

HP Universal CMDB

Versión de software: 10.10

Guía de referencia para el desarrollador

Fecha de publicación del documento: Noviembre 2013

Fecha de lanzamiento del software: Noviembre 2013



Avisos legales

Garantía

Las únicas garantías de los productos y servicios HP se exponen en el certificado de garantía que acompaña a dichos productos y servicios. El presente documento no debe interpretarse como una garantía adicional. HP no es responsable de omisiones, errores técnicos o de edición contenidos en el presente documento.

La información contenida en esta página está sujeta a cambios sin previo aviso.

Leyenda de derechos limitados

Software informático confidencial. Es necesario disponer de una licencia válida de HP para su posesión, uso o copia. De conformidad con FAR 12.211 y 12.212, el Gobierno estadounidense dispone de licencia de software informático de uso comercial, documentación del software informático e información técnica para elementos de uso comercial con arreglo a la licencia estándar para uso comercial del proveedor.

Aviso de copyright

© Copyright 2002 - 2013 Hewlett-Packard Development Company, L.P.

Avisos de marcas comerciales

Adobe™ es una marca comercial de Adobe Systems Incorporated.

Microsoft® y Windows® son marcas comerciales registradas estadounidenses de Microsoft Corporation.

UNIX® es una marca comercial registrada de The Open Group.

Actualizaciones de la documentación

La página de título de este documento contiene la siguiente información de identificación:

- Número de versión del software, que indica la versión del software.
- Fecha de publicación del documento, que cambia cada vez que se actualiza el documento.
- Fecha de lanzamiento del software, que indica la fecha desde la que está disponible esta versión del software.

Para buscar actualizaciones recientes o verificar que está utilizando la edición más reciente de un documento, visite: <http://h20230.www2.hp.com/selfsolve/manuals>

Este sitio requiere que esté registrado como usuario de HP Passport. Para registrarse y obtener un ID de HP Passport, visite: <http://h20229.www2.hp.com/passport-registration.html>

O haga clic en el enlace **New user registration** (Registro de nuevos usuarios) de la página de registro de HP Passport.

Asimismo, recibirá ediciones actualizadas o nuevas si se suscribe al servicio de soporte del producto correspondiente. Póngase en contacto con su representante de ventas de HP para obtener más información.

Soporte

Visite el sitio web HP Software Support Online en: <http://www.hp.com/go/hpsoftwaresupport>

Este sitio web proporciona información de contacto y detalles sobre los productos, servicios y soporte que ofrece HP Software.

HP Software Support Online brinda a los clientes la posibilidad de auto-resolución de problemas. Ofrece una forma rápida y eficaz de acceder a las herramientas de soporte técnico interactivas necesarias para gestionar su negocio. Como cliente preferente de soporte, puede beneficiarse de utilizar el sitio web de soporte para:

- Buscar los documentos de la Base de conocimiento que le interesen
- Enviar y realizar un seguimiento de los casos de soporte y las solicitudes de mejora
- Descargar revisiones de software
- Gestionar contratos de soporte
- Buscar contactos de soporte de HP
- Consultar la información sobre los servicios disponibles
- Participar en debates con otros clientes de software
- Investigar sobre formación de software y registrarse para recibirla

Para acceder a la mayor parte de las áreas de soporte es necesario que se registre como usuario de HP Passport. En muchos casos también será necesario disponer de un contrato de soporte. Para registrarse y obtener un ID de HP Passport, visite:

<http://h20229.www2.hp.com/passport-registration.html>

Para obtener más información sobre los niveles de acceso, visite:

http://h20230.www2.hp.com/new_access_levels.jsp

HP Software Solutions Now accede al sitio web HPSW Solution and Integration Portal. Este sitio le permite explorar las soluciones de productos HP que satisfacen sus necesidades de negocio e incluye una lista completa de integraciones entre productos HP, así como una lista de procesos ITIL. La URL de este sitio web es <http://h20230.www2.hp.com/sc/solutions/index.jsp>

Acerca de esta versión en PDF de la Ayuda en línea

Este documento es una versión en PDF de la Ayuda en línea. Este archivo PDF se incluye para que pueda imprimir con facilidad varios temas de la información de ayuda o leer la Ayuda en línea en formato PDF. Como este contenido se creó originalmente para verse como Ayuda en línea en un navegador web, es posible que algunos temas no estén correctamente formateados. Además, algunos temas pueden no aparecer en esta versión en PDF. Esos temas se pueden imprimir correctamente dentro de la Ayuda en línea.

Contenido

Contenido	4
Creación de adaptadores de detección e integración	11
Capítulo 1: Desarrollo y escritura de adaptadores	12
Información general del desarrollo y escritura de adaptadores	12
Creación de contenido	12
Ciclo de desarrollo del adaptador	13
Data Flow Management e integración	16
Asociación del valor de negocio con el desarrollo de detección	17
Investigación de los requisitos de integración	18
Desarrollo del contenido de integración	21
Desarrollo del contenido de detección	23
Adaptadores de detección y componentes relacionados	23
Separación de adaptadores	24
Implementación de un adaptador de detección	25
Paso 1: Creación de un adaptador	28
Paso 2: Asignación de un trabajo al adaptador	35
Paso 3: Creación de código Jython	37
Configurar ejecución de proceso remoto	37
Capítulo 2: Desarrollo de los adaptadores Jython	39
Referencia de API de HP Data Flow Management	39
Creación de código Jython	39
Utilización de archivos Java JAR externos dentro de Jython	40
Ejecución del código	40
Modificación de secuencias de comandos de serie	41
Estructura del archivo Jython	41
Importaciones	42
Función principal: DiscoveryMain	42
Definición de funciones	43
Generación de resultados por la secuencia de comandos Jython	44

Sintaxis de ObjectStateHolder	44
Enviar grandes cantidades de datos	46
La instancia Framework	47
Encontrar las credenciales correctas (para adaptadores de conexión)	50
Manejo de excepciones desde Java	53
Solución de problemas de migración de Jython Versión 2.1 a 2.5.3	53
Soporte de la localización en los adaptadores Jython	55
Adición de soporte para un nuevo idioma	56
Cambio del idioma predeterminado	57
Determinación del juego de caracteres para codificación	57
Definición de un nuevo trabajo para operar con datos localizados	58
Descodificación de comandos sin una palabra clave	59
Trabajo con paquetes de recursos	60
Referencia de API	61
Registro de código DFM	63
Bibliotecas y utilidades Jython	65
Capítulo 3: Mensajes de error	68
Información general de los mensajes de error	68
Convenciones para la escritura de errores	68
Niveles de gravedad de errores	71
Capítulo 4: Desarrollo de adaptadores de bases de datos genéricas	73
Información general de los adaptadores de bases de datos genéricas	74
Consultas TQL para el adaptador de bases de datos genéricas	74
Conciliación	75
Hibernate como proveedor de JPA	76
Preparar la creación del adaptador	78
Preparación del paquete de adaptadores	83
Configuración del adaptador - Método mínimo	86
Configurar el archivo adapter.conf	87
Ejemplo: Rellenado de un nodo y dirección IP utilizando el método simplificado	87
Configuración del adaptador - Método avanzado	91

Implementación de un complemento	96
Despliegue del adaptador	99
Edición del adaptador	99
Crear un punto de integración	99
Creación de una vista	100
Cálculo de los resultados	100
Visualización de resultados	100
Ver informes	100
Habilitación de archivos de registro	101
Uso de Eclipse para asignar entre atributos CIT y tablas de bases de datos	101
Archivos de configuración de adaptadores	109
Archivo adapter.conf	110
Archivo simplifiedConfiguration.xml	112
Archivo orm.xml	114
Archivo reconciliation_types.txt	128
Archivo reconciliation_rules.txt (para compatibilidad con versiones anteriores)	128
Archivo transformations.txt	130
Archivo discriminator.properties	131
Archivo replication_config.txt	132
Archivo fixed_values.txt	132
Archivo persistence.xml	133
Conectarse a base de datos mediante autenticación NT	134
Convertidores de serie	134
Complementos	140
Ejemplos de configuración	140
Archivos de registro del adaptador	149
Referencias externas	151
Solución de problemas y limitaciones	151
Capítulo 5: Desarrollo de adaptadores Java	153
Información general de Federation Framework	153
Adaptador y asignación de interacción con Federation Framework	158

Federation Framework para consultas TQL federadas	159
Interacciones entre Federation Framework, servidor, adaptador y motor de asignación	160
Flujo de Federation Framework para Rellenado	169
Interfaces del adaptador	170
Depuración de recursos del adaptador	172
Adición de un adaptador para un nuevo origen de datos externo	172
Creación de un adaptador de ejemplo	181
Propiedades y etiquetas de configuración XML	182
La interfaz DataAdapterEnvironment	184
OutputStream openResourceForWriting(String resourceName) throws FileNotFoundException;	184
InputStream openResourceForReading(String resourceName) throws FileNotFoundException;	184
Properties openResourceAsProperties(String propertiesFile) throws IOException;	185
String openResourceAsString(String resourceName) throws IOException;	185
public void saveResourceFromString(String relativeFileName, String value) throws IOException;	186
boolean resourceExists(String resourceName);	186
boolean deleteResource(String resourceName);	187
Collection<Cadena> listResourcesInPath(String path);	187
DataAdapterLogger getLogger();	187
DestinationConfig getDestinationConfig();	188
Int getChunkSize();	188
int getPushChunkSize();	188
ClassModel getLocalClassModel();	188
CustomerInformation getLocalCustomerInformation();	188
Object getSettingValue(String name);	188
Map<String, Object> getAllSettings();	189
boolean isMTEnabled();	189
String getUcmdbServerHostName();	189
Capítulo 6: Desarrollo de los adaptadores de inserción	190
Desarrollo e implementación de los adaptadores de inserción	190

Construcción de un paquete de adaptadores	191
Solución de problemas	194
Prácticas recomendadas de TQL para los adaptadores de inserción	195
Crear asignaciones	195
Compilar un archivo de asignación	195
Preparación de los archivos de asignación	196
Escritura de secuencias de comandos Jython	199
Admisión de la sincronización diferencial	203
Consultas de SQL del adaptador de inserción XML genérico	205
Adaptador genérico de inserción de servicio web	205
Referencia de archivos de asignación	225
Esquema del archivo de asignación	227
Esquema de resultados de asignación	239
Personalización	243
Capítulo 7: Desarrollo de adaptadores de inserción genéricos mejorados	245
Información general	245
El archivo de asignación	245
Groovy Traveler	248
Escritura de secuencias de comandos Groovy	251
Implementar la interfaz de PushConnector	252
Construcción de un paquete de adaptadores	253
Esquema del archivo de asignación	254
Utilización de las API	263
Capítulo 8: Introducción a las API	264
Información general de las API	264
Capítulo 9: API de HP Universal CMDB	265
Convenciones	265
Utilización de las API de HP Universal CMDB	265
Estructura general de una aplicación	266
Colocar el archivo Jar de la API en la ruta de clase	269
Creación de un usuario de integración	269

Casos de uso de la API de UCMDB	271
Ejemplos	272
Capítulo 10: HP Universal CMDB API del servicio web	274
Convenciones	274
Información general de la API de servicio web de HP Universal CMDB	275
Llamada al servicio web	278
Consulta de CMDB	278
Actualización de UCMDB	282
Importación del modelo de clase de UCMDB	283
getClassAncestors	283
getAllClassesHierarchy	284
getCmdbClassDefinition	284
Consulta de análisis de impacto	285
Parámetros generales de UCMDB	285
Parámetros de salida de UCMDB	288
Métodos de consulta de UCMDB	289
executeTopologyQueryByNameWithParameters	289
executeTopologyQueryWithParameters	290
getChangedCIs	291
getCINeighbours	292
getCIsByID	292
getCIsByType	293
getFilteredCIsByType	293
getQueryNameOfView	297
getTopologyQueryExistingResultByName	298
getTopologyQueryResultCountByName	298
pullTopologyMapChunks	299
releaseChunks	300
Métodos de actualización de UCMDB	301
addCIsAndRelations	301
addCustomer	302

deleteCIsAndRelations	302
removeCustomer	303
updateCIsAndRelations	303
Métodos de análisis de impacto de UCMDB	303
calculateImpact	304
getImpactPath	304
getImpactRulesByNamePrefix	305
API del servicio web de estado real	305
Casos de uso de la API de los servicios web de UCMDB	307
Ejemplos	308
Capítulo 11: API de Java de Administración de Data Flow	309
Uso de la API de Java de Administración de Data Flow	309
Capítulo 12: API de servicio web de Administración de Data Flow	311
Información general de la API de servicio web de Administración de Data Flow	311
Convenciones	312
Llamar al servicio web	312
Métodos y estructuras de datos de Administración de Data Flow	312
Estructuras de datos	313
Administración de los métodos de trabajo de detección	314
Gestión de métodos de activación	315
Métodos de datos de sondas y dominios	317
Métodos de datos de credenciales	319
Métodos de actualización de datos	321
Ejemplo de código	323
Ejemplo de adición de credenciales	326
Agradecemos sus comentarios.	330

Creación de adaptadores de detección e integración

Capítulo 1: Desarrollo y escritura de adaptadores

Este capítulo incluye:

Información general del desarrollo y escritura de adaptadores	12
Creación de contenido	12
Desarrollo del contenido de integración	21
Desarrollo del contenido de detección	23
Implementación de un adaptador de detección	25
Paso 1: Creación de un adaptador	28
Paso 2: Asignación de un trabajo al adaptador	35
Paso 3: Creación de código Jython	37
Configurar ejecución de proceso remoto	37

Información general del desarrollo y escritura de adaptadores

Antes de comenzar la planificación real del desarrollo de los nuevos adaptadores, es importante comprender los procesos e interacciones habitualmente asociados a este desarrollo.

Las secciones siguientes le pueden ayudar a comprender lo que necesita saber y hacer para gestionar y ejecutar satisfactoriamente un proyecto de desarrollo de detección.

En este capítulo :

- Se supone un conocimiento práctico de HP Universal CMDB y cierta familiaridad básica con los elementos del sistema. Lo que significa una ayuda en el proceso de aprendizaje y que no proporciona una guía completa.
- Se abarcan las etapas de planificación, investigación e implementación del nuevo contenido detectado para HP Universal CMDB, junto con las directrices y consideraciones que hay que tener en cuenta.
- Debe proporcionarse información sobre las API de clave de Data Flow Management Framework. Para ver toda la documentación sobre las API disponibles, consulte la *Referencia de API de HP Universal CMDB Data Flow Management*. (Hay otras API no formales que, aunque se usan en los adaptadores de serie, pueden estar sujetas a cambios).

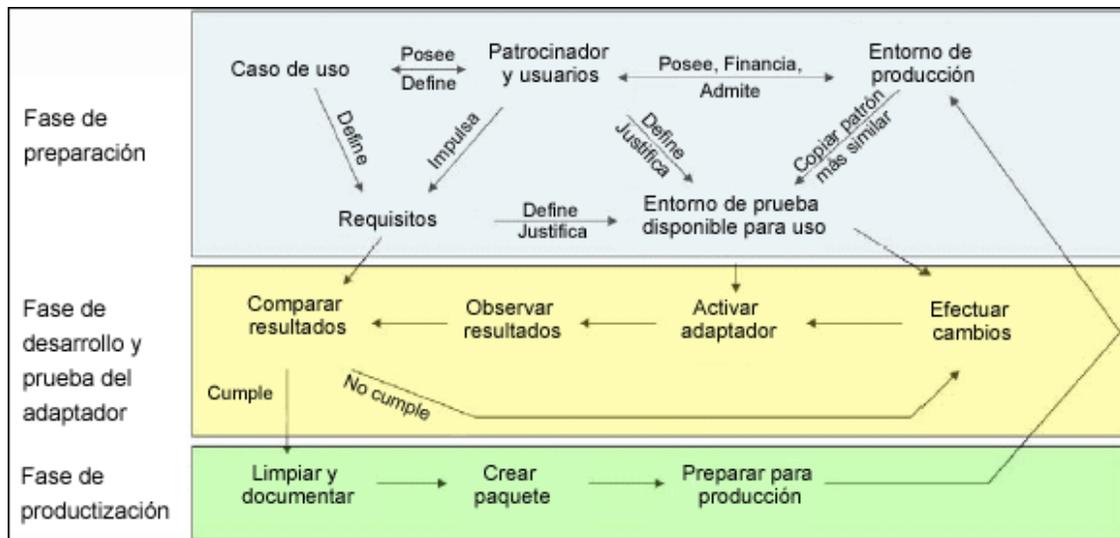
Creación de contenido

Esta sección incluye:

- ["Ciclo de desarrollo del adaptador" abajo](#)
- ["Data Flow Management e integración" en la página 16](#)
- ["Asociación del valor de negocio con el desarrollo de detección" en la página 17](#)
- ["Investigación de los requisitos de integración" en la página 18](#)

Ciclo de desarrollo del adaptador

En la ilustración siguiente se muestra un gráfico de flujo para la escritura de adaptadores. La mayor parte del tiempo se reserva a la sección central, que consiste en un bucle iterativo de desarrollo y prueba.



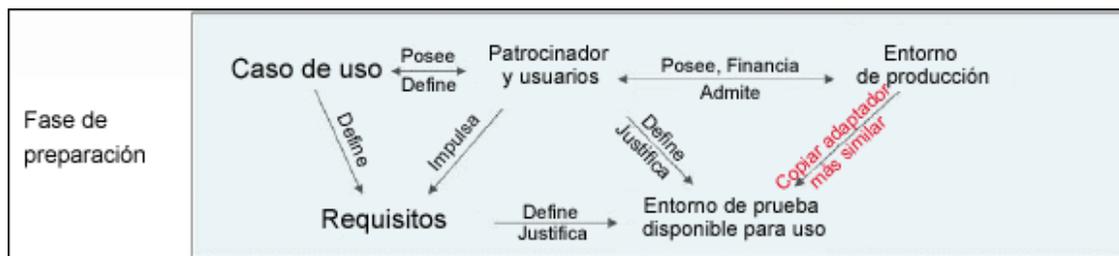
Cada fase del desarrollo del adaptador se construye sobre la anterior.

Cuando esté satisfecho con el aspecto y funcionamiento del adaptador, ya está listo para empaquetarlo. Con el Administrador de paquetes de UCMDb o exportando manualmente los componentes, cree un archivo de paquete *.zip. Como práctica recomendada, debería desplegar y probar este paquete en otro sistema de UCMDb antes de lanzarlo a producción para asegurarse de que todos los componentes están representados y correctamente empaquetados. Para obtener información detallada sobre el empaquetado, consulte "Administrador de paquetes" en *HP Universal CMDB – Guía de administración*.

En las secciones siguientes se amplía cada una de las fases, mostrando los pasos más críticos y las prácticas recomendadas:

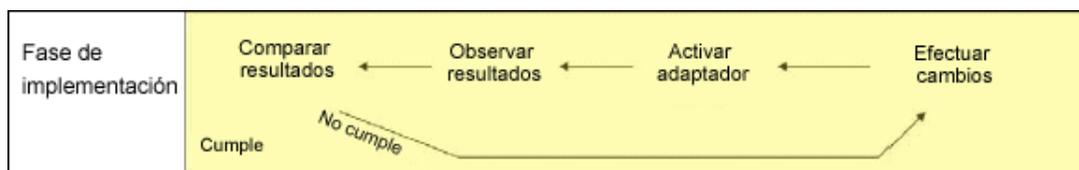
- ["Fase de investigación y preparación" en la página siguiente](#)
- ["Desarrollo y prueba del adaptador" en la página siguiente](#)
- ["Empaquetado y conversión a producto del adaptador" en la página 15](#)

Fase de investigación y preparación



La fase Investigación y preparación abarca las necesidades motrices de negocio y los casos de uso, y también considera la protección de las instalaciones necesarias para desarrollar y probar el adaptador.

1. Al planificar la modificación de un adaptador existente, el primer paso técnico es hacer una copia de seguridad del adaptador y asegurarse de que puede devolverlo a su estado original. Si piensa crear un nuevo adaptador, copie el adaptador más similar y guárdelo con un nombre apropiado. Para obtener más información, consulte "Panel Recursos" en la *HP Universal CMDB – Guía de Administración de Data Flow*.
2. Investigue el método que el adaptador debe utilizar para recopilar los datos:
 - Utilice herramientas o protocolos externos para obtener los datos
 - Desarrolle cómo el adaptador debería crear CI en función de los datos
 - Ahora sabe cómo debería ser un adaptador similar
3. Determine el adaptador más similar en función de:
 - Los mismos CI creados
 - Los mismos protocolos usados (SNMP)
 - El mismo tipo de destino (por tipo de sistema operativo, versiones, etc.)
4. Copie el paquete completo.
5. Descomprima el contenido del paquete en el espacio de trabajo y cambie el nombre de los archivos del adaptador (XML) y Jython (.py).



Desarrollo y prueba del adaptador

La fase Desarrollo y prueba del adaptador es un proceso sumamente iterativo. Cuando el adaptador empieza a tomar forma, comienza a realizar la prueba en los casos de uso finales, a realizar

cambios, a probar de nuevo y a repetir este proceso hasta que el adaptador cumple con los requisitos.

Inicio y preparación de la copia

- Modifique las partes XML del adaptador: Nombre (ID) en línea 1, tipos de CI creados y nombre de la secuencia de comandos Jython llamada.
- Haga que la copia se ejecute con resultados idénticos al adaptador original.
- Defina como comentarios la mayor parte del código, especialmente el código que produzca los resultados críticos.

Desarrollo y prueba

- Utilice otro código de ejemplo para desarrollar cambios
- Pruebe el adaptador ejecutándolo
- Utilice una vista dedicada para validar los resultados complejos, busque validar los resultados sencillos

Empaquetado y conversión a producto del adaptador

La fase Empaquetado y conversión a producto del adaptador constituye la última fase del desarrollo. Como práctica recomendada, se debe realizar una revisión final para limpiar de restos de depuración, documentos y comentarios, para examinar las consideraciones de seguridad, etc., antes de desplazarse al empaquetado. Siempre debería disponer al menos de un documento Léame para explicar el funcionamiento interno del adaptador. Alguien (quizá el mismo usuario) puede tener que examinar este adaptador en el futuro y le resultará de gran ayuda incluso la documentación más limitada.

Limpieza y documentación

- Suprima la depuración
- Marque como comentarios todas las funciones y agregue algunos comentarios de apertura en la sección principal
- Cree un TQL de ejemplo y haga que lo pruebe el usuario

Crear paquete

- Exporte los adaptadores, el TQL y demás con el Administrador de paquetes. Para obtener más información, consulte "Package Manager" en *HP Universal CMDB – Guía de administración*.
- Compruebe cualquier dependencia entre el paquete y otros paquetes, por ejemplo, si los CI creados por esos paquetes son CI de entradas al adaptador.
- Utilice el Administrador de paquetes para crear un archivo zip de paquetes. Para obtener más información, consulte "Package Manager" en *HP Universal CMDB – Guía de administración*.

- Pruebe el despliegue quitando partes del nuevo contenido y volviendo a desplegarlo, o despliegue otro sistema de prueba.

Data Flow Management e integración

Los adaptadores DFM pueden integrarse con otros productos. Tenga en cuenta las siguientes definiciones:

- DFM recopila contenido específico de varios destinos.
- La integración recopila varios tipos de contenido de un sistema.

Tenga en cuenta que estas definiciones no distinguen entre los métodos de colección. Tampoco el DFM. El proceso del desarrollo de un nuevo adaptador es el mismo que el proceso de desarrollo de una nueva integración. Realiza la misma investigación, las mismas elecciones de adaptadores nuevos frente a existentes, escribe los adaptadores de la misma forma, etc. Solo cambian unas pocas cosas:

- La planificación final del adaptador. Los adaptadores de integración pueden ejecutarse con más frecuencia que los de detección, pero depende de los casos de uso.
- CI de entrada:
 - Integración: sin CI de activación para ejecutarse sin entrada: se pasa un nombre de archivo u origen mediante el parámetro del adaptador.
 - Discovery (detección): usa los CI de CMDB regulares como entrada.

En los proyectos de integración, casi siempre se debería reutilizar un adaptador existente. La dirección de la integración (desde HP Universal CMDB a otro producto, o desde otro producto a HP Universal CMDB) puede afectar a su enfoque al desarrollo. Puede copiar para sus propios usos de los paquetes de campos disponibles, utilizando técnicas probadas.

De HP Universal CMDB a otro proyecto:

- Cree un TQL que produzca los CI y las relaciones que se van a exportar.
- Utilice un adaptador de contenedor genérico para ejecutar el TQL y escriba los resultados en un archivo XML para que lo lea el producto externo.

Nota: Para obtener ejemplos de paquetes de campos, póngase en contacto con HP Software Support.

Para integrar otro producto en HP Universal CMDB, dependiendo de cómo el otro producto muestra sus datos, el adaptador de integración actúa de manera diferente:

Tipo de integración	Ejemplo de referencia que se va a reutilizar
Acceder directamente a la base de datos del producto	HP ED
Leer en un archivo csv o xml producido por una exportación	HP ServiceCenter
Acceder a la API de un producto	BMC Atrium/Remedy

Asociación del valor de negocio con el desarrollo de detección

El caso de uso para desarrollar nuevo contenido de detección debe estar orientado por un caso de negocio y un plan para producir un valor de negocio. Es decir, el objetivo de asignar componentes del sistema a los CI y agregarlos a CMDB es proporcionar valor de negocio.

El contenido no siempre se va a utilizar para la asignación de aplicaciones, aunque es un paso intermedio habitual para muchos casos de uso. Con independencia del uso final del contenido, el plan debería responder a las preguntas siguientes:

- ¿Quién es el consumidor? ¿Cómo debería actuar el consumidor en relación a la información proporcionada por los CI (y las relaciones entre ellos)? ¿Cuál es el contexto de negocio en el que se van a visualizar los CI y las relaciones? ¿Es el consumidor de estos CI una persona, un producto o ambos?
- Cuando existe la combinación perfecta de CI y relaciones en CMDB, ¿cómo piensa utilizarlos para producir un valor de negocio?
- ¿Cómo sería la asignación perfecta?
 - ¿Qué término describiría de manera más significativa las relaciones entre cada CI?
 - ¿Qué tipos de CI son los que habría que incluir en primer lugar?
 - ¿Cuál es el uso final y el usuario final de la asignación?
- ¿Cuál sería el diseño de informe perfecto?

Cuando se haya establecido la justificación del negocio, el paso siguiente consiste en plasmar el valor de negocio en un documento. Esto implica representar el mapa perfecto usando una herramienta de dibujo y comprender el impacto y las dependencias entre los CI, informes, cómo se realiza el seguimiento de los cambios, qué cambio es importante, la monitorización, la conformidad y el valor de negocio adicional, tal como requieren los casos de uso.

Este dibujo (o modelo) se conoce como **plano técnico**.

Por ejemplo, si es fundamental para la aplicación saber cuándo ha cambiado un determinado archivo de configuración, éste debe representarse y vincularse al CI apropiado (al que está relacionado) en el mapa dibujado.

Trabaje con un SME (experto en la materia del asunto) del área, que es el usuario final del contenido desarrollado. Este experto debería apuntar a las entidades críticas (los CI con atributos y relaciones) que deben existir en el CMDB para proporcionar valor de negocio.

Un método podría ser proporcionar un cuestionario al propietario de la aplicación (también el SME en este caso). El propietario debería poder especificar los objetivos y el plano técnico anterior. El propietario al menos debe proporcionar una arquitectura actual de la aplicación.

Debería representar únicamente datos críticos y no datos innecesarios: siempre podrá mejorar posteriormente el adaptador. El objetivo debería ser configurar una detección limitada que funcione y proporcione valor. La representación de grandes cantidades de datos produce mapas más impresionantes pero pueden resultar confusos y necesitar mucho tiempo en su desarrollo.

Cuando el modelo y el valor de negocio están claros, continúe al paso siguiente. Esta etapa se puede volver a revisar cuando se proporcione información más concreta en las etapas siguientes.

Investigación de los requisitos de integración

El requisito previo de esta etapa es un **plano técnico** de los CI y relaciones que el DFM tiene que detectar, que deberían incluir los atributos que se van a detectar. Para obtener más información, consulte ["Información general del desarrollo y escritura de adaptadores" en la página 12](#).

Esta sección incluye los siguientes temas:

- ["Modificación de un adaptador existente" abajo](#)
- ["Escribir un nuevo adaptador" en la página siguiente](#)
- ["Investigación del modelo" en la página siguiente](#)
- ["Investigación de tecnología" en la página siguiente](#)
- ["Directivas para elegir formas de acceder a los datos" en la página 20](#)
- ["Resumen " en la página 20](#)

Modificación de un adaptador existente

Puede modificar un adaptador existente cuando existe un adaptador de campo o uno de serie, pero:

- no detecta atributos específicos que son necesarios
- no se está detectando un tipo de destino (OS) específico o se está detectando de manera incorrecta
- no se está detectando o creando una relación específica

Si un adaptador existente realiza parte del trabajo, pero no todo, su primer enfoque debería ser evaluar los adaptadores existentes y comprobar si uno de ellos hace más o menos lo que se precisa; si lo hace, puede modificar el adaptador existente.

También debería evaluar si está disponible algún adaptador de campo existente. Los adaptadores de campo son adaptadores de detección que están disponibles pero que no son de serie. Póngase en contacto con HP Software Support para recibir la lista actual de adaptadores de campo.

Escribir un nuevo adaptador

Hay que desarrollar un nuevo adaptador:

- Cuando es más rápido escribir un adaptador que introducir la información manualmente en CMDB (por lo general, desde 50 a 100 CI y relaciones aproximadamente) o que no sea una tarea que se realice una sola vez.
- Cuando la necesidad justifique el esfuerzo.
- Si los adaptadores de serie o de campo no están disponibles.
- Si los resultados se pueden reutilizar.
- Cuando el entorno de destino o sus datos están disponibles (no puede detectar lo que no puede ver).

Investigación del modelo

- Examine el modelo de clase de UCMDB (Administrador de tipos de CI) y haga corresponder las entidades y relaciones desde su **plano técnico** a los CIT existentes. Se recomienda encarecidamente adherirse al modelo actual para evitar posibles complicaciones durante la actualización de la versión. Si tiene que ampliar el modelo, debería crear nuevos CIT ya que una actualización puede sobrescribir los CIT de serie.
- Si algunas entidades, relaciones o atributos carecen del modelo actual, debería crearlos. Es preferible crear un paquete con estos CIT (que posteriormente también almacenará toda la detección, vistas y otros artefactos relacionados con este paquete), ya que tendrá que implementar estos CIT en cada instalación de HP Universal CMDB.

Investigación de tecnología

Cuando haya verificado que CMDB contiene los CI relevantes, la etapa siguiente consiste en decidir cómo recuperar estos datos de los sistemas relevantes.

La recuperación de los datos por lo general implica la utilización de un protocolo para acceder a una parte de gestión de la administración, los datos reales de la aplicación o archivos de configuración o bases de datos relacionadas con la aplicación. Cualquier origen de datos que pueda proporcionar información sobre un sistema es valioso. La investigación de tecnología requiere tanto un amplio conocimiento del sistema en cuestión y a veces creatividad.

Para las aplicaciones creadas internamente, puede resultar útil proporcionar un cuestionario al propietario de la aplicación. En este formulario el propietario debería enumerar todas las áreas de la aplicación que pueden proporcionar la información necesaria para el plano técnico y los valores de negocio. Esta información debería incluir (pero no estar limitada) las bases de datos de gestión, archivos de configuración, archivos de registros, interfaces de gestión, programas de administración, servicios web, mensajes o eventos enviados, entre otros.

En el caso de los productos diseñados, debería concentrarse en la documentación, foros o soporte del producto. Busque las guías de administración, guías de complementos e integraciones, guías de gestión, entre otras. Si todavía faltan datos en las interfaces de gestión, lea sobre los archivos de configuración de la aplicación, entradas del Registro, archivos de registro, registros de eventos NT y cualquier artefacto de la aplicación que controle su operación correcta.

Directivas para elegir formas de acceder a los datos

Relevancia: Seleccione los orígenes o una combinación de orígenes que proporcionan la mayoría de los datos. Si un único origen proporciona la mayor parte de información mientras que el resto de información está disperso o es difícil de acceder, pruebe a evaluar el valor de la información restante en comparación con el esfuerzo o riesgo de obtenerla. A veces se puede reducir el plano técnico si el valor o el coste no garantiza el esfuerzo invertido.

Reutilización: Si HP Universal CMDB ya incluye un soporte de protocolo de conexión específico, es un buen motivo para usarlo. Significa que DFM Framework puede proporcionar un cliente ya preparado y la configuración para la conexión. En caso contrario, es posible que tenga que invertir en desarrollo de infraestructura. Puede ver los protocolos de conexión de compatibles actualmente de HP Universal CMDB en **Administración de Data Flow > Configuración de Data Flow Probe> panel Dominios y sondas**. Para obtener más información sobre cada protocolo, consulte la sección que describe los protocolos compatibles en la *HP UCMDB Discovery and Integration Content Guide*.

Puede agregar nuevos protocolos añadiendo nuevos CI al modelo. Para obtener más información, póngase en contacto con HP Software Support.

Nota: Para acceder a los datos del Registro de Windows, puede usar WMI o NTCMD.

Seguridad: El acceso a la información suele requerir credenciales (nombre de usuario, contraseña), que se introducen en CMDB y se mantienen protegidas en todo el producto. Si es posible, y si la adición de seguridad no entra en conflicto con otros principios que haya establecido, elija la credencial o protocolo menos sensible que todavía responda a las necesidades de acceso. Por ejemplo, si la información está disponible tanto mediante JMX (interfaz de administración estándar, limitada) y Telnet, es preferible usar JMX ya que proporciona inherentemente acceso limitado y (por lo general) no permite el acceso a la plataforma subyacente.

Comodidad: Algunas interfaces de gestión pueden incluir funciones más avanzadas. Por ejemplo, puede ser más fácil enviar consultas (SQL, WMI) que desplazarse por árboles de información o crear expresiones regulares para análisis.

Audiencia del programador: Aquellos que en último término desarrollarán los adaptadores pueden estar inclinados hacia una determinada tecnología. También se puede tener en cuenta si dos tecnologías proporcionan casi la misma información al mismo coste en otros factores.

Resumen

El resultado de esta etapa es un documento que describe los métodos de acceso y la información relevante que se puede extraer para cada método. El documento también debería contener un mapa desde cada origen a cada dato del plano técnico relevante.

Cada método de acceso debería estar marcado de acuerdo con las instrucciones anteriores. Finalmente, debería tener ahora un plan en relación a qué orígenes se van a detectar y qué

información se va a extraer de cada origen en el modelo del plano técnico (lo que ahora debería estar representado en el modelo de UCMDb correspondiente).

Desarrollo del contenido de integración

Antes de crear una nueva integración, debe comprender cuáles son los requisitos de dicha integración:

- ¿Debe la integración copiar datos en el CMDB? ¿Deben rastrearse los datos por el historial? ¿Es el origen no fiable?

Si su respuesta es sí a estas preguntas, se necesita **Rellenado**.

- ¿Debe la integración federar datos al instante para vistas y consultas TQL? ¿Es crítica la precisión de los cambios en los datos? ¿La cantidad de datos es demasiado grande para copiarse en CMDB, pero la cantidad de datos solicitada suele ser pequeña?

Si su respuesta es sí a estas preguntas, se necesita **Federación**.

- ¿Debe la integración insertar datos en orígenes de datos remotos?

Si su respuesta es sí a estas preguntas, se necesita **Inserción de datos**.

Nota: Los flujos Federación y Rellenado pueden configurarse en la misma integración para una máxima flexibilidad.

Para obtener información detallada sobre los distintos tipos de integraciones, consulte *Integration Studio en HP Universal CMDB – Guía de Administración de Data Flow*.

Hay cinco opciones diferentes disponibles para crear adaptadores de integración:

1. Adaptador Jython:

- El patrón de detección clásico
- Escrito en Jython
- Usado para el relleno

Para obtener más información, consulte "[Desarrollo de los adaptadores Jython](#)" en la [página 39](#).

2. Adaptador Java:

- Adaptador que implementa una de las interfaces de adaptador en Federation Framework SDK.

- Se puede usar para uno o varios procesos de Federación, Rellenado o Inserción de datos (dependiendo de la implementación requerida).
- Escrito en su totalidad en Java, que permite escribir código que se conectará a cualquier origen o destino posible.
- Adecuado para trabajos que se conectan a un único origen o destino de datos.

Para obtener más información, consulte "[Desarrollo de adaptadores Java](#)" en la página 153.

3. Adaptador de base de datos genérica:

- Un adaptador abstracto basado en el adaptador Java que utiliza el SDK de Federation Framework.
- Permite la creación de adaptadores que se conectan a repositorios de datos externos.
- Admite tanto la Federación como el Rellenado (con un complemento Java implementado para admitir los cambios).
- Relativamente fácil de definir, ya que está basado sobre todo en XML y en archivos de configuración de propiedades.
- La configuración principal está basada en un archivo **orm.xml** que establece relaciones entre las clases UCMDB y las columnas de la base de datos.
- Adecuado para trabajos que se conectan a un único origen de datos.

Para obtener más información, consulte "[Desarrollo de adaptadores de bases de datos genéricas](#)" en la página 73.

4. Adaptador genérico de inserción:

- Un adaptador abstracto basado en el adaptador Java (Federation Framework SDK) y el adaptador Jython.
- Permite la creación de adaptadores que insertan datos a destinos remotos.
- Relativamente fácil de definir, ya que solo se necesita definir la relación entre las clases UCMDB y XML, y una secuencia de comandos Jython que introduzca los datos en el destino.
- Adecuado para trabajos que se conectan a un único destino de datos.
- Utilizado para la inserción de datos.

Para obtener más información, consulte "[Desarrollo de los adaptadores de inserción](#)" en la página 190.

5. Adaptador genérico de inserción mejorado:

- Todas las características anteriores del adaptador genérico de inserción
- Un adaptador basado en el elemento raíz
- Asigna una estructura de datos de árbol de UC MDB a una estructura de datos de árbol de destino

Para obtener más información, consulte "[Desarrollo de adaptadores de inserción genéricos mejorados](#)" en la página 245.

En la tabla siguiente se muestran las características de cada adaptador:

Flujo/Adaptador	Adaptador Jython	Adaptador Java	Adaptador DB genérico	Adaptador de inserción genérico	Adaptador genérico de inserción mejorado
Rellenado	✓	✓	✓	✗	✗
Federación	✗	✓	✓	✗	✗
Inserción de datos	✗	✓	✗	✓	✓

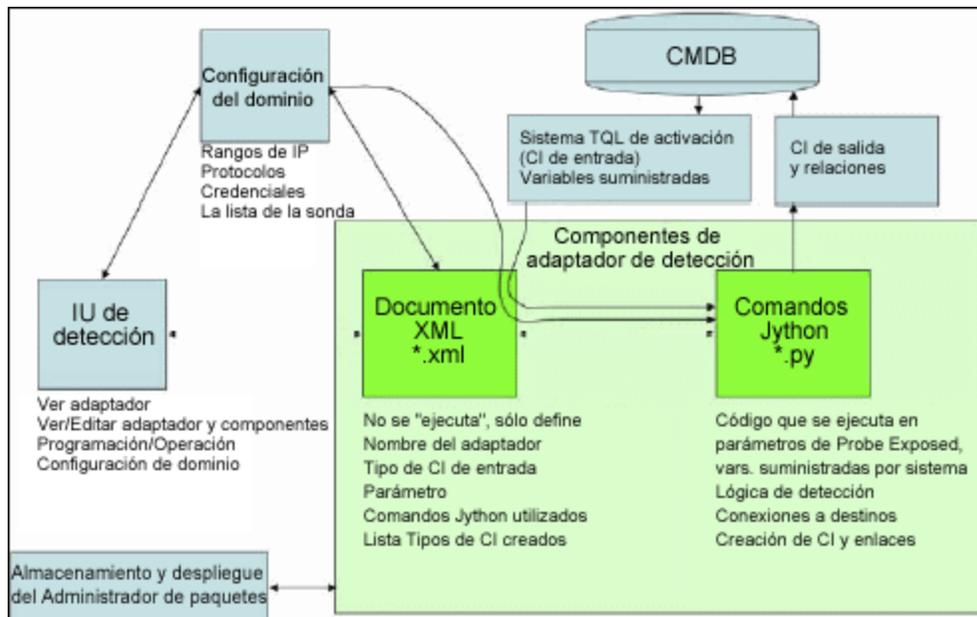
Desarrollo del contenido de detección

Esta sección incluye:

- "[Adaptadores de detección y componentes relacionados](#)" abajo
- "[Separación de adaptadores](#)" en la página siguiente

Adaptadores de detección y componentes relacionados

En el diagrama siguiente se muestran los componentes de un adaptador y los componentes que interactúan con la ejecución de la detección. Los componentes indicados en verde son los adaptadores reales y los componentes en azul son componentes que interactúan con los adaptadores.



Tenga en cuenta que la noción mínima de un adaptador es de dos archivos: un documento XML y una secuencia de comandos Jython. Discovery Framework, incluyendo los CI de entrada, credenciales y bibliotecas proporcionadas por el usuario, se expone al adaptador en tiempo de ejecución. Ambos componentes del adaptador de detección se administran mediante Data Flow Management. Se almacenan operativamente en el mismo CMDB; aunque sigue el paquete externo, no se hace referencia a él en el funcionamiento. El Administrador de paquetes permite la conservación de la nueva capacidad de contenido de detección e integración.

Un TQL proporciona los CI de entrada del adaptador y se exponen a la secuencia de comandos del adaptador en variables suministradas por el sistema. También se suministran parámetros del adaptador como datos de destino, por tanto puede configurar el funcionamiento del adaptador de acuerdo con la función específica del adaptador.

La aplicación DFM se utiliza para crear y probar nuevos adaptadores. Utilice las páginas Universal Discovery, Administración de adaptador y Configuración de Data Flow Probe durante la escritura del adaptador.

Los adaptadores se almacenan y se transportan como paquetes. La aplicación Administrador de paquetes y la consola JMX se utilizan para crear paquetes desde adaptadores recientemente creados y para implementar los adaptadores en nuevos sistemas.

Separación de adaptadores

Una detección completa podría definirse en un único adaptador. Sin embargo, el buen diseño exige que un sistema complejo esté separado en componentes más sencillos y más manejables.

A continuación se describen las directrices y prácticas recomendadas para dividir el proceso del adaptador:

- La detección debería realizarse en etapas. Cada etapa debería estar representada por un adaptador que debe representar un área o nivel del sistema. Los adaptadores deberían depender de la etapa o nivel anterior de detección para continuar la detección del sistema. Por ejemplo, el adaptador A está activado por un resultado TQL del servidor de aplicación y representa el nivel del servidor de aplicación. Como parte de esta representación, se representa un componente de conexión de JDBC. El adaptador B registra un componente de la conexión JDBC como un TQL de activación y utiliza el resultado del adaptador A para acceder al nivel de base de datos (por ejemplo, a través del atributo URL de JDBC) y representa el nivel de la base de datos.
- **El paradigma de conexión de dos fases:** La mayoría de los sistemas requieren credenciales para acceder a sus datos. Esto significa que una combinación de usuario/contraseña tiene que probarse con estos sistemas. El administrador de DFM proporciona la información de las credenciales de una forma segura al sistema y puede ofrecer varias credenciales de inicio de sesión priorizadas. Esto se conoce como **Diccionario de protocolos**. Si por cualquier motivo no se puede acceder al sistema, no hay motivo para seguir realizando la detección. Si la conexión se ha realizado correctamente, será preciso de establecer una forma de indicar qué conjunto de credenciales se ha usado correctamente para poder acceder a la detección en el futuro.

Estas dos fases conducen a una separación de los dos adaptadores en los casos siguientes:

- **Adaptador de conexión:** Es un adaptador que acepta un activador inicial y busca la existente de un agente remoto en ese activador. Lo hace probando todas las entradas del Diccionario de protocolos que coincidan con el tipo de agente. Si se realiza correctamente, este adaptador proporciona como resultado un CI de agente remoto (SNMP, WMI, entre otros), que también apunta a la entrada correcta del Diccionario de protocolos para futuras conexiones. Este CI de agente forma entonces parte de un activador para el adaptador de contenido.
- **Adaptador de contenido:** La condición previa de este adaptador es la conexión correcta del adaptador anterior (condiciones previas especificadas por los TQL). Estos tipos de adaptadores ya no necesitan recorrer todo el Diccionario de protocolos ya que tienen una forma de obtener las credenciales correctas desde el CI de agente remoto y utilizarlas para iniciar sesión en el sistema detectado.
- En la división de detección también pueden influir las diferentes consideraciones de planificación. Por ejemplo, un sistema solo puede consultarse durante las horas no laborables, por tanto aunque pudiera tener sentido unir el adaptador al mismo adaptador que detecta otro sistema, las diferentes programaciones implican que es preciso crear dos adaptadores.
- La detección de distintas interfaces de gestión o tecnologías para detectar el mismo sistema deberían situarse en adaptadores independientes. De esta forma, podrá activar el método de acceso apropiado para cada sistema u organización. Por ejemplo, algunas organizaciones tienen acceso WMI a los equipos, pero no tienen agentes SNMP instalados en ellos.

