Detalles se organizan de acuerdo con los tipos de objeto del modelo de datos común de los elementos de configuración mostrados en el panel Detalles. Para encontrar las vistas de un elemento de configuración concreto, en el panel Detalles, pulse el separador **General** y busque el valor del campo **Tipo de objeto**. A continuación, puede buscar el tipo de objeto en el archivo detail_panel_views.txt, que identifica las vistas correspondientes.

Por ejemplo, el archivo detail_panel_views.txt file incluye la entrada siguiente para el tipo de objeto `DB2 Instance`:

```
########## DB2 Instance...<Layout> ##########
...General.....<Tab Level 1>
......General.....<TabData>
......Db2Instance.General.....<Content>
.........DP_DB2_INSTANCE_GENERAL_V.....<View>
...System.....<Tab Level 1>
......System Info.....<TabData>
......Db2Instance.SystemInfo.....<Content>
.........DP_DB2_INSTANCE_SYSTEM_INFO_V.....<View>
......Profile Registry.....<TabData>
......Db2Instance.GlobalProfileRegistry.....<Content>
.........DP_DB2_INSTANCE_GLOB_PROFREG_V.....<View>
......License Info.....<TabData>
......Db2Instance.LicenseInfo.....<Content>
.........DP_DB2_INSTANCE_LICENSE_INFO_V.....<View>
...Configuration.....<Tab Level 1>
......Configuration.....<TabData>
......Db2Instance.Configuration.....<Content>
.........DP_DB2_INSTANCE_CONFIG_V.....<View>
...Profile Registry.....<Tab Level 1>
......Profile Registry.....<TabData>
......Db2Instance.ProfileRegistry.....<Content>
.........DP_DB2_INSTANCE_PROFILE_REG_V.....<View>
...Databases.....<Tab Level 1>
......Databases.....<TabData>
......Db2Instance.Databases.....<Content>
.........DP_DB2_INSTANCE_DATABASES_V.....<View>
...Remote Databases.....<Tab Level 1>
......Remote Databases.....<TabData>
......Db2Instance.Db2Alias.....<Content>
.........DP_DB2_INSTANCE_DB2_ALIAS_V.....<View>
...Modules.....<Tab Level 1>
......Software Modules.....<TabData>
......AppServer.SoftwareModules.....<Content>
.........DP_APP_SERVER_SW_MODULES_V.....<View>
......Other Modules.....<TabData>
......AppServer.StaticModules.....<Content>
.........DP_APP_SERVER_STATIC_MOD_V.....<View>
...Application Descriptors.....<Tab Level 1>
......Application Descriptors.....<TabData>
......AppServer.ApplicationDescriptors.....<Content>
.........DP_APP_SERVER_APP_DESCRIPS_V.....<View>
...Runtime.....<Tab Level 1>
......Databases.....<TabData>
......Db2Instance.Runtime.....<Content>
.........DP_DB2_INSTANCE_RUNTIME_V.....<View>
```

En este ejemplo, se muestra que el separador General del tipo de objeto `DB2 instance` se corresponde con la vista de base de datos DP_DB2_INSTANCE_GENERAL_V. También muestra las vistas de base de datos adicionales para los separadores restantes (algunos de los cuales se representan con varias vistas).

Puede consultar estas vistas para recuperar información sobre los elementos de configuración que ha descubierto TADDM. Por ejemplo, puede recuperar datos del separador Bases de datos de `DB2 Instance` consultando la vista de base de datos DP_DB2_INSTANCE_DATABASES_V. Si selecciona todos los elementos de esta vista, obtendrá el contenido del separador Base de datos de todas las instancias de DB2 que haya descubierto TADDM. Puede limitar esta consulta a una única instancia de DB2 uniendo la vista DP_DB2_INSTANCE_DATABASES_V a la vista DP_DB2_INSTANCE_GENERAL_V y filtrando por nombre de instancia (consulte el apartado "Consulta de ejemplo" en la página 175).

Para ver más información sobre cómo se definen estas vistas, mire los comentarios del archivo create_detail_panel_views_db2.sql o create_detail_panel_views_oracle.sql. El siguiente bloque de comentarios proporciona información sobre los nodos, o los tipos de objeto del CDM, a los que hace referencia la vista DP_DB2_INSTANCE_DATABASES_V:

```
-- ######## Db2Instance.Databases ########
--
-- View...... DP_DB2_INSTANCE_DATABASES_V
--
-- Node...... 1
-- Node Path..................... Db2Instance
-- Node Class Name............... model.topology.app.db.db2.Db2Instance
-- Node Type..................... Root
-- Node...... 2
-- Node Path..................... Db2Instance._arraydatabases
-- Node Class Name............... model.topology.app.db.db2.Db2Database
-- Node Field Name............... databases
-- Node Type..................... Array Many-to-Many
-- Node...... 3
-- Node Path..................... Db2Instance._arraydatabases.databases
-- Node Class Name............... model.topology.app.db.db2.Db2Database
-- Node Field Name............... databases
-- Node Type..................... Array
```

Nodos y columnas

Para ver cómo se definen las columnas de vista, puede ejecutar una consulta SQL `describe`. Por ejemplo, la consulta siguiente muestra las columnas de la vista DP_DB2_INSTANCE_DATABASES_V:

```
db2 "describe table DP_DB2_INSTANCE_DATABASES_V"

Column                         Type      Type
name                           schema    name               Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
NAME_C3                        SYSIBM    VARCHAR                 192     0 Yes
ALIAS_C3                       SYSIBM    VARCHAR                 192     0 Yes
PK_C3                          SYSIBM    VARCHAR                 192     0 Yes
PK_C1                          SYSIBM    VARCHAR                 192     0 No
```

Todos los nombres de columna acaban con un número que identifica el nodo al que representa la columna. En este ejemplo, se muestra que las columnas NAME_C3, ALIAS_C3 y PK_C3 se refieren al tipo de objeto `Db2Database` y la columna PK_C1 se refiere al tipo de objeto `Db2Instance`.

Nota: En este ejemplo, ninguna columna se refiere al nodo 2, porque la tabla a la que se hace referencia es la tabla de correlación de muchos a muchos que conecta

DB2Instances con DB2Databases. El panel Detalles resume las complejidades de unir estas dos tablas, no es necesario determinar si los datos se almacenan en las tablas base.

Consulta de ejemplo

Con toda esta información, puede ver una lista de bases de datos DB2 definidas para una instancia concreta utilizando esta consulta:

```
select
   db.name_c3
from
   DP_DB2_INSTANCE_GENERAL_V inst
   join DP_DB2_INSTANCE_DATABASES_V db ON (inst.PK_C1 = db.PK_C1)
where
   inst.db2_instance_c1 = 'bg-linux.tivlab.austin.ibm.com:db2inst1'
```

Esta consulta une las vistas correspondientes a los separadores General y Bases de datos, utilizando la columna PK_C1 (que representa la clave primaria de la instancia DB2) y seleccionando los nombres de bases de datos. Los resultados se filtran luego para incluir solo la instancia con la etiqueta `bg-linux.tivlab.austin.ibm.com:db2inst1`.

Vistas personalizadas

Se proporcionan vistas personalizadas para extraer datos de la base de datos del TADDM y para colaborar con la creación de informes. TADDM proporciona dos vistas personalizadas definidas en el archivo `custom-views.xml`. También puede definir vistas, denominadas vistas definidas por el usuario, si las vistas existentes del bloque de creación y las vistas del panel de detalles no le proporcionan todo lo necesario.

Una vista personalizada se define utilizando XML, procesado luego por los scripts de TADDM para producir las sentencias SQL `CREATE VIEW` necesarias. Se crea una vista personalizada a partir de las vistas existentes de bloques de creación, pero los scripts de TADDM determinan cómo unir estas vistas.

TADDM incluye una vista personalizada denominada CM_COMPUTER_SYSTEMS_V, descrita en los ejemplos siguientes. Esta vista proporciona información básica que se puede utilizar en un informe sobre sistemas informáticos descubiertos por TADDM:

- Nombre de host completo
- Fabricante, modelo, número de serie y tipo de chasis
- Tipo, número y velocidad de las CPU
- Tamaño de la RAM
- Sistema operativo
- Direcciones IP de todos los adaptadores
- Capacidad de almacenamiento total y espacio libre para todos los sistemas de archivos

Estos atributos proceden de muchos tipos distintos de objetos del modelo de datos común:

- ComputerSystem
- OperatingSystem
- IPInterface
- IPAddress

- FileSystem

Además, los datos incluyen dos relaciones de muchos a muchos:

- ComputerSystem a IPInterface
- ComputerSystem a FileSystem

Para escribir a mano una consulta sobre esta información, antes debe identificar todas las vistas del bloque de creación que representan las relaciones y los tipos de CDM. A continuación, debe determinar cómo unirlos. Al definir una vista de usuario, puede utilizar los scripts de TADDM para definir automáticamente las uniones necesarias y generar el SQL para crear las vistas necesarias.