Implementación de un adaptador de detección

Una tarea DFM tiene el objetivo de acceder a sistemas remotos (o locales), modelando datos extraídos como CI y guardando los CI al CMDB. La tarea consta de los siguientes pasos:

1. Crear un adaptador.

Puede configurar un archivo de adaptador que almacene el contexto, parámetros y tipos de resultado seleccionando las secuencias de comandos que van a formar parte del adaptador. Para obtener más información, consulte ["Paso 1: Creación de un adaptador" en la página 28.](#)

2. Crear trabajo de detección

Puede configurar un trabajo con información de planificación y una consulta de activación. Para obtener más información, consulte ["Paso 2: Asignación de un trabajo al adaptador" en la página 35.](#)

3. Editar código de detección.

Puede editar el código Jython o Java contenido en los archivos del adaptador y que se refiere a DFM Framework. Para obtener más información, consulte ["Paso 3: Creación de código Jython" en la página 37.](#)

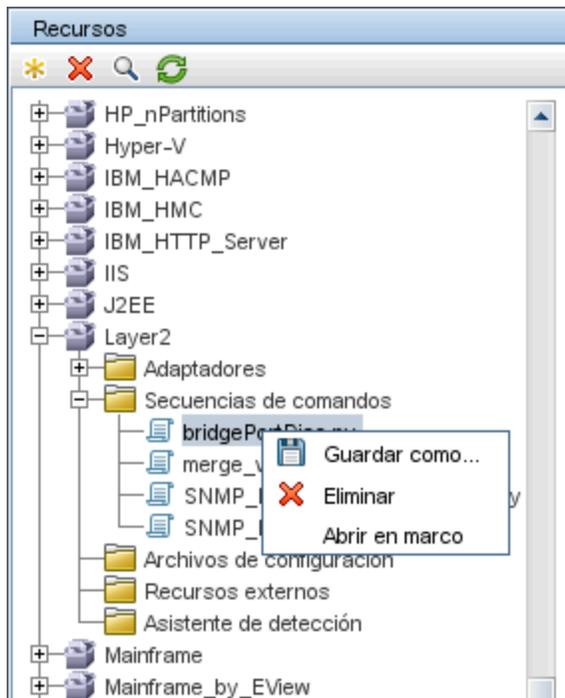
Para escribir nuevos adaptadores, puede crear los componentes anteriores, cada uno de los cuales se vincula automáticamente al componente del paso anterior. Por ejemplo, una vez creado un trabajo y seleccionado el adaptador relevante, el archivo del adaptador se vincula al trabajo.

Código del adaptador

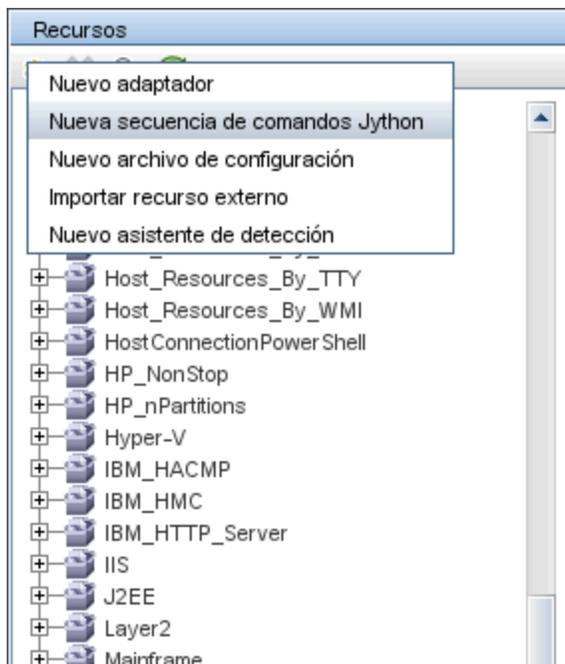
La implementación real de la conexión al sistema remoto, la consulta de los datos y su asignación como datos de CMDB lo realiza el código Jython. Por ejemplo, el código contiene la lógica para la conexión a una base de datos y su extracción de los datos. En este caso, el código espera recibir una URL de JDBC, un nombre de usuario, una contraseña, un puerto, etc. Estos parámetros son específicos para cada instancia de la base de datos que responde a la consulta de TQL. Puede definir estas variables en el adaptador (en los datos del CI de activación) y, cuando se ejecuta el trabajo, estos detalles concretos se pasan al código para su ejecución.

El adaptador puede referirse a este código por un nombre de clase Java o un nombre de secuencias de comandos Jython. En esta sección, se tratará cómo escribir código DFM como secuencias de comandos Jython.

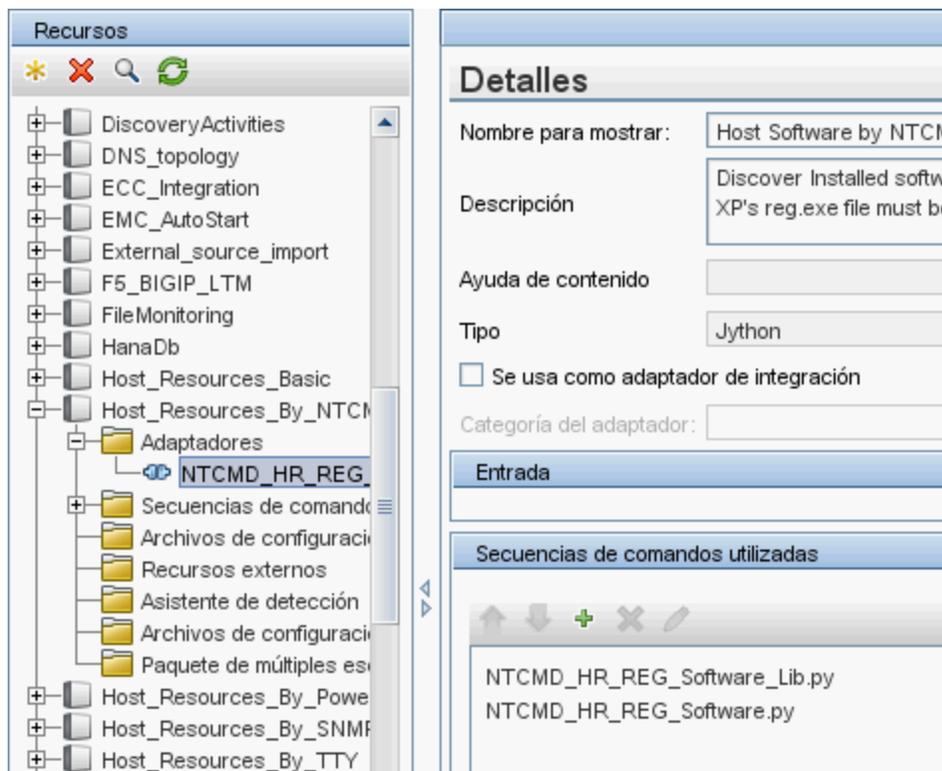
Un adaptador puede contener una lista de secuencias de comandos que se pueden utilizar al ejecutar la detección. Al crear un nuevo adaptador, por lo general creará una nueva secuencia de comandos y la asignará al adaptador. Una nueva secuencia de comandos incluye plantillas básicas, pero puede usar una de las otras secuencias de comandos como plantilla haciendo clic con el botón derecho sobre ella y seleccionando **Guardar como**:



Para obtener más información sobre cómo escribir nuevas secuencias de comandos Jython, consulte "[Paso 3: Creación de código Jython](#)" en la página 37. Puede agregar secuencias de comandos mediante el panel Recursos:



La lista de secuencias de comandos se ejecuta una tras otra, en el orden definido en el adaptador:



Nota: Se puede especificar una secuencia de comandos aunque otra secuencia de comandos la esté utilizando únicamente como biblioteca. En este caso, debe definirse la secuencia de comandos de la biblioteca antes de que la utilice la secuencia de comandos. En este ejemplo, la secuencia de comandos `processdbutils.py` es una biblioteca usada por la última secuencia de comandos `host_processes.py`. Las bibliotecas se distinguen de las secuencias de comandos regulares ejecutables por la ausencia de la función `DiscoveryMain()`.

Paso 1: Creación de un adaptador

Se puede considerar al adaptador como la definición de una función. Esta función define una definición de entrada, ejecuta la lógica en la entrada, define la salida y proporciona un resultado.

Cada adaptador especifica una entrada y una salida: tanto la entrada como la salida son CI de activación específicamente definidos en el adaptador. El adaptador extrae los datos desde el CI de activación de entrada y pasa estos datos como parámetros al código. Los datos procedentes de CI relacionados a veces se pasan también al código. Para obtener más información, consulte Ventana CI relacionados en la *HP Universal CMDB – Guía de Administración de Data Flow*. El código del adaptador es genérico, excepto esos parámetros específicos del CI de activación de entrada que se pasan al código.

Para obtener más información sobre los componentes de entrada, consulte Conceptos de Administración de Data Flow en la *HP Universal CMDB – Guía de Administración de Data Flow*.

Esta sección incluye los siguientes temas:

- "Definición de la entrada del adaptador(CIT de activación y consulta de entrada)" abajo
- "Definición de la salida del adaptador" en la página 31
- "Reemplazar los parámetros del adaptador" en la página 33
- "Reemplazar selección de sonda: opcional" en la página 34
- "Configure una ruta de clase para un proceso remoto: opcional" en la página 35

1. Definición de la entrada del adaptador(CIT de activación y consulta de entrada)

Puede usar los componentes CIT de activación y Consulta de entrada para definir CI específicos como entrada del adaptador:

- El CIT de activación define qué CIT se utiliza como entrada para el adaptador. Por ejemplo, para un adaptador que va a detectar direcciones IP, el CIT de entrada es Red.
- La Consulta de entrada es una consulta regular y editable que define la consulta en CMDB. La Consulta de entrada define restricciones adicionales en el CIT (por ejemplo, si la tarea requiere un atributo `hostID` o `application_ip`) y puede definir más datos de CI, si lo necesita el adaptador.

Si el adaptador requiere información adicional de los CI que están relacionados con el CI de activación, puede agregar nodos adicionales a la consulta de entrada. Para obtener más información, consulte *Cómo agregar nodos de consulta y relaciones a una consulta TQL en la HP Universal CMDB – Guía de modelado*.

- Los datos del CI de activación contienen toda la información requerida en el CI de activación, así como información de otros nodos de la consulta de entrada, si se definen. DFM usa variables para recuperar datos desde los CI. Cuando la tarea se descarga en la prueba, las variables de datos del CI de activación se reemplazan con valores reales que existen en los atributos para instancias de CI reales.
- Si el valor de los datos de destino es una lista, puede definir el número de elementos de la lista que se enviará a la sonda. Para definirlo, agregue una coma después del valor predeterminado seguido del número de elementos. Si no hay ningún valor predeterminado para los datos de destino, introduzca dos puntos.

Por ejemplo, si se introducen los siguientes datos de destino: `name=portId, value=${PHYSICALPORT.root_id:NA:1}` o `name=portId, value=${PHYSICALPORT.root_id: :1}`, a la sonda solo se envía el primer puerto de la lista de puertos.

Ejemplo de sustitución de variables con datos reales:

En este ejemplo, las variables sustituyen los datos del CI **IpAddress** por valores reales que existen en instancias del CI **IpAddress** reales del sistema.

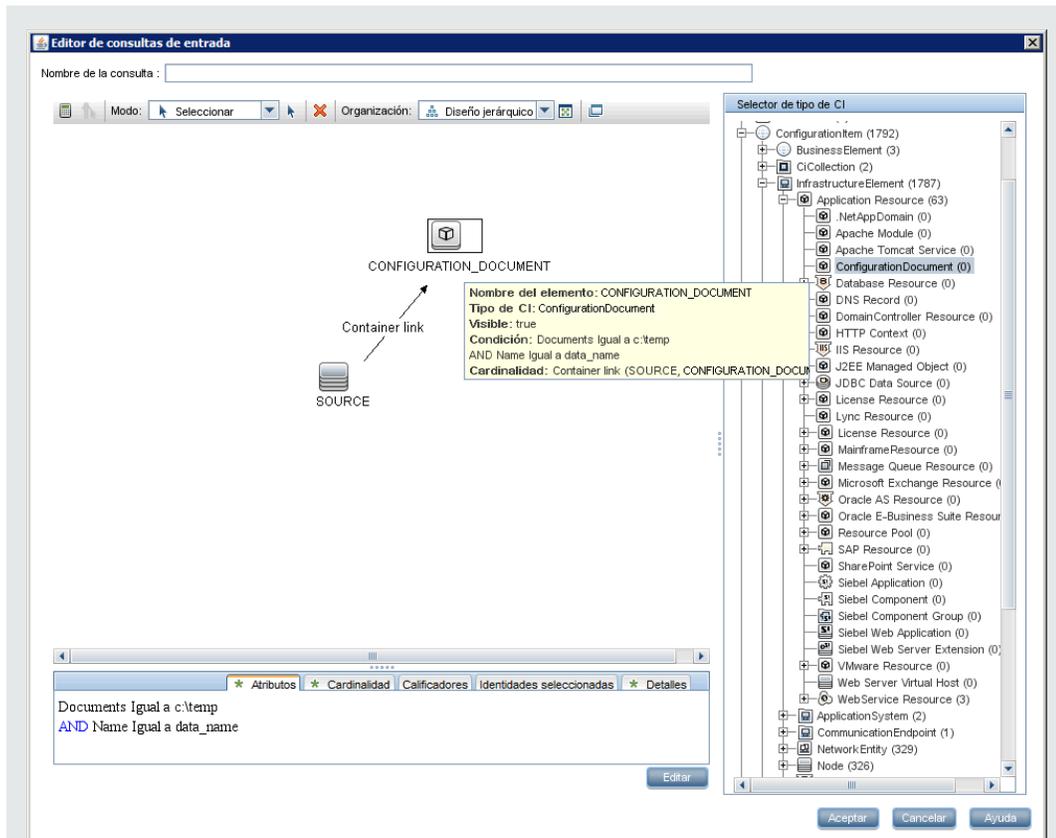
Los datos de CI activados para el CI **IpAddress** incluyen una variable `fileName`. Esta variable permite la sustitución del nodo **CONFIGURATION_FILE** en el TQL de entrada con los valores reales del archivo de configuración ubicado en un host:

Nombre	Valor
Protocol	\${SOURCE.credentials_id}
credentialsId	\${SOURCE.credentials_id}
fileName	\${CONFIGURATION_FILE.data_name}
hostID	\${HOST.root_id}
hostKey	\${SOURCE.host_key}
ip_address	\${SOURCE.application_ip}
path	\${CONFIGURATION_FILE.document_path}

Los datos del CI de activación se cargan en la sonda con todas las variables sustituidas por valores reales. La secuencia de comandos del adaptador incluye un comando para utilizar DFM Framework para recuperar los valores reales de las variables definidas:

```
Framework.getTriggerCIData ('ip_address')
```

Las variables `fileName` y `path` usan los atributos `data_name` y `document_path` desde el nodo **CONFIGURATION_DOCUMENT** (definido en el Editor de consulta de entrada: consulte el ejemplo anterior).



Haga clic en la miniatura para ver una imagen a tamaño completo.

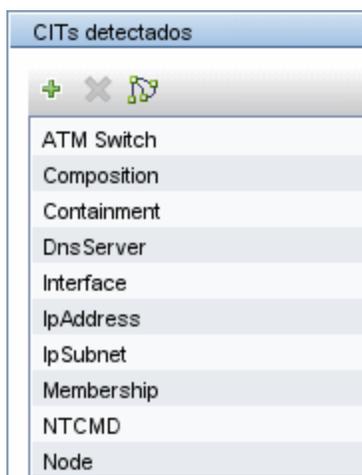
Las variables Protocol, credentialsId e ip_address usan los atributos root_class, credentials_id y application_ip:

Clave	Nombre para most...	Nombre	Tipo	Descripción	Valor predet...	Visible
ack_cleared_time	ack_cleared_time	ack_cleared_time	long			
ack_id	ack_id	ack_id	string			
BODY_ICON	BODY_ICON	BODY_ICON	string	host		
city	City	City	string	City location		✓
codepage	CodePage	CodePage	string	System su...		
contextmenu	Context Menu	Context Menu	string_list	Context me... itCIs		
country	Country	Country	string	Country loc...		✓
credentials_id	Reference to the cre...	Reference to the cre...	string	Reference ...		
data_adminstate	Admin State	Admin State	adminstate...	Admin State	Managed	

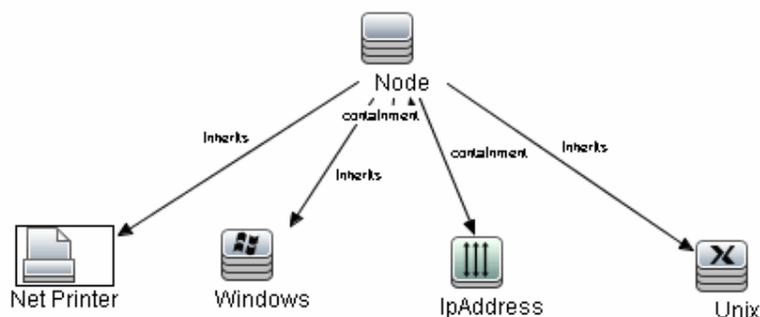
2. Definición de la salida del adaptador

La salida del adaptador es una lista de CI detectados (**Administración de Data Flow > Administración de adaptador > ficha Definición de adaptador > CIT detectados**) y los

vínculos que hay entre ellos:



También puede ver los CIT como un mapa de topología, es decir, los componentes y la forma en que se vinculan (haga clic en el botón **Ver CITs detectados en mapa**):



El código DFM devuelve los CI detectados (es decir, la secuencia de comandos Jython) en formato de `ObjectStateHolderVector` de UCMDB. Para obtener más información, consulte ["Generación de resultados por la secuencia de comandos Jython" en la página 44](#).

Ejemplo de salida del adaptador:

En este ejemplo, define los CIT que van a formar parte de la salida del CI IP.

- a. Acceda a **Administración de Data Flow > Administración de adaptador**.
- b. En el panel Recursos, seleccione **Red > Adaptadores > NSLOOKUP_on_Probe**.
- c. En la ficha Definición de adaptador, localice el panel CITs detectados.
- d. Se enumeran los CIT que van a formar parte de la salida del adaptador. Agregue CIT a la lista, o suprámalos de allí. Para obtener más información, consulte Ficha Definición de adaptador en la *HP Universal CMDB – Guía de Administración de Data Flow*.

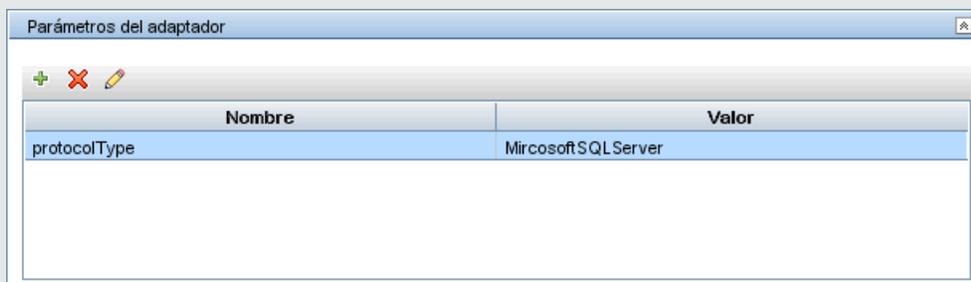
3. Reemplazar los parámetros del adaptador

Para configurar un adaptador para más de un trabajo, puede reemplazar los parámetros del adaptador. Por ejemplo, el adaptador `SQL_NET_Dis_Connection` se utiliza en los trabajos `MSSQL Connection by SQL` y `Oracle Connection by SQL`.

Ejemplo de reemplazo de un parámetro del adaptador:

En este ejemplo se muestra el reemplazo de un parámetro del adaptador de tal forma que se puede usar un adaptador para detectar bases de datos Microsoft SQL Server y Oracle.

- a. Acceda a **Administración de Data Flow > Administración de adaptador**.
- b. En el panel Recursos, seleccione **Database_Basic > Adaptadores > SQL_NET_Dis_Connection**.
- c. En la ficha Definición de adaptador, localice el panel **Parámetros del adaptador**. El parámetro `protocolType` tiene un valor de **all**:



Nombre	Valor
protocolType	MicrosoftSQLServer

- d. Haga clic con el botón derecho en el adaptador **SQL_NET_Dis_Connection_MsSql** y seleccione **Ir a trabajo de detección > MSSQL Connection by SQL**.
- e. Visualice la ficha **Propiedades**. Localice el panel Parámetros:

Parámetros		
Reemplazar	Nombre	Valor
<input checked="" type="checkbox"/>	protocolType	MicrosoftSQLServer

El valor all es reemplazado por el valor MicrosoftSQLServer.

Nota: El trabajo **Oracle Connection by SQL** incluye el mismo parámetro pero el valor se reemplaza por un valor Oracle.

Para obtener más información sobre cómo agregar, eliminar o editar parámetros, consulte Ficha Definición de adaptador en la *HP Universal CMDB – Guía de Administración de Data Flow*.

DFM comienza a buscar instancias de Microsoft SQL Server de acuerdo con este parámetro.

4. Reemplazar selección de sonda: opcional

En el servidor UCMDDB hay un mecanismo de distribución que toma los CI de activación recibidos por UCMDDB y elige automáticamente la sonda que deberá ejecutar el trabajo para cada CI de activación según una de las opciones siguientes.

- **Para el tipo de CI de dirección IP:** tome la sonda definida para esta dirección IP.
- **Para el tipo de CI del software en ejecución:** Utilice los atributos **application_ip** y **application_ip_domain** y elija la sonda que se define para la IP en el dominio correspondiente.
- **Para otros tipos de CI:** tome la dirección IP del nodo según el nodo relacionado del CI (si existe).

La selección automática de sonda se realiza según el nodo relacionado del CI. Tras obtener el nodo relacionado del CI, el mecanismo de distribución elige una de las direcciones IP del nodo y elige la sonda según las definiciones del ámbito de red de la sonda.

En los casos siguientes, tiene que especificar la sonda manualmente y no utilizar el mecanismo de distribución automática:

- Ya sabe qué sonda debe ejecutarse para el adaptador y no es necesario que el mecanismo de distribución automática seleccione la sonda (por ejemplo, si el CI de activación es la puerta de enlace de la sonda).
- La selección automática de sonda podría dar error. Esto puede ocurrir en las siguientes situaciones:

- Un CI de activación no tiene un nodo relacionado (como el CIT de red).
- Un nodo de CI de activación tiene varias direcciones IP, cada una perteneciente a una sonda distinta.

Para especificar manualmente qué sonda se va a usar con el adaptador:

- Seleccione el adaptador y haga clic en la ficha **Configuración de adaptador**.
- En **Opciones de distribución de activador**, seleccione **Reemplazar selección de sonda predeterminada**.
- En el cuadro, introduzca la sonda en uno de los formatos siguientes:

Nombre de sonda	El nombre de la sonda
Dirección IP	La dirección IP de la sonda, que puede definirse en formato IPv4 o IPv6
IP,dominio	Formato IPv4: 16.59.63.86,DefaultDomain Formato IPv6: 2001:0:9d46:953c:34a9:1e6b:f2ff:ffe,CustomDomain
Nombre de dominio	El dominio en el que se debe seleccionar la sonda.

Por ejemplo:

5. Configure una ruta de clase para un proceso remoto: opcional

Para obtener más información, consulte "[Configurar ejecución de proceso remoto](#)" en la [página 37](#).

Paso 2: Asignación de un trabajo al adaptador

Cada adaptador tiene uno o más trabajos asociados que definen la directiva de ejecución. Los trabajos permiten la planificación del mismo adaptador de manera diferente en un conjunto distinto de CI activados y también permiten que se proporcionen distintos parámetros para cada conjunto.

Los trabajos aparecen en el árbol Módulos de detección y esta es la entidad que activa el usuario, como se muestra en la imagen siguiente .



Elegir una consulta de activación

Cada trabajo debe asociarse a los TQL de activación. Estos TQL de activación publican los resultados que se usan como CI de activación de entrada para el adaptador de este trabajo.

Un TQL de activación puede agregar restricciones a una consulta de entrada. Por ejemplo, si los resultados de una consulta de entrada son las IP conectadas a SNMP, los resultados de una consulta de activación pueden ser IP conectadas a SNMP dentro del rango 195.0.0.0-195.0.0.10.

Nota: Una consulta de activación debe hacer referencia a los mismos objetos a los que hace referencia la consulta de entrada. Por ejemplo, si una consulta de entrada consulta IP que ejecutan SNMP, no puede definir una consulta de activación (para el mismo trabajo) para que consulte IP conectadas a un host, porque algunas de las IP pueden no estar conectadas a un objeto SNMP, tal como requiere la consulta de entrada.

Configurar información de planificación

La información de planificación de la sonda especifica cuándo ejecutar el código en CI de activación. Si se activa la casilla **Invocar en nuevos CIs activados inmediatamente**, el código también se ejecuta una vez en cada CI de activación cuando alcanza a la sonda, con independencia de la configuración de planificación futura.



Para cada ocurrencia planificada de cada trabajo, la sonda ejecuta el código en todos los CI de activación acumulados para dicho trabajo. Para obtener más información, consulte "Cuadro de diálogo Planificador de detección" en *HP Universal CMDB – Guía de Administración de Data Flow*.

Reemplazar los parámetros del adaptador

Al configurar un trabajo, puede reemplazar los parámetros del adaptador. Para obtener más información, consulte ["Reemplazar los parámetros del adaptador" en la página 33](#).

Paso 3: Creación de código Jython

HP Universal CMDB utiliza secuencias de comandos Jython para la escritura de adaptadores. Por ejemplo, la secuencia de comandos `SNMP_Connection.py` la utiliza el adaptador `SNMP_NET_Dis_Connection` para probar y conectarse a equipos usando SNMP. Jython es un lenguaje basado en Python y con tecnología Java.

Para obtener más información sobre cómo trabajar en Jython, puede consultar estos sitios Web:

- <http://www.jython.org>
- <http://www.python.org>

Para obtener más información, consulte ["Creación de código Jython" en la página 39](#).

Configurar ejecución de proceso remoto

Puede ejecutar la detección para un trabajo de detección en un proceso independiente del proceso de Data Flow Probe.

Por ejemplo, puede ejecutar el trabajo en un proceso remoto independiente si el trabajo utiliza bibliotecas `.jar` que sean de una versión diferente de las bibliotecas de la sonda o que sean incompatibles con las bibliotecas de la sonda.

También puede ejecutar el trabajo en un proceso remoto independiente si el trabajo consume potencialmente mucha memoria (aporta muchos datos) y preferirá aislar la sonda de problemas potenciales de falta de memoria (**OutOMemory**).

Para configurar un trabajo para ejecutarlo como un proceso remoto, defina los parámetros siguientes en el archivo de configuración de su adaptador:

Parámetro	Descripción
<code>remoteJVMArgs</code>	Parámetros de JVM para el proceso Java remoto.
<code>runInSeparateProcess</code>	Cuando se establece como true , el trabajo de detección se ejecuta en un proceso independiente.

Parámetro	Descripción
remoteJVMClasspath	<p>(Opcional) Permite la personalización de la ruta de clase del proceso remoto, reemplazando la ruta de clase de sonda predeterminada. Esto es útil si puede haber una incompatibilidad de versiones entre los archivos jar de la sonda y los archivos jar personalizados requeridos para la detección definida por el cliente.</p> <p>Si no se ha definido el parámetro remoteJVMClasspath, o se ha dejado vacío, se utiliza la ruta de clase de sonda predeterminada.</p> <p>Si desarrolla un trabajo de detección nuevo y desea asegurarse de que la versión de la biblioteca jar de la sonda no esté en conflicto con las bibliotecas jar del trabajo, debe utilizar al menos la ruta de clase mínima requerida para ejecutar la detección básica. La ruta de clase mínima está definida en DataFlowProbe.properties, en el parámetro basic_discovery_minimal_classpath.</p> <p>Ejemplos de personalización de remoteJVMClasspath:</p> <ul style="list-style-type: none">• Para agregar o agregar como prefijo archivos jar personalizados a la ruta de clase de sonda predeterminada, personalice el parámetro remoteJVMClasspath de la manera siguiente: <code>custom1.jar;%classpath%;custom2.jar -</code> En este caso, se coloca custom1.jar antes de la ruta de clase de sonda predeterminada y custom2.jar se agrega a la ruta de clase de la sonda.• Para utilizar la ruta de clase mínima, personalice el parámetro remoteJVMClasspath de la manera siguiente: <code>custom1.jar;%minimal_classpath%;custom2.jar</code>

Capítulo 2: Desarrollo de los adaptadores Jython

Este capítulo incluye:

Referencia de API de HP Data Flow Management	39
Creación de código Jython	39
Soporte de la localización en los adaptadores Jython	55
Registro de código DFM	63
Bibliotecas y utilidades Jython	65

Referencia de API de HP Data Flow Management

Para obtener toda la información sobre las API disponibles, consulte *Referencia de API de HP Universal CMDB Data Flow Management*. Estos archivos están ubicados en la carpeta siguiente:

<directorio de instalación de UCMDB>\UCMDBServer\deploymdb-docs\docs\eng\APIs\DDM_JavaDoc\index.html

Creación de código Jython

HP Universal CMDB utiliza secuencias de comandos Jython para la escritura de adaptadores. Por ejemplo, la secuencia de comandos **SNMP_Connection.py** la utiliza el adaptador **SNMP_NET_Dis_Connection** para intentar conectarse a equipos usando SNMP. Jython es un lenguaje basado en Python y con tecnología Java.

Para obtener más información sobre cómo trabajar en Jython, puede consultar estos sitios Web:

- <http://www.jython.org>
- <http://www.python.org>

La sección siguiente describe la escritura real del código Jython dentro de DFM Framework. Esta sección aborda específicamente estos puntos de contactos entre la secuencia de comandos Jython y Framework al que llama, y también describe las bibliotecas Jython y utilidades que deberían usarse siempre que sea posible.

Nota:

- Las secuencias de comandos escritas para Universal Discovery deben ser compatibles con Jython versión 2.5.3.

- Para obtener toda la información sobre las API disponibles, consulte *Referencia de API de HP Universal CMDB Data Flow Management*.

Esta sección incluye los siguientes temas:

- ["Utilización de archivos Java JAR externos dentro de Jython" abajo](#)
- ["Ejecución del código" abajo](#)
- ["Modificación de secuencias de comandos de serie" en la página siguiente](#)
- ["Estructura del archivo Jython" en la página siguiente](#)
- ["Generación de resultados por la secuencia de comandos Jython" en la página 44](#)
- ["La instancia Framework" en la página 47](#)
- ["Encontrar las credenciales correctas \(para adaptadores de conexión\)" en la página 50](#)
- ["Manejo de excepciones desde Java" en la página 53](#)

Utilización de archivos Java JAR externos dentro de Jython

Al desarrollar nuevas secuencias de comandos Jython, a veces se necesitan bibliotecas Java externas (archivos JAR) como archivos ejecutables de terceros como archivos de utilidad Java, archivos de conexión como archivos JAR de JDBC Driver o archivos ejecutables (por ejemplo, **nmap.exe** se usa para la detección sin credenciales).

Estos recursos deberían agruparse en el paquete debajo de la carpeta **Recursos externos**. Todos los recursos que se colocan en esta carpeta se envían automáticamente a la sonda que se conecta con el servidor de HP Universal CMDB.

Además, cuando se inicia la detección, todos los recursos de archivos JAR se cargan en la ruta de clase de Jython, haciendo que todas las clases que están dentro de ella se puedan importar y usar.

Ejecución del código

Una vez activado un trabajo, se descarga en la sonda una tarea con toda la información necesaria.

La sonda inicia la ejecución del código de DFM usando la información especificada en la tarea.

El flujo del código Jython inicia la ejecución desde una entrada principal de la secuencia de comandos, ejecuta el código para detectar los CI y proporciona los resultados de un vector de CI detectados.

Modificación de secuencias de comandos de serie

Al realizar modificaciones de secuencias de comandos de serie, realice solo cambios mínimos en ellas y coloque todos los métodos necesarios en una secuencia de comandos externa. Puede realizar el seguimiento de los cambios con más eficacia y, al pasar a una versión de HP Universal CMDB más reciente, el código no se sobrescribe.

Por ejemplo, la siguiente línea de código de una secuencia de comandos de serie llama a un método que calcula un nombre de servidor web de forma específica de aplicación:

```
serverName = iplanet_cspecific.PlugInProcessing(serverName, transportHN, mam_utils)
```

En una secuencia de comandos externa está contenida la lógica más compleja que decide cómo calcular este nombre:

```
# implement customer specific processing for 'servername' attribute of httpplugin
#
def PlugInProcessing(servername, transportHN, mam_utils_handle):
    # support application-specific HTTP plug-in naming
    if servername == "appsrv_instance":
        # servername is supposed to match up with the j2ee server name,
        however some groups do strange things with their
        # iPlanet plug-in files. this is the best work-around we could find.
        # this join can't be done with IP address:port
        # because multiple apps on a web server share the same IP:port for
        # or multiple websphere applications
        logger.debug('httpcontext_webapplicationserver attribute has been
        changed from [' + servername + '] to [' + transportHN[:5] + '] to facilitate
        websphere enrichment')
        servername = transportHN[:5]
    return servername
```

Guarde la secuencia de comandos externa en la carpeta Recursos externos. Para obtener más información, consulte "Panel Recursos" en *HP Universal CMDB – Guía de Administración de Data Flow*. Si agrega esta secuencia de comandos a un paquete, puede usarla también para otros trabajos. Para obtener más información sobre cómo trabajar con el Administrador de paquetes, consulte "Administrador de paquetes" en *HP Universal CMDB – Guía de administración*.

Durante la actualización, el cambio que se realice en una única línea de código se sobrescribe por la nueva versión de la secuencia de comandos de serie, por tanto habrá que sustituir la línea. Sin embargo, la secuencia de comandos externa no se sobrescribe.

Estructura del archivo Jython

El archivo Jython está compuesto por tres partes en un orden específico:

1. Importaciones
2. Función principal - DiscoveryMain
3. Definiciones de funciones (opcional)

A continuación se muestra un ejemplo de una secuencia de comandos Jython:

```
# imports section
from appilog.common.system.types import ObjectStateHolder
from appilog.common.system.types.vectors import ObjectStateHolderVector
# Function definition
def foo:
    # do something
# Main Function
def DiscoveryMain(Framework):
    OSHVResult = ObjectStateHolderVector()
    ## Write implementation to return new result CIs here...
    return OSHVResult
```

Importaciones

Las clases Jython se extienden por espacios de nombres jerárquicos. En la versión 7.0 o posterior, a diferencia de las versiones anteriores, no hay importaciones implícitas y por tanto cada clase que se utiliza debe importarse explícitamente. (Este cambio se realizó por motivos de rendimiento y para permitir una comprensión más sencilla de la secuencia de comandos de Jython al no ocultar ningún detalle necesario).

- Para importar una secuencia de comandos Jython:

```
import logger
```

- Para importar una clase Java:

```
from appilog.collectors.clients import ClientsConsts
```

Función principal: DiscoveryMain

Cada archivo de secuencia de comandos ejecutable de Jython contiene una función principal: DiscoveryMain.

La función DiscoveryMain es la entrada principal a la secuencia de comandos; es la primera función que se ejecuta. La función principal puede llamar a otras funciones que se definen en las secuencias de comandos:

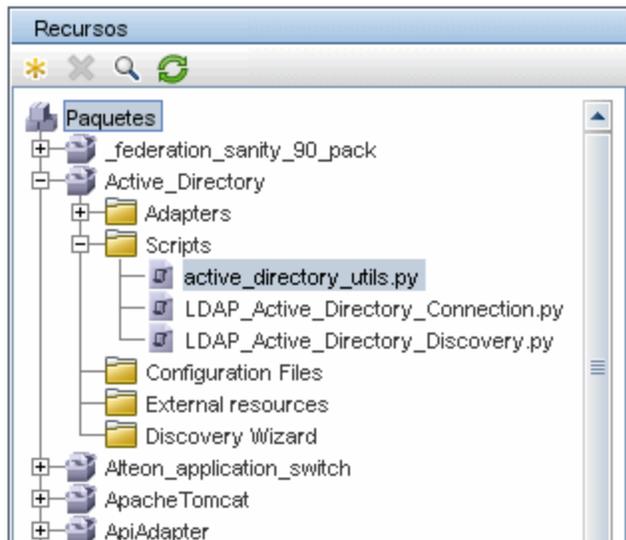
```
def DiscoveryMain(Framework):
```

Se debe especificar el argumento Framework en la definición de la función principal. Este argumento lo utiliza la función principal para recuperar información que se requiere para ejecutar las secuencias de comandos (como información sobre parámetros y CI de activación) y también se pueden usar para informar sobre errores que tienen lugar durante la ejecución de la secuencia de comandos.

Puede crear una secuencia de comandos Jython sin ningún método principal. Dichas secuencias de comandos se usan como secuencias de comandos de biblioteca que se llaman desde otras secuencias de comandos.

Definición de funciones

Cada secuencia de comandos puede contener funciones adicionales que se llaman desde el código principal. Cada una de estas funciones puede llamar a otra función, que existe en la secuencia de comandos actual o en otra (utilice la instrucción `import`). Tenga en cuenta que para usar otra secuencia de comandos, debe agregarla a la sección Scripts del paquete:



Ejemplo de una función que llama a otra función:

En el ejemplo siguiente, el código principal llama al método `doQueryOSUsers(..)` que llama a un método interno `doOSUserOSH(..)`:

```
def doOSUserOSH(name):
    sw_obj = ObjectStateHolder('winosuser')

    sw_obj.setAttribute('data_name', name)
    # return the object
    return sw_obj

def doQueryOSUsers(client, OSHVResult):
    _hostObj = modeling.createHostOSH(client.getIpAddress())
    data_name_mib = '1.3.6.1.4.1.77.1.2.25.1.1,1.3.6.1.4.1.77.1.2.25.1.2, string'
```

```
resultSet = client.executeQuery(data_name_mib)
while resultSet.next():
    UserName = resultSet.getString(2)
    ##### send object #####
    OSUserOSH = doOSUserOSH(UserName)
    OSUserOSH.setContainer(_hostObj)
    OSHVResult.add(OSUserOSH)
def DiscoveryMain/Framework):
    OSHVResult = ObjectStateHolderVector()
    try:
        client = Framework.createClient(Framework.getTriggerCIData(BaseClient.CREDENTIALS_ID))
    except:
        Framework.reportError('Connection failed')
    else:
        doQueryOSUsers(client, OSHVResult)
        client.close()
    return OSHVResult
```

Si esta secuencia de comandos es una biblioteca global relevante a varios adaptadores, puede agregarla a la lista de secuencia de comandos en el archivo de configuración `jythonGlobalLibs.xml`, en lugar de agregarla a cada adaptador (**Administración de adaptador > Panel Recursos > AutoDiscoveryContent > Archivos de configuración**).

Generación de resultados por la secuencia de comandos Jython

Cada secuencia de comandos Jython se ejecuta en un CI de activación específico y finaliza con resultados que devuelve el valor devuelto de la función `DiscoveryMain`.

El resultado de la secuencia de comandos es en realidad un grupo de CI y vínculos que se van a insertar o actualizar en CMDB. La secuencia de comandos devuelve este grupo de CI y vínculos en formato de `ObjectStateHolderVector`.

La clase `ObjectStateHolder` es una forma de representar un objeto o vínculo definido en CMDB. El objeto `ObjectStateHolder` contiene el nombre del CIT y una lista de atributos y sus valores. El objeto `ObjectStateHolderVector` es un vector de instancias de `ObjectStateHolder`.

Sintaxis de ObjectStateHolder

En esta sección se explica cómo generar los resultados de DFM en un modelo de UCMDB.

Ejemplo de configuración de atributos en los CI:

La clase `ObjectStateHolder` describe el gráfico de resultado de DFM. Cada CI y vínculo (relación) se coloca dentro de una instancia de la clase `ObjectStateHolder` en el siguiente ejemplo de código Jython:

```
# siebel application server 1 appServerOSH = ObjectStateHolder('siebelappserver' ) 2
appServerOSH.setStringAttribute('data_name', sblsvrName) 3
appServerOSH.setStringAttribute ('application_ip', ip) 4 appServerOSH.setContainer
(appServerHostOSH)
```

- En la línea 1 se crea un CI de tipo **siebelappserver**.
- En la línea 2 se crea un atributo denominado **data_name** con un valor de **sblsvrName**, que es un conjunto de variables Jython con el valor detectado para el nombre de servidor.
- En la línea 3 se establece un atributo que no es clave actualizado en CMDB.
- En la línea 4 se genera la contención (el resultado es un gráfico). Especifica que este servidor de aplicaciones se encuentra dentro de un host (otra clase `ObjectStateHolder` del ámbito).

Nota: Cada CI que notifica la secuencia de comandos Jython debe incluir valores para todos los atributos clave del tipo CI de los CI.

Ejemplo de relaciones (vínculos):

El siguiente ejemplo de vínculo explica cómo se representa el gráfico:

```
1 linkOSH = ObjectStateHolder('route') 2 linkOSH.setAttribute('link_end1', gatewayOSH) 3
linkOSH.setAttribute('link_end2', appServerOSH)
```

- En la línea 1 se crea el vínculo (que también es de la clase `ObjectStateHolder`. La única diferencia es que `route` es un tipo CI de vínculo).
- En las líneas 2 y 3 se especifican los nodos al final de cada vínculo. Esto se realiza usando los atributos **end1** y **end2** del vínculo que deben especificarse (porque son los atributos clave mínimos de cada vínculo). Los valores de los atributos son instancias de `ObjectStateHolder`. Para obtener más información sobre End 1 y End 2, consulte "Vínculo" en *HP Universal CMDB – Guía de Administración de Data Flow*.

Precaución: un vínculo es direccional. Debería comprobar que los nodos End 1 y End 2 se corresponden con CIT válidos en cada extremo. Si los nodos no son válidos, el objeto de resultado no podrá realizar la validación y no se notificará correctamente. Para obtener más información, consulte "Relaciones de tipo de CI" en *HP Universal CMDB – Guía de modelado*.

Ejemplo de vector (CI de recopilación):

Después de crear objetos con atributos y vínculos con objetos en sus extremos, ahora hay que agruparlos. Para ellos, se agregan a una instancia `ObjectStateHolderVector`, de la forma siguiente:

```
oshvMyResult = ObjectStateHolderVector()  
oshvMyResult.add(appServerOSH)  
oshvMyResult.add(linkOSH)
```

Para obtener más información sobre cómo notificar este resultado compuesto a Framework para que se pueda enviar al servidor de CMDB, consulte el método `sendObjects`.

Cuando el gráfico de resultados se haya unido en una instancia `ObjectStateHolderVector`, debe devolverse a DFM Framework para que se inserte en CMDB. Esto se lleva a cabo devolviendo la instancia `ObjectStateHolderVector` como resultado de la función `DiscoveryMain()`.

Nota: Para obtener más información sobre cómo crear **OSH** para CIT comunes, consulte "[modeling.py](#)" en la página 67.

Enviar grandes cantidades de datos

Es difícil procesar el envío de grandes cantidades de datos (normalmente más de 20 KB) en UCMDB. Los datos de este tamaño deberían dividirse en fragmentos más pequeños antes de enviarlos a UCMDB. A fin de que todos los fragmentos se inserten correctamente en UCMDB, cada fragmento tiene que contener la información de identificación requerida de los CI en el fragmento. Esto es un escenario común al desarrollar integraciones de Jython. El método `sendObjects` se usa para enviar los resultados en fragmentos. Si la secuencia de comandos Jython envía un gran número de resultados (el valor predeterminado es 20.000, pero este valor se puede configurar en el archivo "DataFlowProbe.properties" utilizando la clave **appilog.agent.local.maxTaskResultSize**), debería fragmentar los resultados en función de su topología. Esta fragmentación se debe realizar teniendo en cuenta las reglas de identificación de manera que los resultados se introduzcan correctamente en UCMDB. Si la secuencia de comandos Jython no fragmenta los resultados, la sonda intenta fragmentarlos; sin embargo, esto puede conducir a un rendimiento mediocre si se trata de un conjunto grande de resultados.

Nota: La fragmentación debería usarse para los adaptadores de integración de Jython y no para trabajos normales de detección. Esto es debido a que los trabajos de detección generalmente detectan información concerniente a un activador específico y no envían grandes cantidades de información. Con integraciones de Jython, se detectan grandes cantidades de datos en el activador de la integración.

También es posible utilizar la fragmentación para un número reducido de resultados. En este caso, existe una relación entre los CI de fragmentos distintos y el programador de la secuencia de comandos Jython tiene dos opciones:

- Enviar todo el CI y toda la información de identificación de nuevo en cada segmento que contenga un vínculo al mismo.

- Utilizar el Id. de UCMDB del CI. Para ello, la secuencia de comandos Jython tiene que esperar a que cada fragmento sea procesado en el servidor UCMDB a fin de obtener el Id. de UCMDB. Para habilitar este modo (llamado envío síncrono de resultados), agregue la etiqueta `SendJythonResultsSynchronously` al adaptador. Esta etiqueta garantiza que cuando termine el envío del fragmento, la sonda ya habrá recibido los Id. de UCMDB de los CI del fragmento. El programador del adaptador puede utilizar el Id. de UCMDB para generar el siguiente fragmento. Para usar los Id. de UCMDB, utilice la API de marco `getIdMapping`.

Ejemplo de uso de `getIdMapping`

En el primer fragmento se envían nodos. En el segundo fragmento se envían procesos. El contenedor raíz del proceso es un nodo. En lugar de enviar la totalidad de `objectStateHolder` del nodo en el atributo de proceso `root_container`, puede obtener el Id. de UCMDB del nodo usando la API `getIdMapping` y utilizar solo el Id. de nodo en el atributo de proceso `root_container` para que el fragmento sea menor.

La instancia *Framework*

La instancia de *Framework* es el único argumento que se proporciona en la función principal de la secuencia de comandos Jython. Es una interfaz que se puede usar para recuperar la información que se requiere para ejecutar las secuencias de comandos (como información sobre CI de activación y parámetros del adaptador) y también se puede usar para notificar sobre errores que tienen lugar durante la ejecución de la secuencia de comandos. Para obtener más información, consulte "[Referencia de API de HP Data Flow Management](#)" en la página 39.

El uso correcto de la instancia de *Framework* consiste en pasarla como argumento a todos los métodos que la utilicen.

Ejemplo:

```
def DiscoveryMain(Framework):
    OSHVResult = helperMethod (Framework)
    return OSHVResult
def helperMethod (Framework):
    ....
    probe_name    = Framework.getDestinationAttribute('probe_name')
    ...
    return result
```

En esta sección se describen los usos más importantes de *Framework*:

- "[Framework.getTriggerCIData\(String attributeName\)](#)" en la página siguiente
- "[Framework.createClient\(credentialsId, props\)](#)" en la página siguiente
- "[Framework.getParameter \(String parameterName\)](#)" en la página 49

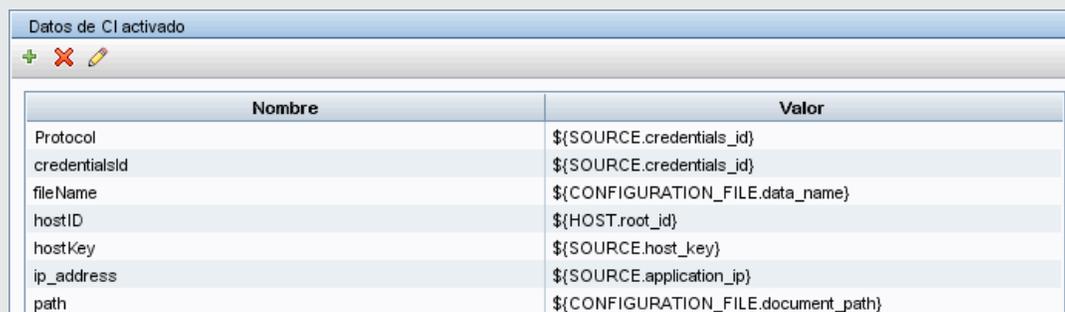
- "[Framework.reportError\(mensaje de cadena\)](#) y [Framework.reportWarning\(mensaje de cadena\)](#)" en la página 50

Framework.getTriggerCIData(String attributeName)

Esta API proporciona el paso intermedio entre los datos del CI de activación definidos en el adaptador y la secuencia de comandos.

Ejemplo de recuperación de información de credenciales:

Solicita la siguiente información sobre datos del CI de activación:



Nombre	Valor
Protocol	\${SOURCE.credentials_id}
credentialsId	\${SOURCE.credentials_id}
fileName	\${CONFIGURATION_FILE.data_name}
hostID	\${HOST.root_id}
hostKey	\${SOURCE.host_key}
ip_address	\${SOURCE.application_ip}
path	\${CONFIGURATION_FILE.document_path}

Para recuperar la información de credenciales de la tarea, utilice esta API:

```
credId = Framework.getTriggerCIData('credentialsId')
```

Framework.createClient(credentialsId, props)

Realice una conexión a un equipo remoto creando un objeto cliente y ejecutando comandos en dicho cliente. Para crear un cliente, recupere la clase `ClientFactory`. El método `getClientFactory()` recibe el tipo de protocolo de cliente solicitado. Las constantes del protocolo se definen en la clase `ClientsConsts`. Para obtener más información sobre credenciales y protocolos admitidos, consulte el manual *HP UCMDB Discovery and Integration Content Guide*.

Ejemplo de creación de una instancia Client para el Id. de credenciales:

Para crear una instancia `Client` para el Id. de credenciales:

```
properties = Properties()
codePage = Framework.getCodePage()
properties.put( BaseAgent.ENCODING, codePage)
client = Framework.createClient(credentialsId ,properties)
```

Ahora puede usar la instancia `Client` para conectarse al equipo o aplicación pertinente.

Ejemplo de creación de un cliente WMI y de la ejecución de una consulta WMI:

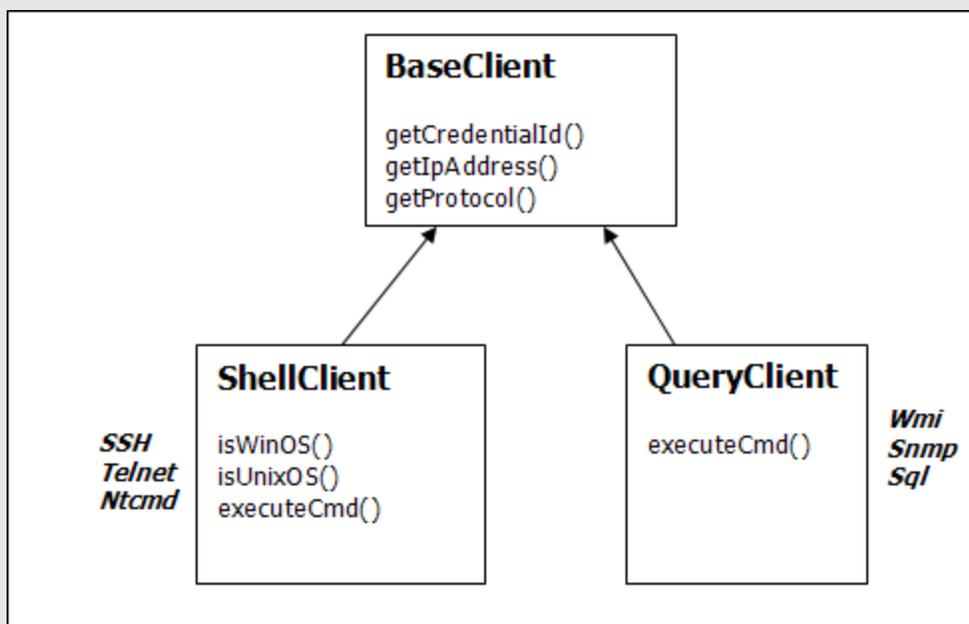
Para crear un cliente WMI y ejecutar una consulta WMI con el cliente:

```
wmiClient = Framework.createClient(credential)
resultSet = wmiClient.executeQuery("SELECT TotalPhysicalMemory
                                   FROM Win32_Logical
                                   MemoryConfiguration")
```

Nota: Para que la API `createClient()` funcione, agregue el siguiente parámetro a los parámetros de datos del CI de activación: **credentialsId = \${SOURCE.credentials_id}** en el panel de datos de CI activados. O bien, puede agregar manualmente el ID de credenciales al llamar a la función:

wmiClient = clientFactory().createClient(credentials_id).

En el diagrama siguiente se muestra la jerarquía de los clientes, con sus API habitualmente admitidos:



Para obtener más información sobre los clientes y sus API compatibles, consulte `BaseClient`, `ShellClient` y `QueryClient` en DFM Framework. Estos archivos están ubicados en la carpeta siguiente:

<directorio raíz de UCMDB>\UCMDBServer\deploy\ucmdb-docs\docs\eng\APIs\DDM_Schema\webframe.html

Framework.getParameter (String parameterName)

Además de recuperar información sobre el CI de activación, con frecuencia necesitará recuperar un valor de los parámetros del adaptador. Por ejemplo:

Parámetros		
Reemplazar	Nombre	Valor
<input checked="" type="checkbox"/>	protocolType	MicrosoftSQLServer

Ejemplo de recuperación de un valor del parámetro protocolType:

Para recuperar el valor del parámetro protocolType desde la secuencia de comandos Jython, utilice la API siguiente:

```
protocolType = Framework.getParameterValue('protocolType')
```

Framework.reportError(mensaje de cadena) y Framework.reportWarning(mensaje de cadena)

Durante la ejecución de una secuencia de comandos pueden ocurrir algunos errores (por ejemplo, fallo de conexión, problemas de hardware, tiempos de espera excedidos). Cuando se detectan estos errores, Framework puede notificar del problema. El mensaje que se notifica llega al servidor y se muestra al usuario.

Ejemplo de una notificación de error y mensaje:

En el ejemplo siguiente se ilustra el uso de la API reportError(<Error Msg>):

```
try:  
    client = Framework.createClient(Framework.getTriggerCIData(BaseClient.CREDENTIALS_ID))  
except:  
    strException = str(sys.exc_info()[1]).strip()  
    Framework.reportError('Connection failed: %s' % strException)
```

Puede utilizar cualquiera de las API (Framework.reportError(String message), Framework.reportWarning(String message)) para informar de un problema. La diferencia entre las dos API es que al informar de un error, la sonda guarda un archivo de registro de comunicaciones con todos los parámetros de la sesión en el sistema de archivos. De esta forma, podrá realizar el seguimiento de la sesión y comprender mejor el error.

Para obtener más información sobre los mensajes de error, consulte "[Mensajes de error](#)" en la [página 68](#).

Encontrar las credenciales correctas (para adaptadores de conexión)

Un adaptador que intenta conectarse a un sistema remoto necesita probar todas las credenciales posibles. Uno de los parámetros necesarios al crear un cliente es el Id. de credenciales. La

secuencia de comandos de credenciales obtiene acceso a posibles conjuntos de credenciales y los prueba uno por uno usando el método `Framework.getAvailableProtocols()`. Cuando un conjunto de credenciales es correcto, el adaptador notifica un objeto de conexión de CI en el host de este CI de activación (con el Id. de credenciales que coincida con la dirección IP) a CMDB. Los adaptadores siguientes pueden usar directamente este CI de objeto de conexión para conectarse al conjunto de credenciales (es decir, los adaptadores no tienen que probar de nuevo todas las credenciales posibles).

Nota: El acceso a datos confidenciales (contraseñas, claves privadas, etc.) está bloqueado para los siguientes tipos de protocolo:

sshprotocol, ntadminprotocol, as400protocol, vmwareprotocol, wmiprotocol, vcloudprotocol, sapjmxprotocol, websphereprotocol, siebelgtwyprotocol, sapprotocol, ldapprotocol, udaprotocol, ntcmdprotocol, snmpprotocol, jbossprotocol, telnetprotocol, powershellprotocol, sqlprotocol, weblogicprotocol

La utilización de estos tipos de protocolo debe hacerse mediante clientes dedicados.

En el ejemplo siguiente se muestra cómo obtener todas las entradas del protocolo SNMP. Obsérvese que aquí el IP se obtiene de los datos del CI de activación (# Get the Trigger CI data values).

La secuencia de comandos de conexión solicita todas las credenciales de protocolos posibles (# Go over all the protocol credentials) y las prueba en un bucle hasta que una sea satisfactoria (`resultVector`). Para obtener más información, consulte la entrada **paradigma de conexión de dos fases** en ["Separación de adaptadores" en la página 24](#).

Ejemplo

```
import logger
import netutils
import sys
import errorcodes
import errorobject

# Java imports
from java.util import Properties
from com.hp.ucmdb.discovery.common import CollectorsConstants
from appilog.common.system.types.vectors import ObjectStateHolderVector
from com.hp.ucmdb.discovery.library.clients import ClientsConsts
from com.hp.ucmdb.discovery.library.scope import DomainScopeManager

TRUE = 1
FALSE = 0

def mainFunction(Framework, isClient, ip_address = None):
    _vector = ObjectStateHolderVector()
    errStr = ''
    ip_domain = Framework.getDestinationAttribute('ip_domain')
```

```
# Get the Trigger CI data values
ip_address = Framework.getDestinationAttribute('ip_address')

if (ip_domain == None):
    ip_domain = DomainScopeManager.getDomainByIp(ip_address, None)

protocols = netutils.getAvailableProtocols(Framework, ClientsConsts.SNMP_PROTOCOL_NAME, ip_address, ip_domain)
if len(protocols) == 0:
    errStr = 'No credentials defined for the triggered ip'
    logger.debug(errStr)
    errObj = errorobject.createError(errorcodes.NO_CREDENTIALS_FOR_TRIGGERED_IP, [ClientsConsts.SNMP_PROTOCOL_NAME], errStr)
    return (_vector, errObj)

connected = 0
# Go over all the protocol credentials
for protocol in protocols:
    client = None
    try:
        try:
            logger.debug('try to get snmp agent for: %s:%s' % (ip_address, ip_domain))
            if (isClient == TRUE):
                properties = Properties()
                properties.setProperty(CollectorsConstants.DESTINATION_DATA_IP_ADDRESS, ip_address)
                properties.setProperty(CollectorsConstants.DESTINATION_DATA_IP_DOMAIN, ip_domain)
                client = Framework.createClient(protocol, properties)
            else:
                properties = Properties()
                properties.setProperty(CollectorsConstants.DESTINATION_DATA_IP_ADDRESS, ip_address)
                client = Framework.createClient(protocol, properties)

            logger.debug('Running test connection queries')
            testConnection(client)
            Framework.saveState(protocol)
            logger.debug('got snmp agent for: %s:%s' % (ip_address, ip_domain))
            isMultiOid = client.supportMultiOid()
            logger.debug('snmp server isMultiOid state=%s' % isMultiOid)

            client.close()
            client = None
```

```
        except:
            if client != None:
                client.close()
                client = None
            logger.debugException('Unexpected SNMP_AGENT Exceptio
n:')
            lastExceptionStr = str(sys.exc_info()[1]).strip()
        finally:
            if client != None:
                client.close()
                client = None

    return (_vector, error)
```

Manejo de excepciones desde Java

Algunas clases Java lanzan una excepción tras un error. Se recomienda capturar la excepción y manejarla; en caso contrario, puede hacer que el adaptador finalice de forma inesperada.

Al capturar una excepción conocida, en la mayoría de los casos debería imprimir su seguimiento de la pila en el registro y enviar un mensaje adecuado a la interfaz de usuario.

Nota: Es muy importante importar la clase de excepción base de Java tal como se muestra en el siguiente ejemplo debido a la presencia de la clase de excepción base en Pitón con el mismo nombre.

```
from java.lang import Exception as JException
try:
    client = Framework.createClient(Framework.getTriggerCIData(BaseClient.CREDENTIALIA
LS_ID))
except JException, ex:
    # procesar solo excepciones java
    Framework.reportError('Error de conexión')
    logger.debugException(str(ex))
    return
```

Si la excepción no es irreuperable y la secuencia de comandos puede continuar, debería omitir la llamada al método `reportError()` y dejar que continúe la secuencia de comandos.

Solución de problemas de migración de Jython Versión 2.1 a 2.5.3

Universal Discovery ahora utiliza la versión de Jython 2.5.3. Todas las secuencias de comandos de serie se han migrado correctamente. Si ha desarrollado sus propias secuencias de comandos Jython antes de esta actualización para su uso por Discovery, puede encontrar los problemas siguientes y tener que realizar las correcciones indicadas.

Nota: Debe ser un programador experimentado en Jython para realizar estos cambios.

Formateado de cadena

- **Mensaje de error:** `TypeError: argumento int obligatorio`
- **Posible causa:** Uso del formateado de cadena en entero decimal de variable de cadena que contiene datos enteros.
- **Código de Jython 2.1 problemático:**

```
variable = "43"  
print "%d" % variable
```

- **Código de Jython 2.5.3 correcto:**

```
variable = "43"  
print "%s" % variable
```

o

```
variable = "43"  
print "%d" % int(variable)
```

Comprobación de tipo de cadena

Es posible que el código siguiente no funcione correctamente si la entrada contiene cadenas Unicode:

- **Código de Jython 2.1 problemático:** `isinstance(unicodeStringVariable, '')`
- **Código de Jython 2.5.3 correcto:** `Isinstance(unicodeStringVariable, basestring)`

La comparación debe hacerse con `basestring` para probar si un objeto es una instancia de `str` o `Unicode`.

Carácter no ASCII en el archivo

- **Mensaje de error:**
`SyntaxError: Non-ASCII character in file 'x', , but no encoding declared; see http://www.python.org/peps/pep-0263.html for details`
- **Código de Jython 2.5.3 correcto:** (añada esto a la primera línea del archivo)

```
# coding: utf-8
```

Importar subpaquetes

- **Mensaje de error:**
`AttributeError: 'module' object has no attribute 'sub_package_name'`

- **Posible causa:** Un subpaquete se importa sin especificar explícitamente el nombre del subpaquete en la instrucción de importación.

- **Código de Jython 2.1 problemático:**

```
import a
print dir(a.b)
```

El subpaquete no se importa explícitamente.

- **Código de Jython 2.5.3 correcto:**

```
import a.b

o

from a import b
```

Cambios de iterador

A partir de Jython 2.2, se utiliza el método `__iter__` para realizar un bucle a través de una recopilación en el alcance de un bloque **for-in**. El iterador debería aplicar el método `next`, que devuelve un elemento apropiado o lanzar el error **StopIteration** si ha llegado al final de la recopilación. Si no se ha aplicado el método `__iter__`, el método `getitem` se usa en su lugar.

Generación de excepciones

- **El método de generar excepciones de Jython 2.1 es obsoleto:**
`raise Exception, 'Error al obtener el contenido del archivo'`
- **Método recomendado de Jython 2.5.3 para generar excepciones:**
`raise Exception('Error al obtener el contenido del archivo')`

Soporte de la localización en los adaptadores Jython

La característica de configuración regional multilingüe permite a DFM funcionar con los distintos idiomas de los sistemas operativos y permitir las personalizaciones apropiadas en tiempo de ejecución.

Esta sección incluye:

- ["Adición de soporte para un nuevo idioma" en la página siguiente](#)
- ["Cambio del idioma predeterminado" en la página 57](#)
- ["Determinación del juego de caracteres para codificación" en la página 57](#)
- ["Definición de un nuevo trabajo para operar con datos localizados" en la página 58](#)
- ["Descodificación de comandos sin una palabra clave" en la página 59](#)

- ["Trabajo con paquetes de recursos" en la página 60](#)
- ["Referencia de API" en la página 61](#)

Adición de soporte para un nuevo idioma

En esta tarea se describe la manera de agregar soporte para un nuevo idioma.

Esta tarea incluye los siguientes pasos:

- ["Agregar un paquete de recursos \(archivos *.properties\)" abajo](#)
- ["Declarar y registrar el objeto de idioma" abajo](#)

1. Agregar un paquete de recursos (archivos *.properties)

Agregue un paquete de recursos en función del trabajo que se va a realizar. En la tabla siguiente se enumeran los trabajos de DFM y el paquete de recursos que se utiliza en cada trabajo:

Trabajo	Nombre básico del paquete de recursos
Monitorización de archivos por shell	langFileMonitoring
Recursos y aplicaciones del host por shell	langHost_Resources_By_TTY, langTCP
Hosts por shell usando NSLOOKUP en servidor DNS	langNetwork
Conexión host por shell	langNetwork
Recopilar datos de red por shell o SNMP	langTCP
Recursos y aplicaciones del host por SNMP	langTCP
Conexión de Microsoft Exchange por NTCMD, topología de Microsoft Exchange por NTCMD	msExchange
Clúster MS por NTCMD	langMsCluster

Para más información acerca de los paquetes, consulte ["Trabajo con paquetes de recursos" en la página 60](#).

2. Declarar y registrar el objeto de idioma

Para definir un nuevo idioma, agregue las dos líneas de código siguientes a la secuencia de comandos `shellutils.py`, que actualmente contiene la lista de todos los idiomas admitidos. La secuencia de comandos se incluye en el paquete `AutoDiscoveryContent`. Para ver la secuencia de comandos, acceda a la ventana Administración de adaptador. Para obtener más información, consulte "Ventana Administración de adaptadores" en *HP Universal CMDB –*

Guía de Administración de Data Flow.

- a. Declare el idioma de la forma siguiente:

```
LANG_RUSSIAN = Language(LOCALE_RUSSIAN, 'rus', ('Cp866', 'Cp1251'), (1049, ), 866)
```

Para obtener más información acerca del idioma de clase, consulte "[Referencia de API](#)" en la [página 61](#). Para obtener más información sobre el objeto Class Locale, consulte <http://java.sun.com/j2se/1.5.0/docs/api/java/util/Locale.html>. Puede usar una configuración regional existente o definir una nueva.

- b. Registre el idioma agregándolo a la colección siguiente:

```
LANGUAGES = (LANG_ENGLISH, LANG_GERMAN, LANG_SPANISH, LANG_RUSSIAN, LANG_JAPANESE)
```

Cambio del idioma predeterminado

Si el idioma del sistema operativo no se puede determinar, se utiliza el predeterminado. El idioma predeterminado se especifica en el archivo **shellutils.py**.

```
#default language for fallback  
DEFAULT_LANGUAGE = LANG_ENGLISH
```

Para cambiar el idioma predeterminado, debe inicializar la variable DEFAULT_LANGUAGE con otro idioma. Para obtener más información, consulte "[Adición de soporte para un nuevo idioma](#)" en la [página precedente](#).

Determinación del juego de caracteres para codificación

El juego de caracteres adecuado para la salida del comando de decodificación se determina en tiempo de ejecución. La solución multilingüe se basa en los hechos y suposiciones siguientes:

1. Es posible determinar el idioma del sistema operativo de manera independiente a la configuración regional, por ejemplo, ejecutando el comando **chcp** en Windows o el comando **locale** en Linux.
2. El idioma-codificación de la relación se conoce bien y se puede definir estáticamente. Por ejemplo, el ruso admite dos de las codificaciones más populares: Cp866 y Windows-1251.
3. Se prefiere un juego de caracteres para cada idioma, por ejemplo, el juego de caracteres preferible para el ruso es Cp866. Esto significa que la mayoría de los comandos producen salidas en esta codificación.
4. La codificación en la que se proporciona la siguiente salida del comando es impredecible, pero es una de las posibles codificaciones de un idioma dado. Por ejemplo, al trabajar con un equipo Windows con una configuración local rusa, el sistema proporciona la salida del comando **ver** en Cp866, pero el comando **ipconfig** se proporciona en Windows-1251.

5. Un comando conocido produce palabras clave conocidas en su salida. Por ejemplo, el comando **ipconfig** contiene la forma traducida de la cadena **IP-Address**. Por tanto la salida del comando **ipconfig** contiene **IP-Address** para el sistema operativo en inglés, **IP-Адрес** para el sistema operativo en ruso, **IP-Adresse** para el sistema operativo en alemán, y así sucesivamente.

Una vez detectado en qué idioma se produce la salida del comando (# 1), los posibles juegos de caracteres están limitados a uno o dos (# 2). Es más, se sabe las palabras clave contenidas en esta salida (# 5).

La solución, por tanto, es descodificar la salida del comando con una de las posibles codificaciones buscando una palabra clave en el resultado. Si se encuentra la palabra clave, se considera correcto el juego de caracteres actual.

Definición de un nuevo trabajo para operar con datos localizados

En esta tarea se describe cómo escribir un nuevo trabajo que puede operar con datos localizados.

Las secuencias de comandos por lo general ejecutan comandos y analizan su salida. Para recibir esta salida de comando de forma correctamente descodificada, hay que usar la API para la clase **ShellUtils**. Para obtener más información, consulte "[Información general de la API de servicio web de HP Universal CMDB](#)" en la página 275.

Por lo general el código adopta la forma siguiente:

```
client = Framework.createClient(protocol, properties)
shellUtils = shellutils.ShellUtils(client)
languageBundle = shellutils.getLanguageBundle ('langNetwork', shellutils.osLanguage, Framework)
strWindowsIPAddress = languageBundle.getString('windows_ipconfig_str_ip_addresses')
ipconfigOutput = shellUtils.executeCommandAndDecode('ipconfig /all', strWindowsIPAddress)
#Do work with output here
```

1. Cree un cliente:

```
client = Framework.createClient(protocol, properties)
```

2. Cree una instancia de la clase **ShellUtils** y agréguela el idioma del sistema operativo. Si no se agrega el idioma, se utiliza el idioma predeterminado (por lo general, el inglés):

```
shellUtils = shellutils.ShellUtils(client)
```

Durante la inicialización del objeto, DFM detecta automáticamente el idioma del equipo y establece la codificación preferible para el objeto Language predefinido. La codificación preferible es la primera instancia que aparece en la lista de codificación.

3. Recupere el paquete de recursos apropiado desde **shellclient** mediante el método **getLanguageBundle**:

```
languageBundle = shellutils.getLanguageBundle ('langNetwork', shellUtils.osLanguage, Framework)
```

4. Recupere una palabra clave del paquete de recursos, adecuado para un comando particular:

```
strWindowsIPAddress = languageBundle.getString('windows_ipconfig_str_ip_address')
```

5. Llame al método **executeCommandAndDecode** y pásese la palabra clave en el objeto **ShellUtils**:

```
ipconfigOutput = shellUtils.executeCommandAndDecode('ipconfig /all', strWindowsIPAddress)
```

También se necesita el objeto `ShellUtils object` para vincular un usuario a la referencia de API (donde este método se describe de forma detallada).

6. Analice la salida de la manera habitual.

Descodificación de comandos sin una palabra clave

El método actual para la localización utiliza una palabra clave para decodificar todas las salidas de los comandos. Para obtener más información, consulte el paso sobre la recuperación de una palabra clave del paquete de recursos en ["Definición de un nuevo trabajo para operar con datos localizados" en la página precedente](#).

Sin embargo, otro enfoque utiliza una palabra clave para decodificar solo la primera salida del comando y, a continuación, descodifica comandos adicionales con el juego de caracteres usado para descodificar el primer comandos. Para ello, utilice los métodos **getCharsetName** y **useCharset** del objeto **ShellUtils**.

El caso de uso regular funciona de la forma siguiente:

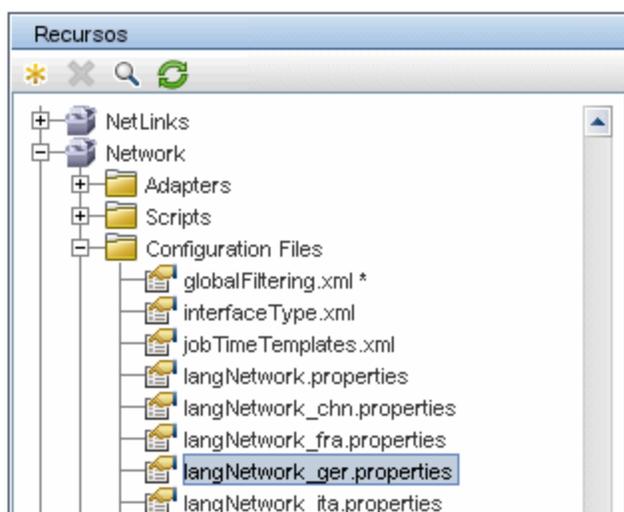
1. Llame al método **executeCommandAndDecode** una vez.
2. Obtenga el nombre del último juego de caracteres usado con el método **getCharsetName**.
3. De manera predeterminada, haga que **shellUtils** utilice este juego de caracteres llamando al método **useCharset** en el objeto **ShellUtils**.
4. Llame al método **execCmd** de **ShellUtils** una o más veces. La salida se devuelve con el juego de caracteres especificado en el paso anterior. No tiene lugar ninguna operación de descodificación adicional.

Trabajo con paquetes de recursos

Un paquete de recursos es un archivo que lleva la extensión `properties` (***.properties**). Se puede considerar un archivo de propiedades con un diccionario que almacena datos en el formato `clave = valor`. Cada fila de un archivo de propiedades contiene una asociación `clave = valor`. La principal funcionalidad de un paquete de recursos es devolver un valor por su clave.

Los paquetes de recursos se localizan en el equipo de sondas:

C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryConfigFiles. Se descargan desde el servidor de UCMDB como cualquier otro archivo de configuración. Se pueden editar, agregar o suprimir en la ventana Recursos. Para obtener más información, consulte "Panel Archivo de configuración" en *HP Universal CMDB – Guía de Administración de Data Flow*.



Al detectar un destino, DFM por lo general tiene que analizar el texto desde una salida de comando o contenido de archivo. Este análisis suele estar basado en una expresión regular. Los distintos idiomas requieren el uso de diferentes expresiones regulares para su análisis. Para que se escriba código una vez para todos los idiomas, se deben extraer todos los datos específicos del idioma en paquetes de recursos. Hay un paquete de recursos para cada idioma. (Aunque es posible que un paquete de recursos contenga datos para distintos idiomas, en DFM cada paquete de recursos contiene datos solamente para un idioma).

La misma secuencia de comandos Jython no incluye datos específicos del idioma, codificados de forma rígida (por ejemplo, expresiones regulares específicas del idioma). La secuencia de comandos determina el idioma del sistema remoto, carga el paquete de recursos apropiado y obtiene todos los datos específicos del idioma por clave específica.

En DFM, los paquetes de recursos adoptan un formato de nombre específico: `<nombre_básico>_<identificador_idioma>.properties`, por ejemplo, `langNetwork_spa.properties`. (El paquete de recursos predeterminado adopta el formato siguiente: `<nombre_básico>.properties`, por ejemplo, `langNetwork.properties`.)

El formato del `nombre_básico` refleja el propósito previsto de este paquete. Por ejemplo, **langMsCluster** significa el paquete de recursos contiene recursos específicos del idioma utilizados por los trabajos de MS Cluster.

El formato `identificador_idioma` es un acrónimo de tres letras utilizado para identificar el idioma. Por ejemplo, `rus` representa el idioma ruso y `ger` el idioma alemán. Este identificador de idioma se incluye en la declaración del objeto `Language`.

Referencia de API

Esta sección incluye:

- ["Clase `Language`" abajo](#)
- ["Método `executeCommandAndDecode`" en la página siguiente](#)
- ["Método `getCharsetName`" en la página siguiente](#)
- ["Método `useCharset`" en la página siguiente](#)
- ["Método `getLanguageBundle`" en la página siguiente](#)
- ["Campo `osLanguage`" en la página 63](#)

Clase `Language`

Esta clase encapsula información sobre el idioma, como el postfijo del paquete de recursos, la codificación posible, entre otra.

Campos

Nombre	Descripción
<code>configuración regional</code>	Objeto Java que representa la configuración regional.
<code>bundlePostfix</code>	El postfijo del paquete de recursos. Este postfijo se utiliza en los nombres de los archivos de paquetes de recursos para identificar el idioma. Por ejemplo, el paquete <code>langNetwork_ger.properties</code> incluye el postfijo del paquete <code>ger</code> .
<code>charsets</code>	Los juegos de caracteres utilizados para codificar este idioma. Cada idioma puede tener varios juegos de caracteres. Por ejemplo, el idioma ruso se suele codificar con la codificación <code>Cp866</code> y <code>Windows-1251</code> .
<code>wmiCodes</code>	Lista de códigos WMI utilizados por el sistema operativo Microsoft Windows para identificar el idioma. Todos los códigos posibles se enumeran en http://msdn.microsoft.com/en-us/library/aa394239(VS.85).aspx (sección <code>OSLanguage</code>). Uno de los métodos para identificar el idioma del sistema operativo es consultar el sistema operativo de la clase WMI para la propiedad <code>OSLanguage</code> .
<code>codepage</code>	Página de códigos utilizada con un idioma específico. Por ejemplo, se utiliza <code>866</code> para los equipos en ruso y <code>437</code> para los equipos en inglés. Uno de los métodos para identificar el idioma del sistema operativo es recuperar su página de códigos predeterminada (por ejemplo, por el comando <code>OSLanguage</code>).

Método `executeCommandAndDecode`

Este método se ha previsto para su uso con las secuencias de comandos Jython de lógica de negocio. Encapsula la operación de descodificación y devuelve una salida descodificada de comandos.

Argumentos

Nombre	Descripción
<code>cmd</code>	El comando real que se va a ejecutar.
<code>keyword</code>	La palabra clave que se va a usar en la operación de descodificación.
<code>framework</code>	El objeto Framework pasado en todas las secuencias de comandos Jython ejecutables de DFM.
<code>timeout</code>	El tiempo de espera agotado del comando.
<code>waitForTimeout</code>	Especifica si el cliente debe esperar a que se agote el tiempo de espera.
<code>useSudo</code>	Especifica si se debe usar <code>sudo</code> (solo relevante para clientes de equipos UNIX).
<code>language</code>	Permite especificar directamente el idioma en lugar de detectarlo automáticamente.

Método `getCharsetName`

Este método devuelve el nombre del juego de caracteres últimamente utilizado.

Método `useCharset`

Este método establece el juego de caracteres en la instancia `ShellUtils`, que utilice este juego de caracteres para la descodificación de datos inicial.

Argumentos

Nombre	Descripción
<code>charsetName</code>	El nombre del juego de caracteres, por ejemplo, <code>windows-1251</code> o <code>UTF-8</code> .

Consulte también ["Método `getCharsetName`" arriba](#)

Método `getLanguageBundle`

Este método debería utilizarse para obtener el paquete de recursos correcto. Reemplaza a la API siguiente:

```
Framework.getEnvironmentInformation().getBundle(...)
```

Argumentos

Nombre	Descripción
baseName	El nombre del paquete sin el sufijo de idioma, por ejemplo, langNetwork.
language	El objeto de idioma. Aquí debe pasarse <code>ShellUtils.osLanguage</code> .
framework	El objeto común Framework que se pasa en todas las secuencias de comandos Jython ejecutables de DFM.

Campo `osLanguage`

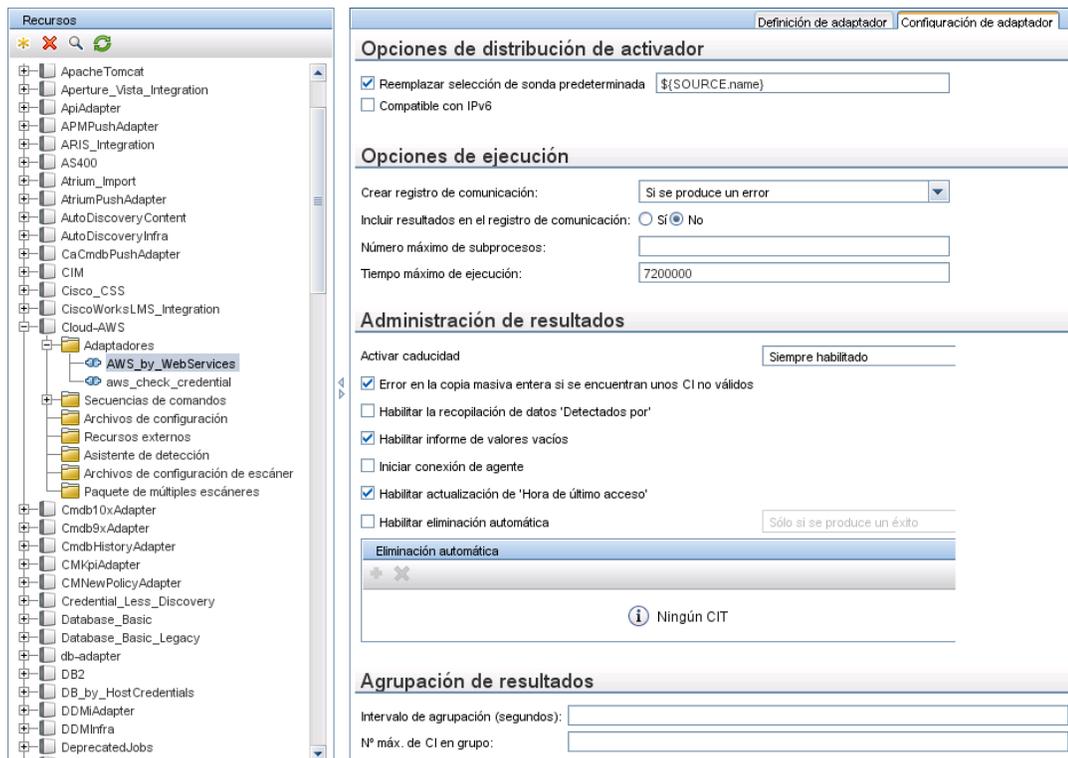
Este campo contiene un objeto que representa el idioma.

Registro de código DFM

Puede ser muy útil registrar una ejecución completa, incluyendo todos los parámetros, por ejemplo, al depurar y probar el código. En esta tarea se describe cómo registrar una ejecución completa con todas las variables relevantes. Es más, puede ver información de depuración adicional que por lo general no se imprime en los archivos de registros, incluso en el nivel de depuración.

Para registrar código DFM:

1. Acceda a **Administración de Data Flow > Universal Discovery**. Haga clic con el botón derecho en el trabajo cuya ejecución debe registrarse y seleccione **Ir a adaptador** para abrir la aplicación Administración de adaptador.
2. Busque el panel **Opciones de ejecución** en la ficha **Configuración de adaptador**, como se muestra a continuación.



3. Cambie el cuadro **Crear registro de comunicación** a **Siempre**. Para obtener más información acerca de cómo establecer las opciones de registro, consulte "Panel Opciones de ejecución" en *HP Universal CMDB – Guía de Administración de Data Flow*.

El ejemplo siguiente es el archivo de registro XML que se crea cuando se ejecuta el trabajo **Conexión de host por shell** y el cuadro **Crear registro de comunicación** se establece en **Siempre** o **Si se produce un error**:

Nombre del trabajo	Datos de CI de activación	
<pre>- <execution jobId="Host Connection by Shell" destinationid="0e9787433d65e4a68839bfa8b224c92d"> - <destination> <destinationData name="ip_domain">DefaultDomain</destinationData> <destinationData name="hostId" /> <destinationData name="ip_address">16.59.63.34</destinationData> <destinationData name="id">0e9787433d65e4a68839bfa8b224c92d</destinationData> </destination></pre>		

En el ejemplo siguiente se muestra el mensaje y los parámetros del seguimiento de la pila:

Seguimiento de la pila	
<pre>- <exec start="18:41:55" duration="2062" type="ssh" credentialsId="f464999bdf5a1e1407b479b6f730d5b"> <cmd>[CDATA: client_connect]</cmd> <result IS_NULL="Y" /> - <error class="com.hp.ucmdb.discovery.probe.services.dynamic.agents.SSHAgentException"> <message>[CDATA: Failed to connect: Error connecting: Connection refused: connect]</message> <stacktrace> <frame class="com.hp.ucmdb.discovery.probe.services.dynamic.agents.SSHAgent" method="connect" file="SSHAgent.java"> <frame class="com.hp.ucmdb.discovery.probe.clients.shell.SSHClient" method="createWrapper" file="SSHClient.java"> <frame class="com.hp.ucmdb.discovery.probe.clients.BaseClient" method="initPrivate" file="BaseClient.java"></pre>	

Bibliotecas y utilidades Jython

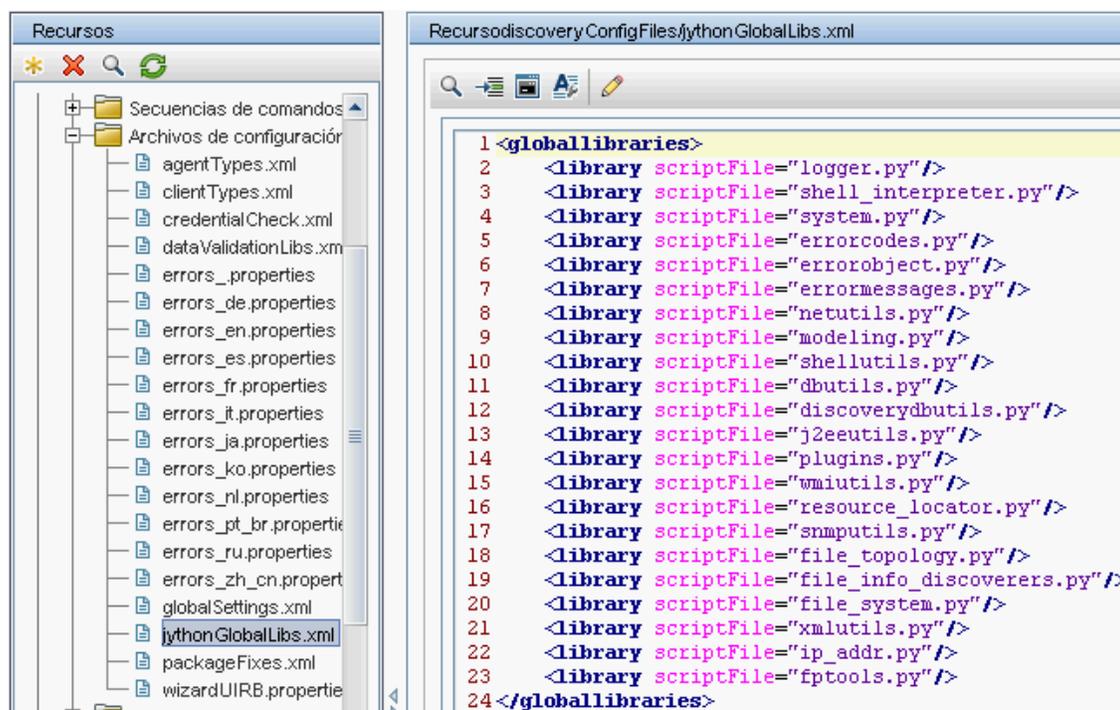
En los adaptadores se utilizan con mucha frecuencia varias secuencias de comandos de utilidades. Estas secuencias de comandos forman parte del paquete `AutoDiscovery` y están localizadas en: `C:\hp\UCMDB\DataFlowProbe\runtime\probeManager\discoveryScripts` con las demás secuencias de comandos que se han descargado de la sonda.

Nota: La carpeta `discoveryScript` se crea dinámicamente cuando la sonda comienza a trabajar.

Para usar una de las secuencias de comandos de utilidad, agregue la siguiente línea de importación a la sección de importación de la secuencia de comandos:

```
import <nombre de secuencia de comandos>
```

La biblioteca `AutoDiscovery Python` contiene secuencias de comandos de la utilidad `Jython`. Estas secuencias de comandos de la biblioteca se consideran una biblioteca externa de DFM. Se definen en el archivo `jythonGlobalLibs.xml` (ubicado en la carpeta **Configuration Files**).



Todas las secuencias de comandos que aparecen en el archivo `jythonGlobalLibs.xml` se cargan de manera predeterminada en el inicio de la sonda, por tanto no es preciso usarlas explícitamente en la definición del adaptador.

Esta sección incluye los siguientes temas:

- ["logger.py" abajo](#)
- ["modeling.py" en la página siguiente](#)
- ["netutils.py" en la página siguiente](#)
- ["shellutils.py" en la página siguiente](#)

logger.py

La secuencia de comandos **logger.py** contiene utilidades de registro y funciones de la aplicación auxiliar para la generación de informes de errores. Puede llamar a sus API de depuración, información y error para escribir en los archivos de registro. Los mensajes de registro se registran en **C:\hp\UCMDB\DataFlowProbe\runtime\log**.

Los mensajes se introducen en el archivo de registro de acuerdo con el nivel de depuración definido por el anexador PATTERNS_DEBUG del archivo

C:\hp\UCMDB\DataFlowProbe\conf\log\probeMgrLog4j.properties. (De manera predeterminada, el nivel es DEBUG). Para obtener más información, consulte ["Niveles de gravedad de errores" en la página 71](#).

```
#####  
#####          PATTERNS_DEBUG log          #####  
#####  
#####  
log4j.category.PATTERNS_DEBUG=DEBUG, PATTERNS_DEBUG  
log4j.appender.PATTERNS_DEBUG=org.apache.log4j.RollingFileAppender  
log4j.appender.PATTERNS_DEBUG.File=C:\hp\UCMDB\DataFlowProbe\runtime\log\pro  
beMgr-patternsDebug.log  
log4j.appender.PATTERNS_DEBUG.Append=true  
log4j.appender.PATTERNS_DEBUG.MaxFileSize=15MB  
log4j.appender.PATTERNS_DEBUG.Threshold=DEBUG  
log4j.appender.PATTERNS_DEBUG.MaxBackupIndex=10  
log4j.appender.PATTERNS_DEBUG.layout=org.apache.log4j.PatternLayout  
log4j.appender.PATTERNS_DEBUG.layout.ConversionPattern=%d [%-5p] [%t] - %m  
%n  
log4j.appender.PATTERNS_DEBUG.encoding=UTF-8
```

Los mensajes de información y error también aparecen en la consola del símbolo del sistema.

Hay dos conjuntos de API:

- `logger.<debug/info/warn/error>`
- `logger.<debugException/infoException/warnException/errorException>`

El primer conjunto envía la concatenación de todos sus argumentos de cadena al nivel de registro apropiado y el segundo conjunto envía la concatenación así como envía el seguimiento de la pila de la última excepción lanzada para proporcionar más información, por ejemplo:

```
logger.debug('found the result')  
logger.errorException('Error in discovery')
```

modeling.py

La secuencia de comandos **modeling.py** contiene API para crear hosts, IP, CI de proceso, etc. Estas API permiten la creación de objetos comunes y facilitan la lectura del código. Por ejemplo:

```
ipOSH= modeling.createIpOSH(ip)  
host = modeling.createHostOSH(ip_address)  
member1 = modeling.createLinkOSH('member', ipOSH, networkOSH)
```

netutils.py

La biblioteca **netutils.py** se utiliza para recuperar información de la red y de TCP, como la recuperación de nombres del sistema operativo, comprobación de la validez de la dirección MAC, comprobación de la validez de la dirección IP, etc. Por ejemplo:

```
dnsName = netutils.getHostName(ip, ip)  
isValidIp = netutils.isValidIp(ip_address)  
address = netutils.getHostAddress(hostName)
```

shellutils.py

La biblioteca **shellutils.py** proporciona una API para ejecutar comandos de shell y recuperar el estado final de un comando ejecutado, y permite la ejecución de varios comandos en función de dicho estado final. La biblioteca se inicializa con un cliente shell y utiliza el cliente para ejecutar comandos y recuperar resultados. Por ejemplo:

```
ttyClient = Framework.createClient(Framework.getTriggerCIData(BaseClient.CREDENTIALS_ID), Props)  
clientShUtils = shellutils.ShellUtils(ttyClient)  
if (clientShUtils.isWinOs()):  
    logger.debug ('discovering Windows..')
```

Capítulo 3: Mensajes de error

Este capítulo incluye:

Información general de los mensajes de error	68
Convenciones para la escritura de errores	68
Niveles de gravedad de errores	71

Información general de los mensajes de error

Durante la detección, se pueden revelar muchos errores, por ejemplo, fallos de conexión, problemas de hardware, excepciones, tiempos de espera agotados, entre otros. Estos errores se muestran en la ventana Universal Discovery siempre que el flujo de detección regular no se realiza correctamente. Puede profundizar en el CI de activación que causó el problema para ver el mismo mensaje de error.

DFM diferencia entre errores que se pueden omitir a veces (por ejemplo, un host al que no se puede tener acceso) y errores que hay que resolver (por ejemplo, problemas de credenciales o configuración que falta o archivos DLL). Es más, DFM notifica los errores una vez, incluso si el mismo error tiene lugar en sucesivas ejecuciones, y notifica un error aunque solo tenga lugar una vez.

Al crear un paquete, puede añadir mensajes apropiados como recursos al paquete. Durante el despliegue del paquete, los mensajes también se despliegan en la ubicación correcta. Los mensajes deben cumplir con las convenciones, tal como se describe en "[Convenciones para la escritura de errores](#)" abajo.

DFM admite mensajes de error en varios idiomas. Puede traducir los mensajes que escriba para que aparezcan en el idioma local.

Para obtener información detallada sobre la búsqueda de errores, consulte Progreso y resultados de la detección en la *HP Universal CMDB – Guía de Administración de Data Flow*.