XML de vistas definidas por el usuario

Las vistas definidas por el usuario, al igual que las vistas personalizadas, se crean a partir de una definición XML. Para crear una vista definida por el usuario, siga estos pasos:

1. Copie el archivo `custom-views.xml` del directorio `$COLLATION_HOME/etc/views` en el directorio `$COLLATION_HOME/bin`.
2. Cambie el nombre del archivo `custom-views.xml` por `user-views.xml`.
3. Cambie el nombre de vista CM_COMPUTER_SYSTEMS_V y CM_APP_SERVERS_PER_HOST_V del archivo `user-views.xml`. TADDM reserva estas vistas y no se deben sobrescribir. Modifique el resto del archivo según sea necesario.

Elemento	Atributos	Elementos contenidos
`view`	**`className`** Nombre de clase de objeto del modelo base de la vista. **`viewName`** Nombre de la vista. El nombre debe ser una serie que empiece por `CM_` y termine en `_V`, con una longitud máxima de 30 caracteres. Evite los nombres que ya se están utilizando. **`includePrimaryKeys`** Si las claves primarias se deben incluir como columnas. Debe ser `true` o `false`. Especifique `true` si la vista se va a unir a otras vistas.	`field`
`field`	Ninguno	`nested` `plain`
`nested`	**`className`** Nombre de clase del objeto de modelo del anidado. **`fieldName`** Nombre de campo del anidado	`nested` `plain`

Elemento	Atributos	Elementos contenidos
`plain`	**`fieldName`** Nombre de campo del plano **`nameInView`** Nombre de la columna que se expondrá en la vista de base de datos. La longitud máxima es de 30 caracteres. Evite las palabras reservadas de DB2 u Oracle. **`displayType`** Tipo de valor. Este atributo es opcional. Especifique uno de los valores siguientes: **`speed`** Valor en MHz **`memory`** Valor en KB, MB o GB **`mBytes`** Valor en MB **`date`** Indicación de fecha y hora en formato `AAAA-MM-DD-HH24:MI:SS`, utilizada para los campos que contienen tiempo en milisegundos **`networkSpeed`** Valor en megabits por segundo **`StorageGBytes`** Valor en GB	Ninguno

El XML describe dos tipos de campos, ambos contenidos en el elemento `field`:

- El campo *plano* representa un atributo de un objeto de modelo. Por ejemplo, el siguiente campo plano especifica que el atributo de nombre de dominio completo de `ComputerSystem` aparece en la vista como columna `FQDN`:

  ```
  <plain fieldName="fqdn" nameInView="FDQN"/>
  ```

- Un campo *anidado* representa una relación entre objetos de modelo. Por ejemplo, el siguiente campo anidado describe la relación entre `ComputerSystem` y `OperatingSystem` a través del atributo `OSRunning` del tipo de objeto `ComputerSystem`:

  ```
  <field>
    <nested className="com.collation.platform.model.topology.sys.OperatingSystem"
    fieldName="OSRunning">
      <plain fieldName="OSName" nameInView="OS_NAME"/>
    </nested>
  </field>
  ```

  En el campo anidado, un campo plano especifica que el nombre del sistema operativo aparece en la vista como columna `OS_NAME`.