Para obtener más información sobre cómo establecer los registros de comunicación, consulte "Panel Opciones de ejecución" en la *HP Universal CMDB – Guía de Administración de Data Flow*.

Convenciones para la escritura de errores

- Los errores se identifican por un código del mensaje de error y una matriz de argumentos (**int**, **String[]**). Una combinación de un código de mensaje y una matriz de argumentos define un error específico. La matriz de parámetros puede ser un valor nulo.
- Cada código de error se asigna a un **mensaje breve**, que es una cadena fija, y un **mensaje detallado**, que es una cadena de plantilla que contiene cero o más argumentos. Se asume la correspondencia entre el número de argumentos de la plantilla y el número real de parámetros.

Ejemplo de código de mensaje de error:

10234 puede representar un error con el mensaje breve:

```
Error de conexión
```

y el mensaje detallado:

```
No se puede conectar al protocolo {0} debido a un tiempo de espera agotado de {1} mseg
```

donde

{0} = el primer argumento: un nombre de protocolo

{1} = el segundo argumento: la duración del tiempo de espera agotado en mseg

Esta sección incluye también los siguientes temas:

- ["Contenido de los archivos de propiedades" abajo](#)
- ["Archivo de propiedades de los mensajes de error" abajo](#)
- ["Convenciones de nomenclatura de configuración regional" en la página siguiente](#)
- ["Códigos de mensajes de error" en la página siguiente](#)
- ["Errores de contenido sin clasificar" en la página 71](#)
- ["Cambios en Framework" en la página 71](#)

Contenido de los archivos de propiedades

Un archivo de propiedades debería contener dos claves para cada código de mensaje de error. Por ejemplo, para el error 45:

- **DDM_ERROR_MESSAGE_SHORT_45**. Descripción breve del error.
- **DDM_ERROR_MESSAGE_LONG_45**. Descripción larga del error (puede contener parámetros, por ejemplo, **{0}**,**{1}**).

Archivo de propiedades de los mensajes de error

Un archivo de propiedades contiene una asignación entre un código de mensaje de error y dos mensajes (breve y detallado).

Cuando se despliega un archivo de propiedades, sus datos se fusionan con datos existentes, se añaden los nuevos códigos de mensaje mientras que se reemplazan los códigos de mensajes antiguos.

Los archivos de propiedades de infraestructura forman parte del paquete **AutoDiscoveryInfra**.

Convenciones de nomenclatura de configuración regional

- Para la configuración regional predeterminada: **<file name>.properties.errors**
- Para una configuración regional específica: **<file name>_xx.properties.errors**

donde **xx** es la configuración regional (por ejemplo, **infraerr_fr.properties.errors** o **infraerr_en_us.properties.errors**).

Códigos de mensajes de error

Los siguientes códigos de error se incluyen de manera predeterminada con HP Universal CMDB. Puede agregar sus propios mensajes de error a esta lista.

Nombre de error	Código de error	Descripción
Interno	100-199	Se resuelven sobre todo en excepciones producidas durante las ejecuciones de las secuencias de comandos Jython
Conexión	200-299	Error en la conexión, sin agente en el equipo destino, destino no alcanzable, etc.
Relacionado con credenciales	300-399	Permiso denegado, intento de conexión bloqueado debido a la falta de credenciales
Tiempo de espera	400-499	Tiempo de espera agotado durante la conexión/comando
Comportamiento inesperado o no válido	500-599	Archivos de configuración que faltan, interrupciones inesperadas, etc.
Recuperación de información	600-699	Falta información en equipos de destino, error al solicitar información al agente, etc.
Relacionado con recursos	700-799	Errores relacionados con falta de memoria o clientes que no se han liberado correctamente
Análisis	800-899	Error al analizar texto
Codificación	900	Error en la entrada, codificación no admitida
Relacionado con SQL	901-903, 924	Errores recibidos en operaciones SQL
Relacionado con HTTP	904-909	Errores generados durante las conexiones HTTP, analizados desde códigos de error HTTP.
Específico de aplicación	910-923	Error notificado debido a problemas específicos de la aplicación, por ejemplo, versión LSOF incorrecta, no se encontraron gestores de cola, etc.

Errores de contenido sin clasificar

Para admitir contenido antiguo sin causar una regresión, la aplicación y los métodos relevantes de SDK gestionan errores del código de mensaje 100 (es decir, un error de secuencia de comandos sin clasificar) de manera diferente.

Estos errores no están agrupados (es decir, no se consideran errores del mismo tipo) por su código de mensaje, sino que se agrupan por el contenido del mensaje. Es decir, si una secuencia de comandos notifica un error por métodos antiguos y obsoletos (con una cadena de mensajes y sin un código de error), todos los mensajes reciben el mismo código de error, pero en la aplicación o en métodos SDK relevantes, los distintos mensajes se muestran como errores diferentes.

Cambios en Framework

(com.hp.ucmdb.discovery.library.execution.BaseFramework)

Los métodos siguientes se agregan a la interfaz:

- `void reportError(int msgCode, String[] params);`
- `void reportWarning(int msgCode, String[] params);`
- `void reportFatal(int msgCode, String[] params);`

Los siguientes métodos antiguos todavía se admiten por compatibilidad con versiones anteriores, pero se marcan como obsoletos:

- `void reportError(String message);`
- `void reportWarning (String message);`
- `void reportFatal (String message);`

Niveles de gravedad de errores

Cuando un adaptador finaliza la ejecución en un CI de activación, devuelve un estado. Si no se notifica de ningún error o advertencia, el estado es **Success** (Correcto).

Los niveles de gravedad se enumeran aquí desde el ámbito más restringido al más amplio:

Errores graves

Este nivel notifica errores graves como un problema con la infraestructura, archivos DLL que faltan o excepciones:

- Error al generar la tarea (no se encuentra la sonda, no se encuentran las variables, etc.)
- No es posible ejecutar la secuencia de comandos
- El procesamiento de los resultados produce un error en el servidor y los datos no se escriben en el CMDB

Errores

Este nivel notifica problemas que hacen que DFM no recupere los datos. Revise estos errores ya que por lo general requieren que se tome alguna medida (por ejemplo, aumentar el tiempo de espera, cambiar un intervalo, cambiar un parámetro, agregar otra credencial de usuario, etc.).

- En aquellos casos donde pueda ayudar la intervención del usuario, se informa de un error, ya sea un problema de credenciales o redes que pueda necesitar información adicional. (No son errores de detección sino de configuración).
- Error interno, por lo general debido a un comportamiento inesperado del equipo o aplicación detectado, por ejemplo, archivos de configuración que faltan, entre otros.

Advertencias

Cuando una ejecución se realiza de manera correcta pero pueden surgir problemas no graves que deberían ser tenidos en cuenta, DFM marca la gravedad como **Advertencia**. Puede examinar esos CI para ver si faltan datos antes de comenzar una sesión de depuración más detallada. La **advertencia** puede incluir mensajes sobre la falta de un agente instalado en un host remoto, o que datos no válidos han causado que un atributo se calcule incorrectamente.

- Agente de conexión que falta (SNMP, WMI)
- La detección se realiza correctamente, pero no se detecta toda la información disponible

Capítulo 4: Desarrollo de adaptadores de bases de datos genéricas

Este capítulo incluye:

Información general de los adaptadores de bases de datos genéricas	74
Consultas TQL para el adaptador de bases de datos genéricas	74
Conciliación	75
Hibernate como proveedor de JPA	76
Preparar la creación del adaptador	78
Preparación del paquete de adaptadores	83
Configuración del adaptador - Método mínimo	86
Configuración del adaptador - Método avanzado	91
Implementación de un complemento	96
Despliegue del adaptador	99
Edición del adaptador	99
Crear un punto de integración	99
Creación de una vista	100
Cálculo de los resultados	100
Visualización de resultados	100
Ver informes	100
Habilitación de archivos de registro	101
Uso de Eclipse para asignar entre atributos CIT y tablas de bases de datos	101
Archivos de configuración de adaptadores	109
Convertidores de serie	134
Complementos	140
Ejemplos de configuración	140
Archivos de registro del adaptador	149
Referencias externas	151
Solución de problemas y limitaciones	151

Información general de los adaptadores de bases de datos genéricas

El propósito de la plataforma de adaptadores de bases de datos genéricas es crear adaptadores que puedan integrarse con sistemas de gestión de bases de datos relacionales (RDBMS) y ejecutar consultas de TQL y trabajos de relleno en la base de datos. Los RDBMS admitidos por el adaptador de bases de datos genéricas son Oracle, Microsoft SQL Server y MySQL.

Esta versión de la implementación de adaptadores de bases de datos está basada en un estándar JPA (API de persistencia Java) con la biblioteca Hibernate ORM como proveedor de persistencia.

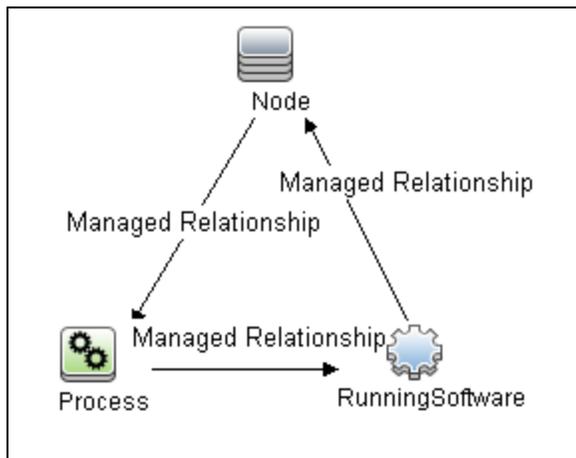
Consultas TQL para el adaptador de bases de datos genéricas

Para los trabajos de relleno, cada diseño necesario de un CI debe comprobarse en el cuadro de diálogo Configuración de diseño en Modeling Studio. Para obtener más información, consulte "Cuadro de diálogo Propiedades de nodo de consulta/relación" en *HP Universal CMDB – Guía de modelado*. Es importante tener en cuenta que un CI puede necesitar un atributo para ser identificado y sin esos atributos, el CI no se podrá agregar a UCMDB.

Hay las limitaciones siguientes en las consultas TQL calculadas únicamente por el adaptador de bases de datos genéricas:

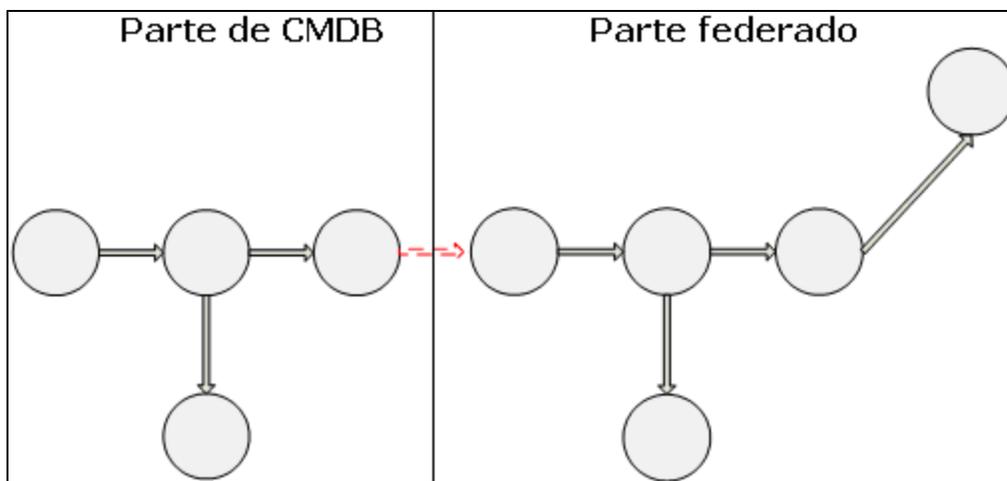
- no se admiten los subgráficos
- no se admiten las relaciones compuestas
- no se admiten los ciclos o partes de ciclos

La siguiente consulta TQL es un ejemplo de un ciclo:



- No se admite el diseño de funciones.

- No se admite la cardinalidad 0..0.
- No se admite la relación Join.
- No se admiten las condiciones del calificador.
- Para conectar dos CI, debe haber una relación en forma de tabla o clave externa en el origen de la base de datos externa.



Conciliación

La conciliación se realiza como parte del cálculo de TQL en el adaptador. Para que tenga lugar la conciliación, el CMDB se asigna a una entidad federada llamada CIT de conciliación.

Asignación. Cada atributo de CMDB se asigna a una columna en el origen de datos.

Aunque la asignación se realiza directamente, también se admiten las funciones de transformación en los datos asignados. Puede agregar nuevas funciones mediante el código Java (por ejemplo, minúscula, mayúscula). El propósito de estas funciones es habilitar las conversiones de valor (valores almacenados en CMDB en un formato y en la base de datos federada en otro formato).

Nota:

- Para conectar CMDB y el origen de la base de datos externa, debe haber una asociación adecuada en la base de datos. Para obtener más información, consulte ["Requisitos previos" en la página 78](#).
- También se admite la conciliación con el Id. de CMDB
- También se admite la conciliación con el Id. global.

Hibernate como proveedor de JPA

Hibernate es una herramienta de asignación relacional de objetos (OR), que permite la asignación de clases Java a tablas en varios tipos de bases de datos relacionales (por ejemplo, Oracle y Microsoft SQL Server). Para obtener más información, consulte ["Limitaciones funcionales" en la página 151](#).

En una asignación elemental, cada clase Java está asignada a una única tabla. Una asignación más avanzada permite la asignación de herencia (como ocurre en la base de datos de CMDB).

Otras funciones admitidas incluyen la asignación de una clase a varias tablas, soporte para recopilaciones y asociaciones de tipos uno-a-uno, uno-a-varios y varios-a-uno. Para obtener más información, consulte ["Asociaciones" en la página 78](#) a continuación.

En el contexto que nos ocupa, no hay necesidad de crear clases Java. La asignación se define desde los CIT del modelo de clase de CMDB a las tablas de la base de datos.

Esta sección incluye también los siguientes temas:

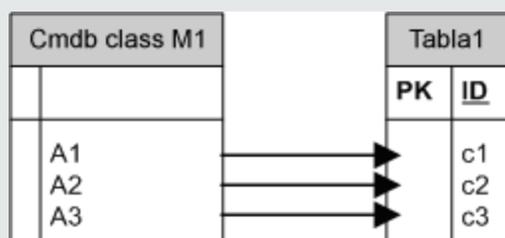
- ["Ejemplos de asignación relacional de objetos" abajo](#)
- ["Asociaciones" en la página 78](#)
- ["Facilidad de uso" en la página 78](#)

Ejemplos de asignación relacional de objetos

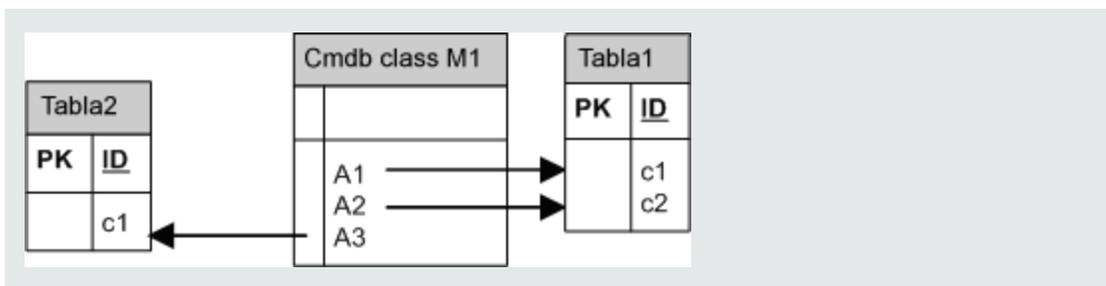
En los ejemplos siguientes se describe la asignación relacional de objetos:

Ejemplo de 1 clase de CMDB asignada a una tabla de la base de datos:

La clase M1, con atributos A1, A2 y A3, se asigna a la tabla 1 columnas c1, c2 y c3. Esto significa que cualquier instancia de M1 tiene una fila correspondiente en la tabla 1.

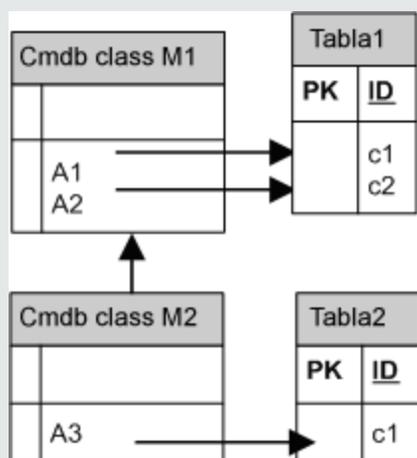


Ejemplo de 1 clase de CMDB asignada a dos tablas de la base de datos:



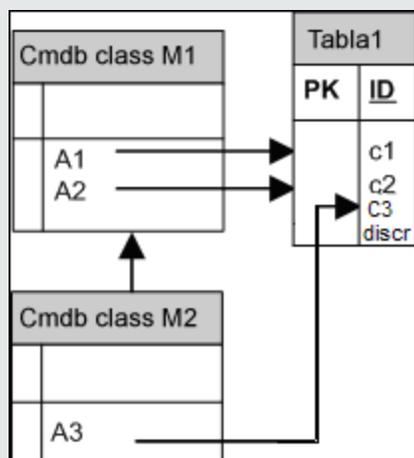
Ejemplo de herencia:

Este caso se utiliza en CMDB, donde cada clase tiene su propia tabla de base de datos.



Ejemplo de herencia de una única tabla con discriminador:

Una jerarquía completa de clases se asigna a una única tabla de la base de datos, cuyas columnas constan de un superconjunto de todos los atributos de las clases asignadas. La tabla también contiene una columna adicional (Discriminator), cuyo valor indica qué clase específica debería asignarse a esta entrada.



Asociaciones

Hay tres tipos de asociaciones: uno-a-varios, varios-a-uno y varios-a-varios. Para realizar la conexión entre los distintos objetos de la base de datos, debe definirse una de estas asociaciones usando una columna de clave externa (para el caso uno-a-varios) o una tabla de asignación (para el caso varios-a-varios).

Facilidad de uso

Como el esquema JPA es muy amplio, se incluye un archivo XML simplificado para facilitar la definición de asociaciones.

El caso de uso para utilizar este archivo XML es como sigue: Los datos federados se modelan en una clase federada. Esta clase tiene relaciones varios-a-uno en una clase de CMDB no federada. Además, solo hay un posible tipo de relación entre la clase federada y la clase no federada.

Preparar la creación del adaptador

Esta tarea describe las preparaciones que son necesarias para crear un adaptador.

Nota: Puede ver ejemplos del adaptador de bases de datos genéricas en la API de UCMDB. Específicamente, la muestra del adaptador DDMi contiene un archivo **orm.xml** complicado, así como implementaciones para algunas interfaces de complementos.

Esta tarea incluye los siguientes pasos:

- ["Requisitos previos" abajo](#)
- ["Crear un tipo de CI" en la página 80](#)
- ["Crear una relación" en la página 80](#)

1. Requisitos previos

Para validar que puede utilizar el adaptador de la base de datos con su base de datos, compruebe lo siguiente:

- Las clases de conciliación y sus atributos (también conocidos como multinodos) están en la base de datos. Por ejemplo, si la conciliación se ejecuta por nombre de nodo, compruebe que hay una tabla que contiene una columna con nombres de nodos. Si la conciliación se ejecuta de acuerdo con el nodo `cmdb_id`, compruebe que hay una columna con unos Id. de CMDB que coincidan con los Id. de CMDB de los nodos de CMDB. Para más información sobre la conciliación, consulte ["Conciliación" en la página 75](#).

ID	NAME	IP_ADDRESS
31	BABA	16.59.33.60
33	ext3.devlab.ad	16.59.59.116

ID	NAME	IP_ADDRESS
46	LABM1MAM15	16.59.58.188
72	cert-3-j2ee	16.59.57.100
102	labm1sun03.devlab.ad	16.59.58.45
114	LABM2PCOE73	16.59.66.79
116	CUT	16.59.41.214
117	labm1hp4.devlab.ad	16.59.60.182

- Para correlacionar dos CIT con una relación, debe haber datos de correlación entre las tablas de CIT. La correlación puede ser por columna de clave externa o por tabla de asignación. Por ejemplo, para establecer una correlación entre el nodo y el ticket, tiene que haber una tabla de tickets que contenga el Id. del nodo, una columna en la tabla de nodos con el Id. de ticket que está conectado a él o una tabla de asignación cuyo end1 sea el Id. de nodo y end2 sea el Id. de ticket. Para obtener más información sobre los datos de correlación, consulte ["Hibernate como proveedor de JPA" en la página 76](#).

En la tabla siguiente se muestra la columna NODE_ID de la clave externa:

NODE_ID	CARD_ID	CARD_TYPE	CARD_NAME
2015	1	Controladora de bus serie	Intel 82801EB USB Universal Host Controller
3581	2	Sistema	Intel 631xESB/6321ESB/3100 Chipset LPC
3581	3	Display (Visualización)	ATI ES1000
3581	4	Periférico del sistema base	HP ProLiant iLO 2 Legacy Support Function

- Cada CIT se puede asignar a una o más tablas. Para asignar un CIT a más de una tabla, compruebe que hay una tabla principal cuyas claves principales existen en las otras tablas, y que hay una única columna de valor.

Por ejemplo, un ticket se asigna a dos tablas: ticket1 y ticket2. La primera tabla tiene las columnas c1 y c2 y la segunda tabla tiene las columnas c3 y c4. Para permitir que se consideren como una tabla, ambos deben tener la misma clave principal. Además, la primera clave de la tabla principal puede ser una columna de la segunda tabla.

En el ejemplo siguiente, las tablas comparten la misma clave principal denominada CARD_ID:

CARD_ID	CARD_TYPE	CARD_NAME
1	Controladora de bus serie	Intel 82801EB USB Universal Host Controller
2	Sistema	Intel 631xESB/6321ESB/3100 Chipset LPC
3	Display (Visualización)	ATI ES1000
4	Periférico del sistema base	HP ProLiant iLO 2 Legacy Support Function

CARD_ID	CARD_VENDOR
1	Hewlett-Packard Company
2	(Controlador de host USB estándar)
3	Hewlett-Packard Company
4	(Dispositivos del sistema estándar)
5	Hewlett-Packard Company

2. Crear un tipo de CI

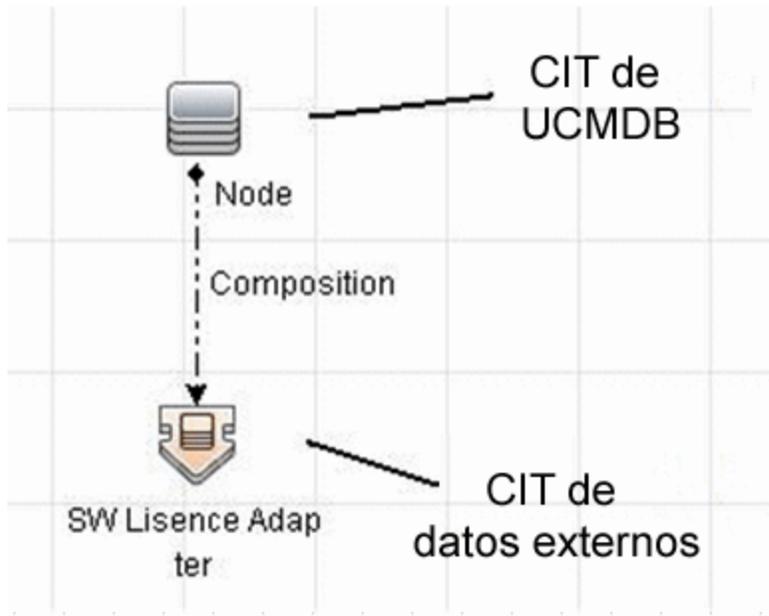
En este paso, va a crear un CIT que representa los datos del RDBMS (el origen de datos externo).

- a. En UCMDB, acceda al Administrador de tipos de CI y cree un nuevo tipo de CI. Para obtener más información, consulte "Cómo crear un tipo de CI" en *HP Universal CMDB – Guía de modelado*.
- b. Agregue los atributos necesarios al CIT, como la hora del último acceso, proveedor, etc. Son los atributos que el adaptador recuperará del origen de datos externo e introducirá en las vistas de CMDB.

3. Crear una relación

En este paso, agregará una relación entre el CIT de UCMDB y el nuevo CIT que representa los datos que se van a federar desde el origen de datos externo.

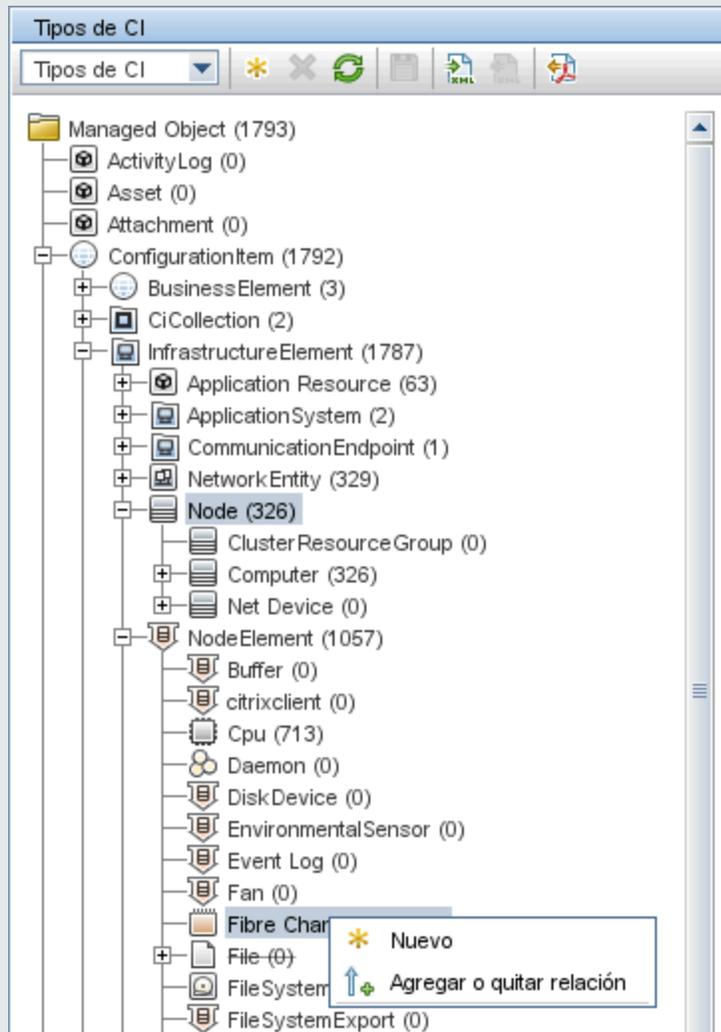
Agregue relaciones adecuadas y válidas al nuevo CIT. Para obtener más información, consulte "Cuadro de diálogo Agregar o quitar relación" en *HP Universal CMDB – Guía de modelado*.



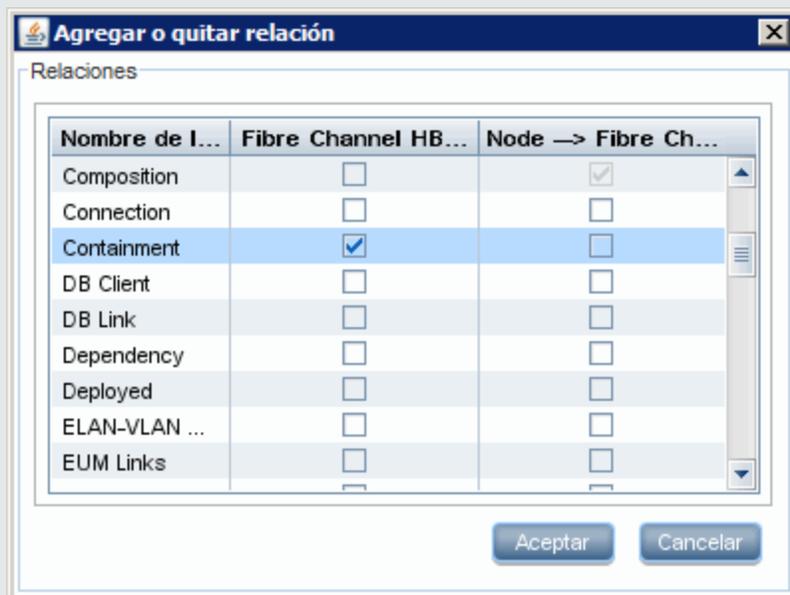
Nota: En esta etapa, todavía no puede ver los datos federados ni rellenar los datos externos, ya que todavía no ha definido el método para introducir los datos.

Ejemplo de la creación de una relación de contención:

- a. En el Administrador de CIT, seleccione los dos CIT:



- b. Cree una relación **Containment** entre los dos CIT:



Preparación del paquete de adaptadores

En este paso, localizará y configurará el paquete del adaptador de bases de datos genéricas.

1. Localice el paquete **db-adapter.zip** en la carpeta **C:\hp\UCMDB\UCMDBServer\content\adapters**.
2. Extraiga el paquete en un directorio temporal local.
3. Edite el archivo XML del adaptador:
 - Abra el archivo **discoveryPatterns\db_adapter.xml** en un editor de texto.
 - Localice el atributo **adapter id** y reemplace el nombre:

```
<pattern id="MyAdapter" displayLabel="My Adapter" xsi:noNamespaceSchemaLocation="../../Patterns.xsd" description="Discovery Pattern Description"
        schemaVersion="9.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" displayName="UCMDB API Population">
```

Si el adaptador admite el relleno, debería agregarse la capacidad siguiente al elemento **<adapter-capabilities>**:

```
<support-replicatioin-data>  
  <source>  
    <changes-source>  
  </source>  
</support-replicatioin-data>
```

La etiqueta o ID del monitor en la lista de adaptadores en el panel Punto de integración de HP Universal CMDB.

Al crear un adaptador de BD genérico, no es necesario editar la etiqueta **changes-source** en la etiqueta **support-replicatioin-data**. Si se implementa el complemento **FcmdbPluginForSyncGetChangesTopology**, se devolverá la topología modificada de la última ejecución. Si no se implementa el complemento, se devolverá la topología completa y se realizará la eliminación automática según los CI devueltos.

Para obtener más información sobre cómo rellenar CMDB con datos, consulte "Página de Integration Studio" en la *HP Universal CMDB – Guía de Administración de Data Flow..*

- Si el adaptador está utilizando el motor de asignación desde la versión 8.x (lo que significa que no está usando el nuevo motor de asignación de conciliación), reemplace el siguiente elemento:

```
<default-mapping-engine>
```

con:

```
<default-mapping-engine>com.hp.ucmdb.federation.  
mappingEngine.AdapterMappingEngine</default-mapping-engine>
```

Para revertir al nuevo motor de asignación, devuelva el elemento al siguiente valor:

```
<default-mapping-engine>
```

- Localice la definición **category**:

```
<category>Generic</category>
```

Cambie el nombre de categoría **Generic** a la categoría que prefiera.

Nota: Los adaptadores cuyas categorías se especifican como **Generic** no se enumeran en Integration Studio cuando crea un nuevo punto de integración.

- La conexión con la base de datos puede describirse mediante un nombre de usuario (esquema), contraseña, tipo de base de datos, nombre de máquina host de base de datos y nombre de base de datos o SID.

Para este tipo de conexión, los parámetros tienen los elementos siguientes en la sección **parameter** del archivo XML del adaptador:

```
<parámetros>
  <!--The description attribute may be written in simple text or HTML.-->
  <!--The host attribute is treated as a special case by UCMDDB-->
  <!--and will automatically select the probe name (if possible)-->
  <!--according to this attribute's value.-->
  <!--Display name and description may be overwritten by I18N values-->
    <parameter name="host" display-name="Hostname/IP" type="string"
description="The host name or IP address of the remote machine" mandatory="false" order-index="10" />
    <parameter name="port" display-name="Port" type="integer" description="El puerto de conexión del equipo remoto" mandatory="false" order-index="11" />
    <parameter name="dbtype" display-name="DB Type" type="string" description="El tipo de base de datos" valid-values="Oracle;SQLServer;MySQL;BO" mandatory="false" order-index="13">Oracle</parameter>
    <parameter name="dbname" display-name="DB Name/SID" type="string" description="El nombre de la base de datos o su SID (en el caso de Oracle)" mandatory="false" order-index="13" />
    <parameter name="credentialsId" display-name="Credentials ID" type="integer" description="The credentials to be used" mandatory="true" order-index="12" />
</parameters>
```

Nota: Esta es la configuración predeterminada. Por consiguiente, el archivo **db_adapter.xml** ya contiene esta definición.

Hay situaciones en las que la conexión con la base de datos no puede configurarse de esta manera. Por ejemplo, la conexión con Oracle RAC o la conexión mediante un controlador de base de datos distinto del proporcionado con CMDDB.

Para estas situaciones, puede describir la conexión utilizando el nombre de usuario (esquema), la contraseña y una cadena de URL de conexión.

Para definir esto, edite la sección de parámetros XML del adaptador de la manera siguiente:

```
<parámetros>
  <!--The description attribute may be written in simple text or HTML.-->
  <!--The host attribute is treated as a special case by CMDDBRTSM-->
  <!--and will automatically select the probe name (if possible)-->
  <!--according to this attribute's value.-->
```

```

    <!--Display name and description may be overwritten by I18N values-->
    <parameter name="url" display-name="Connection String" type="string"
description="The connection string to connect to the database" mandatory
="true" order-index="10" />
    <parameter name="credentialsId" display-name="Credentials ID" type=
"integer" description="The credentials to be used" mandatory="true" order-i
ndex="12" />
</parameters>

```

Un ejemplo de una dirección URL que se conecta a un Oracle RAC mediante el controlador Data Direct con la configuración de fábrica es:

```
jdbc:mercury:oracle://labm3amdb17:1521;ServiceName=RACQA;AlternateServers=
(labm3amdb18:1521);LoadBalancing=true.
```

4. En el directorio temporal, abra la carpeta **adapterCode** y cambie el nombre de **GenericDBAdapter** al valor de **adapter id** que se usó en el paso anterior.

Esta carpeta contiene los archivos jar que ejecutan la lógica de federación, por ejemplo, el nombre del adaptador, la consulta y las clases en CMDB y los campos de RDBMS que admite el adaptador.

5. Configure el adaptador como se precise. Para obtener más información, consulte ["Configuración del adaptador - Método mínimo" abajo](#).
6. Cree un archivo *.zip con el mismo nombre que haya dado al atributo **adapter id**, tal como se ha descrito en el paso ["Edite el archivo XML del adaptador:" en la página 83](#).

Nota: El archivo **descriptor.xml** es un archivo predeterminado que existe en cada paquete.

7. Guarde el nuevo paquete que ha creado en el paso anterior. El directorio predeterminado para los adaptadores es: **C:\hp\UCMDB\UCMDBServer\content\adapters**.

Configuración del adaptador - Método mínimo

El método simplificado (mínimo) es un método para crear el archivo de asignación **simplifiedConfiguration.xml** utilizado por el adaptador. Este método permite un relleno o federación de tipo básico de un CIT único.

Las instrucciones que se proporcionan en esta sección describen un método de asignar el modelo de clase para ciertos tipos de CI en CMDB a un RDBMS.

Todos los archivos de configuración mencionados en esta sección se encuentran en el paquete **db-adapter.zip** de la carpeta **C:\hp\UCMDB\UCMDBServer\content\adapters** que extrajo en ["Preparación del paquete de adaptadores" en la página 83](#).

Nota: El archivo **orm.xml** que se ha generado automáticamente como resultado de ejecutar este método es un buen ejemplo que se puede usar al trabajar con el método avanzado.

Debe usar este método mínimo cuando tenga que:

- Federar/rellenar un único nodo, como un atributo de nodo.
- Mostrar las capacidades del adaptador de bases de datos genéricas.

Este método:

- admite solo la federación/rellenado de un nodo
- admite solo relaciones virtuales varios-a-uno

Configurar el archivo `adapter.conf`

Para cambiar la configuración en el archivo `adapter.conf` de manera que el adaptador utilice el método de configuración simplificado:

1. Abra el archivo **`adapter.conf`** en un editor de texto.
2. Localice la siguiente línea: **`use.simplified.xml.config=<true/false>`**.
3. Cámbielo a **`use.simplified.xml.config=true`**.

Ejemplo: Rellenado de un nodo y dirección IP utilizando el método simplificado

Este ejemplo muestra cómo rellenar un **Nodo** relacionado por un vínculo de tipo Containment con **Dirección IP** en UCMDB. RDBMS tiene una tabla denominada **simpleNode** que contiene datos en el nombre, el nodo y la dirección IP del equipo.

El contenido de la tabla **simpleNode** se muestra a continuación:

	host_id	host_name	note	ip_address
	1	Comp1	Test Computer 1	12.33.211.52
	2	Comp2	Test Computer 2	12.33.211.53
	3	Comp3	Test Computer 3	12.33.211.54
	4	Comp4	Test Computer 4	12.33.211.55

El relleno se realiza en tres fases, de la forma siguiente:

1. ["Crear simplifiedConfiguration.xml" en la página siguiente](#)
2. ["Crear el TQL" en la página 89](#)
3. ["Crear un punto de integración" en la página 91](#)

Crear `simplifiedConfiguration.xml`

Cree `simplifiedConfiguration.xml` de la forma siguiente:

1. Cree una entidad `cmdb-class` de la forma siguiente:

```
<cmdb-class cmdb-class-name="node" default-table-name="simpleNode">
```

El tipo de CI es `node` y el nombre de tabla de RDBMS es `simpleNode`.

2. Establezca `primary-key` de la tabla de la forma siguiente:

```
<primary-key column-name="host_id"/>
```

Esta clave principal es equivalente al Id. de entidad en el archivo `orm.xml`.

3. Establezca la regla `reconciliation-by-two-nodes` de la forma siguiente:

```
<reconciliation-by-two-nodes connected-node-cmdb-class-name="ip_address" cmd  
b-link-type="containment">
```

Esta etiqueta define la relación entre los tipos de CI de `Node` e `IpAddress`. El tipo de relación es el vínculo de tipo `Containment`. Los dos tipos de CI conectados realizan la conciliación. La asignación de atributos del nodo conectado (en este caso, `IpAddress`) se define en el atributo `connected-node`.

4. Agregue la condición `or` entre los atributos de conciliación de la forma siguiente:

```
<or is-ordered="true">
```

Esta etiqueta define una relación OR entre los atributos de conciliación, lo que significa que el primer atributo de conciliación que sea `true` establecerá la regla de conciliación entera en `true`.

5. Agregue los siguientes atributos:

```
<attribute cmdb-attribute-name="name" column-name="host_name" ignore-case="t  
rue"/>
```

Esta etiqueta establece una asignación entre `node.name` en UCMDB y la columna `host_name` de la tabla `simpleNode`.

Haga lo mismo con el atributo `data_note`:

```
<attribute cmdb-attribute-name="data_note" column-name="note" ignore-case="t  
rue"/>
```

Agregue el atributo de nodo conectado:

```
<connected-node-attribute cmdb-attribute-name="name" column-name="ip_adres  
s"/>
```

Esta etiqueta establece una asignación entre `ip_address.name` y la columna `ip_address` de la tabla `simpleNode`.

6. Cierre la etiqueta abierta por orden:

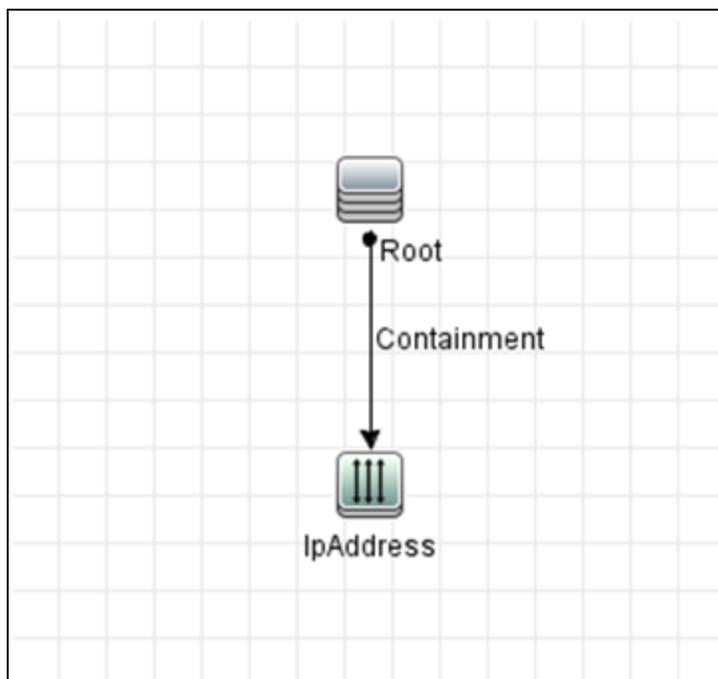
```
</or>  
</reconciliation-by-two-nodes>  
</cmdb-class>
```

El contenido del archivo `simplifiedConfiguration.xml` aparecen ahora como se indica a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>  
<generic-db-adapter-config xmlns:xsi="http://www.w3.org/2001/  
XMLSchema-instance" xsi:noNamespaceSchemaLocation="META-CONF/simplifiedConfig  
uration.xsd">  
  <cmdb-class cmdb-class-name="node" default-table-name="simpleNode">  
    <primary-key column-name="host_id"/>  
    <reconciliation-by-two-nodes connected-node-cmdb-class-name="ip_address" cmd  
b-link-type="containment">  
      <or is-ordered="true">  
        <attribute cmdb-attribute-name="name" column-name="host_name" ignore-  
case="true"/>  
        <attribute cmdb-attribute-name="data_note" column-name="note" ignore-  
case="true"/>  
        <connected-node-attribute cmdb-attribute-name="name" column-name="ip_a  
ddress"/>  
      </or>  
    </reconciliation-by-two-nodes>  
  </cmdb-class>  
</generic-db-adapter-config>
```

Crear el TQL

El TQL será un **nodo** conectado por un vínculo de tipo `Containment` a `ip_address`. El nodo deberá estar marcado como **root**, como se muestra a continuación.



Para crear el TQL:

1. Vaya a **Modelado > Modeling Studio**.
2. Haga clic en el botón **Nuevo** y cree una consulta nueva.
3. Vaya a la ficha Tipos de CI y arrastre el tipo de CI Node y el tipo de CI IPAddress a la pantalla TQL.
4. Conecte **Node** e **IPAddress** con una relación de tipo Containment.
5. Haga clic con el botón derecho en el elemento **Node** y seleccione Propiedades del nodo de consulta.
6. Cambie **Nombre del elemento** a **Root**.
7. Vaya a la ficha **Diseño de elementos**. Seleccione **Atributos específicos** como la condición de atributos. Seleccione **Nombre** y **Nota** de la ventana Atributos disponibles y muévalos a la ventana Atributos específicos.
8. Haga clic con el botón derecho en el elemento **IPAddress** y seleccione Propiedades del nodo de consulta.
9. Vaya a la ficha **Diseño de elementos**. Seleccione **Atributos específicos** como la condición de atributos. Seleccione **Nombre** de la ventana Atributos disponibles y muévalo a la ventana Atributos específicos.
10. Guarde el TQL.

Crear un punto de integración

Cree el punto de integración de la forma siguiente:

1. Vaya a **Administración de Data Flow > Integration Studio** y haga clic en el botón **Nuevo punto de integración**.
2. Inserte los detalles del punto de integración y haga clic en **Aceptar**.
3. En la ficha Rellenado, seleccione el botón **Nuevo trabajo de integración** y agregue el TQL creado previamente.
4. Guarde el punto de integración y haga clic en el botón **Ejecutar sincronización completa**.

Configuración del adaptador - Método avanzado

Estos archivos de configuración se encuentran en el paquete **db-adapter.zip** de la carpeta **C:\hp\UCMDB\UCMDBServer\content\adapters** que extrajo al preparar el paquete del adaptador. Para obtener más información, consulte "[Preparación del paquete de adaptadores](#)" en la [página 83](#).

Esta tarea incluye los siguientes pasos:

- "[Configuración del archivo orm.xml](#)" abajo
- "[Configurar el archivo reconciliation_rules.txt](#)" en la [página 95](#)

Configuración del archivo orm.xml

En este paso, asigna los CIT y las relaciones de CMDDB a las tablas del RDBMS.

1. Abra el archivo **orm.xml** en un editor de texto.

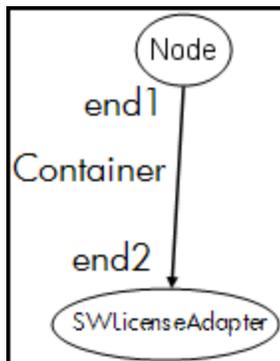
De manera predeterminada, este archivo contiene una plantilla que puede usar para asignar tantos CIT y relaciones como sea necesario.

Nota: No edite el archivo **orm.xml** en ninguna versión del Bloc de notas de Microsoft Corporation. Utilice Notepad++, UltraEdit o cualquier otro editor de texto de otros fabricantes.

2. Realice los cambios en el archivo de acuerdo con las entidades de datos que se van a asignar. Para obtener más información, consulte los ejemplos siguientes.

En el archivo **orm.xml** se pueden asignar los tipos de relaciones siguientes:

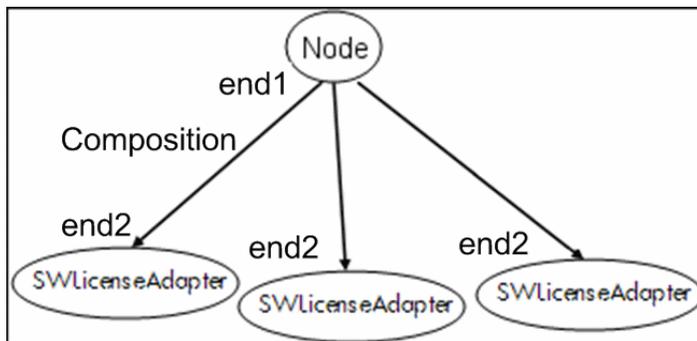
- Uno a uno (One-to-one):



El código para este tipo de relación es:

```
<one-to-one name="end1" target-entity="node">  
    <join-column name="Device_ID" >  
</one-to-one>  
<one-to-one name="end2" target-entity="sw_sub_component">  
    <join-column name="Device_ID" >  
    <join-column name="Version_ID" >  
</one-to-one>
```

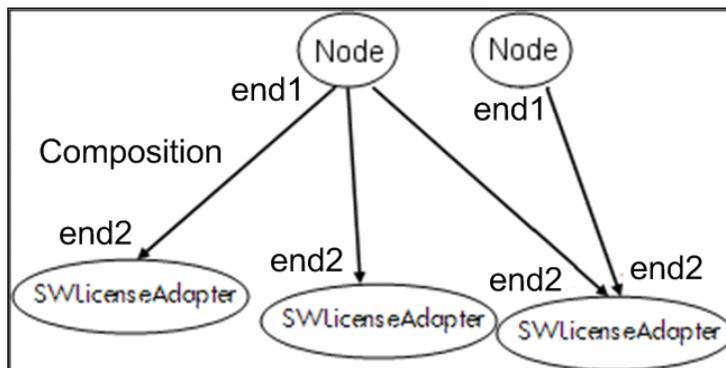
- Varios a uno (Many to one):



El código para este tipo de relación es:

```
<many-to-one name="end1" target-entity="node">  
    <join-column name="Device_ID" >  
</many-to-one>  
<one-to-one name="end2" target-entity="sw_sub_component">  
    <join-column name="Device_ID" >  
    <join-column name="Version_ID" >  
</one-to-one>
```

- Varios a varios (Many to many):



El código para este tipo de relación es:

```
<many-to-one name="end1" target-entity="node">  
  <join-column name="Device_ID" >  
</many-to-one>  
<many-to-one name="end2" target-entity="sw_sub_component">  
  <join-column name="Device_ID" >  
  <join-column name="Version_ID" >  
</many-to-one>
```

Para obtener más información sobre las convenciones de nomenclatura, consulte ["Convenciones de nomenclatura" en la página 118](#).

Ejemplo de asignación de entidades entre el modelo de datos y el RDBMS:

Nota: Los atributos que no tienen que ser configurados se omiten en los ejemplos siguientes.

- La clase del CIT de CMDB:
`<entity class="generic_db_adapter.node">`
- El nombre de la tabla del RDBMS:
`<table name="Device"/>`
- El nombre de columna del identificador exclusivo de la tabla RDBMS:
`<column name="Device ID"/>`
- El nombre del atributo del CIT de CMDB:
`<basic name="name">`
- El nombre del campo de la tabla del origen de datos externo:
`<column name="Device_Name"/>`
- El nombre del nuevo CIT que ha creado en ["Crear un tipo de CI" en la página 80](#):
`<entity class="generic_db_adapter.MyAdapter">`
- El nombre de la tabla correspondiente del RDBMS:
`<table name="SW_License"/>`
- La identidad única del RDBMS:
- El nombre de atributo del CIT de CMDB y el nombre del atributo correspondiente del RDBMS:

Ejemplo de asignación de relaciones entre el modelo de datos y el RDBMS:

- La clase de la relación de CMDB:

```
<entity class="generic_db_adapter.node_containment_MyAdapter">
```

- El nombre de la tabla de RDBMS donde se realiza la relación:

```
<table name="MyAdapter"/>
```

- El Id. exclusivo del RDBMS:

```
<id name="id1">  
    <column updatable="false" insertable="false"  
name="Device_ID">  
        <generated-value strategy="TABLE"/>  
</id>  
<id name="id2">  
    <column updatable="false" insertable="false"  
name="Version_ID">  
        <generated-value strategy="TABLE"/>  
</id>
```

- El tipo de relación y el CIT de CMDB:

```
<many-to-one target-entity="node" name="end1">
```

- Los campos de clave principal y la clave externa del RDBMS:

```
<join-column updatable="false" insertable="false"  
referenced-column-name="[column_name]" name="Device_ID" />
```

Configurar el archivo `reconciliation_rules.txt`

En este paso puede definir las reglas por las que el adaptador reconcilia el CMDB y el RDBMS (solo si se utiliza el motor de asignación, por compatibilidad con la versión 8.x):

1. Abra **META-INF\reconciliation_rules.txt** en un editor de texto.
2. Realice los cambios en el archivo de acuerdo con el CIT que está asignando. Por ejemplo, para asignar un CIT de nodo, utilice la siguiente expresión:

```
multinode[node] ordered expression[^name]
```

Nota:

- Si los datos de la base de datos distinguen mayúsculas de minúsculas, no elimine el carácter de control (^).
- Compruebe que cada corchete de apertura tiene un corchete de cierre correspondiente.

Para obtener más información, consulte "[Archivo reconciliation_rules.txt \(para compatibilidad con versiones anteriores\)](#)" en la página 128.

Implementación de un complemento

Esta tarea describe cómo implementar y desplegar un adaptador de bases de datos genéricas con complementos.

Nota: Antes de escribir un complemento para un adaptador, asegúrese de que ha completado todos los pasos necesarios de "[Preparación del paquete de adaptadores](#)" en la página 83.

1. Opción 1: Escribir un complemento basado en Java
 - a. Copie los siguientes archivos jar desde el directorio de instalación del servidor UCMDDB en la ruta de clase de desarrollo:
 - Copie el archivo **db-interfaces.jar** y el archivo **db-interfaces-javadoc.jar** de la carpeta **tools\adapter-dev-kit\db-adapter-framework**.
 - Copie el archivo **federation-api.jar** y el archivo **federation-api-javadoc.jar** de la carpeta **tools\adapter-dev-kit\SampleAdapters\production-lib**.

Nota: En los archivos **db-interfaces-javadoc.jar** y **federation-api-javadoc.jar** y en la documentación en línea podrá encontrar más información sobre cómo desarrollar un complemento:

- **C:\hp\UCMDDB\UCMDDBServer\deploy\ucmdb-docs\docs\eng\APIs\DBAdapterFramework_JavaAPI\index.html**
 - **C:\hp\UCMDDB\UCMDDBServer\deploy\ucmdb-docs\docs\eng\APIs\Federation_JavaAPI\index.html**
- b. Escriba una clase Java que implemente la interfaz Java del complemento. Las interfaces se definen en el archivo **db-interfaces.jar**. La tabla siguiente especifica la interfaz que debe implementarse para cada complemento:

Tipo de complemento	Nombre de interfaz	Método
Sincronizar la topología completa	FcmdbPluginForSyncGetFullTopology	getFullTopology
Sincronizar cambios	FcmdbPluginForSyncGetChangesTopology	getChangesTopology
Sincronizar diseño	FcmdbPluginForSyncGetLayout	getLayout
Recuperar consultas admitidas	FcmdbPluginForSyncGetSupportedQueries	getSupportedQueries
Modificar la definición de consulta TQL y los resultados	FcmdbPluginGetTopologyCmdbFormat	getTopologyCmdbFormat
Modificar peticiones de diseño para CI	FcmdbPluginGetCIsLayout	getCisLayout
Modificar peticiones de diseño para vínculos	FcmdbPluginGetRelationsLayout	getRelationsLayout
Id. de devolución	FcmdbPluginPushBackIds	getPushBackIdsSQL

La clase del complemento debe tener un constructor público predeterminado. Además, todas las interfaces muestran un método llamado `initPlugin`. Este método se llama antes de cualquier otro método y se utiliza para inicializar al adaptador con el objeto de entorno del adaptador continente.

Si se implementa **FcmdbPluginForSyncGetChangesTopology**, hay dos maneras diferentes de informar de los cambios:

- **Informar de la topología raíz entera en todo momento.** Según esta topología, la función de eliminación automática encuentra los CI que deben quitarse. En este caso, la función de eliminación automática debe habilitarse utilizando lo siguiente:

```
<autoDeleteCITs isEnabled="true">
    <CIT>link                </CIT>
    <CIT>object              </CIT>
</autoDeleteCITs>
```

- **Informar de cada instancia de CI que se ha quitado/actualizado.** En este caso, la función de eliminación automática debe habilitarse utilizando lo siguiente:

```
<autoDeleteCITs isEnabled="false">
    <CIT>link                </CIT>
    <CIT>object              </CIT>
</autoDeleteCITs>
```

- Asegúrese de que dispone de los archivos JAR del SDK de Federación y los JAR del adaptador de bases de datos genéricas antes de compilar el código Java. El SDK de Federación es el archivo **federation_api.jar**, que se puede encontrar en el directorio **C:\hp\UCMDB\UCMDBServer\lib**
 - Empaquete su clase en un archivo jar y colóquela en la carpeta `adapterCode\<Nombre adaptador>` en el paquete del adaptador, antes de desplegarla.
- Opción 2: Escribir un complemento basado en Groovy
 - Cree un archivo de código Groovy (MyPlugin.groovy) en el menú Administración de adaptador, bajo los archivos de configuración de paquete del adaptador.
 - En la clase Groovy, implemente las interfaces adecuadas. Las interfaces se definen en el archivo `db-interfaces.jar`, consulte la tabla mostrada anteriormente.
 - Los complementos se configuran con el archivo **plugins.txt**, situado en la carpeta **META-INF** del adaptador.

A continuación se muestra un ejemplo del archivo desde el adaptador DDMi:

```
# mandatory plugin to sync full topology
[getFullTopology]
com.hp.ucmdb.adapters.ed.plugins.replication.EDReplicationPlugin
# mandatory plugin to sync changes in topology
[getChangesTopology]
com.hp.ucmdb.adapters.ed.plugins.replication.EDReplicationPlugin
# mandatory plugin to sync layout
[getLayout]
com.hp.ucmdb.adapters.ed.plugins.replication.EDReplicationPlugin
# plugin to get supported queries in sync. If not defined return all tq1s
names
[getSupportedQueries]
# internal not mandatory plugin to change tq1 definition and tq1 result
[getTopologyCmdbFormat]
# internal not mandatory plugin to change layout request and CIs result
[getCisLayout]
# internal not mandatory plugin to change layout request and relations
```

```
result
[getRelationsLayout]
# internal not mandatory plugin to change action on pushBackIds
[pushBackIds]
```

Leyenda:

#: línea de comentarios.

[<Tipo de adaptador>]: comienzo de la sección de definición para un tipo de adaptador específico.

Puede ser una línea vacía debajo de cada [<Tipo de adaptador>], lo que significa que no hay ninguna clase de complemento asociado, o se puede mostrar el nombre completo de la clase de complemento.

4. Empaquete el adaptador con el nuevo archivo jar y el archivo **plugins.xml** actualizado. El resto de los archivos del paquete debería ser el mismo que en cualquier adaptador basado en el adaptador de bases de datos genéricas.

Despliegue del adaptador

1. En UC MDB, acceda al Administrador de paquetes. Para obtener más información, consulte "Página Administrador de paquetes" en *HP Universal CMDB – Guía de administración*.
2. Pulse el icono **Desplegar paquetes en servidor (desde un disco local)**  y busque el paquete del adaptador. Seleccione el paquete y haga clic en **Abrir**; a continuación, haga clic en **Desplegar** para mostrar el paquete en el Administrador de paquetes.
3. Seleccione el paquete de la lista y haga clic en el icono **Ver recursos del paquete**  para comprobar que el contenido del paquete lo reconoce el Administrador de paquetes.

Edición del adaptador

Cuando haya creado y desplegado el adaptador, puede editarlo en UC MDB. Para obtener más información, consulte "Administración de adaptadores" en la *HP Universal CMDB – Guía de Administración de Data Flow*.

Crear un punto de integración

En este paso, comprobará que la federación funciona. Es decir, que la conexión es válida y el archivo XML es válido. Sin embargo, esta comprobación no verifica que el XML se está asignando a los campos correctos del RDBMS.

1. En UCMDB, acceda a Integration Studio (**Administración de Data Flow > Integration Studio**).
2. Cree un punto de integración. Para obtener más información, consulte "Cuadro de diálogo Nuevo punto de integración/Editar punto de integración" en *HP Universal CMDB – Guía de Administración de Data Flow*.

La ficha Federación muestra todos los CIT que se pueden federar usando este punto de integración. Para obtener más información, consulte "Ficha Federación" en *HP Universal CMDB – Guía de Administración de Data Flow*.

Creación de una vista

En este paso va a crear una vista que le permitirá ver instancias del CIT.

1. En UCMDB, acceda a Modeling Studio (**Modelado > Modeling Studio**).
2. Cree una vista. Para obtener más información, consulte "Cómo crear una vista de patrón" en *HP Universal CMDB – Guía de modelado*.

Cálculo de los resultados

En este paso, comprobará los resultados.

1. En UCMDB, acceda a Modeling Studio (**Modelado > Modeling Studio**).
2. Abra una vista.
3. Calcule los resultados haciendo clic en el botón **Calcular número de resultados de la consulta** .
4. Haga clic en el botón **Vista previa** para ver los CI de la vista.

Visualización de resultados

En este paso, verá los resultados y depurará los problemas del procedimiento. Por ejemplo, si no se muestra nada en la vista, compruebe las definiciones en el archivo **orm.xml**; suprima los atributos de la relación y vuelva a cambiar el adaptador.

1. En UCMDB, acceda a IT Universe Manager (**Modelado > IT Universe Manager**).
2. Seleccione un CI. La ficha Propiedades muestra los resultados de la federación.

Ver informes

En este paso, visualizará los informes de topología. Para obtener más información, consulte "Información general de los informes de topología" en *HP Universal CMDB – Guía de modelado*.

Habilitación de archivos de registro

Consulte los archivos de registro para comprender los flujos de cálculo, el ciclo de vida del adaptador y para ver la información de depuración. Para obtener más información, consulte ["Archivos de registro del adaptador" en la página 149](#).

Uso de Eclipse para asignar entre atributos CIT y tablas de bases de datos

Precaución: Este procedimiento está dirigido a usuarios con un conocimiento avanzado del desarrollo de contenido. Si tiene alguna duda, póngase en contacto con HP Software Support.

En esta tarea se describe cómo instalar y usar el complemento JPA, incluido con la edición J2EE de Eclipse, para:

- Habilite la asignación gráfica entre los atributos de clase de CMDB y las columnas de la tabla de la base de datos.
- Habilite la edición manual del archivo de asignación (`orm.xml`), mientras proporciona precisión. La comprobación de precisión incluye una comprobación de sintaxis así como verificación de que los atributos de la clase y columnas de la tabla de la base de datos asignada se indican correctamente.
- Habilite el despliegue del archivo de asignación en el servidor de CMDB y para ver los errores, como una comprobación de precisión adicional.
- Defina una consulta de ejemplo en el servidor de CMDB y ejecútelo directamente en Eclipse, para probar el archivo de asignación.

La versión 1.1 del complemento es compatible con UCMDDB versión 9.01 o posterior y Eclipse IDE for Java EE Developers, versión 1.2.2.20100217-2310 o posterior.

Esta tarea incluye los siguientes pasos:

- ["Requisitos previos" en la página siguiente](#)
- ["Instalación" en la página siguiente](#)
- ["Preparación del entorno de trabajo" en la página 103](#)
- ["Creación de un adaptador" en la página 103](#)
- ["Configuración del complemento CMDB" en la página 103](#)
- ["Importación del modelo de clase de UCMDDB" en la página 104](#)
- ["Construcción del archivo ORM: Asignación de clases de UCMDDB a tablas de la base de datos"](#)

en la página 104

- "Asignación de Id." en la página 105
- "Asignación de atributos" en la página 105
- "Asignación de un vínculo válido" en la página 105
- "Construcción del archivo ORM: uso de tablas secundarias" en la página 106
- "Definición de una tabla secundaria" en la página 106
- "Asignación de un atributo a una tabla secundaria" en la página 107
- "Utilización de un archivo ORM existente como base" en la página 107
- "Importación de un archivo ORM existente desde un adaptador" en la página 107
- "Comprobación de la precisión del archivo orm.xml - Comprobación de precisión incorporada" en la página 108
- "Creación de un nuevo punto de integración" en la página 108
- "Despliegue el archivo ORM en el CMDDB" en la página 108
- "Ejecución de una consulta TQL de ejemplo" en la página 108

1. Requisitos previos

Instale la última actualización de **Java Runtime Environment (JRE) 6** en el equipo en donde se ejecutará Eclipse desde el sitio siguiente:

<http://java.sun.com/javase/downloads/index.jsp>.

2. Instalación

- a. Descargue y extraiga **Eclipse IDE for Java EE Developers** de <http://www.eclipse.org/downloads> en una carpeta local, por ejemplo, **C:\Archivos de programa\eclipse**.
- b. Copie **com.hp.plugin.import_cmdb_model_1.0.jar** de **C:\hp\UCMDB\UCMDBServer\tools\db-adapter-eclipse-plugin\bin** en **C:\Program Files\Eclipse\plugins**.
- c. Inicie **C:\Archivos de programa\Eclipse\eclipse.exe**. Si aparece un mensaje relativo a que no se encuentra la máquina virtual Java, inicie **eclipse.exe** con la siguiente línea de comandos:

```
"C:\Program Files\eclipse\eclipse.exe" -vm "<JRE installation folder>\bin"
```

3. Preparación del entorno de trabajo

En este paso, configurará el espacio de trabajo, base de datos, conexiones y propiedades del controlador.

- a. Extraiga el archivo **workspaces_gdb.zip** de **C:\hp\UCMDB\UCMDBServer\tools\db-adapter-eclipse-plugin\workspace** en **C:\Documents and Settings\All Users**.

Nota: Debe usar la ruta exacta de la carpeta. Si descomprime el archivo en la ruta errónea o deja el archivo sin descomprimir, el procedimiento no funcionará.

- b. En Eclipse, elija **File > Switch Workspace > Other**:

Si está trabajando con:

- SQL Server, seleccione la carpeta siguiente: **C:\Documents and Settings\All Users\workspace_gdb_sqlserver**.
- MySQL, seleccione la carpeta siguiente: **C:\Documents and Settings\All Users\workspace_gdb_mysql**.
- Oracle, seleccione la carpeta siguiente: **C:\Documents and Settings\All Users\workspace_gdb_oracle**.

- c. Haga clic en **Aceptar**.

- d. En Eclipse, muestre la vista Project Explorer y seleccione **<Active project> > JPA Content > persistence.xml > <nombre proyecto activo> > orm.xml**.

- e. En la vista Data Source Explorer (panel inferior izquierdo), haga clic con el botón derecho en la conexión de la base de datos y seleccione el menú **Properties**.

- f. En el cuadro de diálogo **Properties for <Connection name>**, seleccione **Common** y seleccione la casilla **Connect every time the workbench is started**. Seleccione **Driver Properties** y rellene las propiedades de conexión. Haga clic en **Test Connection** y compruebe que la conexión funciona. Haga clic en **Aceptar**.

- g. En la vista Data Source Explorer, haga clic con el botón derecho la conexión de la base de datos y haga clic en **Connect**. Se muestra un árbol que contiene los esquemas y tablas de la base de datos debajo del icono de la conexión de la base de datos.

4. Creación de un adaptador

Cree un adaptador usando las directrices indicadas en "[Paso 1: Creación de un adaptador](#)" en la [página 28](#).

5. Configuración del complemento C MDB

- a. En Eclipse, haga clic en **UCMDB > Settings** para abrir el cuadro de diálogo **CMDB Settings**.
- b. Si no está seleccionado, seleccione el proyecto JPA recientemente creado como proyecto activo.
- c. Introduzca el nombre de host de CMDB, por ejemplo, **localhost** o **labm1.itdep1**. No es preciso incluir el número de puerto o el prefijo `http://` en la dirección.
- d. Rellene el nombre de usuario y la contraseña para acceder a la API de CMDB, por lo general **admin/admin**.
- e. Asegúrese de que la carpeta **C:\hp** en el servidor de CMDB se asigna como una unidad de red.
- f. Seleccione la carpeta base del adaptador relevante en **C:\hp**. La carpeta base es la única que contiene el archivo **dbAdapter.jar** y la subcarpeta **META-INF**. Su ruta debería ser **C:\hp\UCMDB\UCMDBServer\runtime\fcmdb\CodeBase\<nombre de adaptador>**. Compruebe que no hay ninguna barra invertida (\) al final.

6. Importación del modelo de clase de UCMDB

En este paso, seleccione los CIT que se van a asignar como entidades JPA.

- a. Haga clic en **UCMDB > Import CMDB Class Model** para abrir el cuadro de diálogo **CI Type Selection**.
- b. Seleccione los tipos de CI que pretende asignar como entidades de JPA. Haga clic en **Aceptar**. Los tipos de CI se importan como clases Java. Compruebe que aparecen en la carpeta **src** del proyecto activo.

7. Construcción del archivo ORM: Asignación de clases de UCMDB a tablas de la base de datos

En este paso, va a asignar las clases Java (las que ha importado en el paso anterior) a las tablas de la base de datos.

- a. Asegúrese de que la conexión de la base de datos está conectada. Haga clic con el botón derecho en el proyecto activo (llamado **myProject** de manera predeterminada) en Project Explorer. Seleccione la vista JPA, seleccione la casilla **Override default schema from connection** y seleccione el esquema de base de datos relevante. Haga clic en **Aceptar**.
- b. Asigne un CIT: En la vista JPA Structure, haga clic con el botón derecho la rama **Entity Mappings** y seleccione **Add Class**. Se abre el cuadro de diálogo **Add Persistent Class**. No cambie el campo **Map as (Entity)**.
- c. Haga clic en **Browse** y seleccione la clase UCMDB que se va a asignar (todas las clases de UCMDB pertenecen al paquete **generic_db_adapter**).
- d. Haga clic en **OK** en ambos cuadros de diálogo. La clase seleccionada se muestra bajo la

rama **Entity Mappings** en la vista JPA Structure.

Nota: Si la entidad aparece sin un árbol de atributos, haga clic con el botón derecho en el proyecto activo de la vista Project Explorer. Elija **Close** y después **Open**.

- e. En la vista JPA Details, seleccione la tabla de la base de datos principal a la que deberá asignarse la clase de UCMDB. Deje los demás campos sin modificar.

8. Asignación de Id.

De acuerdo con los estándares de JPA, cada clase persistente debe tener al menos un atributo de Id. Para las clases de UCMDB, puede asignar hasta tres atributos como ID. Los posibles atributos de ID se llaman **id1**, **id2** y **id3**.

Para asignar un atributo de Id.:

- a. Expanda la clase correspondiente debajo de la rama **Entity Mappings** en la vista JPA Structure, haga clic con el botón derecho en el atributo relevante (por ejemplo, **id1**) y seleccione **Add Attribute to XML and Map...**
- b. Se abre el cuadro de diálogo **Add Persistent Attribute**. Seleccione **Id** en el campo **Map as** y haga clic en **OK**.
- c. En la vista JPA Details, seleccione la columna de la tabla de la base de datos a la que deberá asignarse el campo Id.

9. Asignación de atributos

En este paso, puede asignar atributos a las columnas de la base de datos.

- a. Expanda la clase correspondiente debajo de la rama **Entity Mappings** en la vista JPA Structure, haga clic con el botón derecho en el atributo relevante (por ejemplo, **host_hostname**) y seleccione **Add Attribute to XML and Map...**
- b. Se abre el cuadro de diálogo **Add Persistent Attribute**. Seleccione **Basic** en el campo **Map as** y haga clic en **OK**.
- c. En la vista JPA Details, seleccione la columna de la tabla de la base de datos a la que deberá asignarse el campo de atributo.

10. Asignación de un vínculo válido

Realice los pasos descritos en el paso "[Construcción del archivo ORM: Asignación de clases de UCMDB a tablas de la base de datos](#)" en la página precedente para asignar una clase de UCMDB que denote un vínculo válido. El nombre de cada una de estas clases tiene la estructura siguiente: **<nombre entidad end1>_<nombre vínculo>_<nombre entidad end 2>**. Por ejemplo, un vínculo **Contains** entre un host y una ubicación se indica por una clase Java cuyo nombre es **generic_db_adapter.host_contains_location**. Para obtener más información, consulte "[Archivo reconciliation_rules.txt \(para compatibilidad con versiones](#)

anteriores)" en la página 128.

- a. Asigne los atributos de Id. de la clase de vínculo, tal como se describe en "[Asignación de Id.](#)" en la página precedente. Para cada atributo de Id., expanda el grupo de casillas de verificación **Details** en la vista JPA Details y borre las casillas **Insertable** y **Updateable**.
- b. Asigne los atributos **end1** y **end2** de la clase de vínculo de la forma siguiente: Para cada uno de los atributos **end1** y **end2** de la clase de vínculo:
 - Expande la clase correspondiente debajo de la rama **Entity Mappings** en la vista JPA Structure, haga clic con el botón derecho en el atributo relevante (por ejemplo, **end1**) y seleccione **Add Attribute to XML and Map...**
 - En el cuadro de diálogo **Add Persistent Attribute**, seleccione **Many to One** o **One to One** en el campo **Map as**.
 - Seleccione **Many to One** si el CI **end1** o **end2** especificado puede tener varios vínculos de este tipo. En caso contrario, seleccione **One to One**. Por ejemplo, para un vínculo **host_contains_ip** el extremo **host** debería asignarse como **Many to One**, ya que un host puede tener varios IP y el extremo **ip** debería asignarse como **One to One**, ya que un IP puede tener solo un único host.
 - En la vista JPA Details, seleccione **Target entity**, por ejemplo, **generic_db_adapter.host**.
 - En la sección **Join Columns** de la vista JPA Details, marque **Override Default**. Haga clic en **Edit**. En el cuadro de diálogo **Edit Join Column**, seleccione la columna de la clave externa de la tabla de la base de datos de vínculos a una entrada en la tabla de la entidad de destino **end1/end2**. Si el nombre de columna referenciado en la tabla de la entidad de destino **end1/end2** está asignado a su atributo de Id., deje **Referenced Column Name** sin cambiar. En caso contrario, seleccione el nombre de la columna a la que apunta la columna de la clave externa. Borre las casillas **Insertable** y **Updatable** y haga clic en **OK**.
 - Si la entidad de destino **end1/end2** tiene más de un Id., haga clic en el botón **Add** para agregar columnas de unión adicionales y asignarlas de la misma forma que se describió en el paso anterior.

11. Construcción del archivo ORM: uso de tablas secundarias

JPA permite que una clase Java se asigne a más de una tabla de la base de datos. Por ejemplo, **Host** se puede asignar a la tabla **Device** para habilitar la persistencia de la mayoría de sus atributos y para la tabla **NetworkNames** para habilitar la persistencia de **host_hostName**. En este caso, **Device** es la tabla principal y **NetworkNames** es la tabla secundaria. Se puede definir cualquier número de tablas secundarias. La única condición es que tiene que haber una relación uno-a-uno entre las entradas de las tablas principal y secundaria.

12. Definición de una tabla secundaria

Seleccione la clase apropiada en la vista JPA Structure. En la vista **JPA Details**, acceda a la sección **Secondary Tables** y haga clic en **Add**. En el cuadro de diálogo **Add Secondary Table**, seleccione la tabla secundaria apropiada. Deje los otros campos sin modificar.

Si la tabla principal y la secundaria no tienen las mismas claves principales, configure las columnas de unión en la sección **Primary Key Join Columns** de la vista **JPA Details**.

13. Asignación de un atributo a una tabla secundaria

Puede asignar un atributo de clase a un campo de una tabla secundaria de la forma siguiente:

- a. Asigne el atributo, tal como se describe en ["Asignación de atributos" en la página 105](#).
- b. En la sección **Column** de la vista JPA Details, seleccione el nombre de la tabla secundaria en el campo **Table** para reemplazar el valor predeterminado.

14. Utilización de un archivo ORM existente como base

Para usar un archivo **orm.xml** existente como base para el que está desarrollando, ejecute los pasos siguientes:

- a. Compruebe que todos los CIT asignados en el archivo **orm.xml** se importan en el proyecto Eclipse activo.
- b. Seleccione y copie todo o parte de las asignaciones de entidades del archivo existente.
- c. Seleccione la ficha **Source** del archivo **orm.xml** en la perspectiva JPA de Eclipse.
- d. Pegue las asignaciones de entidades copiadas en la etiqueta **<entity-mappings>** del archivo **orm.xml**, debajo de la etiqueta **<schema>**. Asegúrese de que la etiqueta schema se configura tal como se describe en el paso ["Construcción del archivo ORM: Asignación de clases de UCMDDB a tablas de la base de datos" en la página 104](#). Todas las entidades pegadas ahora aparecen en la vista JPA Structure. De ahora en adelante, las asignaciones se pueden editar tanto gráfica como manualmente a través del código xml del archivo **orm.xml**.
- e. Haga clic en **Guardar**.

15. Importación de un archivo ORM existente desde un adaptador

Si ya existe un adaptador, se puede usar el complemento de Eclipse para editar gráficamente su archivo ORM. Importe el archivo **orm.xml** en Eclipse, edítelo usando el complemento y, a continuación, vuelva a implementarlo en el equipo UCMDDB. Para importar el archivo ORM, haga clic en el botón en la barra de herramientas de Eclipse. Se muestra un cuadro de diálogo de confirmación. Haga clic en **Aceptar**. El archivo ORM se copia desde el equipo de UCMDDB al proyecto Eclipse activo y todas las clases relevantes se importan desde el modelo de clase de UCMDDB.

Si las clases relevantes no aparecen en la vista JPA Structure, haga clic con el botón derecho en el proyecto activo de la vista Project Explorer, elija **Close** y, a continuación, **Open**.

De ahora en adelante, el archivo ORM se puede editar gráficamente usando Eclipse y, a continuación, implementarlo de nuevo en el equipo de UCMDDB, tal como se describe en ["Despliegue el archivo ORM en el CMDB" abajo](#).

16. **Comprobación de la precisión del archivo orm.xml - Comprobación de precisión incorporada**

El complemento JPA de Eclipse comprueba si hay algún error y lo marca en el archivo **orm.xml**. Se comprueban tanto los errores de sintaxis (por ejemplo, nombre de etiqueta erróneo, etiqueta sin cerrar, Id. que falta) como los errores de asignación (por ejemplo, nombre del atributo o nombre de cambio de la tabla de base de datos erróneos). Si hay errores, su descripción aparece en la vista **Problems**.

17. **Creación de un nuevo punto de integración**

Si no hay ningún punto de integración en el CMDB para este adaptador, puede crearla en Integration Studio. Para obtener más información, consulte "Integration Studio" en *HP Universal CMDB – Guía de Administración de Data Flow*.

Rellene el nombre del punto de integración en el cuadro de diálogo que se muestra. El archivo **orm.xml** se copia en la carpeta del adaptador. Se crea un punto de integración con todos los tipos de CI importados como sus clases admitidas, excepto por los CIT multinodo, si están configurados en el archivo **reconciliation_rules.txt**. Para obtener más información, consulte ["Archivo reconciliation_rules.txt \(para compatibilidad con versiones anteriores\)" en la página 128](#).

18. **Despliegue el archivo ORM en el CMDB**

Guarda el archivo **orm.xml** y despléguelo en el servidor UCMDDB pulsando **UCMDDB > Deploy ORM**. El archivo **orm.xml** se copia en la carpeta del adaptador y este se recarga. El resultado de la operación se muestra en el cuadro de diálogo **Operation Result**. Si ocurre cualquier error durante el proceso de recarga, se muestra el seguimiento de la pila de excepción Java en el cuadro de diálogo. Si no se ha definido todavía un punto de integración usando el adaptador, no se detecta ningún error de asignación después del despliegue.

19. **Ejecución de una consulta TQL de ejemplo**

- a. Defina una consulta (no una vista) en Modeling Studio. Para obtener más información, consulte "Modeling Studio" en *HP Universal CMDB – Guía de modelado*.
- b. Cree un punto de integración mediante el adaptador que creó en el paso ["Creación de un nuevo punto de integración" arriba](#). Para obtener más información, consulte "Cuadro de diálogo Nuevo punto de integración/Editar punto de integración" en *HP Universal CMDB – Guía de Administración de Data Flow*.
- c. Durante la creación del adaptador, compruebe que los tipos de CI que deben participar en la consulta se admiten en este punto de integración.
- d. Al configurar el complemento CMDB, utilice este nombre de consulta de ejemplo en el cuadro de diálogo Configuración. Para obtener más información, consulte el paso ["Configuración del complemento CMDB" en la página 103](#).

- e. Haga clic en el botón **Ejecutar consulta** para ejecutar una consulta de ejemplo y comprobar si devuelve los resultados requeridos usando el archivo **orm.xml** recientemente creado.

Archivos de configuración de adaptadores

Los archivos que se tratan en esta sección están ubicados en el paquete **db-adapter.zip** en la carpeta **C:\hp\UCMDB\UCMDBServer\content\adapters**.

En esta sección se describen los siguientes archivos de configuración:

- ["Archivo adapter.conf"](#) en la página siguiente
- ["Archivo simplifiedConfiguration.xml"](#) en la página 112
- ["Archivo orm.xml"](#) en la página 114
- ["Archivo reconciliation_types.txt"](#) en la página 128
- ["Archivo reconciliation_rules.txt \(para compatibilidad con versiones anteriores\)"](#) en la página 128
- ["Archivo transformations.txt"](#) en la página 130
- ["Archivo discriminator.properties"](#) en la página 131
- ["Archivo replication_config.txt"](#) en la página 132
- ["Archivo fixed_values.txt"](#) en la página 132
- ["Archivo persistence.xml"](#) en la página 133

Configuración general

- **adapter.conf**. El archivo de configuración del adaptador. Para obtener más información, consulte ["Archivo adapter.conf"](#) en la página siguiente.

Configuración sencilla

- **simplifiedConfiguration.xml**. Archivo de configuración que reemplaza **orm.xml**, **transformations.txt** y **reconciliation_rules.txt** con menos capacidades. Para obtener más información, consulte ["Archivo simplifiedConfiguration.xml"](#) en la página 112.

Configuración avanzada

- **orm.xml**. El archivo de asignación relacional de objetos en el que asigna entre los CIT de C MDB y las tablas de base de datos. Para obtener más información, consulte ["Archivo orm.xml"](#) en la página 114.
- **reconciliation_rules.txt**. Contiene las reglas de conciliación. Para obtener más información,

consulte "[Archivo reconciliation_rules.txt \(para compatibilidad con versiones anteriores\)](#)" en la [página 128](#).

- **transformations.txt.** Archivo de transformaciones en el que se especifican los convertidores que se van a aplicar para convertir el valor de CMDB al valor de la base de datos, y viceversa. Para obtener más información, consulte "[Archivo transformations.txt](#)" en la [página 130](#).
- **Discriminator.properties.** Este archivo asigna cada tipo de CI admitido a una lista separada por comas de posibles valores correspondientes. Para obtener más información, consulte "[Archivo discriminator.properties](#)" en la [página 131](#).
- **Replication_config.txt.** Este archivo contiene una lista separada por comas de tipos de CI y relaciones, cuyas condiciones de propiedades se admiten en el complemento de replicación. Para obtener más información, consulte "[Archivo replication_config.txt](#)" en la [página 132](#).
- **Fixed_values.txt.** Este archivo le permite configurar valores fijos para atributos específicos de ciertos CIT. Para obtener más información, consulte "[Archivo fixed_values.txt](#)" en la [página 132](#).

Configuración de Hibernate

- **persistence.xml.** Se utiliza para reemplazar las configuraciones de Hibernate de serie. Para obtener más información, consulte "[Archivo persistence.xml](#)" en la [página 133](#).

Activación del soporte de tabla temporal para el adaptador

La activación de las tablas temporales permite que el adaptador trabaje con más eficacia con la base de datos remota, reduciendo así la tensión en la base de datos y la red y mejorando también el rendimiento.

Para habilitar el soporte de tablas temporales en el adaptador de bases de datos generales, deben cumplirse las siguientes condiciones:

- Las credenciales proporcionadas para conectarse a la base de datos, incluyen permiso para crear, modificar y eliminar tablas temporales.
- Configure los ajustes siguientes en el archivo de configuración adapter.conf:

```
temp.tables.enabled=true
```

```
performance.enable.single.sql=true
```

Nota: Las tablas temporales solo son compatibles con Microsoft SQL y Oracle.

Archivo adapter.conf

Este archivo contiene la configuración siguiente:

- **use.simplified.xml.config=false.true**: utiliza `simplifiedConfiguration.xml`.

Nota: El uso de este archivo significa que `orm.xml`, `transformations.txt` y `reconciliation_rules.txt` se reemplazan con menos capacidades.

- **dal.ids.chunk.size=300**. No cambie este valor.
- **dal.use.persistence.xml=false.true**: el adaptador lee la configuración de Hibernate de `persistence.xml`.

Nota: No se recomienda reemplazar la configuración de Hibernate.

- **performance.memory.id.filtering=true**. Cuando GDBA ejecuta TQLS, en algunos casos puede recuperarse un gran número de ID y devolverlos a la base de datos mediante SQL. Para evitar este trabajo excesivo y mejorar el rendimiento, GDBA intenta leer toda la vista/tabla y filtra los resultados que hay en la memoria.
- **id.reconciliation.cmdb.id.type=string/bytes**. Al asignar el adaptador de bases de datos genéricas mediante la conciliación de Id., puede asignar **cmdb_ida** un tipo columna **string** o **bytes/raw** cambiando la propiedad **META-INF/ adapter.conf**.
- **performance.enable.single.sql=true**. Este parámetro es opcional. Si no aparece en el archivo, su valor predeterminado es **true**. Si es **true**, el adaptador de bases de datos genéricas intenta generar una única sentencia SQL para cada consulta que se ejecute (sea para rellenado o una consulta federada). El uso de una sola sentencia SQL mejora el rendimiento y el consumo de memoria del adaptador de bases de datos genéricas. Si es **false**, el adaptador de bases de datos genéricas genera varias sentencias SQL, que pueden tardar más tiempo y consumir más memoria que una sola. Aunque este atributo se establezca como **true**, el adaptador no genera una única sentencia SQL en las situaciones siguientes:
 - La base de datos a la que se conecta el adaptador no es un servidor Oracle o SQL.
 - La TQL que se ejecuta contiene una condición de cardinalidad distinta de 0..* y 1..* (por ejemplo, si existe una condición de cardinalidad como 2..* o 0..2).
- **in.expression.size.limit=950** (predeterminado). Este parámetro divide la expresión 'IN' del SQL ejecutado, cuando se alcance el límite de tamaño de la lista de argumentos.
- **stringlist.delimiter.of.<Nombre de CIT>.<Nombre de atributo>=<delimitador>**. Para asignar un atributo de lista de cadenas a una columna de la base de datos en el adaptador de bases de datos genéricas, el atributo tiene que asignarse a una columna de tipo string que contenga una lista de valores concatenados. Por ejemplo, para asignar el atributo **policy_category** con el tipo de CI **policy**, y la columna string contenga una lista de valores: `valor1##valor2##valor3` (que define una lista de 3 valores `valor1`, `valor2`, `valor3`), utilice la configuración: **stringlist.delimiter.of.policy.policy_category=##**.

- **temp.tables.enabled=true**. Permite el uso de tablas temporales para mejorar el rendimiento. Solo está disponible cuando **performance.enable.single.sql** está habilitado (solo es compatible con Microsoft SQL y Oracle). Pueden ser necesarios ciertos permisos en el servidor de bases de datos.
- **temp.tables.min.value=50**. Define el número de valores de condición (o Id.) que son necesarios para usar las tablas temporales.

Archivo *simplifiedConfiguration.xml*

Este archivo se usa para la asignación simple de clases de UCMDB a tablas de bases de datos. Para acceder a la plantilla para editar el archivo, vaya a **Administración de adaptador > db-adapter > Archivos de configuración**.

Esta sección incluye los siguientes temas:

- ["Plantilla del archivo simplifiedConfiguration.xml" abajo](#)
- ["Limitaciones" en la página 114](#)

Plantilla del archivo *simplifiedConfiguration.xml*

- La propiedad **CMDB-class-name** es el tipo de multinodo (el nodo al que los CIT federados se conectan en el TQL):

```
<?xml version="1.0" encoding="UTF-8"?>
<generic-DB-adapter-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="META-CONF/simplifiedConfiguration.xsd">
  <CMDB-class CMDB-class-name="node" default-table-name="[table_name]">
    <primary-key column-name="[column_name]"/>
  </CMDB-class>
</generic-DB-adapter-config>
```

- **reconciliation-by-two-nodes**. La conciliación se puede realizar usando uno o dos nodos. En este ejemplo de caso, la conciliación usa dos nodos.
- **connected-node-CMDB-class-name**. El segundo tipo de clase que se necesita en el TQL de conciliación.
- **CMDB-link-type**. El tipo de relación que se necesita en el TQL de conciliación.
- **link-direction**. La dirección de la relación del TQL de conciliación (desde `node` a `ip_address` o desde `ip_address` a `node`):

```
<reconciliation-by-two-nodes connected-node-CMDB-class-name="ip_address" CMDB-
link-type="containment" link-direction="main-to-connected">
```

La expresión de conciliación se realiza en la forma de OR y cada OR incluye AND.

- **is-ordered**. Determina si la conciliación se realiza en forma ordenada o con una comparación OR.

```
<or is-ordered="true">
```

Si la propiedad de conciliación se recupera desde la clase principal (el multinodo), utilice la etiqueta **attribute**, en caso contrario, utilice la etiqueta **connected-node-attribute**.

- **ignore-case.true**: cuando los datos del modelo de clase de UC MDB se comparan con los datos del RDBMS, no se distingue entre mayúsculas y minúsculas:

```
<attribute CMDB-attribute-name="name" column-name="[column_name]" ignore-case="true"/>
```

El nombre de columna es el nombre de la columna de la clave externa (la columna con valores que apuntan a la columna de la clave principal multinodo).

Si la columna de la clave principal está compuesta por varias columnas, es preciso que haya varias columnas de claves externas, una para cada columna clave principal.

```
<foreign-primary-key column-name="[column_name]" CMDB-class-primary-key-column="[column_name]"/>
```

Si hay pocas columnas de claves principales, duplique esta columna.

```
<primary-key column-name="[column_name]"/>
```

- Las propiedades **from-CMDB-converter** y **to-CMDB-converter** son clases Java que implementan las interfaces siguientes:

- com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.FcldbDalTransformerFromExternalDB
- com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.FcldbDalTransformerToExternalDB

Utilice estos convertidores si el valor de CMDB y de la base de datos no son los mismos.

En este ejemplo, `GenericEnumTransformer` se usa para convertir el enumerador de acuerdo con el archivo XML que se escribe dentro del paréntesis (**generic-enum-transformer-example.xml**):

```
<attribute CMDB-attribute-name="[nombre de atributo CMDB]" column-name="[nombre de columna]" from-CMDB-converter="com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.GenericEnumTransformer(generic-enum-transformer-example.xml)" to-CMDB-converter="com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.GenericEnumTransformer(generic-enum-transformer-example.xml)" />
```

```
<attribute CMDB-attribute-name="[nombre_atributo_CMDB]" column-name="[nombre_columna]" />
```

```
<attribute CMDB-attribute-name="[nombre_atributo_CMDB]" column-name="[nombre_
columna]" />

</class>

</generic-DB-adapter-config>
```

Limitaciones

- Se puede usar para asignar solo consultas TQL que contengan un nodo (en el origen de la base de datos). Por ejemplo, puede ejecutar un `node > ticket` y una consulta TQL `ticket`. Para llevar la jerarquía de nodos desde la base de datos, debe usar el archivo **orm.xml** avanzado.
- Solo se admiten las relaciones uno-a-varios. Por ejemplo, puede llevar uno o más tickets en cada nodo. No puede llevar tickets que pertenecen a más de un nodo.
- No puede conectar la misma clase a tipos diferentes de CIT de CMDB. Por ejemplo, si define que `ticket` está conectado a `node`, no puede conectarse también a `application`.

Archivo *orm.xml*

Este archivo se usa para asignar los CTI de CMDB a las tablas de bases de datos.

En el directorio

C:\hp\UCMDB\UCMDBServer\runtime\fcmdb\CodeBase\GenericDBAdapter\META-INF se encuentra una plantilla que se puede usar para crear un archivo nuevo.

Para editar el archivo XML en un adaptador implementado, vaya a **Administración de adaptador > db-adapter > Archivos de configuración**.

Esta sección incluye los siguientes temas:

- ["Plantilla del archivo orm.xml" abajo](#)
- ["Varios archivos ORM" en la página 118](#)
- ["Convenciones de nomenclatura" en la página 118](#)
- ["Uso de instrucciones SQL integradas en lugar de nombres de tablas" en la página 118](#)
- ["Esquema orm.xml" en la página 119](#)
- ["Ejemplo de creación del archivo orm.xml" en la página 124](#)
- ["Configuración de un orm.xml específico para cada versión remota del producto " en la página 128](#)

Plantilla del archivo *orm.xml*

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

```
<entity-mappings xmlns="http://java.sun.com/xml/ns/persistence/orm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0" xsi:schemaLocation="http://java.sun.com/xml/ns/persistence/orm http://java.sun.com/xml/ns/persistence/orm_1_0.xsd">
  <description>Generic DB adapter orm</description>
```

No cambie el nombre del paquete.

```
<package>generic_db_adapter</package>
```

entity. El nombre del CTI de CMDB. Esta es la entidad multinodo.

Asegúrese de que **class** incluye un prefijo **generic_db_adapter**.

```
<entity class="generic_db_adapter.node">
  <table name="[table_name]"/>
```

Utilice una tabla secundaria si la entidad está asignada a más de una tabla.

```
<secondary-table name=""/>
<attributes>
```

Para la herencia de una única tabla con discriminador, utilice el código siguiente:

```
<inheritance strategy="SINGLE_TABLE"/>
<discriminator-value>node</discriminator-value>
<discriminator-column name="[column_name]"/>
```

Los atributos con la etiqueta **id** son las columnas de claves principales. Asegúrese de que la convención de nomenclatura para estas columnas de claves principales son **idX** (id1, id2, etc.) donde **X** es el índice de columnas de la clave principal.

```
<id name="id1">
```

Cambie solo el nombre de columna de la clave principal.

```
      <column updatable="false" insertable="false" name="[column_name]"/>
      <generated-value strategy="TABLE"/>
    </id>
```

basic. Se utiliza para declarar los atributos de CMDB. Asegúrese de editar solo las propiedades **name** y **column_name**.