La definición XML completa de la vista CM_COMPUTER_SYSTEMS_V es la siguiente:

```
<views>
  <!-- ComputerSystems with OS, Filesystems -->
  <view className="com.collation.platform.model.topology.sys.ComputerSystem"
  viewName="CM_COMPUTER_SYSTEMS_V" includePrimaryKeys="false">
    <field>
      <plain fieldName="fqdn" nameInView="FDQN"/>
    </field>
    <field>
      <nested className="com.collation.platform.model.topology.net.IpInterface"
      fieldName="ipInterfaces">
        <nested className="com.collation.platform.model.topology.net.IpAddress"
        fieldName="ipAddress">
          <plain fieldName="dotNotation" nameInView="IP_ADDRESS"/>
          <plain fieldName="stringNotation" nameInView="IP_ADDRESS_STRING"/>
        </nested>
      </nested>
    </field>
    <field>
      <nested className="com.collation.platform.model.topology.sys.OperatingSystem"
      fieldName="OSRunning">
        <plain fieldName="OSName" nameInView="OS_NAME"/>
      </nested>
    </field>
    <field>
      <plain fieldName="CPUType" nameInView="CPU_TYPE"/>
      <plain fieldName="numCPUs" nameInView="NUM_CPUS"/>
      <plain displayType="speed" fieldName="CPUSpeed" nameInView="CPU_SPEED"/>
      <plain displayType="memory" fieldName="memorySize" nameInView="MEMORY_SIZE"/>
      <plain fieldName="primaryMACAddress" nameInView="PRIMARY_MAC_ADDRESS"/>
      <plain fieldName="serialNumber" nameInView="SERIAL_NUMBER"/>
      <plain fieldName="manufacturer" nameInView="MANUFACTURER"/>
      <plain fieldName="model" nameInView="MODEL"/>
      <plain fieldName="type" nameInView="SYSTEM_TYPE"/>
    </field>
    <field>
      <nested className="com.collation.platform.model.topology.sys.FileSystem"
      fieldName="fileSystems">
        <plain displayType="mBytes" fieldName="capacity" nameInView="CAPACITY"/>
        <plain displayType="mBytes" fieldName="availableSpace"
        nameInView="AVAILABLE_SPACE"/>
        <plain fieldName="displayName" nameInView="FILE_SYSTEM"/>
      </nested>
    </field>
  </view>

  <!-- AppServers with host -->
  <view className="com.collation.platform.model.topology.app.AppServer"
  viewName="CM_APP_SERVERS_PER_HOST_V" includePrimaryKeys="false">
    <field>
      <nested className="com.collation.platform.model.topology.sys.ComputerSystem"
      fieldName="host">
        <plain fieldName="fqdn" nameInView="FQDN"/>
        <nested className="com.collation.platform.model.topology.net.IpInterface"
        fieldName="ipInterfaces">
          <nested className="com.collation.platform.model.topology.net.IpAddress"
          fieldName="ipAddress">
            <plain fieldName="dotNotation" nameInView="IP_ADDRESS"/>
            <plain fieldName="stringNotation" nameInView="IP_ADDRESS_STRING"/>
          </nested>
        </nested>
      </nested>
    </field>

    <field>
      <plain fieldName="jdoClassName" nameInView="CLASS_NAME"/>
      <plain fieldName="objectType" nameInView="TYPE"/>
      <plain fieldName="productName" nameInView="PRODUCT_NAME"/>
```

```
      <plain fieldName="productVersion" nameInView="PRODUCT_VERSION"/>
      <plain fieldName="keyName" nameInView="KEY_NAME"/>
      <plain fieldName="name" nameInView="NAME"/>
    </field>
  </view>
</views>
```

Adición de vistas de usuario a la base de datos

Una vez definidas las vistas de usuario en el archivo `user-views.xml`, siga estos pasos para crear las vistas en la base de datos:

1. En un indicador de mandatos, vaya al directorio `$COLLATION_HOME/bin`.
2. Ejecute uno de los mandatos siguientes para crear los scripts SQL necesarios:
   - Sistemas UNIX y Linux: `user_views.sh scripts`
   - Sistemas Windows: `user_views scripts`

   Este mandato crea los archivos siguientes:
   - `create_custom_views_db2.sql`
   - `create_custom_views_oracle.sql`
   - `create_custom_views_db2zos.sql`
   - `drop_custom_views_db2.sql`
   - `drop_custom_views_oracle.sql`
   - `drop_custom_views_db2zos.sql`
3. Ejecute uno de los mandatos siguientes para crear las vistas en su base de datos:
   - Sistemas UNIX y Linux: `user_views.sh recreate`
   - Sistemas Windows: `user_views recreate`

   Este mandato ejecuta el script SQL correspondiente a su tipo de base de datos.

Después de ejecutar estos mandatos, las vistas de usuario están disponibles para consultas en la base de datos. Las consultas SQL que implemente deben proporcionar el necesario filtrado de datos devuelto desde estas vistas (por ejemplo, utilizando la cláusula SQL `WHERE`).

Modificación de las vistas de usuario

Para modificar las vistas de usuario que existen en la base de datos:

1. Edite el archivo `user-views.xml` para realizar los cambios necesarios.
2. Repita el proceso de creación de las vistas utilizando el mandato **`user_views`**. Este mandato genera y ejecuta automáticamente los scripts SQL correctos para soltar y volver a crear las vistas modificadas.

   Nota: Si cambia el nombre de un usuario, debe soltar a mano la vista con el nombre original para poder ejecutar los mandatos y crear la vista con el nuevo nombre.

Supresión de vistas de usuario

Para suprimir una vista de usuario de la base de datos, ejecute el mandato DROP de SQL correspondiente a la base de datos. Los mandatos DROP para vistas de usuario los genera el mandato **`user_views`** en los archivos siguientes:

- `drop_custom_views_db2.sql`
- `drop_custom_views_oracle.sql`
- `drop_custom_views_db2zos.sql`

Vistas de atributos ampliados

La herramienta de vista de atributos ampliados genera vistas de bases de datos que hacen referencia a los datos de atributos ampliados.