```
        <basic name="name">
            <column updatable="false" insertable="false" name="[column_name]"/>
        </basic>
```

Para la herencia de una única tabla con discriminador, asigne las clases ampliadas de la forma siguiente:

```
    <entity name="[cmdb_class_name]" class="generic_db_adapter.nt" name="nt">
        <discriminator-value>nt </discriminator-value>
        <attributes>
    </entity>
    <entity class="generic_db_adapter.unix" name="unix">
        <discriminator-value>unix</discriminator-value>
        <attributes>
    </entity>
    <entity name="[CMDB_class_name]" class="generic_db_adapter.[CMDB_class_name]">
        <table name="[default_table_name]"/>
        <secondary-table name=""/>
        <attributes>
            <id name="id1">
                <column updatable="false" insertable="false" name="[column_name]"/>
                <generated-value strategy="TABLE"/>
            </id>
            <id name="id2">
                <column updatable="false" insertable="false" name="[column_name]"/>
                <generated-value strategy="TABLE"/>
            </id>
            <id name="id3">
                <column updatable="false" insertable="false" name="[column_name]"/>
                <generated-value strategy="TABLE"/>
            </id>
```

En el ejemplo siguiente se muestra un nombre de atributo de CMDB sin prefijo:

```
        <basic name="[CMDB_attribute_name]">
            <column updatable="false" insertable="false" name="[column_name]"/>
        </basic>
        <basic name="[CMDB_attribute_name]">
            <column updatable="false" insertable="false" name="[column_
```

```
name]"/>
    </basic>
    <basic name="[CMDB_attribute_name]">
        <column updatable="false" insertable="false" name="[column_n
ame]"/>
    </basic>
</attributes>
</entity>
```

Esta es la entidad de relación. La convención de nomenclatura es **end1Type_linkType_end2Type**. En este ejemplo **end1Type** es **node** y el **linkType** es **composition**.

```
<entity name="node_composition_[CMDB_class_name]" class="generic_db_adap
ter.node_composition_[CMDB_class_name]">
    <table name="[default_table_name]"/>
    <attributes>
        <id name="id1">
            <column updatable="false" insertable="false" name="[column_n
ame]"/>
            <generated-value strategy="TABLE"/>
        </id>
```

La entidad de destino es la entidad a la que está apuntando esta propiedad. En este ejemplo, **end1** se asigna a la entidad **node**.

many-to-one. Varias relaciones se pueden conectar a un nodo.

join-column. La columna que contiene los Id. **end1** (los Id. de entidades de destino).

referenced-column-name. El nombre de columna en la entidad de destino (**node**) que contiene los Id. que se usan en la columna de unión.

```
<many-to-one target-entity="node" name="end1">
    <join-column updatable="false" insertable="false" referenced-
column-name="[column_name]" name="[column_name]"/>
</many-to-one>
```

one-to-one. Una relación se puede conectar a una **[CMDB_class_name]**.

```
<one-to-one target-entity="[CMDB_class_name]" name="end2">
    <join-column updatable="false" insertable="false" referenced-
column-name="" name="[column_name]"/>
</one-to-one>
</attributes>
</entity>
</entity-mappings>
```

node attribute. Esto es un ejemplo de cómo agregar un atributo de nodo.

```
<entity class="generic_db_adapter.host_node">
  <discriminator-value>host_node</discriminator-value>
  <attributes/>
</entity>
<entity class="generic_db_adapter.nt">
  <discriminator-value>nt</discriminator-value>
  <attributes>
    <basic name="nt_servicepack">
      <column updatable="false" insertable="false" name="specific_type_value"/>
    </basic>
  </attributes>
</entity>
```

Varios archivos ORM

Se admiten varios archivos de asignación. Cada archivo de asignación debe finalizar con **orm.xml**. Todos los archivos de asignación deberían colocarse debajo de la carpeta META-INF del adaptador.

Convenciones de nomenclatura

- En cada entidad, la propiedad de clase debe coincidir con la propiedad de nombre con el prefijo `generic_db_adapter`.
- Las columnas de claves principales deben tomar los nombres del formulario **idX** donde **X = 1, 2, ...**, de acuerdo con el número de claves principales de la tabla.
- Los nombres de atributos deben coincidir con los nombres de atributos de la clase, distinguiendo mayúsculas y minúsculas.
- El nombre de la relación toma la forma de `end1Type_linkType_end2Type`.
- Los CIT de CMDB, que también son palabras reservadas de Java, deben llevar el prefijo **gdba_**. Por ejemplo, para el CIT de CMDB **goto**, la entidad ORM debe llamarse **gdba_goto**.

Uso de instrucciones SQL integradas en lugar de nombres de tablas

Puede asignar entidades en cláusulas `select` integradas en lugar de en tablas de bases de datos. Es el equivalente a definir una vista en la base de datos y a asignar una entidad en esta vista. Por ejemplo:

```
<entity class="generic_db_adapter.node">
```

```
<table name="(select d.id as id1, d.name as name , d.os as host_os f  
rom  
Device d)" />
```

En este ejemplo, los atributos de nodo deberían asignarse a las columnas id1, name y host_os, en lugar de id, name y os.

Se aplican las siguientes limitaciones:

- La instrucción SQL integrada está disponible solo cuando se usa Hibernate como proveedor de JPA.
- Es obligatorio el uso de corchetes redondos en torno a la cláusula select de SQL integrada.
- El elemento **<schema>** no debería estar presente en el archivo **orm.xml**. En el caso de Microsoft SQL Server 2005, significa que todos los nombres de tablas deberían llevar el prefijo dbo., en lugar de definirlos globalmente con **<schema>dbo</schema>**.

Esquema orm.xml

En la tabla siguiente se explican los elementos comunes del archivo **orm.xml**. El esquema completo se puede encontrar en http://java.sun.com/xml/ns/persistence/orm_1_0.xsd. La lista no es completa y explica principalmente el comportamiento específico del API de persistencia Java estándar para el adaptador de bases de datos genéricos.

Nombre y ruta del elemento	Descripción	Atributos
entity-mappings	El elemento raíz del documento de asignación de la entidad. Este elemento debe ser exactamente el mismo que el dado en los archivos de ejemplo GDBA.	
descripción (entity-mappings)	Una descripción libre del documento de asignación de la entidad. (Opcional)	

Nombre y ruta del elemento	Descripción	Atributos
paquete (entity-mappings)	El nombre del paquete Java que contendrá las clases de asignación. Siempre debería contener el texto <code>generic_db_adapter</code> .	<ol style="list-style-type: none"> Nombre: name Descripción: El nombre del tipo de CI de UCMDB al que está asignada la entidad. Si esta entidad se asigna a un vínculo de CMDB, el nombre de la entidad debería tener el formato <code>end_1>_link_name>_end_2></code>. <code><end_1>_<link_name>_<end_2></code>. Por ejemplo, <code>node_composition_cpu</code> define una entidad que se asignará al vínculo de composición entre un nodo y un CPU. Si el nombre del tipo de CI es el mismo que el nombre de la clase Java sin el prefijo del paquete, este campo se puede omitir. ¿Es necesario?: Opcional Tipo: Cadena Nombre: class Descripción: El nombre completo de la clase Java que se creará para esta entidad de la base de datos. El nombre del paquete de la clase Java debería ser el mismo que el nombre dado en el elemento <code>package</code>. Puede no utilizar palabras reservadas Java, como <code>interface</code> o <code>switch</code>, como nombre de clase. En ese caso, agregue el prefijo <code>gdba_</code> al nombre (por tanto la interfaz será <code>generic_db_adapter.gdba_interface</code>). ¿Es necesario?: necesario Tipo: Cadena

Nombre y ruta del elemento	Descripción	Atributos
<p>table (entity-mappings>entity)</p>	<p>Este elemento define la tabla principal de la entidad de la base de datos. Solo puede aparecer una vez. Necesario.</p>	<p>Nombre: name Descripción: El nombre de la tabla principal. Si el nombre de la tabla no contiene el esquema al que pertenece, la tabla se buscará solo en el esquema del usuario que se usó para crear el punto de integración. También puede ser cualquier instrucción SELECT válida. Si es una instrucción SELECT, debe ir entre paréntesis. ¿Es necesario?: necesario Tipo: Cadena</p>
<p>secondary-table (entity-mappings > entity)</p>	<p>Este elemento se puede usar para definir una tabla secundaria para la entidad de la base de datos. Esta tabla debe estar conectada a la tabla principal con una relación uno-a-uno. Puede definir más de una tabla secundaria. Opcional.</p>	<p>Nombre: name Descripción: El nombre de la tabla secundaria. Si el nombre de la tabla no contiene el esquema al que pertenece, la tabla se buscará solo en el esquema del usuario que se usó para crear el punto de integración. También puede ser cualquier instrucción SELECT válida. Si es una instrucción SELECT, debe ir entre paréntesis. ¿Es necesario?: necesario Tipo: Cadena</p>
<p>primary-key-join-column (entity-mappings > entity > secondary-table)</p>	<p>Si la tabla secundaria y la tabla principal no están conectadas usando campos con el mismo nombre, este elemento define el nombre del campo de la clave principal en la tabla secundaria que necesita conectarse al campo de clave principal de la tabla principal.</p>	<p>Nombre: name Descripción: El nombre del campo de la tabla principal en la tabla secundaria. Si este elemento no existe, se asume que el campo de clave principal tiene el mismo nombre que el campo de la clave principal de la tabla principal. ¿Es necesario?: Opcional Tipo: Cadena</p>

Nombre y ruta del elemento	Descripción	Atributos
<p>inheritance (entity-mappings > entity)</p>	<p>Si la entidad actual es la entidad principal de una familia de entidades de bases de datos, utilice este elemento para marcarlo como tal. Opcional.</p>	<p>Nombre: strategy Descripción: Define la forma en que se implementa la herencia en la base de datos. ¿Es necesario?: necesario Tipo: Uno de los valores siguientes:</p> <ul style="list-style-type: none"> • SINGLE_TABLE: Esta entidad y todas las entidades secundarias existen en la misma tabla. • JOINED: Las entidades secundarias son tablas unidas. • TABLE_PER_CLASS: Cada entidad está completamente definida en una tabla separada.
<p>discriminator-column (entity-mappings > entity)</p>	<p>Si la herencia es de tipo SINGLE_TABLE, este elemento se usa para definir el nombre del campo usado para determinar el tipo de entidad para cada fila.</p>	<p>Nombre: name Descripción: El nombre de la columna discriminador. ¿Es necesario?: necesario Tipo: Cadena</p>
<p>discriminator-value (entity-mappings > entity)</p>	<p>Este elemento define el tipo de entidad específica en el árbol de herencia. Este nombre necesita ser el mismo que el nombre definido en el archivo discriminator.properties para el grupo de valor de este tipo de entidad específica.</p>	
<p>attributes (entity-mappings > entity)</p>	<p>El elemento raíz de todas las asignaciones de atributos para una entidad.</p>	

Nombre y ruta del elemento	Descripción	Atributos
<p>id</p> <p>(entity-mappings > entity attributes)</p>	<p>Este elemento define el campo de clave para la entidad. Debe haber por lo menos un campo de Id. definido. Si existe más de un elemento de id, sus campos crean una clave compuesta para la entidad. Debería intentar y evitar claves compuestas para las entidades de CI (no para vínculos).</p>	<p>Nombre: name Descripción: Una cadena de tipo idX, donde X es un número entre 1 y 9. El primer Id. debería marcarse como id1, el segundo como id2 y así sucesivamente. NO es el nombre del atributo de clave de UCMDB. ¿Es necesario?: necesario Tipo: Cadena</p>
<p>basic</p> <p>(entity-mappings > entity attributes)</p>	<p>Este elemento define una asignación entre un campo de la tabla, que no forma parte de la clave principal de la tabla, y un atributo UCMDB.</p>	<p>Nombre: name Descripción: El nombre del atributo de UCMDB para el que se asigna el campo. Este atributo debe existir en el CI de UCMDB al que está asignada la entidad actual. ¿Es necesario?: necesario Tipo: Cadena</p>
<p>column</p> <p>(entity-mappings > entity > attributes > id</p> <p>-O BIEN-</p> <p>(entity-mappings > entity > attributes > basic)</p>	<p>Define el nombre de la columna de la tabla para la asignación básica o un campo de Id.</p>	<ol style="list-style-type: none"> Nombre: name Descripción: El nombre del campo. ¿Es necesario?: necesario Tipo: Cadena Nombre: table Descripción: El nombre de la tabla al que pertenece el campo. Esto debe ser la tabla principal o bien una de las tablas secundarias definidas para la entidad. Si se omite este atributo, se asume que el campo pertenece a la tabla principal. ¿Es necesario?: Opcional Tipo: Cadena

Nombre y ruta del elemento	Descripción	Atributos
one-to-one (entity-mappings > entity > attributes)	Define una columna cuyo valor está en otra tabla, y las dos tablas se conectan usando una relación uno-a-uno. Este elemento solo se admite para asignaciones de entidades de vínculo y no para los otros tipos de CI. Esta es la única forma de definir una asignación entre una tabla y un vínculo de UCMDB.	<ol style="list-style-type: none"> Nombre: name Descripción: Cuál de los dos extremos representa este campo. ¿Es necesario?: necesario Tipo: Bien <code>end1</code> o <code>end2</code> Nombre: target-entity Descripción: El nombre de la entidad a la que se refiere el extremo. ¿Es necesario?: necesario Tipo: Uno de los nombres de entidad definidos en el documento de asignación de entidades.
join-column (entity-mappings > entity attributes > one-to-one)	Define la forma de unir la entidad de destino definida en el elemento uno-a-uno de primer nivel y la entidad actual.	<ol style="list-style-type: none"> Nombre: name Descripción: El nombre del campo de la tabla actual que se usará para realizar la unión uno-a-uno. ¿Es necesario?: necesario Tipo: Cadena Nombre: name Descripción: El nombre de un campo en la entidad conjunta mediante el cual se realiza la unión. Si este atributo se omite, asume que la tabla unida tiene una columna con el mismo nombre que el campo definido en el atributo de nombre. ¿Es necesario?: Opcional Tipo: Cadena

Ejemplo de creación del archivo orm.xml

El ejemplo aquí presentado muestra cómo crear el **archivo orm.xml**. En este ejemplo, las tablas de SQL de una base de datos remota se asignan a tipos de CI del UCMDB.

En las tablas que se encuentran en la base de datos remota con el formato que se muestra a continuación, rellene la tabla **Hosts** con nodos, la tabla **IP_Addresses** con direcciones IP, y después cree vínculos entre los nodos y las direcciones IP como se muestra a continuación:

Tabla de hosts

host_name	host_id
Test1	1
Test2	2
Test3	3

IP_Addresses Table

ip_address	ip_id
10.1.1.1	1
10.2.2.2	2
10.3.3.2	3
10.4.4.4	4

Tabla Host_IP_Link (vínculos entre nodos y direcciones IP)

host_id	ip_id
1	1
2	2
2	3
3	4

La clave principal de la tabla de **Hosts** es el campo **host_id** y la clave principal de la tabla **IP_Addresses** es el campo **ip_id**. En la tabla **Host_IP_Link**, **host_id** e **ip_id** son claves externas de las tablas **Hosts** e **IP_Addresses**.

De acuerdo con las tablas anteriores, cree el archivo **orm.xml** como se indica en los pasos siguientes. Las entidades utilizadas en este ejemplo son **node**, **ip_address** y **node_containment_ip_address**

1. Cree la entidad **node** asignando **host_id** de la tabla **Hosts** de la forma siguiente:

```
<entity-mappings xmlns="http://java.sun.com/xml/ns/persistence/orm"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
  xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
  /orm_1_0.xsd">
  <description>test_integration</description>
  <package>generic_db_adapter</package>
```

```
<entity class="generic_db_adapter.node">
  <table name="Hosts"/>
  <attributes>
    <id name="id1">
      <column updatable="false" insertable="false" name="
        host_id"/>
      <generated-value strategy="TABLE"/>
    </id>
    <basic name="name">
      <column updatable="false" insertable="false" name="
        host_name"/>
    </basic>
  </attributes>
</entity>
```

La entidad **class** debe ser un tipo de CI que ya existe en UCMDB. La tabla **name** es la tabla contenida en la base de datos que incluye un Id. y la información del host. El atributo de Id. se requiere para identificar hosts específicos y se usará más tarde en la asignación. En este ejemplo, el atributo **name** de esta entidad se rellena con la columna **host_name** en la tabla Hosts.

2. En la siguiente entidad, asigne direcciones IP de la tabla de interfaces:

```
<entity name="ip_address" class="generic_db_adapter.ip_address">
  <table name="IP_Addresses"/>
  <attributes>
    <id name="id1">
      <column insertable="false" updatable="false" name="ip_id"/>
      <generated-value strategy="TABLE"/>
    </id>
    <basic name="name">
      <column updatable="false" insertable="false" name="ip_address"/>
    </basic>
  </attributes>
```

```
</entity>
```

3. A continuación, el vínculo entre el nodo y la dirección IP debe crearse mediante la tabla de asignación y debe hacer referencia al campo **ip_id** (aunque podría hacer referencia, si se desea, a los campos **host_id** e **ip_id**).

```
<entity name="node_containment_ip_address"  
  class="generic_db_adapter.node_containment_ip_address">  
  <table name="Host_IP_Link"/>  
  <attributes>  
    <id name="id1">  
      <column updatable="false" insertable="false" name="ip_id"/>  
      <generated-value strategy="TABLE"/>  
    </id>  
    <many-to-one target-entity="node" name="end1">  
      <join-column name="host_id"/>  
    </many-to-one>  
    <one-to-one target-entity="ip_address" name="end2">  
      <join-column name="ip_id"/>  
    </one-to-one>  
  </attributes>  
</entity>
```

El nombre de entidad para el contenedor tiene el formato: [end1 CIT]_[link CIT]_[end2 CIT]. Así, para este ejemplo, dado que el tipo de CI de vínculo es **containment**, el nombre de entidad para el contenedor es: **node_containment_ip_address** y la clase de entidad es **generic_db_adapter.node_containment_ip_address**. El **id**. se requiere en este bloque de código y, aunque este ejemplo funciona con un único **id**. de la interfaz, las dos columnas podrían referirse a **id1** e **id2**. En ese caso el código sería:

```
<id name="id1">  
  <column updatable="false" insertable="false" name="ip_id"/>  
  <generated-value strategy="TABLE"/>  
</id>
```

```
<id name="id2">
  <column updatable="false" insertable="false" name="host_id"/>
  <generated-value strategy="TABLE"/>
</id>
```

Los dos extremos de este vínculo son 'many-to-one' y 'one-to-one', lo que significa que cada dirección IP estará vinculada a un nodo, pero un nodo podrá vincularse a varias direcciones IP. Las columnas incluidas proceden de la tabla de vínculos y hacen referencia a las tablas de hosts e interfaces.

Configuración de un orm.xml específico para cada versión remota del producto

Es posible configurar un archivo **orm.xml** específico para que el adaptador utilice un **orm.xml** determinado para una versión remota del producto. Por ejemplo, si el almacén de datos remoto tiene dos versiones del producto x e y, para cada una de las versiones puede haber una asignación diferente de entidades.

Para configurar un archivo orm.xml específico por versión remota del producto:

1. Agregue un parámetro al archivo **adapter.xml** denominado **version** y especifique los valores de versión posibles como **valid-values**.
2. En el paquete de adaptador, bajo la carpeta META-INF, cree una carpeta llamada **VersionOrm**.
3. En la carpeta **VersionOrm**, cree un archivo **orm.xml** para cada versión específica. El nombre de archivo debe contener el prefijo de la versión. Por ejemplo, si la versión se denomina **x**, el nombre de archivo debería ser **x_orm.xml**.

Nota: El archivo **orm.xml** en la carpeta META-INF se carga para cualquier versión remota del producto, independientemente de si se crea un archivo **orm.xml** específico para una versión remota del producto. Puede tener entidades que se asignan de la misma manera para todas las versiones.

Archivo *reconciliation_types.txt*

A partir de UCMDDB 10.00, el archivo **reconciliation_types.txt** ya no es relevante. Puede utilizarse cualquier CIT para la conciliación. El motor de federación ejecuta la asignación automáticamente.

Archivo *reconciliation_rules.txt* (para compatibilidad con versiones anteriores)

Este archivo se usa para configurar las reglas de conciliación si desea ejecutar la conciliación cuando se configura DBMappingEngine en el adaptador. Si no usa DBMappingEngine, se utiliza el

mecanismo de conciliación UCMDb genérico y no es preciso configurar este archivo.

Cada fila del archivo representa una regla. Por ejemplo:

```
multinode[node] expression[^node.name OR ip_address.name] end1_type[node]  
end2_type[ip_address] link_type[containment]
```

El multinodo se rellena con el nombre del multinodo (el CIT de CMDB que está conectado al CIT de la base de datos federada en la consulta TQL).

Esta expresión incluye la lógica que decide si dos multinodos son iguales (un multinodo en el CMDB y el otro en el origen de la base de datos).

La expresión está compuesta de OR o AND.

La convención concerniente a los nombres de atributos en la parte de expresión es [className].[attributeName]. Por ejemplo, attributeName en la clase ip_address se escribe ip_address.name.

Para una correspondencia ordenada (si la primera subexpresión OR devuelve una respuesta indicando que los multinodos no son iguales, no se compara la segunda subexpresión OR), utilice `ordered expression` en lugar de `expression`.

Para ignorar la distinción de mayúsculas y minúsculas durante una comparación, utilice el signo de control (^).

Los parámetros `end1_type`, `end2_type` y `link_type` solo se usan si la consulta TQL de conciliación contiene dos nodos y no solo un multinodo. En este caso, la consulta TQL de conciliación es `end1_type > (link_type) > end2_type`.

No es preciso agregar el diseño relevante cuando se toma de la expresión.

Tipos de reglas de conciliación

Las reglas de conciliación toman la forma de condiciones OR y AND. Puede definir estas reglas en varios nodos diferentes (por ejemplo, el nodo se identifica por `name from nodeAND/ORname from ip_address`).

Las opciones siguientes encuentran una correspondencia:

- **Correspondencia ordenada.** La expresión de conciliación se lee de izquierda a derecha. Dos subexpresiones OR se consideran iguales si tienen valores y son iguales. Dos subexpresiones OR se consideran no iguales si ambos tienen valores y no son iguales. Para cualquier otro caso no hay decisión y se busca la igualdad en la siguiente subexpresión OR.

name from node OR from ip_address. Si tanto CMDB como el origen de datos incluyen `name` y son iguales, los nodos se consideran iguales. Si ambos incluyen `name` pero no son iguales, los nodos se consideran que no son iguales sin probar el `name` de `ip_address`. Si el `name of node` falta en CMDB o en el origen de datos, se comprueba el `name of ip_address`.

- **Correspondencia regular.** Si hay igualdad en una de las expresiones OR, tanto el CMDB y el origen de datos se consideran iguales.

name from node OR from ip_address. Si no hay correspondencia en `name of node`, `name of ip_address` se comprueba por igualdad.

En las conciliaciones complejas, donde se modela la entidad de conciliación en el modelo de clase con varios CIT con relaciones (como nodo), la asignación de un nodo de superconjunto incluye todos los atributos relevantes de todos los CIT modelados.

Nota: Como resultado, existe una limitación de que todos los atributos de conciliación en el origen de datos deberían residir en tablas que comparten la misma clave principal.

Otra limitación indica que la consulta TQL de conciliación no debería tener más de dos nodos. Por ejemplo, la consulta TQL `node > ticket` tiene un nodo en el CMDB y un ticket en el origen de datos.

Para reconciliar los resultados, `name` debe recuperarse desde el nodo o de `ip_address`.

Si el `name` en CMDB está en el formato `*.m.com`, puede utilizarse un convertidor desde CMDB a la base de datos federada, y viceversa, para convertir estos valores.

Se usa la columna `node_id` de la tabla del ticket de la base de datos para conectar entre las entidades (la asociación definida también se puede realizar en una tabla de nodo):

Nodo de BD		BD de IP_Address	
PK	node_id	PK	ip_id
	nombre		nombre

Ticket de BD	
PK	ticket_id
	node_id

Nota: Las tres tablas deben formar parte del origen del RDBMS federado y no de la base de datos de CMDB.

Archivo *transformations.txt*

Este archivo contiene todas las definiciones de convertidor.

El formato es que cada línea contiene una nueva definición.

Plantilla del archivo *transformations.xml*

```
entity[[CMDB_class_name]] attribute[[CMDB_attribute_name]] to_DB_class[com.mercury.topaz.fcmbd.adapters.dbAdapter.dal.transform.impl.GenericEnumTransformer(generic-enum-transformer-example.xml)]
from_DB_class[com.mercury.topaz.fcmbd.adapters.dbAdapter.dal.transform.impl.GenericEnumTransformer(generic-enum-transformer-example.xml)]
```

entity. El nombre de entidad tal como aparece en el archivo `orm.xml`.

attribute. El nombre de atributo tal como aparece en el archivo `orm.xml`.

to_DB_class. El nombre completo de una clase que implementa la interfaz `com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.FcldbDalTransformerToExternalDB`. Los elementos entre paréntesis se incluyen en este constructor de clase. Utilice este convertidor para transformar valores de CMDB en valores de base de datos, por ejemplo, para anexar el sufijo `.com` a cada nombre de nodo.

from_DB_class. El nombre completo de una clase que implementa la interfaz `com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.FcldbDalTransformerFromExternalDB`. Los elementos entre paréntesis se incluyen en este constructor de clase. Utilice este convertidor para transformar valores de base de datos en valores de CMDB, por ejemplo, para anexar el sufijo `.com` a cada nombre de nodo.

Para obtener más información, consulte ["Convertidores de serie" en la página 134](#).

Archivo discriminator.properties

Este archivo asigna cada tipo de CI admitido (que también se usa como un valor de discriminador en `orm.xml`) a una lista separada por comas de posibles valores correspondientes de la columna discriminador, o una condición que coincida con valores posibles de la columna discriminador.

Si se utiliza una condición, utilice la sintaxis: `like(condición)`, donde `condición` es una cadena que puede contener los comodines siguientes:

- `%` (signo de porcentaje): permite que coincida con cualquier cadena de cualquier longitud (incluso una cadena de longitud cero)
- `_` (subrayado): permite que coincida con un solo carácter

Por ejemplo, `like(%unix%)` coincidirá con `unix`, `linux`, `unix-aix`, etc. Las condiciones de `like` solo pueden aplicarse a columnas de tipo cadena.

También puede tener un solo valor discriminador asignado a cualquier valor que no pertenezca a otro discriminador indicando `'all-other'`.

Si el adaptador que está creando usa capacidades de discriminador, debe definir todos los valores de discriminador en el archivo **`discriminator.properties`**.

Ejemplo de asignación del discriminador:

Por ejemplo, el adaptador admite los tipos de CI `node`, `nt` y `unix`, y la base de datos contiene una sola tabla denominada `t_nodes` que contiene una columna denominada **`type`**. Si el tipo es `10001`, la fila representa un nodo; si el tipo es `10004`, representa una máquina `unix`, etc. El archivo **`discriminator.properties`** podría tener el aspecto siguiente:

```
node=10001, 10005
nt=10002,10003
unix=2%
mainframe=all-other
```

El archivo **`orm.xml`** incluye el código siguiente:

```
<entity class="generic_db_adapter.node" >
  <table name="t_nodes" />
  ...
  <inheritance strategy="SINGLE_TABLE"/>
  <discriminator-value>node</discriminator-value>
  <discriminator-column name="type" />
  ...
</entity>
<entity class="generic_db_adapter.nt" name="nt">
  <discriminator-value>nt      </discriminator-value>
  <attributes>
</entity>
<entity class="generic_db_adapter.unix" name="unix">
  <discriminator-value>unix</discriminator-value>
  <attributes>
</entity>
```

El atributo `discriminator_column` se calcula de la forma siguiente:

- Si **type** contiene 10002 o 10003 para una entrada determinada, la entrada se asigna al CIT **nt**.
- Si **type** contiene 10001 o 10005 para una entrada determinada, la entrada se asigna al CIT **node**.
- Si **type** empieza por 2 o para una entrada determinada, la entrada se asigna al CIT **unix**.
- Cualquier otro valor en la columna **type** se asigna al CIT **mainframe**.

Nota: El CIT **node** también es el elemento de primer nivel de **nt** y **unix**.

Archivo replication_config.txt

Este archivo contiene una lista separada por comas de tipos de CI y relaciones, cuyas condiciones de propiedades se admiten en el complemento de replicación. Para obtener más información, consulte ["Complementos" en la página 140](#).

Archivo fixed_values.txt

Este archivo le permite configurar valores fijos para atributos específicos de ciertos CIT. De esta forma, a cada uno de estos atributos se le puede asignar un valor fijo que no está almacenado en la base de datos.

El archivo debería contener cero o más entradas del formato siguiente:

```
entity[<entityName>] attribute[<attributeName>] value[<value>]
```

Por ejemplo:

```
entity[ip_address] attribute[ip_domain] value[DefaultDomain]
```

El archivo también admite una lista de constantes. Para definir una lista de constantes, utilice la sintaxis siguiente:

```
entity[<entityName>] attribute[<attributeName>] value[<{<Va11>, <Va12>, <Va13>, .
.. }>]
```

Archivo *persistence.xml*

Este archivo se usa para reemplazar la configuración predeterminada de Hibernate y para agregar soporte para tipos de bases de datos que no son de serie (los tipos de bases de datos de serie son Oracle Server, Microsoft SQL Server y MySQL).

Si necesita admitir un nuevo tipo de base de datos, asegúrese de que proporciona un proveedor de grupos de conexión (el valor predeterminado es `c3p0`) y un controlador JDBC para su base de datos (ponga los archivos `*.jar` en la carpeta del adaptador).

Para ver todos los valores disponibles de Hibernate que se pueden cambiar, consulte la clase **org.hibernate.cfg.Environment** (para obtener más información, consulte <http://www.hibernate.org>.)

Ejemplo del archivo *persistence.xml*:

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation=
"http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd" version="1.
0">
  <!-- Don't change this value -->
  <persistence-unit name="GenericDBAdapter">
    <properties>
      <!-- Don't change this value -->
      <property name="hibernate.archive.autodetection" value="class,
        hbm" />
      <!--The driver class name"/-->
      <property name="hibernate.connection.driver_class" value="com.me
rcury.
        jdbc.MercOracleDriver" />
      <!--The connection url"/-->
      <property name="hibernate.connection.url" value="jdbc:mercury:or
acle:
        //artist:1521;sid=cmdb2" />
      <!--DB login credentials"/-->
      <property name="hibernate.connection.username" value="CMDB"/>
      <property name="hibernate.connection.password" value="CMDB"/>
      <!--connection pool properties"/-->
      <property name="hibernate.c3p0.min_size" value="5"/>
      <property name="hibernate.c3p0.max_size" value="20"/>
      <property name="hibernate.c3p0.timeout" value="300"/>
```

```
<property name="hibernate.c3p0.max_statements" value="50"/>
<property name="hibernate.c3p0.idle_test_period" value="3000"/>
<!--The dialect to use-->
<property name="hibernate.dialect" value="org.hibernate.dialect.
    OracleDialect" />
</properties>
</persistence-unit>
</persistence>
```

Conectarse a base de datos mediante autenticación NT

Es posible conectarse a un servidor MS SQL Server que requiera la autenticación NT. Para ello, se necesita un controlador que pueda analizar el dominio (es decir, un controlador JDBC JTDS).

La autenticación se realiza de acuerdo a los parámetros proporcionados (dominio, nombre de usuario, contraseña), y no con las credenciales de NT del proceso que se ejecuta actualmente.

1. En el archivo **persistence.xml**, edite las siguientes propiedades como se indica a continuación:

```
<!--The driver class name"/-->
<property name="hibernate.connection.driver_class"
value="net.sourceforge.jtds.jdbc.Driver"/>
<property name="hibernate.connection.url" value="jdbc:jtds:sqlserver://[nombre de host]:
[port];DatabaseName=[nombre de base de datos];domain=[dominio]"/>
<!--DB login credentials"/-->
<property name="hibernate.connection.username" value="[nombre de usuario]"/>
<property name="hibernate.connection.password" value="[contraseña]"/>
```

2. Coloque el archivo del controlador JDBC en: **<carpeta de instalación de sonda>\lib**.
3. Reinicie la sonda.

Convertidores de serie

Puede usar los siguientes convertidores (transformadores) para convertir consultas federadas y trabajos de replicación a y desde datos de base de datos.

Esta sección incluye los siguientes temas:

- ["Convertidores de serie" arriba](#)
- ["Convertidor SuffixTransformer" en la página 138](#)
- ["Convertidor PrefixTransformer" en la página 138](#)

- ["Convertidor BytesToStringTransformer" en la página 138](#)
- ["Convertidor StringDelimitedListTransformer " en la página 138](#)
- ["El convertidor personalizado" en la página 139](#)

Convertidor enum-transformer

Este convertidor usa un archivo XML que se le incluye como parámetro de entrada.

El archivo XML asigna entre valores de CMDB codificados de forma rígida y valores de base de datos (enums). Si uno de los valores no existe, puede optar por devolver el mismo valor, devolver un valor nulo o lanzar una excepción.

El transformador realiza una comparación entre dos cadenas mediante un método que distinga o que no distinga entre mayúsculas y minúsculas. El comportamiento predeterminado es distinguir mayúsculas de minúsculas. Para definir que el uso no distinga entre mayúsculas y minúsculas: `case-sensitive="false"` en el elemento `enum-transformer`.

Utilice un archivo de asignación XML para cada atributo de entidad.

Nota: Este convertidor se puede usar para los campos `to_DB_class` y `from_DB_class` del archivo **transformations.txt**.

XSD de archivo de entrada:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="enum-transformer">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="value" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="db-type" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="integer"/>
            <xs:enumeration value="long"/>
            <xs:enumeration value="float"/>
            <xs:enumeration value="double"/>
            <xs:enumeration value="boolean"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
        <xs:enumeration value="string"/>
        <xs:enumeration value="date"/>
        <xs:enumeration value="xml"/>
        <xs:enumeration value="bytes"/>
    </xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="cmdb-type" use="required">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="integer"/>
            <xs:enumeration value="long"/>
            <xs:enumeration value="float"/>
            <xs:enumeration value="double"/>
            <xs:enumeration value="boolean"/>
            <xs:enumeration value="string"/>
            <xs:enumeration value="date"/>
            <xs:enumeration value="xml"/>
            <xs:enumeration value="bytes"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="non-existing-value-action" use="required">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="return-null"/>
            <xs:enumeration value="return-original"/>
            <xs:enumeration value="throw-exception"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
```

```
<xs:attribute name="case-sensitive" use="optional">
  <xs:simpleType>
    <xs:restriction base="xs:boolean">
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:complexType>
</xs:element>
<xs:element name="value">
  <xs:complexType>
    <xs:attribute name="cldb-value" type="xs:string" use="required"/>
    <xs:attribute name="external-db-value" type="xs:string" use="required"/>
    <xs:attribute name="is-cldb-value-null" type="xs:boolean" use="optional"/>
    <xs:attribute name="is-db-value-null" type="xs:boolean" use="optional"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Ejemplo de conversión de un valor 'sys' en un valor 'System':

En este ejemplo, el valor `sys` de CMDB se transforma en un valor `System` en la base de datos federada y el valor `System` de la base de datos federada se transforma en el valor `sys` en CMDB.

Si el valor no existe en el archivo XML (por ejemplo, la cadena `demo`), el convertidor devuelve el mismo valor de entrada que recibe.

```
<enum-transformer CMDB-type="string" DB-type="string" non-existing-value-action="return-original" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../META-CONF/generic-enum-transformer.xsd">
  <value CMDB-value="sys" external-DB-value="System"/>
</enum-transformer>
```

Ejemplo de conversión de un valor externo o de CMDB en un valor nulo:

En este ejemplo, el valor de `NNN` en la base de datos remota se transforma en un valor nulo en la base de datos de CMDB.

```
<value cldb-value="null" is-cldb-value-null="true" external-db-value="NNN"/>
```

En este ejemplo, el valor de **000** en CMDB se transforma en un valor nulo en la base de datos remota.

```
<value cmdb-value="000" external-db-value="null" is-db-value-null="true"/>
```

Convertidor SuffixTransformer

Este convertidor se usa para agregar o suprimir sufijos desde el valor de CMDB o el de origen de la base de datos federada.

Hay dos implementaciones:

- **com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.AdapterToCmdbAddSuffixTransformer**. Agrega el sufijo (dado como entrada) al convertir un valor de base de datos federada en un valor de CMDB y suprime el sufijo al convertir del valor de CMDB en un valor de la base de datos federada.
- **com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.AdapterToCmdbRemoveSuffixTransformer**. Quita el sufijo (dado como entrada) al convertir un valor de base de datos federada en un valor de CMDB y agrega el sufijo al convertir del valor de CMDB en un valor de la base de datos federada.

Convertidor PrefixTransformer

Este convertidor se usa para agregar o suprimir un prefijo desde el valor de CMDB o el de origen de la base de datos federada.

Hay dos implementaciones:

- **com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.AdapterToCmdbAddPrefixTransformer**. Agrega el prefijo (dado como entrada) al convertir un valor de base de datos federada en un valor de CMDB y suprime el prefijo al convertir del valor de CMDB en un valor de la base de datos federada.
- **com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.AdapterToCmdbRemovePrefixTransformer**. Quita el prefijo (dado como entrada) al convertir un valor de base de datos federada en un valor de CMDB y agrega el prefijo al convertir del valor de CMDB en un valor de la base de datos federada.

Convertidor BytesToStringTransformer

Este convertidor se utiliza para convertir matrices de bytes en el CMDB en su representación de cadena del origen de la base de datos federada.

El convertidor es:

com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.CmdbToAdapterBytesToStringTransformer.

Convertidor StringDelimitedListTransformer

Este convertidor se utiliza para transformar una sola lista de cadenas a una lista de enteros/cadenas en CMDB.

El convertidor es: **com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.StringDelimitedListTransformer**.

El convertidor personalizado

Es posible escribir su propio convertidor personalizado (transformador) desde cero. Esto le permite crear cualquier convertidor necesario para sus necesidades.

Existen dos formas de escribir un convertidor personalizado:

1. Escriba un convertidor de Java compilado
 - a. Cree un proyecto Java en un IDE Java (como Eclipse, IntelliJ o Netbeans).
 - b. Agregue `federation-api.jar` y `db-interfaces.jar` a la ruta de clase.
 - c. Cree una clase Java que implementa las siguientes interfaces (de **db-interfaces.jar**):
 - `FcmdbDalTransformerFromExternalDB`
 - `FcmdbDalTransformerValuesToExternalDB`
 - `FcmdbDalTransformerInit`
 - d. Compile el proyecto y cree un archivo jar.
 - e. Coloque el archivo jar en el paquete del adaptador (bajo `adapterCode<Id. de adaptador>`)
 - f. Despliegue el paquete.
 - g. Agregue el nuevo nombre de clase del convertidor al archivo **transformations.txt**.
2. Escriba un convertidor (basado en secuencia de comandos) de Groovy

Un ejemplo se encuentra en el paquete GDBA original, **GroovyExampleTransformer.groovy**.

- a. Cree un archivo Groovy en el paquete del adaptador (bajo `adapterCode\<Id. de adaptador>`). Puede hacerlo directamente utilizando el menú Administración de adaptador.
- b. Cree una clase Groovy que implemente las siguientes interfaces (de **db-interfaces.jar**):
 - `FcmdbDalTransformerFromExternalDB`
 - `FcmdbDalTransformerValuesToExternalDB`
 - `FcmdbDalTransformerInit`
- c. Agregue el nuevo nombre de clase Groovy del convertidor al archivo **transformations.txt** como corresponda.

Nota: Groovy es un lenguaje de secuencias de comandos que amplía Java. El código Java regular también es código Groovy válido.

Complementos

El adaptador de bases de datos genéricas admite los complementos siguientes:

- Un complemento opcional para la completa sincronización de topología.
- Un complemento opcional para sincronizar los campos en la topología. Si no se implementa ningún complemento para sincronizar los cambios, es posible realizar una sincronización diferencial, pero dicha sincronización será en realidad una sincronización completa.
- Un complemento opcional para sincronizar el diseño.
- Se requiere un complemento opcional para recuperar las consultas admitidas para sincronización. Si este complemento no se define, se devuelven todos los nombres TQL.
- Un complemento interno, opcional para cambiar la definición TQL y el resultado TQL.
- Un complemento interno, opcional para cambiar la petición de diseño y el resultado de los CI.
- Un complemento interno, opcional para cambiar la petición de diseño y el resultado de los CI.
- Un complemento interno opcional para cambiar la acción de devolver los Id.

Para obtener más información acerca de implementar y desplegar complementos, consulte ["Implementación de un complemento" en la página 96](#).

Ejemplos de configuración

En esta sección se dan ejemplos de configuraciones.

Esta sección incluye los siguientes temas:

- ["Caso de uso" abajo](#)
- ["Conciliación de un único nodo" en la página siguiente](#)
- ["Conciliación de dos nodos" en la página 143](#)
- ["Uso de una clave principal que contiene más de una columna" en la página 146](#)
- ["Uso de transformaciones" en la página 148](#)

Caso de uso

Una consulta TQL es:

```
node > (composition) > card
```

donde:

- **node** es la entidad de CMDB
- **card** es la entidad de origen de la base de datos federada
- **composition** es la relación entre ellas

El ejemplo se ejecuta en la base de datos ED. Los ED nodes se almacenan en la tabla Device y card se almacena en la tabla hwCards. En los ejemplos siguientes, card siempre se asigna de la misma forma.

Conciliación de un único nodo

En este ejemplo la conciliación se ejecuta en la propiedad name.

Definición simplificada

La conciliación se realiza por node y se resalta con la etiqueta especial **CMDB-class**.

```
<?xml version="1.0" encoding="UTF-8"?>
<generic-DB-adapter-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="META-CONF/simplifiedConfiguration.xml">
  <CMDB-class CMDB-class-name="node" default-table-name="Device">
    <primary-key column-name="Device_ID"/>
    <reconciliation-by-single-node>
      <or>
        <attribute CMDB-attribute-name="name" column-name="Device_Name"/>
      </or>
    </reconciliation-by-single-node>
  </CMDB-class>
  <class CMDB-class-name="card" default-table-name="hwCards" connected-CMDB-class-name="node" link-class-name="composition">
    <foreign-primary-key column-name="Device_ID" CMDB-class-primary-key-column="Device_ID"
    <primary-key column-name="hwCards_Seq"/>
    <attribute CMDB-attribute-name="card_class" column-name="hwCardClasses"/>
    <attribute CMDB-attribute-name="card_vendor" column-name="hwCardVendor"/>
    <attribute CMDB-attribute-name="card_name" column-name="hwCardName"/>
  </class>
</generic-DB-adapter-config>
```

Definición avanzada

Archivo orm.xml

Preste atención a la adición de la asignación de relaciones. Para obtener información detallada, consulte la sección de definición en ["Archivo orm.xml" en la página 114](#).

Ejemplo del archivo orm.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<entity-mappings xmlns="http://java.sun.com/xml/ns/persistence/orm" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/
xml/ns/
persistence/orm http://java.sun.com/xml/ns/persistence/orm_1_0.xsd" version=
"1.0">
  <description>Generic DB adapter orm</description>
  <package>generic_db_adapter</package>
  <entity class="generic_db_adapter.node" >
    <table name="Device"/>
    <attributes>
      <id name="id1">
        <column name="Device_ID"
          insertable="false"
          updatable="false"/>
        <generated-value strategy="TABLE"/>
      </id>
      <basic name="name">
        <column name="Device_Name"/>
      </basic>
    </attributes>
  </entity>
  <entity class="generic_db_adapter.card" >
    <table name="hwCards"/>
    <attributes>
      <id name="id1">
        <column name="hwCards_Seq" insertable="false"
          updatable="false"/>
        <generated-value strategy="TABLE"/>
      </id>
      <basic name="card_class">
        <column name="hwCardClass" insertable="false"
          updatable="false"/>
      </basic>
      <basic name="card_vendor">
        <column name="hwCardVendor" insertable="false"
          updatable="false"/>
      </basic>
      <basic name="card_name">
        <column name="hwCardName" insertable="false"
          updatable="false"/>
      </basic>
    </attributes>
  </entity>
```

```
<entity class="generic_db_adapter.node_composition_card" >
  <table name="hwCards"/>
  <attributes>
    <id name="id1">
      <column name="hwCards_Seq" insertable="false"
        updatable="false"/>
      <generated-value strategy="TABLE"/>
    </id>
    <many-to-one name="end1" target-entity="node">
      <join-column name="Device_ID" insertable="false"
        updatable="false"/>
    </many-to-one>
    <one-to-one name="end2" target-entity="card">
      <join-column name="hwCards_Seq"
        referenced-column-name="hwCards_Seq" insertable="
        false" updatable="false"/>
    </one-to-one>
  </attributes>
</entity>
</entity-mappings>
```

Archivo reconciliation_rules.txt

Para obtener más información, consulte ["Archivo reconciliation_rules.txt \(para compatibilidad con versiones anteriores\)" en la página 128](#).