Para cada objeto de modelo que tiene atributos ampliados, la herramienta crea un script SQL que, al ejecutarse, crea dos vistas de atributos ampliados:

- `EA_model_V`, que tiene columnas correspondientes a los atributos ampliados. Cada atributo ampliado puede unirse a la vista de bloques de creación correspondiente, `BB_modelo_V`, mediante la columna `PK_C`.
- Fix Pack 1 `BE_modelo_V` que tiene columnas correspondientes a los atributos del modelo de datos comunes y a los atributos ampliados. Cuando se produce un conflicto entre los nombres de columna, es posible que sea porque las columnas de atributos ampliados no estén presentes.

Todos los atributos ampliados que hay en el mismo tipo de objeto de modelo están en la misma vista de base de datos.

Los scripts dependen de las definiciones actuales de los atributos ampliados. La herramienta no comprueba los atributos que se hayan creado y eliminado, aunque dichos atributos con valores estén aún asignados a algunos objetos. Si modifica un atributo ampliado, debe descartarse la vista existente antes de utilizar la herramienta de vista de atributos ampliados para crear un script SQL actualizado y una vista de atributos actualizada.

Fix Pack 3

Modificación de tipo de datos

En TADDM 7.3.0.2 y anterior, el script `extattr_views.sh` crea vistas de atributos ampliadas con columnas de tipo CLOB. Desde 7.3.0.3, las vistas contienen datos de tipos específicos, como por ejemplo VARCHAR o SMALLINT. La tabla siguiente proporciona los nuevos tipos de columnas en las vistas de atributos ampliadas en las bases de datos DB2 y Oracle.

Tabla 42. Tipos de columnas de vistas de atributos ampliados en bases de datos DB2 y Oracle

Tipo de GUI	Tipo de columna en DB2	Tipo de columna en Oracle
Serie	VARCHAR(32000)	VARCHAR2(4000)
Carácter	VARCHAR(4)	VARCHAR2(4)
Coma flotante de precisión doble	DOUBLE	NUMBER
Coma flotante	REAL	NUMBER
Booleano	SMALLINT	NUMBER(1)
Entero	INTEGER	NUMBER
Entero corto	SMALLINT	NUMBER
Entero grande	BIGINT	NUMBER

Tal modificación puede afectar a la integración con productos que utilizan las vistas de base de datos de TADDM para atributos ampliados. Por ejemplo, si utiliza informes BIRT, se podrían generar errores si las columnas de atributos ampliados no se difunden a un tipo de datos específico, por ejemplo, VARCHAR.

Nota: En TADDM 7.3.0.2 y anteriores, se mostrarán los atributos ampliados del tipo booleano como `false` o `true` en las vistas de base de datos. En TADDM 7.3.0.3, y posteriores, el valor es un entero, ya sea `0` o `1`. Significa que los informes BIRT y Cognos que se crearon antes de este cambio no son compatibles con los nuevos tipos de datos.

Sintaxis del mandato extattr_views.sh

Para utilizar la herramienta extattr_view, ejecute el mandato `extattr_views.sh` con un parámetro de línea de mandatos adecuado. El mandato `extattr_views.sh` se encuentra en el directorio `$COLLATION_HOME/bin`.

Sintaxis del mandato

extattr_views.sh *parámetro*

Parámetros

`scripts`
Crea los siguientes scripts SQL:

- `create_extattr_views_`*`tipo_bd`*`.sql`
- `drop_extattr_views_`*`tipo_bd`*`.sql`

donde *tipo_bd* es uno de los siguientes tipos de base de datos:

- db2
- oracle
- db2zos

`create`
Crea las vistas.

`remove`
Descarta las vistas. Los scripts SQL correspondientes no se eliminan.

Ejecución de la herramienta de vista de atributos ampliados

Puede utilizar la herramienta de vista de atributos ampliados para generar una vista de base de datos que se corresponda con un atributo ampliado existente.

Procedimiento

Para crear una vista correspondiente a un atributo ampliado, complete los pasos siguientes:

1. Cree un atributo ampliado en el objeto de modelo.
2. Utilice la herramienta de vista de atributos ampliados para crear los scripts SQL necesarios:
   ```
   extattr_views.sh scripts
   ```
3. Utilice la herramienta de vista de atributos ampliados para crear la vista:
   ```
   extattr_views.sh create
   ```
4. Opcional: Consulte los datos mediante un mandato SQL.