```
multinode[node] expression[node.name]
```

Archivo transformation.txt

Este archivo sigue vacío, ya que no es necesario convertir valores en este ejemplo.

Conciliación de dos nodos

En este ejemplo, la conciliación se calcula de acuerdo con la propiedad name de node y de ip_address con distintas variantes.

La consulta TQL de conciliación es **node > (containment) > ip_address**.

Definición simplificada

La conciliación se realiza por name de node OR de ip_address:

```
<?xml version="1.0" encoding="UTF-8"?>
<generic-DB-adapter-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../META-CONF/simplifiedConfiguration.xsd">
  <CMDB-class CMDB-class-name="node" default-table-name="Device">
    <primary-key column-name="Device_ID"/>
    <reconciliation-by-two-nodes connected-node-CMDB-class-name="ip_
```

```
address" CMDB-link-type="containment">
    <o>
        <attribute CMDB-attribute-name="name" column-name="Device_Na
me"/>
        <connected-node-attribute CMDB-attribute-name="name" column-
name="Device_PREFERREDIPAddress"/>
    </or>
</reconciliation-by-two-nodes>
</CMDB-class>
<class CMDB-class-name="card" default-table-name="hwCards" connected-CMD
B-class-name="node" link-class-name="containment">
    <foreign-primary-key column-name="Device_ID" CMDB-class-primary-key-
column="Device_ID"/>
    <primary-key column-name="hwCards_Seq"/>
    <attribute CMDB-attribute-name="card_class" column-name="hwCardClas
s"/>
    <attribute CMDB-attribute-name="card_vendor" column-name="hwCardVend
or"/>
    <attribute CMDB-attribute-name="card_name" column-name="hwCardName"/>
</class>
</generic-DB-adapter-config>
```

La conciliación se realiza por name de node AND de ip_address:

```
<?xml version="1.0" encoding="UTF-8"?>
<generic-DB-adapter-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-insta
nce" xsi:noNamespaceSchemaLocation="..META-CONF/simplifiedConfiguration.xs
sd">
    <CMDB-class CMDB-class-name="node" default-table-name="Device">
        <primary-key column-name="Device_ID"/>
        <reconciliation-by-two-nodes connected-node-CMDB-class-name="ip_addr
ess" CMDB-link-type="containment">
            <and>
                <attribute CMDB-attribute-name="name" column-name="Device_Na
me"/>
                <connected-node-attribute CMDB-attribute-name="name" column-
name="Device_PREFERREDIPAddress"/>
            </and>
        </reconciliation-by-two-nodes>
    </CMDB-class>
    <class CMDB-class-name="card" default-table-name="hwCards" connected-CMD
B-class-name="node" link-class-name="containment">
        <foreign-primary-key column-name="Device_ID" CMDB-class-primary-key-
column="Device_ID"/>
        <primary-key column-name="hwCards_Seq"/>
        <attribute CMDB-attribute-name="card_class" column-name="hwCardClas
s"/>
```

```
        <attribute CMDB-attribute-name="card_vendor" column-name="hwCardVendor"/>
        <attribute CMDB-attribute-name="card_name" column-name="hwCardName"/>
    </class>
</generic-DB-adapter-config>
```

La conciliación se realiza por name de ip_address:

```
<?xml version="1.0" encoding="UTF-8"?>
<generic-DB-adapter-config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="../META-CONF/simplifiedConfiguration.xsd">
    <CMDB-class CMDB-class-name="node" default-table-name="Device">
        <primary-key column-name="Device_ID"/>
        <reconciliation-by-two-nodes connected-node-CMDB-class-name="ip_address" CMDB-link-type="containment">
            <O>
                <connected-node-attribute CMDB-attribute-name="name" column-name="Device_PREFERREDIPAddress"/>
            </or>
        </reconciliation-by-two-nodes>
    </CMDB-class>
    <class CMDB-class-name="card" default-table-name="hwCards" connected-CMDB-class-name="node" link-class-name="containment">
        <foreign-primary-key column-name="Device_ID" CMDB-class-primary-key-column="Device_ID"/>
        <primary-key column-name="hwCards_Seq"/>
        <attribute CMDB-attribute-name="card_class" column-name="hwCardClasses"/>
        <attribute CMDB-attribute-name="card_vendor" column-name="hwCardVendor"/>
        <attribute CMDB-attribute-name="card_name" column-name="hwCardName"/>
    </class>
</generic-DB-adapter-config>
```

Definición avanzada

Archivo orm.xml

Como la expresión de la conciliación no se define en este archivo, se debe usar la misma versión para cualquier expresión de conciliación.

Archivo reconciliation_rules.txt

Para obtener más información, consulte "[Archivo reconciliation_rules.txt \(para compatibilidad con versiones anteriores\)](#)" en la página 128.

```
multinode[node] expression[ip_address.name OR node.name] end1_type[node]
```

```
end2_type[ip_address] link_type[containment]

multinode[node] expression[ip_address.name AND node.name] end1_type[node] en
d2_type[ip_address] link_type[containment]

multinode[node] expression[ip_address.name] end1_type[node] end2_type[ip_add
ress] link_type[containment]
```

Archivo transformation.txt

Este archivo sigue vacío, ya que no es necesario convertir valores en este ejemplo.

Uso de una clave principal que contiene más de una columna

Si la clave principal está compuesta por más de una columna, se añade el código siguiente a las definiciones XML:

Definición simplificada

Hay más de una etiqueta de clave principal y para cada columna hay una etiqueta.

```
<class CMDB-class-name="card" default-table-name="hwCards" connected-CMD
B-class-name="node" link-class-name="containment">
  <foreign-primary-key column-name="Device_ID" CMDB-class-primary-key-
column="Device_ID"/>
  <primary-key column-name="Device_ID"/>
  <primary-key column-name="hwBusesSupported_Seq"/>
  <primary-key column-name="hwCards_Seq"/>
  <attribute CMDB-attribute-name="card_class" column-name="hwCardClas
s"/>
  <attribute CMDB-attribute-name="card_vendor" column-name="hwCardVend
or"/>
  <attribute CMDB-attribute-name="card_name" column-name="hwCardName"/>
</class>
```

Definición avanzada

Archivo orm.xml

Se añade una nueva entidad `id` que se asigna a las columnas de las claves principales. Las entidades que usan esta entidad `id` deben agregar una etiqueta especial.

Si utiliza una clave externa (etiqueta `join-column`) para esta clave principal, debe asignar entre cada columna de la clave externa a una columna de la clave principal.

Para obtener más información, consulte ["Archivo orm.xml" en la página 114](#).

Ejemplo del archivo orm.xml:

```
<entity class="generic_db_adapter.card" >
  <table name="hwCards"/>
  <attributes>
```

```
        <id name="id1">
            <column name="Device_ID" insertable="false" updatable="false" />
        </id>
        <generated-value strategy="TABLE" />
    </id>
    <id name="id2">
        <column name="hwBusesSupported_Seq" insertable="false" updatable="false" />
    </id>
    <generated-value strategy="TABLE" />
    </id>
    <id name="id3">
        <column name="hwCards_Seq" insertable="false" updatable="false" />
    </id>
    <generated-value strategy="TABLE" />
    </id>
<entity class="generic_db_adapter.node_containment_card" >
    <table name="hwCards" />
    <attributes>
        <id name="id1">
            <column name="Device_ID" insertable="false" updatable="false" />
        </id>
        <generated-value strategy="TABLE" />
    </id>
    <id name="id2">
        <column name="hwBusesSupported_Seq" insertable="false" updatable="false" />
    </id>
    <generated-value strategy="TABLE" />
    </id>
    <id name="id3">
        <column name="hwCards_Seq" insertable="false" updatable="false" />
    </id>
    <generated-value strategy="TABLE" />
    </id>
    <many-to-one name="end1" target-entity="node">
        <join-column name="Device_ID" insertable="false" updatable="false" />
    </many-to-one>
    <one-to-one name="end2" target-entity="card">
        <join-column name="Device_ID" referenced-column-name="Device_ID" insertable="false" updatable="false" />
        <join-column name="hwBusesSupported_Seq" referenced-column-name="hwBusesSupported_Seq" insertable="false" updatable="false" />
        <join-column name="hwCards_Seq" referenced-column-name="hwCards_Seq" insertable="false" updatable="false" />
    </one-to-one>
</attributes>
```

```
</entity>  
</entity-mappings>
```

Uso de transformaciones

En el ejemplo siguiente, el transformador **enum** genérico se convierte de valores 1, 2, 3 a valores a, b, c respectivamente en la columna name.

El archivo de asignación es `generic-enum-transformer-example.xml`.

```
<enum-transformer CMDB-type="string" DB-type="string" non-existing-value-action="return-original" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="META-CONF/generic-enum-transformer.xsd">  
  <value CMDB-value="1" external-DB-value="a"/>  
  <value CMDB-value="2" external-DB-value="b"/>  
  <value CMDB-value="3" external-DB-value="c"/>  
</enum-transformer>
```

Definición simplificada

```
<CMDB-class CMDB-class-name="node" default-table-name="Device">  
  <primary-key column-name="Device_ID"/>  
  <reconciliation-by-two-nodes connected-node-CMDB-class-name="ip_address" CMDB-link-type="containment">  
    <o>  
      <attribute CMDB-attribute-name="name" column-name="Device_Name" from-CMDB-converter="com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.GenericEnumTransformer(generic-enum-transformer-example.xml)" to-CMDB-converter="com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.GenericEnumTransformer(generic-enum-transformer-example.xml)" />  
      <connected-node-attribute CMDB-attribute-name="name" column-name="Device_PreferredIPAddress" />  
    </or>  
  </reconciliation-by-two-nodes>  
</CMDB-class>
```

Definición avanzada

Solo hay un cambio en el archivo **transformation.txt**.

Archivo **transformation.txt**

Asegúrese de que los nombres de atributos y los nombres de entidad son iguales en el archivo `orm.xml`.

```
entity[node] attribute[name]  
to_DB_class[com.mercury.topaz.fcldb.adapters.dbAdapter.dal.transform.impl.
```

```
GenericEnumTransformer(generic-enum-transformer-example.xml)] from_DB_class  
[com.mercury.topaz.fcmdb.adapters.dbAdapter.dal.transform.impl.  
GenericEnumTransformer(generic-enum-transformer-example.xml)]
```

Archivos de registro del adaptador

Consulte los archivos de registro siguientes para comprender los flujos de cálculo y el ciclo de vida del adaptador, y para ver la información de depuración.

Esta sección incluye los siguientes temas:

- ["Niveles de registro." abajo](#)
- ["Ubicaciones de registro." abajo](#)

Niveles de registro.

Puede configurar el nivel de registro para cada uno de los registros.

En un editor de texto, abra el archivo **C:\hp\UCMDB\UCMDBServer\conf\log\fcmdb.gdba.properties**

El nivel de registro predeterminado es **ERROR**:

```
#loglevel can be any of DEBUG INFO WARN ERROR FATAL  
loglevel=ERROR
```

- Para aumentar el nivel de registro para todos los archivos de registro, cambie **loglevel=ERROR** a **loglevel=DEBUG** o **loglevel=INFO**.
- Para cambiar el nivel de registro para un archivo específico, cambie la línea de categoría **log4j** específica en consecuencia. Por ejemplo, para cambiar el nivel de registro de **fcmdb.gdba.dal.sql.log** a **INFO**, cambie

```
log4j.category.fcmdb.gdba.dal.SQL=${loglevel},fcmdb.gdba.dal.SQL.appender
```

por:

```
log4j.category.fcmdb.gdba.dal.SQL=INFO,fcmdb.gdba.dal.SQL.appender
```

Ubicaciones de registro.

Los archivos de registro se encuentran en el directorio **C:\hp\UCMDB\UCMDBServer\runtime\log**.

- **Fcmdb.gdba.log**

El registro del ciclo de vida del adaptador. Da detalles sobre cuándo se ha iniciado o detenido el adaptador, y cuáles son los CIT que admite este adaptador.

Consulte errores de iniciación (carga/descarga del adaptador).

- **fcmdb.log**

Consulte las excepciones.

- **cmdb.log**

Consulte las excepciones.

- **Fcmdb.gdba.mapping.engine.log**

El registro del motor de asignación. Da detalles sobre la consulta TQL de conciliación que usa el motor de asignación, así como las topologías de conciliación que se comparan durante la fase de conexión.

Consulte este registro cuando una consulta TQL no da resultados aunque sepa que hay CI relevantes en la base de datos, o los resultados no sean los previstos (compruebe la conciliación).

- **Fcmdb.gdba.TQL.log**

El registro de TQL. Da detalles sobre las consultas TQL y sus resultados.

Consulte este registro cuando una consulta TQL no devuelve resultados y el registro del motor de asignación muestra que no hay resultados en el origen de datos federado.

- **Fcmdb.gdba.dal.log**

El registro del ciclo de vida de DAL. Da detalles sobre la generación de CIT y detalles de la conexión de la base de datos.

Consulte este registro cuando no puede conectarse a la base de datos o cuando hay CIT o atributos no admitidos por la consulta.

- **Fcmdb.gdba.dal.command.log**

El registro de operaciones de DAL. Da detalles sobre las operaciones internas de DAL que se han llamado. (Este registro es similar a `cmdb.dal.command.log`).

- **Fcmdb.gdba.dal.SQL.log**

El registro de consultas SQL de DAL. Da detalles sobre las JPAQL (consultas SQL orientadas a objetos) llamadas y sus resultados.

Consulte este registro cuando no puede conectarse a la base de datos o cuando hay CIT o atributos no admitidos por la consulta.

- **Fcmdb.gdba.hibernate.log**

El registro de Hibernate. Da detalles sobre las consultas SQL que se ejecutan, el análisis de cada JPAQL en SQL, los resultados de las consultas, los datos concernientes a la caché de

Hibernate, etc. Para más información sobre Hibernate, consulte "[Hibernate como proveedor de JPA](#)" en la página 76.

Referencias externas

Para obtener más información sobre la especificación JavaBeans 3.0, consulte <http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html>.

Solución de problemas y limitaciones

En esta sección se describe la solución de problemas y las limitaciones del adaptador de bases de datos genéricas.

Limitaciones generales

Al actualizar un paquete de adaptador, para editar los archivos de plantillas, utilice Notepad++, UltraEdit, o cualquier otro editor de texto de otros fabricantes que no sea Bloc de notas (cualquier versión) de Microsoft Corporation. De esta forma se evitará el uso de símbolos especiales, que pueden hacer que falle el despliegue del paquete preparado.

Limitaciones JPA

- Todas las tablas deben tener una columna de claves principales.
- Los nombres de atributo de la clase CMDB deben seguir la convención de nomenclatura de JavaBeans (por ejemplo, los nombres deben comenzar con minúscula).
- Dos CI que están conectados con una relación en el modelo de clase deben tener una asociación directa en la base de datos (por ejemplo, si `node` está conectado a `ticket` debe haber una clave externa o tabla de unión que los conecte).
- Varias tablas que están asignadas al mismo CIT deben compartir la misma tabla de claves principales.

Limitaciones funcionales

- No puede crear una relación manual entre el CMDB y los CIT federados. Para poder definir relaciones virtuales, se debe definir una lógica de relación especial (puede estar basada en las propiedades de la clase federada).
- Los CIT federados no pueden activar CIT en una regla de impacto, pero se pueden incluir en una consulta TQL de análisis de impacto.
- Un CIT federado puede formar parte de un TQL de enriquecimiento, pero no se puede usar en el nodo en el que se realiza el enriquecimiento (no puede agregar, actualizar o eliminar el CIT federado).
- No se admite el uso de un calificador de clase en una condición.

- No se admiten los subgráficos.
- No se admiten las relaciones compuestas.
- El id de CMDB de CI externo está compuesto de su clave principal y no de sus atributos de clave.
- Una columna de tipo bytes no se puede usar como columna de clave principal en Microsoft SQL Server.
- El cálculo de la consulta TQL falla si las condiciones de atributos que se definen en un nodo federado no tenían asignados sus nombres en el archivo **orm.xml**.

Capítulo 5: Desarrollo de adaptadores Java

Este capítulo incluye:

Información general de Federation Framework	153
Adaptador y asignación de interacción con Federation Framework	158
Federation Framework para consultas TQL federadas	159
Interacciones entre Federation Framework, servidor, adaptador y motor de asignación	160
Flujo de Federation Framework para Rellenado	169
Interfaces del adaptador	170
Depuración de recursos del adaptador	172
Adición de un adaptador para un nuevo origen de datos externo	172
Creación de un adaptador de ejemplo	181
Propiedades y etiquetas de configuración XML	182
La interfaz DataAdapterEnvironment	184

Información general de Federation Framework

Nota:

- El término **relación** es equivalente al término **vínculo**.
- El término **CI** es equivalente al término **objeto**.
- Un gráfico es una colección de nodos y vínculos.

La funcionalidad Federation Framework utilice una API para recuperar la información de orígenes federados. Federation Framework proporciona tres capacidades principales:

- **Federación** sobre la marcha. Todas las consultas se ejecutan en los repositorios de datos originales y los resultados se generan sobre la marcha en CMDB.
- **Rellenado**. Rellena los datos (datos topológicos y propiedades de CI) en CMDB desde un origen de datos externo.
- **Inserción de datos**. Inserta los datos (datos topológicos y propiedades de CI) desde el CMDB local a un origen de datos remoto.

Todos los tipos de acciones requieren un adaptador para cada repositorio de datos, que puede proporcionar las capacidades específicas del repositorio de datos y recuperar o actualizar los datos requeridos. Cada petición del repositorio de datos se realiza a través de este adaptador.

Esta sección incluye también los siguientes temas:

- ["Federación sobre la marcha" abajo](#)
- ["Inserción de datos" en la página siguiente](#)
- ["Rellenado" en la página 156](#)

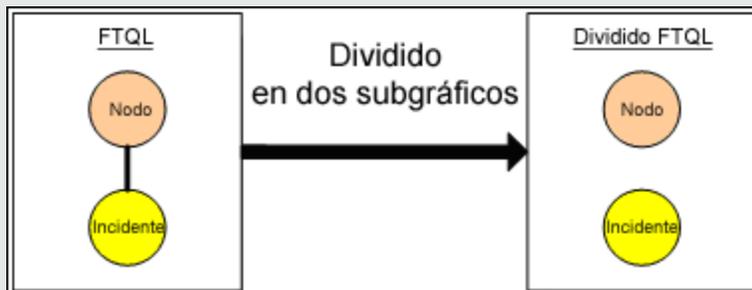
Federación sobre la marcha

Las consultas TQL federadas permiten la recuperación de los datos desde un repositorio de datos externo sin replicar sus datos.

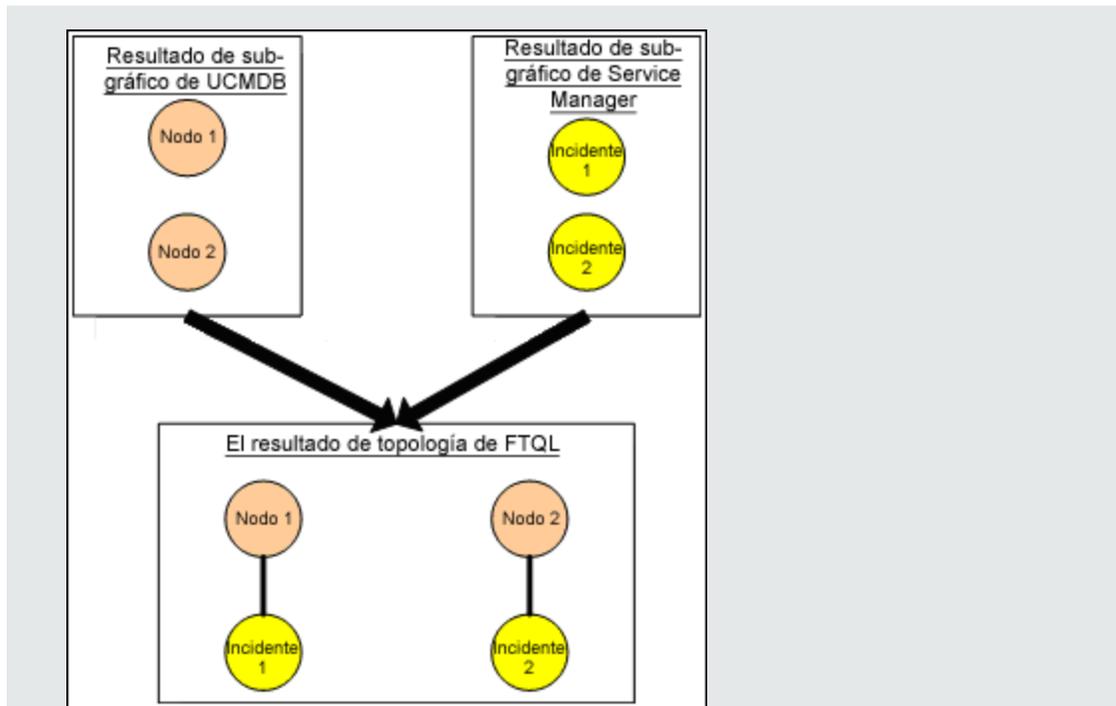
Una consulta TQL federada usa adaptadores que representan repositorios de datos externos para crear relaciones externas entre CI de diferentes repositorios de datos externos y los CI de UCMDB.

Ejemplo de flujo de federación sobre la marcha:

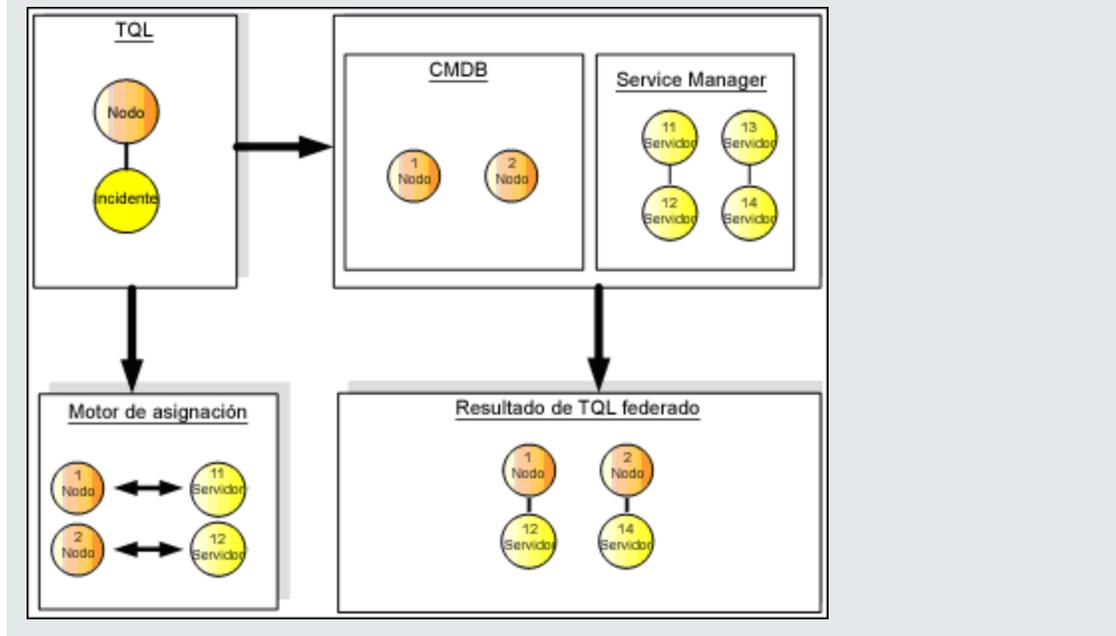
1. Federation Framework divide una consulta TQL federada en varios subgráficos, donde todos los nodos de un subgráfico se refieren al mismo repositorio de datos. Cada subgráfico está conectado a los otros subgráficos por una relación virtual (pero él mismo no contiene relaciones virtuales).



2. Después de que la consulta TQL federada se divida en subgráficos, Federation Framework calcula cada topología de subgráfico y conecta dos subgráficos apropiados al crear relaciones virtuales entre los nodos apropiados.



3. Después de que se calcule la topología TQL federada, Federation Framework recupera un diseño para el resultado de la topología.



Inserción de datos

Puede usar el flujo de inserción de datos para sincronizar los datos desde el CMDB local actual a un servicio remoto o un repositorio de datos de destino.

En la inserción de datos, los repositorios de datos se dividen en dos categorías: origen (CMDB local) y destino. Los datos se recuperan desde el repositorio de datos de origen y se actualizan en el repositorio de datos de destino. El proceso de inserción de datos se basa en los nombres de consulta, lo que significa que los datos se sincronizan entre los repositorios de datos de origen (CMDB local) y destino y se recupera un nombre de consulta TQL desde el CMDB local.

El flujo del proceso de inserción de datos incluye los pasos siguientes:

1. Recuperación del resultado de la topología con firmas desde el repositorio de datos de origen.
2. Comparación de los nuevos resultados con los anteriores.
3. Recuperación de un diseño completo (es decir, todas las propiedades de CI) de los CI y relaciones, solo para resultados modificados.
4. Actualización del repositorio de datos de destino con el diseño completo recibido de CI y relaciones. Si se elimina cualquier CI o relación en el repositorio de datos de origen y la consulta es exclusiva, el proceso de replicación suprime también los CI o relaciones en el repositorio de datos de destino.

CMDB tiene dos orígenes de datos ocultos (**hiddenRMIDataSource** y **hiddenChangesDataSource**), que siempre son el origen de datos "de origen" en los flujos de inserción de datos. Para implementar un nuevo adaptador para los flujos de inserción de datos, solo hay que implementar el adaptador "destino".

Rellenado

Puede usar el flujo de relleno para rellenar CMDB con datos de orígenes externos.

El flujo siempre usa un origen de datos "de origen" para recuperar los datos e inserta los datos recuperados a la sonda en un proceso similar al flujo de un trabajo de detección.

Para implementar un nuevo adaptador para flujos de relleno, solo tiene que implementar el adaptador de origen, ya que Data Flow Probe actúa como el destino.

El adaptador del flujo de relleno se ejecuta en la sonda. La depuración y el registro debería realizarse en la sonda y no en CMDB.

El flujo de relleno se basa en los nombres de consultas, es decir, los datos se sincronizan entre el repositorio de datos de origen y Data Flow Probe y se recupera por un nombre de consulta en el repositorio de datos de origen. Por ejemplo, en UCMDB, el nombre de la consulta es el nombre de la consulta TQL. Sin embargo, en otro repositorio de datos, el nombre de la consulta puede ser un nombre de código que devuelve datos. El adaptador está diseñado para manejar correctamente el nombre de consulta.

Cada trabajo se puede definir como un trabajo exclusivo. Esto significa que los CI y la relaciones en los resultados del trabajo son únicos en el CMDB local y que ninguna otra consulta puede llevarlos al destino. El adaptador del repositorio de datos de origen admite consultas específicas y puede recuperar los datos desde este repositorio de datos. El adaptador del repositorio de datos de destino permite la actualización de datos recuperados en este repositorio de datos.

Flujo SourceDataAdapter

- Recupera el resultado de la topología con firmas desde el repositorio de datos de origen.
- Compara los nuevos resultados con los anteriores.
- Recupera un diseño completo (es decir, todas las propiedades de CI) de los CI y relaciones, solo para resultados modificados.
- Actualiza el repositorio de datos de destino con el diseño completo recibido de CI y relaciones. Si se elimina cualquier CI o relación en el repositorio de datos de origen y la consulta es exclusiva, el proceso de replicación suprime también los CI o relaciones en el repositorio de datos de destino.

Flujo SourceChangesDataAdapter

- Recupera el resultado de topología que tuvo lugar desde la última fecha dada.
- Recupera un diseño completo (es decir, todas las propiedades de CI) de los CI y relaciones, solo para resultados modificados.
- Actualiza el repositorio de datos de destino con el diseño completo recibido de CI y relaciones. Si se elimina cualquier CI o relación en el repositorio de datos de origen y la consulta es exclusiva, el proceso de replicación suprime también los CI o relaciones en el repositorio de datos de destino.

Flujo PopulateDataAdapter

- Recupera la topología completa con el resultado del diseño solicitado.
- Usa el mecanismo de fragmentos de topología para recuperar los datos en fragmentos.
- La sonda quita los datos que ya se trajeron en ejecuciones anteriores.
- Actualiza el repositorio de datos de destino con el diseño recibido de CI y relaciones. Si se elimina cualquier CI o relación en el repositorio de datos de origen y la consulta es exclusiva, el proceso de replicación suprime también los CI o relaciones en el repositorio de datos de destino.

Flujo PopulateChangesDataAdapter

- Recupera la topología con el resultado del diseño solicitado que tiene cambios desde la última ejecución.
- Usa el mecanismo de fragmentos de topología para recuperar los datos en fragmentos.
- La sonda quita los datos que ya se trajeron en ejecuciones anteriores (incluyendo este flujo).
- Actualiza el repositorio de datos de destino con el diseño recibido de CI y relaciones. Si se elimina cualquier CI o relación en el repositorio de datos de origen y la consulta es exclusiva, el proceso de replicación suprime también los CI o relaciones en el repositorio de datos de destino.

Flujo de relleno basado en instancia

Si el adaptador se ha definido para admitir un flujo basado en instancia (mediante la etiqueta **<datos basados en instancia>**, tal como se describe en ["Propiedades y etiquetas de configuración XML" en la página 182](#)), el motor de rellenado encuentra automáticamente los CI eliminados dentro de la instancia y los elimina de UCMDB (suponiendo que la eliminación esté permitida para el trabajo de rellenado específico). Cada instancia debe tener un CI raíz, marcado en la definición de TQL con el nombre **Root**. Cada vez que un CI raíz es correcto, su instancia completa (todos los CI conectados a ella) se compara con la última vez que se envió a UCMDB y cualquier CI que estuviese conectado a la raíz pero que ahora no esté conectado a ella se eliminará de UCMDB. Para que el adaptador admita correctamente un flujo basado en instancia, cualquier cambio en cualquier CI o atributo en toda la instancia debe activar un reenvío de toda la instancia a UCMDB.

Adaptador y asignación de interacción con Federation Framework

Un adaptador es una entidad en UCMDB que representa datos externos (datos no guardados en UCMDB). En los flujos federados, todas las interacciones con los orígenes de datos externos se realizan mediante adaptadores. Las interfaces del adaptador y el flujo de interacción Federation Framework son diferentes para la réplica y las consultas TQL federadas.

Esta sección incluye también los siguientes temas:

- ["Ciclo de vida del adaptador" abajo](#)
- ["Métodos de ayuda del adaptador" abajo](#)

Ciclo de vida del adaptador

Se crea una instancia del adaptador para cada repositorio de datos externos. El adaptador inicia su ciclo de vida con la primera acción aplicada a él (como **calcular TQL** o **recuperar/actualizar datos**). Cuando se llama al método **start**, el adaptador recibe información medioambiental, como la configuración del repositorio de datos, el registro, etc. El ciclo de vida del adaptador termina cuando el repositorio de datos se elimina de la configuración y se llama al método **shutdown**. Esto significa que el adaptador tiene control del estado y puede contener la conexión con el repositorio de datos externo si es necesario.

Métodos de ayuda del adaptador

El adaptador tiene varios métodos de ayuda que pueden agregar configuraciones del repositorio de datos externo. Estos métodos no forman parte del ciclo de vida del adaptador y crean un adaptador cada vez que se les llama.

- El primer método comprueba la conexión al repositorio de datos externo para una determinada configuración. **testConnection** puede ejecutarse en el servidor UCMDB o Data Flow Probe, según el tipo de adaptador.
- El segundo método solo es relevante para el adaptador de origen y devuelve las consultas admitidas para réplica. (Este método solo se ejecuta en la sonda).
- El tercer método solo es relevante para los flujos de federación y rellenado, y devuelve clases

externas admitidas por el repositorio de datos externo. (Este método se ejecuta en el servidor de UCMDB).

Todos estos métodos se usan al crear o ver configuraciones de integración.

Federation Framework para consultas TQL federadas

Esta sección incluye los siguientes temas:

- ["Definiciones y términos" abajo](#)
- ["Motor de asignación" abajo](#)
- ["Adaptador federado" en la página siguiente](#)

Consulte ["Interacciones entre Federation Framework, servidor, adaptador y motor de asignación" en la página siguiente](#) para ver los diagramas que ilustran las interacciones entre Federation Framework, UCMDB, el adaptador y el motor de asignación.

Definiciones y términos

Datos de conciliación. La regla para comparar los CI del tipo especificado que se reciben desde CMDB y el repositorio de datos externo. La regla de conciliación puede ser de tres tipos:

- **Conciliación de ID.** Solo puede usarse si el repositorio de datos externo contiene el Id. de CMDB de los objetos de conciliación.
- **Conciliación de la propiedad.** Se usa cuando la comparación puede realizarse mediante propiedades solo del tipo CI de conciliación.
- **Conciliación de topologías.** Se usa cuando se necesitan las propiedades de CIT adicionales (no solo de CIT de conciliación) para realizar una comparación de los CI de conciliación. Por ejemplo, puede realizar una conciliación del tipo de nodo por la propiedad name, que pertenece al CIT ip_address .

Objeto de conciliación. El adaptador crea el objeto de acuerdo con los datos de conciliación recibidos. Este objeto debe hacer referencia a un CI externo y lo usa el motor de asignación para conectar entre los CI externos y los CI de CMDB.

Tipo de CI de conciliación. El tipo de CI que representa objetos de conciliación. Estos CI deben almacenarse en los repositorios de datos externos y CMDB.

Motor de asignación. Un componente que identifica relaciones entre CI de diferentes repositorios de datos con una relación virtual entre ellos. La identificación se realiza reconciliando los objetos de conciliación de CMDB y los objetos de conciliación de CI externos.

Motor de asignación

Federation Framework usa el motor de asignación para calcular la consulta TQL federada. El motor de asignación conecta entre los CI que se reciben de diferentes repositorios de datos y se conectan mediante relaciones virtuales. El motor de asignación también proporciona datos de conciliación para la relación virtual. Un extremo de la relación virtual debe hacer referencia al CMDB. Este

extremo es un tipo de conciliación. Para el cálculo de los dos subgráficos, puede iniciarse una relación virtual desde cualquier nodo del extremo.

Adaptador federado

El adaptador federado proporciona dos tipos de datos de repositorios de datos externos: Datos de CI externos y objetos de conciliación que pertenecen a unos CI externos.

- **Datos de CI externos.** Datos externos que no existen en CMDB. Son los datos del destino del repositorio de datos externo.
- **Datos de objetos de conciliación.** Datos auxiliares usados por Federation Framework para conectar los CI de CMDB y los datos externos. Cada objeto de conciliación debe hacer referencia a un CI externo. El tipo de objetos de conciliación es el tipo (o subtipo) de uno de los extremos de la relación virtual del que se recuperan los datos. Los objetos de conciliación deben ajustar el adaptador recibido a los datos de conciliación. El objeto de conciliación puede ser de uno de los tres tipos siguientes: `IdReconciliationObject`, `PropertyReconciliationObject` o `TopologyReconciliationObject`.

En las interfaces basadas en `DataAdapter` (`DataAdapter`, `PopulateDataAdapter` y `PopulateChangesDataAdapter`), se solicita la conciliación como parte de la definición de la consulta.

Interacciones entre Federation Framework, servidor, adaptador y motor de asignación

Los siguientes diagramas ilustran las interacciones entre Federation Framework, el servidor UCMDDB, el adaptador y el motor de asignación. La consulta TQL federada en los diagramas de ejemplo solo tiene una relación virtual, por lo que solo UCMDDB y un repositorio de datos externo están implicados en la consulta TQL federada.

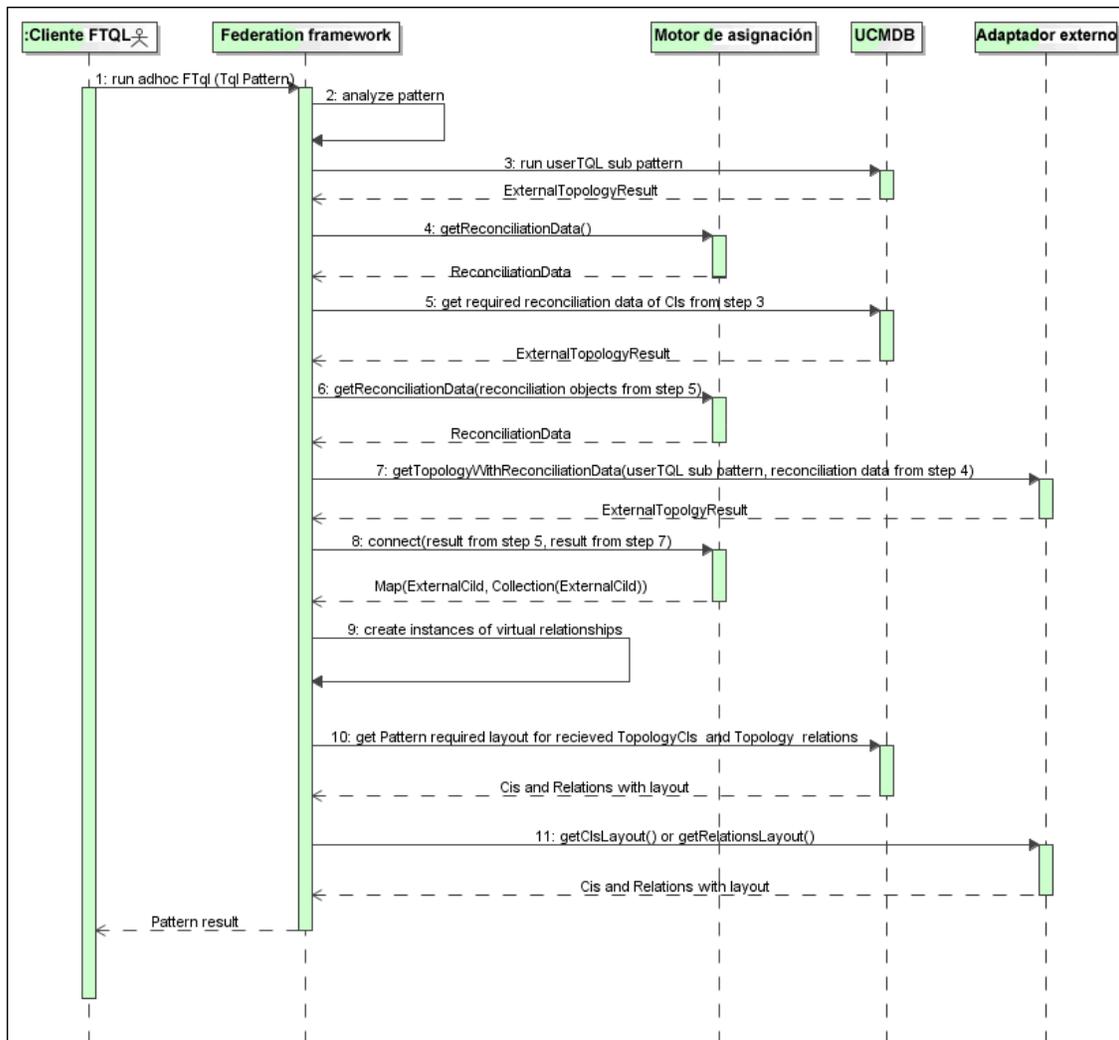
Esta sección incluye los siguientes temas:

- ["El cálculo empieza en el extremo del servidor" abajo](#)
- ["El cálculo empieza en el extremo del adaptador externo" en la página 163](#)
- ["Ejemplo del flujo de Federation Framework para consultas TQL federadas" en la página 164](#)

En el primer diagrama, el cálculo empieza en UCMDDB y en el segundo diagrama, en el adaptador externo. Cada paso del diagrama incluye referencias a la llamada al método adecuada del adaptador o la interfaz del motor de asignación.

El cálculo empieza en el extremo del servidor

El siguiente diagrama de secuencia ilustra las interacciones entre Federation Framework, UCMDDB, el adaptador y el motor de asignación. La consulta TQL federada en el diagrama de ejemplo solo tiene una relación virtual, por lo que solo UCMDDB y un repositorio de datos externo están implicados en la consulta TQL federada.

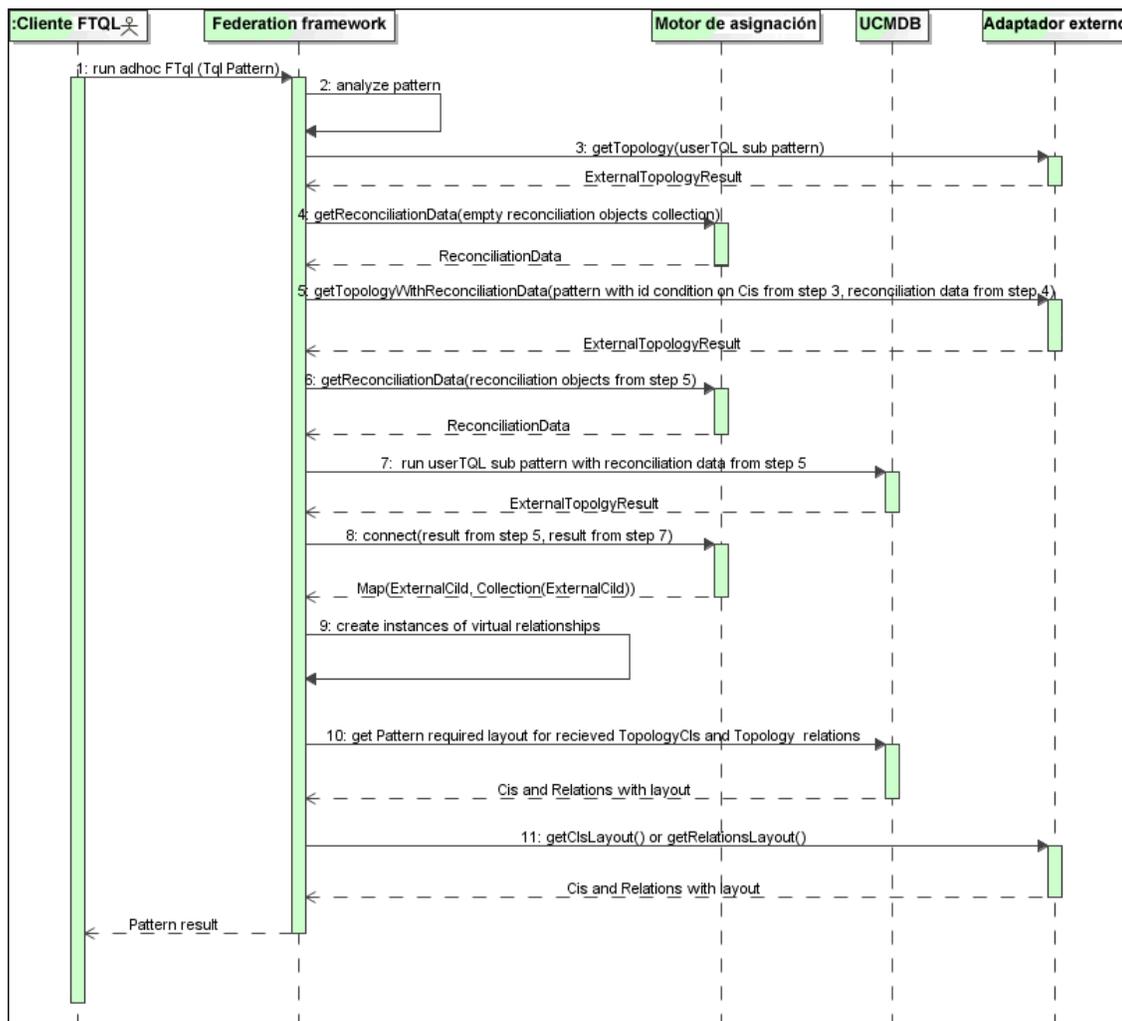


Los números de esta imagen se explican a continuación:

Número	Explicación
1	Federation Framework recibe una llamada para un cálculo TQL federado.
2	Federation Framework analiza el adaptador, encuentra la relación virtual y divide el TQL original en dos subadaptadores; uno para UCMDB y otro para el repositorio de datos externo.
3	Federation Framework solicita la topología del subTQL desde UCMDB.

Número	Explicación
4	<p>Después de recibir el resultado de la topología, Federation Framework llama al motor de asignación adecuado para la relación virtual actual y solicita los datos de conciliación. El parámetro <code>reconciliationObject</code> está vacío en esta fase; es decir que no se añadió ninguna condición a los datos de conciliación en esta llamada. Los datos de conciliación devueltos definen los datos necesarios para comparar los CI de conciliación en UC MDB con el repositorio de datos externo. Los datos de conciliación pueden ser de uno de los siguientes tipos:</p> <ul style="list-style-type: none">• IdReconciliationData. Los CI se reconcilian de acuerdo a su Id.• PropertyReconciliationData. Los CI se reconcilian de acuerdo a las propiedades de uno de los CI.• TopologyReconciliationData. Los CI se reconcilian de acuerdo con la topología (por ejemplo, para reconciliar los CI de nodo, también es necesaria la dirección IP de IP).
5	<p>Federation Framework solicita datos de conciliación para los CI de los extremos de la relación virtual que se recibieron en el paso "3" en la página precedente de UC MDB.</p>
6	<p>Federation Framework llama al motor de asignación para recuperar los datos de conciliación. En este estado (por contraste con el paso "3" en la página precedente), el motor de asignación recibe los objetos de conciliación desde el paso "5" arriba como parámetros. El motor de asignación convierte el objeto de conciliación recibido en la condición de los datos de conciliación.</p>
7	<p>Federation Framework solicita la topología del subTQL desde el repositorio de datos externo. El adaptador externo recibe los datos de conciliación desde el paso "6" arriba como un parámetro.</p>
8	<p>Federation Framework llama al motor de asignación para conectar los resultados recibidos. El parámetro <code>firstResult</code> es el resultado de topología externa recibido de UC MDB en el paso "5" arriba y el parámetro <code>secondResult</code> es el resultado de topología externa recibido del adaptador externo en el paso "7" arriba. El motor de asignación devuelve un mapa en que el Id. de CI externo del primer repositorio de datos (UC MDB en este caso) se ha asignado a los Id. de CI externos del segundo repositorio de datos (externo).</p>
9	<p>Para cada asignación, Federation Framework crea una relación virtual.</p>
10	<p>Después del cálculo de los resultados de la consulta TQL federada (solo en la fase de topología), Federation Framework recupera el diseño de TQL original para los CI resultantes y las relaciones desde los repositorios de datos adecuados.</p>

El cálculo empieza en el extremo del adaptador externo



Los números de esta imagen se explican a continuación:

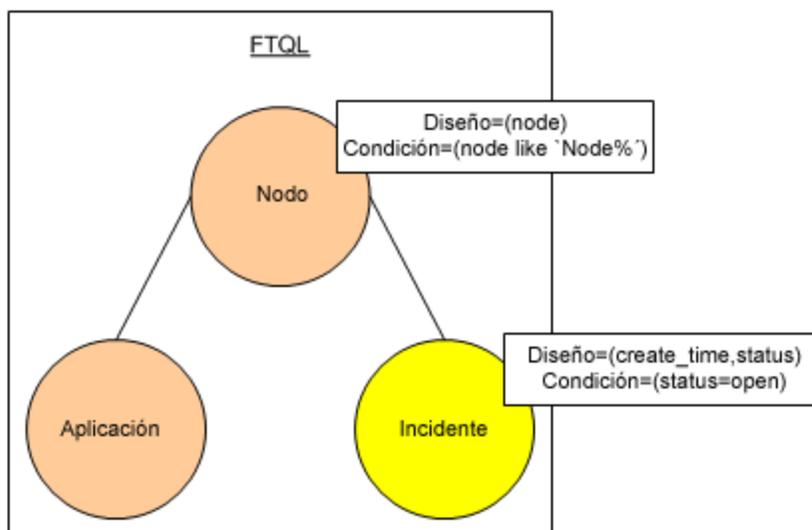
Número	Explicación
1	Federation Framework recibe una llamada de un cálculo de TQL federado.
2	Federation Framework analiza el adaptador, encuentra la relación virtual y divide el TQL original en dos subadaptadores; uno para UCMDB y otro para el repositorio de datos externo.
3	Federation Framework solicita la topología del subTQL del adaptador externo. Se supone que el ExternalTopologyResult devuelto no contiene ningún objeto de conciliación, ya que los datos de conciliación no forman parte de la solicitud.

Número	Explicación
4	<p>Después de recibir los resultados de la topología Federation Framework llama al motor de asignación adecuado con la relación virtual actual y solicita los datos de conciliación. El parámetro <code>reconciliationObjects</code> está vacío en este estado; es decir, que no se añade ninguna condición a los datos de conciliación en esta llamada. Los datos de conciliación devueltos definen los datos necesarios para comparar los CI de conciliación en UCMDDB con el repositorio de datos externo. Los datos de conciliación pueden ser de uno de los tres tipos siguientes:</p> <ul style="list-style-type: none">• IdReconciliationData. Los CI se reconcilian de acuerdo a su Id.• PropertyReconciliationData. Los CI se reconcilian de acuerdo a las propiedades de uno de los CI.• TopologyReconciliationData. Los CI se reconcilian de acuerdo con la topología (por ejemplo, para reconciliar los CI de nodo, también es necesaria la dirección IP de IP).
5	<p>Federation Framework solicita objetos de conciliación de los CI que se recibieron en el paso 3 desde el repositorio de datos externo. Federation Framework llama al método getTopologyWithReconciliationData() en el adaptador externo, donde la topología solicitada es una topología de un nodo con los CI recibidos en el paso 3 como la condición ID y los datos de conciliación del paso 4.</p>
6	<p>Federation Framework llama al motor de asignación para recuperar los datos de conciliación. En este estado (por contraste con el paso 3), el motor de asignación recibe los objetos de conciliación desde el paso 5 como parámetros. El motor de asignación convierte el objeto de conciliación recibido en la condición de los datos de conciliación.</p>
7	<p>Federation Framework solicita la topología del subTQL con datos de conciliación del paso 6 de UCMDDB.</p>
8	<p>Federation Framework llama al motor de asignación para conectar los resultados recibidos. El parámetro <code>firstResult</code> es el resultado de la topología externa recibido del adaptador externo en el paso 5 y el parámetro <code>secondResult</code> es el resultado de la topología externa recibido de UCMDDB7. El motor de asignación devuelve un mapa en el que el Id. de CI externo del primer repositorio de datos (en este caso el repositorio de datos externo) se asigna a los Id. de CI externo desde el segundo repositorio de datos (UCMDDB).</p>
9	<p>Para cada asignación, Federation Framework crea una relación virtual.</p>
10	<p>Después del cálculo de los resultados de la consulta TQL federada (solo en la fase de topología), Federation Framework recupera el diseño de TQL original para los CI resultantes y las relaciones desde los repositorios de datos adecuados.</p>

Ejemplo del flujo de Federation Framework para consultas TQL federadas

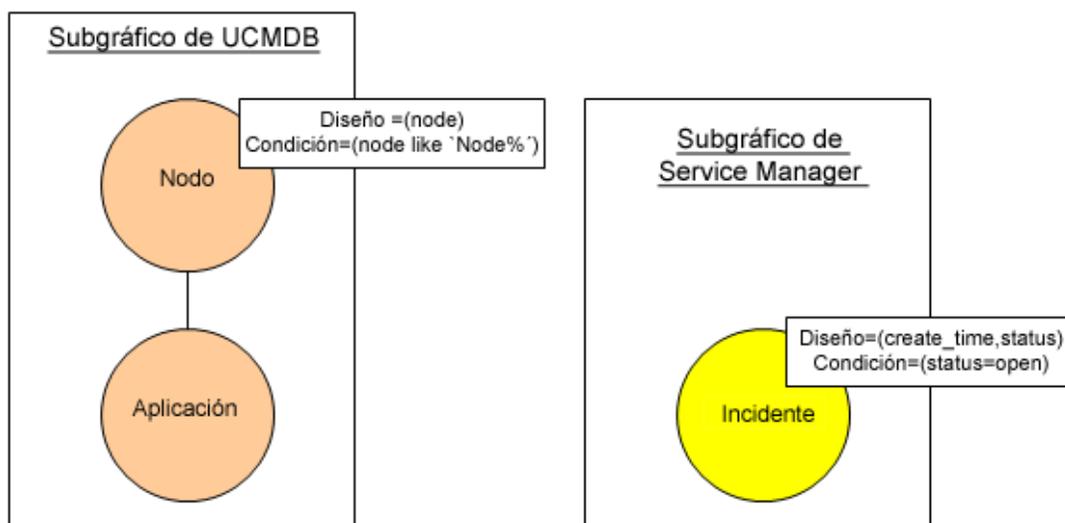
En este ejemplo se explica el modo de ver todos los incidentes abiertos en nodos específicos. El

repositorio de datos ServiceCenter es el repositorio de datos externo. Las instancias del nodo se almacenan en UCMDB y las instancias del incidente se almacenan en ServiceCenter. Se considera que para conectar las instancias del incidente al nodo adecuado, son necesarias las propiedades `node` y `ip_address` del host y la dirección IP. Estas son las propiedades de conciliación que identifican los nodos de ServiceCenter en UCMDB.

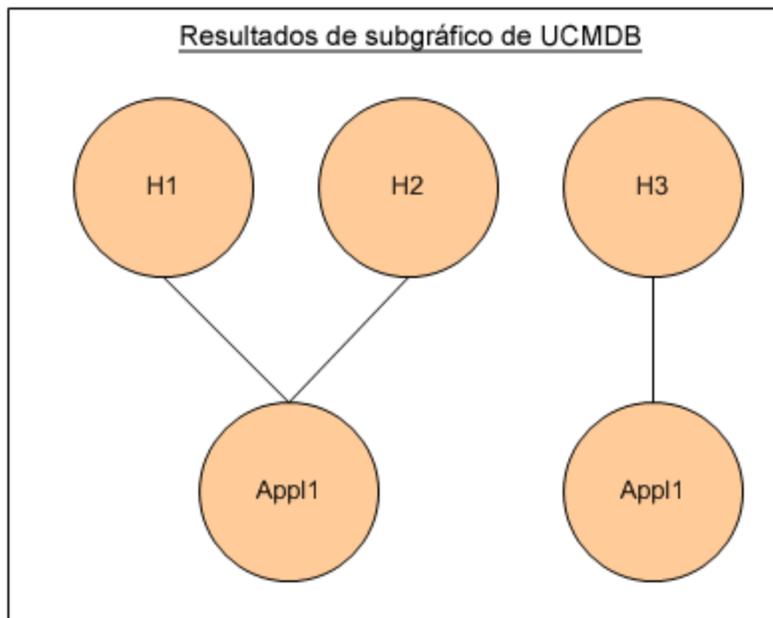


Nota: Para la federación de atributos, se llama al método **getTopology** del adaptador. Los datos de conciliación se adaptan en el TQL del usuario (en este caso, el elemento CI).

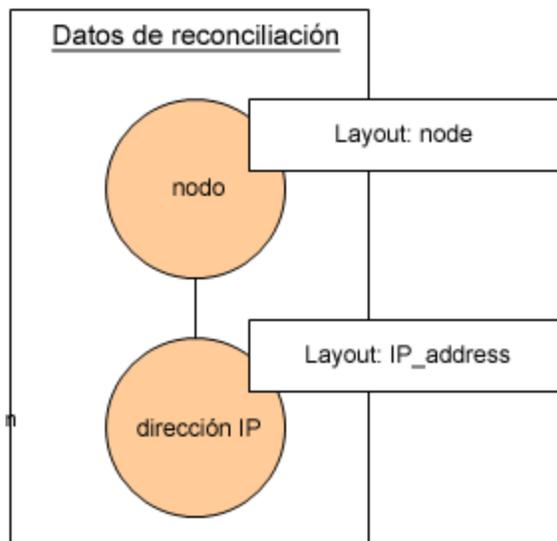
1. Después de analizar el adaptador, Federation Framework reconoce la relación virtual entre `Node` e `Incident` y divide la consulta TQL federada en dos subgráficos:



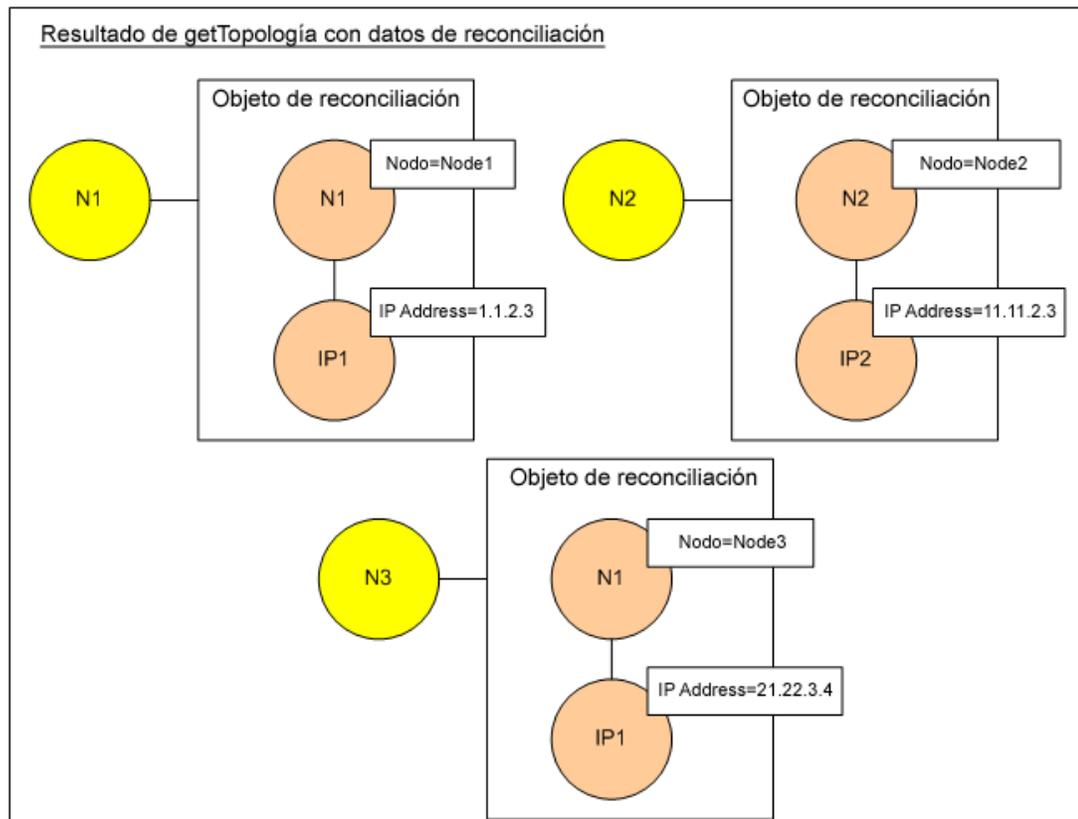
2. Federation Framework ejecuta el subgráfico UCMDB para solicitar la topología y recibe los siguientes resultados:



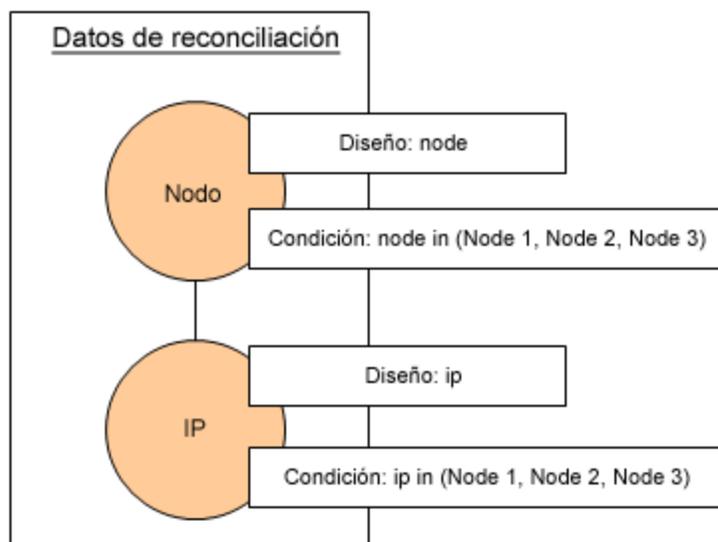
3. Federation Framework solicita, desde el motor de asignación adecuado, los datos de conciliación del primer repositorio de datos (UCMDB) que contiene la información para conectar entre los datos recibidos de dos repositorios de datos. En este caso, los datos de conciliación son:



4. Federation Framework crea una consulta de topología de un nodo con las condiciones Node e ID en ella del resultado anterior (node en H1, H2, H3) y ejecute esta consulta con los datos de conciliación necesarios en UCMDB. El resultado incluye los CI Node relevantes para la condición ID y el objeto de conciliación adecuado para cada CI:

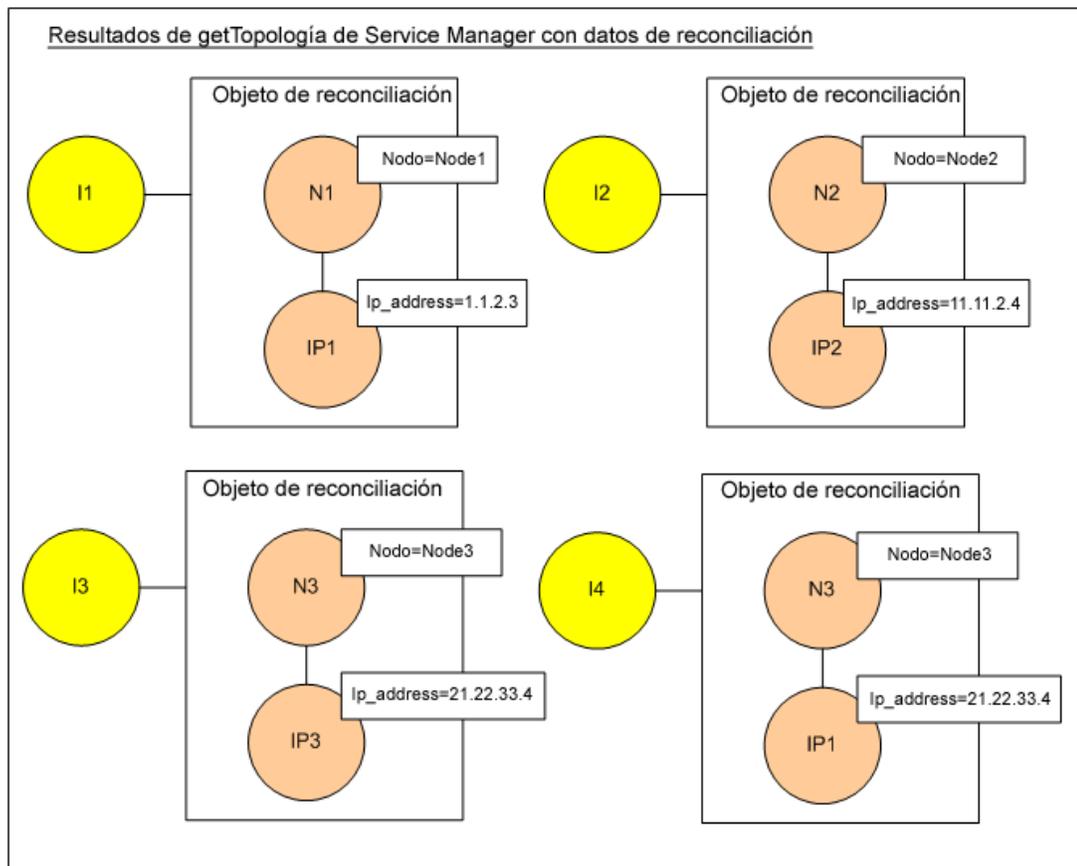


5. Los datos de conciliación de ServiceCenter deben contener una condición para node e ip derivada de los objetos de conciliación recibidos de UCMDB:

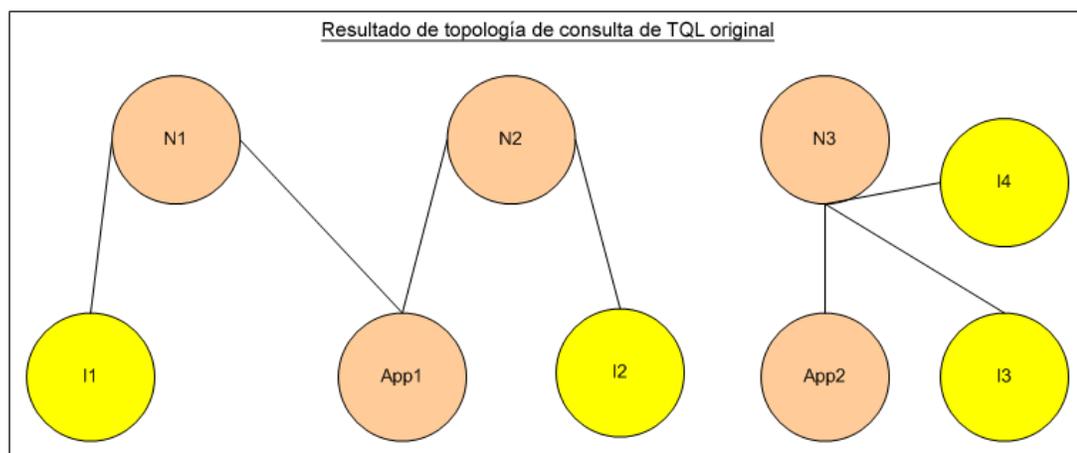


6. Federation Framework ejecuta el subgráfico ServiceCenter con los datos de conciliación para solicitar la topología y los objetos de conciliación adecuados, y recibe los siguientes

resultados:



7. El resultado después de la conexión en el motor de asignación y de crear relaciones virtuales es:



8. Federation Framework solicita el diseño del TQL original para las instancias recibidas de UCMDb y ServiceCenter.

Flujo de Federation Framework para Rellenado

Esta sección incluye los siguientes temas:

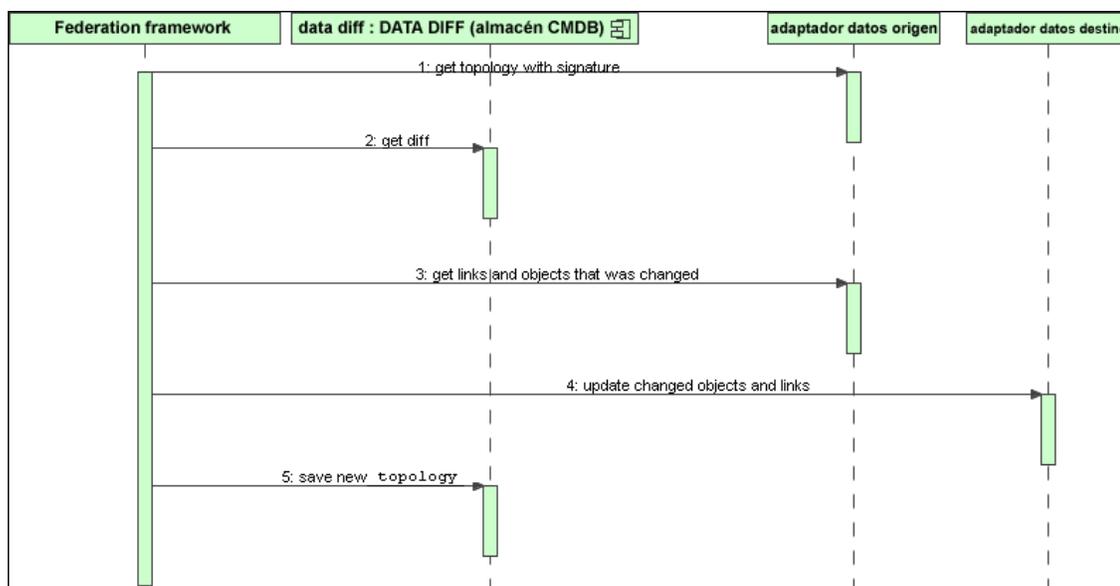
- ["Definiciones y términos" abajo](#)
- ["Diagrama de flujos" abajo](#)

Definiciones y términos

Firma. Denota el estado de las propiedades en el CI. Si se realizan cambios en los valores de la propiedad en un CI, la firma del CI también debe cambiarse. La firma del CI ayuda a detectar si un CI ha cambiado sin recuperar y comparar toda las propiedades del CI. El CI y la firma del CI se proporcionan mediante el adaptador adecuado. El adaptador responsable de cambiar la firma del CI cuando se modifican las propiedades del CI.

Diagrama de flujos

En el siguiente diagrama de secuencias se ilustra la interacción entre Federation Framework y los adaptadores de origen y destino en un flujo de relleno:



1. Federation Framework recibe la topología para el resultado de la consulta del adaptador de origen. El adaptador reconoce la consulta por su nombre y la ejecuta en el repositorio de datos externo. El resultado de la topología contiene el ID y la firma de cada CI y la relación en el resultado. El ID es el ID lógico que define el CI como exclusivo en el repositorio de datos externo. La firma debe modificarse si se modifica el CI o la relación.
2. Federation Framework usa las firmas para comparar los resultados de las consultas de topología recién recibidos con los guardados, y determinar los CI que han cambiado.
3. Cuando Federation Framework encuentra los CI y las relaciones que han cambiado, llama al adaptador de origen con los Id. de los CI cambiados y las relaciones como un parámetro para

recuperar su diseño completo.

4. Federation Framework envía la actualización al adaptador de destino. El adaptador de destino actualiza el origen de datos externo con los datos recibidos.
5. Después de la actualización, Federation Framework guarda el resultado de la última consulta.

Interfaces del adaptador

Esta sección incluye los siguientes temas:

- ["Definiciones y términos" abajo](#)
- ["Interfaces del adaptador para consultas TQL federadas" abajo](#)

Definiciones y términos

Relación externa. La relación entre dos tipos de CI externos admitidos por el mismo adaptador.

Interfaces del adaptador para consultas TQL federadas

Use la interfaz del adaptador adecuada para cada adaptador, como se indica a continuación.

- Se utiliza una **interfaz de topología de un solo nodo** cuando el adaptador no admite ninguna relación externa; es decir, el adaptador no va a recibir ninguna solicitud con más de un CI externo. Los datos de conciliación necesarios para finalizar la operación pueden describirse como consulta compleja (consulte [SingleNodeFederationTopologyReconciliationAdapter](#) a continuación).

Todas las interfaces SingleNode se crean para simplificar el flujo de trabajo; para los casos en los que necesite usar una consulta más amplia, use la interfaz **FederationTopologyAdapter**.

- Se utiliza una **interfaz FederationTopologyAdapter** para definir los adaptadores que admiten consultas federadas complejas. La solicitud de conciliación en estos adaptadores forma parte del parámetro **QueryDefinition**.

El motor de federación utiliza datos de conciliación para conectar los datos federados a los CI locales adecuados. Es posible recuperar los datos de conciliación en más de una solicitud (calculados de forma repetida según los resultados). En este caso, el adaptador recibe una solicitud solo con los datos de conciliación.

Interfaces SingleNode

Las siguientes interfaces tienen distintos tipos de datos de conciliación:

- **SingleNodeFederationIdReconciliationAdapter.** Se usa si el adaptador admite un **TQL de un solo nodo** y el ID calcula la conciliación entre los repositorios de datos.
- **SingleNodeFederationPropertyReconciliationAdapter.** Se usa si el adaptador admite un **TQL de un solo nodo** y la conciliación entre los repositorios de datos se realiza mediante las propiedades de un CI.

- **SingleNodeFederationTopologyReconciliationAdapter.** Se usa si el adaptador admite un **TQL de un solo nodo** y la conciliación entre los repositorios de datos se realiza mediante la topología. El adaptador debe admitir el caso en que el elemento de consulta está vacío y solo se solicita la topología de conciliación.

Interfaces del adaptador de datos

- **FederationTopologyAdapter.** Use este adaptador para admitir consultas TQL federadas complejas. Permite la mayor diversidad. El adaptador debe admitir el caso en que la definición de consulta es describir solamente datos de conciliación.
- **PopulateDataAdapter.** Use este adaptador para admitir consultas TQL federadas complejas y los flujos de relleno. En un flujo de relleno, este adaptador recupera todo el conjunto de datos y permite que la sonda filtre la diferencia desde la última ejecución del trabajo.
- **PopulateChangesDataAdapter.** Use este adaptador para admitir consultas TQL federadas complejas y los flujos de relleno. En un flujo de relleno, este adaptador admite la recuperación solo de los cambios que se produjeron desde la última ejecución del trabajo.

Nota: Al desarrollar un adaptador que pueda devolver grandes conjuntos de datos, es importante permitir la fragmentación implementando la interfaz `ChunkGetter`. Consulte el documento Java del adaptador específico para obtener más información.

Interfaces de informes de recursos

Las siguientes interfaces permiten que el adaptador informe de los recursos que se pueden configurar para personalizar el comportamiento del adaptador. Esto le permite editar estos recursos directamente desde Integration Studio. Estas interfaces deben usarse además de las interfaces de adaptador regulares mencionadas anteriormente.

- **PopulationQueriesResourcesLocator.** Define qué recursos pueden editarse para cada consulta de relleno específica.
- **PushQueriesResourceLocator.** Define qué recursos pueden editarse para cada consulta de inserción de datos específica.
- **GeneralResourcesLocator.** Define qué recursos generales puede editarse en este adaptador.

Interfaces adicionales

- **SortResultDataAdapter.** Se usa si se pueden ordenar los CI en el repositorio de datos externo.
- **FunctionalLayoutDataAdapter.** Se usa si se puede calcular el diseño funcional del repositorio de datos externo.

Interfaces del adaptador para la sincronización

- **SourceDataAdapter.** Se usa para los adaptadores de origen en flujos de relleno.
- **TargetDataAdapter.** Se usa para los adaptadores de destino en los flujos de inserción de datos.

Depuración de recursos del adaptador

Esta tarea describe cómo utilizar la consola JMX para crear, ver y suprimir recursos de estado del adaptador (cualquier recurso creando mediante los métodos de manipulación de recursos en la interfaz `DataAdapterEnvironment`, que se guardan en la base de datos de UCMDB o la base de datos de la sonda) para fines de depuración y desarrollo.

1. Inicie el explorador web y especifique la dirección del servidor, del siguiente modo:

- En el servidor UCMDB: `http://localhost:8080/jmx-console`
- Para la sonda: `http://localhost:1977`

Es posible que tenga que iniciar sesión con un nombre de usuario y una contraseña.

2. Para usar la página JMX MBEAN View, realice una de las siguientes acciones:

- En el servidor UCMDB: haga clic en **UCMDB:service=FCMDB Adapter State Resource Services**
- En la sonda: haga clic en **type=AdapterStateResources**

3. Introduzca valores en las operaciones que desea utilizar y haga clic en **Invocar**.

Adición de un adaptador para un nuevo origen de datos externo

Esta tarea explica el modo de definir un adaptador para que admita un nuevo origen de datos externo.

Esta tarea incluye los siguientes pasos:

- ["Requisitos previos" en la página siguiente](#)
- ["Definición de relaciones válidas para relaciones virtuales" en la página siguiente](#)
- ["Definición de la configuración de un adaptador" en la página 174](#)
- ["Definición de las clases admitidas" en la página 178](#)
- ["Implementación del adaptador" en la página 179](#)
- ["Definición de las reglas de conciliación o implementación del motor de asignación" en la página 179](#)
- ["Adición de archivos Jar necesarios para la implementación en la ruta de clase" en la página 180](#)

- ["Despliegue del adaptador" en la página 180](#)
- ["Actualización del adaptador" en la página 181](#)

1. Requisitos previos

Clases de adaptador admitidas por el modelo para los CI y relaciones en el modelo de datos UCMDb. Como programador de adaptadores, debe:

- tener conocimientos de la jerarquía de los tipos de CI de UCMDb para entender el modo en que los CIT externos están relacionados con los CIT de UCMDb.
- modelar los CIT externos en el modelo de clase de UCMDb.
- Añadir las definiciones de los nuevos tipos de CI y sus relaciones
- definir relaciones válidas en el modelo de clase de UCMDb para las relaciones válidas entre clases internas del adaptador. (Los CIT pueden colocarse en cualquier nivel del árbol del modelo de clase de UCMDb).

El modelo debe ser el mismo, independientemente del tipo de federación (sobre la marcha o réplica). Para ver información detallada acerca de la adición de nuevas definiciones de CIT al modelo de clase UCMDb, consulte "Cómo trabajar con el Selector de CI" en la publicación *HP Universal CMDB – Guía de modelado*.

Para que el adaptador admita atributos federados en los CIT, añada este CIT a las clases admitidas con atributos admitidos y la regla de conciliación para este CIT.

2. Definición de relaciones válidas para relaciones virtuales

Nota: Esta sección solo es relevante para la federación.

Para recuperar los CIT federados que están conectados a los CIT del CMDB local, debe haber una definición de vínculo válida entre los dos CIT del CMDB.

- a. Cree un archivo XML de vínculos válidos que contiene estos vínculos (si aún no existen).
- b. Añada el archivo XML de vínculos al paquete del adaptador en la carpeta `\validlinks`. Para obtener más información, consulte "Administrador de paquetes" en *HP Universal CMDB – Guía de administración*.

Ejemplo de definición de relaciones válidas:

En el siguiente ejemplo, la relación de tipo `containment` entre instancias de tipo `node` e instancias de tipo `myclass1` es una definición de relaciones válida.