Ejemplo

Por ejemplo, si crea un atributo ampliado llamado `'SUPPORT_AREA'` en el tipo de modelo ComputerSystem, al ejecutar la herramienta de vista de atributos ampliados se crearán dos vistas:

- EA_COMPUTERSYSTEM40_V
  La nueva vista tendrá las siguientes columnas:
  - PK_C, que contiene la clave primaria.
  - SUPPORT_AREA_C, que contiene los atributos ampliados.
- Fix Pack 1 BE_COMPUTERSYSTEM40_V
  La nueva vista tendrá las siguientes columnas:
  - Todas las columnas de la vista de bloque de creación BB_COMPUTERSYSTEM40_V, por ejemplo PK_C, NAME_C, VMID_C.
  - SUPPORT_AREA_C, que contiene los atributos ampliados, sólo si no existe conflicto entre el nombre de columna y las columnas de la vista de bloque de creación BB_COMPUTERSYSTEM40_V.

Puede utilizar los siguientes mandatos SQL para consultar los datos:

```
SELECT
      T1.FQDN_C,
      T2.SUPPORT_AREA_C
     FROM
        BB_COMPUTERSYSTEM40_V T1,
        EA_COMPUTERSYSTEM40_V T2
     WHERE
        T1.PK_C = T2.PK_C
```

Fix Pack 1

```
SELECT
     FQDN_C, SUPPORT_AREA_C
    FROM
     BE_COMPUTERSYSTEM40_V
```

Nota: Fix Pack 1 Si se produce un conflicto entre los nombres de columna, la herramienta intenta añadir una cadena de categoría de atributo ampliado. Si así no se resuelve el problema, la columna de atributo ampliado no se incluye en las vistas BE_model_V.

Diccionario de datos de TADDM

El diccionario de datos de TADDM es una recopilación de páginas HTML generadas automáticamente que proporcionan una correlación entre la información del modelo de datos común (CDM) y la información de la base de datos de TADDM.

Acceso al diccionario de datos

El diccionario de datos está disponible en las ubicaciones siguientes del servidor de almacenamiento (en un despliegue del servidor en modalidad continua), un servidor de sincronización (en un despliegue de servidor de sincronización) o un servidor de dominio (en un despliegue de servidor de dominio):

- En http://*hostservidortaddm*:*puerto*/cdm/datadictionary/, por ejemplo http://1.123.123.12:9430/cdm/datadictionary/
- En el archivo taddm-data-dictionary.zip que se encuentra en los directorios $COLLATION_HOME/sdk/datadictionary y $COLLATION_HOME/deploy-tomcat/cdm/datadictionary (TADDM 7.3.0) o $COLLATION_HOME/apps/cdm/datadictionary (TADDM 7.3.0.1 y posterior).

  Para utilizar el archivo taddm-data-dictionary.zip, siga estos pasos:

1. Extraiga el contenido del archivo `taddm-data-dictionary.zip` en una ubicación de su elección.
2. Extraiga el archivo `$COLLATION_HOME/sdk/doc/model/CDMWebsite.zip` en el directorio `data-dictionary/cdm` de la estructura extraída del diccionario de datos.

Índices

El diccionario de datos incluye los índices siguientes:

Índice de vistas del bloque de creación

El índice está disponible en:

`http://`*`hostservidortaddm`*`:`*`puerto`*`/cdm/datadictionary/bb-views/index.html`

En el índice, pulse el nombre de una vista del bloque de creación. Aparecerá la información siguiente:

- La tabla de base de datos de TADDM desde la que se llena la correspondiente vista del bloque de creación. Todos los nombres de tabla enlazan con una definición de la tabla de base de datos.
- Columnas de la vista correspondiente del bloque de creación. Cada nombre de columna enlaza con la definición del CDM del atributo representado por la columna.

Índice del objeto de modelo

El índice está disponible en:

`http://`*`hostservidortaddm`*`:`*`puerto`*`/cdm/datadictionary/model-object/index.html`

En el índice, pulse el nombre de una clase de CDM. Aparecerá la información siguiente:

- Tablas de base de datos de TADDM que contienen la clase correspondiente del CDM. Todos los nombres de tabla enlazan con una definición de la tabla de base de datos.
- Vistas de bloque de creación que contienen la clase del CDM correspondiente. Todos los nombres de vista del bloque de creación enlazan con una definición de la vista del bloque de creación.

Índice de tablas del objeto de modelo

El índice está disponible en:

`http://`*`hostservidortaddm`*`:`*`puerto`*`/cdm/datadictionary/cdm-tables/index.html`

Desde el índice, pulse el nombre de una tabla de base de datos de TADDM. Aparecerá la información siguiente:

- Clase del CDM que declara la tabla de base de datos correspondiente. El nombre de clase del CDM enlaza con una definición de la clase.
- Clases del CDM que contiene la tabla de base de datos correspondiente. Todos los nombres de clase del CDM enlazan con una definición de la clase.
- Columnas en la tabla de base de datos correspondiente. Cada nombre de columna enlaza con la definición del CDM del atributo representado por la columna.

Índice de datos descubiertos por sensores

El índice está disponible en:

`http://`*`hostservidortaddm`*`:`*`puerto`*`/cdm/datadictionary/sensors/index.html`

Desde el índice, pulse el nombre de un sensor. Aparecerá la información siguiente:

- Información general sobre el sensor.
- Clase del CDM de objetos de modelo descubiertos por el sensor. Todos los nombres de clase del CDM enlazan con una definición de la clase.
- Atributos de las clases del CDM e información sobre su disponibilidad en el contexto del sensor.

Datos potenciales para descubrir el índice

El índice está disponible en:

`http://hostservidortaddm:puerto/cdm/datadictionary/sensors/potentialData.html`

Desde el índice, pulse el nombre de una categoría. Se mostrarán los nombres de los tipos de datos que pertenecen a la categoría correspondiente.

Pulse el nombre de un tipo de datos. Aparecerá la información siguiente:

- Información general sobre el sensor que descubre el tipo de datos correspondiente
- Clase del CDM de objetos de modelo descubiertos por el sensor. Todos los nombres de clase del CDM enlazan con una definición de la clase.
- Atributos de las clases del CDM

Índices adicionales

En el directorio `datadictionary/cdm` de Diccionario de datos, también puede encontrar los siguientes índices:

- Índice de clases en `http://hostservidortaddm:puerto/cdm/datadictionary/cdm/classes/$index.htm`
- Índice de interfaces en `http://hostservidortaddm:puerto/cdm/datadictionary/cdm/interfaces/$index.htm`
- Índice de atributos en `http://hostservidortaddm:puerto/cdm/datadictionary/cdm/attributes/$index.htm`
- Índice de relaciones en `http://hostservidortaddm:puerto/cdm/datadictionary/cdm/relationships/$index.htm`
- Índice de tipo de datos en `http://hostservidortaddm:puerto/cdm/datadictionary/cdm/datatypes/$index.htm`
- Índice de reglas de denominación en `http://hostservidortaddm:puerto/cdm/datadictionary/cdm/namingrules/$index.htm`

Información Javadoc sobre TADDM

Puede utilizar la información del Javadoc que se incluye con TADDM para obtener más información acerca de las API que hay disponibles.

Los archivos comprimidos siguientes contienen información Javadoc sobre TADDM:

$COLLATION_HOME/sdk/doc/api/oalapi-javadoc.zip

Este archivo contiene información acerca de las API Java de TADDM que hay disponibles.

$COLLATION_HOME/sdk/doc/api/taddmapi-javadoc.zip

Este archivo contiene información acerca de las API Java de TADDM que hay disponibles.

$COLLATION_HOME/sdk/doc/capabilities/capabilities-javadoc.zip
Este archivo contiene información acerca de la funcionalidad de las prestaciones que se pueden utilizar.

$COLLATION_HOME/sdk/doc/model/model-javadoc.zip
Este archivo contiene información acerca de los objetos de modelo de datos común utilizado por TADDM.

Avisos

Esta información se ha desarrollado para los productos y servicios ofrecidos en los Estados Unidos. Es posible que IBM no ofrezca los productos, los servicios o las funciones mencionados en otros países. Póngase en contacto con el representante de IBM local para obtener información acerca de los productos y servicios que se encuentran actualmente disponibles en su zona. Cualquier referencia a un producto, programa o servicio de IBM no pretende indicar ni implicar que solo se pueda utilizar ese producto, programa o servicio de IBM. En su lugar, puede utilizarse cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ningún derecho de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de esos productos, programas o servicios que no son de IBM.

IBM puede tener patentes o solicitudes de patentes pendientes que cubran temas descritos en este documento. La entrega de este documento no le garantiza licencias para estas patentes. Puede enviar solicitudes de licencia por escrito a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 EE.UU.

Para obtener solicitudes de licencia sobre información de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM en su país o envíe solicitudes por escrito a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japón

El párrafo siguiente no afecta al Reino Unido ni cualquier otro país donde estas condiciones no concuerden con la legislación local vigente:

INTERNATIONAL BUSINESS MACHINES CORPORATION FACILITA ESTA PUBLICACIÓN "TAL COMO ESTÁ" SIN GARANTÍA DE NINGÚN TIPO, YA SEA EXPRESA O IMPLÍCITA, INCLUYENDO, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE NO VIOLACIÓN, COMERCIABILIDAD O ADECUACIÓN A UN OBJETIVO CONCRETO.

Algunos estados no permiten declaración de limitación de responsabilidad de garantías expresas o implícitas en determinadas transacciones, por lo tanto, es posible que esta afirmación no le afecte.

Esta información podría incluir inexactitudes técnicas o errores tipográficos. La información contenida en este documento se modifica de manera periódica; estas modificaciones se incorporarán a las nuevas ediciones de la publicación. IBM puede realizar mejoras y/o cambios en los productos y/o los programas descritos en esta publicación en cualquier momento sin previo aviso.

Toda referencia contenida en esta información a sitios web que no sean de IBM se facilita solo por conveniencia, pero en ningún caso sirven como apoyo a dichos sitios web. Los materiales de esos sitios web no forman parte de los materiales de este producto de IBM y el uso que haga de ellos quedará por su cuenta y riesgo.

IBM puede utilizar o distribuir la información que usted proporcione de la manera que considere apropiada sin que ello le cree a usted ninguna obligación.

Los titulares de licencias de este programa que deseen obtener información al respecto con el fin de permitir: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido este) y (ii) el uso compartido de la información que se ha intercambiado, deben ponerse en contacto con:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 EE.UU.

Esta información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos el pago de una cantidad.

IBM proporciona el programa bajo licencia que se describe en este documento y todo el material bajo licencia disponible para él según los términos de las Condiciones Generales de International Business Machines S.A., el acuerdo internacional de licencia de programa o cualquier acuerdo equivalente entre las partes.

Todos los datos de rendimiento que contiene este documento se han determinado en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar de manera significativa. Es posible que algunas mediciones se hayan realizado en sistemas a nivel de desarrollo y no hay garantía de que dichas medidas sean las mismas en sistemas puestos a disposición general. Además, algunas mediciones pueden haberse estimado por extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información sobre los productos que no son de IBM se ha obtenido de los proveedores de dichos productos, sus declaraciones publicadas u otras fuentes públicas disponibles. IBM no ha probado esos productos, por lo no puede confirmar la corrección de su rendimiento, su compatibilidad ni otras afirmaciones relacionadas con productos que no sean de IBM. Las preguntas sobre las capacidades de los productos que no sean de IBM deben dirigirse a los proveedores de dichos productos.

Todas las declaraciones sobre la dirección o las intenciones futuras de IBM están sujetas a modificaciones o a retirada sin previo aviso, y representan solo objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones empresariales diarias. Para ilustrarlos lo mejor posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente fortuita.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en código fuente, que ilustran técnicas de programación en diferentes plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo como desee, sin pago alguno a IBM, con la intención de desarrollar, utilizar, comercializar o distribuir programas de aplicación de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni implicar la fiabilidad, capacidad de servicio o función de estos programas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier manera sin realizar ningún pago a IBM a fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación que se adecúen a las interfaces de programación de aplicaciones de IBM.

Cada copia o fragmento de estos programas de ejemplo, o cualquier trabajo derivado, debe incluir un aviso de copyright como el siguiente:

© (nombre de su empresa) (año). Algunos fragmentos de este código se derivan de los programas de ejemplo de IBM Corp. © Copyright IBM Corp. _enter the year or years_. Reservados todos los derechos.

Si consulta esta información como copia software, las fotografías e ilustraciones en color podrían no mostrarse.

Marca registradas

IBM, el logotipo de IBM e ibm.com son marcas comerciales o marcas registradas de International Business Machines Corp., en muchas jurisdicciones de todo el mundo. Otros productos y nombres de servicio pueden ser marcas comerciales de IBM o de otras empresas. Hay disponible una lista de marcas registradas de IBM en la página web de “Información sobre copyright y marcas registradas” en http://www.ibm.com/legal/copytrade.shtml.

ITIL es una marca registrada y una marca de comunidad registrada del Office of Government Commerce, y está registrada en U.S. Patent and Trademark Office.

IT Infrastructure Library es una marca registrada de la Agencia Central de Informática y Telecomunicaciones, que forma parte de la Oficina de Comercio Gubernamental.

Java y todas las marcas registradas y logotipos basados en Java son marcas comerciales o marcas comerciales registradas de Oracle y/o sus filiales.

Linux es una marca registrada de Linus Torvalds en Estados Unidos y en otros países.

Microsoft y Windows son marcas registradas de Microsoft Corporation en Estados Unidos y en otros países.

UNIX es una marca registrada de The Open Group en Estados Unidos y en otros países.

Otros nombres de empresas, productos y servicios pueden ser marcas registradas o de servicio de terceros.

Impreso en España