```
<Valid-Links>
  <Valid-Link>
    <Class-Ref class-name="containment">
      <End1 class-name="node">
        <End2 class-name="myclass1">
          <Valid-Link-Qualifiers>
        </Valid-Link-Qualifiers>
      </End2>
    </End1>
  </Valid-Link>
</Valid-Links>
```

3. Definición de la configuración de un adaptador

- a. Vaya a **Administración de adaptador**.
- b. Haga clic en el botón **Crear recurso nuevo**  y seleccione **Nuevo adaptador**.
- c. En el cuadro de diálogo Nuevo adaptador, seleccione **Integración y Adaptador Java**.
- d. Haga clic con el botón derecho en el adaptador que ha creado y seleccione **Editar origen de adaptador** en el menú contextual.
- e. Edite las siguientes etiquetas XML:

```
<pattern xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" id="newAdapterIdName"
xsi:noNamespaceSchemaLocation="../../Patterns.xsd"
description="Adapter Description" schemaVersion="9.0"
displayName="New Adapter Display Name">

<deletable>true</deletable>

<discoveredClasses>

<discoveredClass>link</discoveredClass>

<discoveredClass>object</discoveredClass>

</discoveredClasses>

<taskInfo
className="com.hp.ucmdb.discovery.probe.services.dynamic.cor
e.
AdapterService">

<params
className="com.hp.ucmdb.discovery.probe.services.dynamic.cor
e.
AdapterServiceParams" enableAging="true"
```

```
enableDebugging="false" enableRecording=
"false" autoDeleteOnErrors="success" recordResult="false"
maxThreads="1" patternType="java_adapter"
maxThreadRuntime="2520000">

<className>com.yourCompany.adapter.MyAdapter.MyAdapterClass
</className>

</params>

<destinationInfo
className="com.hp.ucmdb.discovery.probe.tasks.BaseDestination
Data">

<!-- check -->

<destinationData name="adapterId"
description="">${ADAPTER.adapter_id}</destinationData>

<destinationData name="attributeValues"
description="">${SOURCE.attribute_values}</destinationData>

<destinationData name="credentialsId"
description="">${SOURCE.credentials_id}</destinationData>

<destinationData name="destinationId"
description="">${SOURCE.destination_id}</destinationData>

</destinationInfo>

<resultMechanism isEnabled="true">

<autoDeleteCITs isEnabled="true">

<CIT>link</CIT>

<CIT>object</CIT>

</autoDeleteCITs>

</resultMechanism>

</taskInfo>

<adapterInfo>

<adapter-capabilities>
```

```
<support-federated-query>

<!--<supported-classes/> <!--see the section about supported
classes-->

<topology>

<pattern-topology /> <!--or <one-node-topology> -->

</topology>

</support-federated-query>

<!--<support-replicatioin-data>

<source>

<changes-source/>

</source>

<target/>

</adapter-capabilities>

<default-mapping-engine/>

<queries />

<removedAttributes />

<full-population-days-interval>-1</full-population-days-
interval>

</adapterInfo>

<inputClass>destination_config</inputClass>

<protocols />

<parámetros>

<!--The description attribute may be written in simple text
or HTML.-->

<!--The host attribute is treated as a special case by UC MDB-
->
```

```
<!--and will automatically select the probe name (if
possible)-->

<!--according to this attribute's value.-->

<parameter name="credentialsId" description="Special type of
property, handled by UCMDB for credentials menu"
type="integer" display-name="Credentials ID" mandatory="true"
order-index="12" />

<parameter name="host" description="The host name or IP
address of the remote machine" type="string" display-
name="Hostname/IP" mandatory="false" order-index="10" />

<parameter name="port" description="The remote machine's
connection port" type="integer" display-name="Port"
mandatory="false" order-index="11" />

</parameters>

<parameter name="myatt" description="is my att true?"
type="string" display-name="My Att" mandatory="false" order-
index="15" valid-values="True;False"/>True</parameters>

<collectDiscoveredByInfo>true</collectDiscoveredByInfo>

<integration isEnabled="true">

<category >My Category</category>

</integration>

<overrideDomain>${SOURCE.probe_name}</overrideDomain>

<inputTQL>

<resource:XmlResourceWrapper
xmlns:resource="http://www.hp.com/ucmdb/1-0-
0/ResourceDefinition" xmlns:ns4="http://www.hp.com/ucmdb/1-0-
0/ViewDefinition" xmlns:tql="http://www.hp.com/ucmdb/1-0-
0/TopologyQueryLanguage">

<resource xsi:type="tql:Query" group-id="2" priority="low"
is-live="true" owner="Input TQL" name="Input TQL">

<tql:node class="adapter_config" id="-11" name="ADAPTER" />
```

```
<tql:node class="destination_config" id="-10" name="SOURCE"
/>

<tql:link to="ADAPTER" from="SOURCE" class="fcmdb_conf_
aggregation" id="-12" name="fcmdb_conf_aggregation" />

</resource>

</resource:XmlResourceWrapper>

</inputTQL>

<permissions />

</pattern>
```

Para obtener información detallada acerca de las etiquetas XML, consulte "[Propiedades y etiquetas de configuración XML](#)" en la página 182.

4. Definición de las clases admitidas

Defina las clases admitidas del código del adaptador implementando el método *getSupportedClasses()* o usando el archivo XML de patrón.

```
<supported-classes>
  <supported-class name="HistoryChange" is-derived="false" is-reconcili
ation-supported="false" federation-not-supported="false" is-id-reconcilia
tion-supported="false">
    <supported-conditions>
      <attribute-operators attribute-name="change_create_time">
        <operator>GREATER</operator>
        <operator>LESS</operator>
        <operator>GREATER_OR_EQUAL</operator>
        <operator>LESS_OR_EQUAL</operator>
        <operator>CHANGED_DURING</operator>
      </attribute-operators>
    </supported-conditions>
  </supported-class>
```

name	Nombre del tipo de CI
is-derived	Especifica si esta definición incluye todos los secundarios heredados

is-reconciliation-supported	Especifica si esta clase se usa para la conciliación
is-id-reconciliation-supported	Especifica si esta clase se usa para id-reconciliation
federation-not-supported	Especifica si este CIT no debe permitirse para la federación (bloqueando algunos CIT; por ejemplo, un CIT definido únicamente para la federación)
<supported-conditions>	Especifica las condiciones admitidas para cada atributo

5. Implementación del adaptador

Seleccione la clase de implementación del adaptador correcta de acuerdo con sus capacidades definidas. La clase de implementación del adaptador implementa las interfaces adecuadas según las capacidades definidas.

Si el adaptador implementa **getTopologyWithReconciliationData()** y las capacidades del adaptador incluyen la capacidad de utilizarlo como punto de partida, el adaptador debe admitir también la topología solicitante con datos de conciliación sin condiciones (tipo único). En este caso el adaptador debe devolver los datos completos de conciliación de los resultados encontrados.

La compatibilidad de la conciliación de adaptadores puede definirse según **global_id**, en cuyo caso **global_id** debe definirse como parte de los atributos de conciliación en las clases admitidas por el adaptador. Si la compatibilidad de la conciliación de adaptadores se define según **global_id**, **getTopologyWithReconciliationData()** debe devolver **global_id** como parte de las propiedades de los objetos de conciliación. UCMDB utiliza **global_id** para la conciliación de los resultados de federación para un CIT en lugar de la regla de identificación.

Parte de la API de federación es la interfaz `DataAdapterEnvironment`. Esta interfaz representa el entorno del adaptador de datos. Contiene la API de entorno necesaria para que funcione el adaptador. Para obtener más información sobre la interfaz `DataAdapterEnvironment`, consulte "[La interfaz DataAdapterEnvironment](#)" en la página 184.

6. Definición de las reglas de conciliación o implementación del motor de asignación

Si el adaptador admite consultas TQL federadas, tiene dos opciones para definir el motor de asignación:

- Use el motor de asignación predeterminado de , que usa las reglas de conciliación internas de CMDB para la asignación. Para usarlo, deje la etiqueta XML **<default-mapping-engine>** vacía.
- Escriba su propio motor de asignación implementando la interfaz del motor de asignación y

colocando el archivo JAR con el resto del código del adaptador. Para ello, utilice la siguiente etiqueta XML: **<default-mapping-engine>com.yourcompany.map.MyMappingEngine</default-mapping-engine>**

7. Adición de archivos Jar necesarios para la implementación en la ruta de clase

Para implementar las clases, añada el archivo **federation_api.jar** a la ruta de clase de su editor de código.

8. Despliegue del adaptador

Implante el paquete del adaptador. Para obtener información general acerca de la implementación de un paquete, consulte "Administrador de paquetes" en la *HP Universal CMDB – Guía de administración*.

El paquete debe contener las siguientes entidades:

- Nueva definición de CIT (opcional):
 - Usado solo si el adaptador admite nuevos tipos de CI que aún no existen en UCMDB.
 - Las nuevas definiciones de CIT se encuentran en la carpeta `class` del paquete.
- Nueva definición del tipo de datos (opcional):
 - Usado solo si los nuevos CIT requieren nuevos tipos de datos.
 - Las nuevas definiciones de tipos de datos se encuentran en la carpeta `typedef` del paquete.
- Nueva definición de relaciones válidas (opcional):
 - Usado solo si el adaptador admite el TQL federado.
 - Las nuevas definiciones de relaciones válidas se encuentran en la carpeta `validlinks` del paquete.
- El archivo XML de configuración del patrón debe encontrarse en la carpeta `discoveryPatterns` del paquete.
- **Descriptor**. Defina las definiciones del paquete.
- Coloque sus clases compiladas (normalmente un archivo jar) en el paquete bajo la carpeta `adapterCode\<id_adaptador>`.

Nota: El nombre de la carpeta `id_adaptador` tiene el mismo valor que en la configuración del adaptador.

- Si crea su propio archivo de configuración, debe colocar el archivo en el paquete bajo la carpeta `adapterCode\<id_adaptador>`.

9. Actualización del adaptador

Se pueden realizar cambios en cualquiera de los archivos no binarios del adaptador en el módulo Administración de adaptador. Al realizar cambios en los archivos de configuración del módulo Administración de adaptador, el adaptador se vuelve a cargar con las nuevas configuraciones.

También se pueden realizar actualizaciones editando los archivos en el paquete (archivos binarios y no binarios) y volviendo a implantar el paquete mediante el Administrador de paquetes. Para obtener más información, consulte "Cómo desplegar un paquete" en *HP Universal CMDB – Guía de administración*.

Creación de un adaptador de ejemplo

En este ejemplo se ilustra el modo de crear un adaptador de ejemplo. Esta tarea incluye los siguientes pasos:

- ["Selección de la lógica del adaptador" abajo](#)
- ["Carga del proyecto" abajo](#)

1. Selección de la lógica del adaptador

Al implementar un adaptador, es necesario elegir el modo de gestionar la lógica de la condición en la implementación (condiciones de la propiedad, condiciones de ID, condiciones de conciliación y condiciones de vínculos).

- a. Recupere todos los datos en la memoria del adaptador y deje que seleccione o filtre las instancias de CI necesarias.
- b. Convierta todas las condiciones en el idioma de origen de los datos y deje que filtre y seleccione los datos. Por ejemplo:
 - Convierta la condición en una consulta SQL.
 - Convierta la condición en un objeto de filtro API de Java.
- c. Filtre algunos de los datos del servicio remoto y que el adaptador seleccione y filtre el resto.

En el ejemplo MyAdapter, se usa la lógica de la opción *a*.

2. Carga del proyecto

Copie los archivos en la carpeta `C:\hp\UCMDB\UCMDBServer\tools\adapter-dev-kit\SampleAdapters` y siga las instrucciones que aparecen en los archivos Léame.

Nota: Si usa un adaptador con conjuntos de datos grandes, puede ser necesario usar la caché y la indexación para mejorar el rendimiento para la federación.

La documentación de javadocs en línea está disponible en:

C:\hp\UCMDB\UCMDBServer\deploy\ucmdb-docs\docs\eng\APIs\DBAdapterFramework_JavaAPI\index.html

Propiedades y etiquetas de configuración XML

<code>id="newAdapterIdName"</code>	Define el nombre real del adaptador. Usado para búsquedas en carpetas y registros
<code>displayName="New Adapter Display Name"</code>	Define el nombre de la pantalla del adaptador, como aparece en la UI.
<code><className>...</className></code>	Define la interfaz del adaptador que implementa la clase Java.
<code><category >My Category</category></code>	Define la categoría del adaptador.
<code><parámetros></code>	Define las propiedades de la configuración que están disponibles en la UI al configurar un nuevo punto de integración.
<code>name</code>	Nombre de la propiedad (usado principalmente por el código)
<code>description</code>	Pista de visualización de la propiedad
<code>type</code>	Cadena o número entero (use los valores válidos con las cadenas para Booleano).
<code>display-name</code>	Nombre de la propiedad en la UI.
<code>mandatory</code>	Especifica si esta propiedad de configuración es obligatoria para el usuario.
<code>order-index</code>	Orden de colocación de la propiedad (pequeño = arriba)
<code>valid-values</code>	Lista de los posibles valores inválidos separados por caracteres ';' (por ejemplo, <code>valid-values="Oracle;SQLServer;MySQL"</code> or <code>valid-values="True;False"</code>).
<code><adapterInfo></code>	Contiene la definición de las capacidades y la configuración estática del adaptador.

	<support-federated-query>	Define este adaptador como capaz de federación.
	<start-point-adapter>	Especifica que este adaptador es el punto de inicio para cálculo de la consulta TQL.
	<one-node-topology>	Capacidad de las consultas federadas con un nodo de consulta federado.
	<pattern-topology>	Posibilidad de federar consultas complejas.
	<support-replicatioin-data>	Define la capacidad de ejecutar inserción de datos y flujos de relleno.
	<source>	Este adaptador puede usarse para flujos de relleno.
	<push-back-ids>	Devolver el Id. global del CI a la columna global_id de la tabla (debe definirse en orm.xml). Este comportamiento puede sustituirse implementando el complemento FcmdbPluginPushBackIds .
	<changes-source>	Este adaptador puede usarse para flujos de cambios en el relleno.
	<instance-based-data>	Esta etiqueta define que el adaptador admite un flujo de relleno basado en una instancia.
	<target>	Este adaptador puede usarse para los flujos de inserción de datos.
	<default-mapping-engine>	Permite definir un motor de asignación para el adaptador (de forma predeterminada, el adaptador usa el motor de asignación predeterminado). Para cualquier otro motor de asignación, introduzca el nombre de la clase de implementación del motor de asignación
	<removedAttributes>	Fuerza la eliminación de atributos específicos del resultado.
	<full-population-days-interval>	Especifica cuándo se ejecutará un trabajo de relleno completo, en lugar de un trabajo diferencial (cada 'x' días). Usa el mecanismo de envejecimiento junto con el flujo de cambios.
	<adapter-settings>	Lista de ajustes del adaptador.

	<code><list.attributes.for.set></code>	Determina qué atributos sustituyen al valor anterior (si existe).
--	--	---

La interfaz **DataAdapterEnvironment**

OutputStream openResourceForWriting(String resourceName) throws FileNotFoundException;

Este método abre un recurso con un nombre determinado para escritura. Se utiliza para guardar los datos persistentes para la integración. Debería utilizarse este método en lugar de tratar de cargar archivos usando métodos java. El usuario debe asegurarse de que se cierra la secuencia al terminar de escribir en la secuencia. `close()/flush()` guardará el recurso. Este método crea un recurso de ejecución (no puede sobrescribir los archivos incluidos en el paquete del adaptador).

Parámetro

- **resourceName:** El nombre del recurso que se va a recuperar. Este nombre debe ser único en todas las integraciones del mismo adaptador.

Valor de devolución

Devuelve una secuencia en la que escribir.

Excepciones

- Este método genera la excepción *FileNotFoundException* si el tipo de recurso es archivo y el archivo no existe, si el recurso es un directorio en lugar de un archivo regular o si por algún otro motivo no se puede abrir el recurso para lectura.
- Este método genera la excepción *SecurityException* si existe un administrador de seguridad y su método *checkRead* deniega el acceso al archivo.

InputStream openResourceForReading(String resourceName) throws FileNotFoundException;

Este método abre un recurso con un nombre determinado para lectura. Se utiliza para leer los datos persistentes para la integración. Este método debería utilizarse en lugar de tratar de cargar un archivo usando métodos java. El usuario debe asegurarse de que se cierra la secuencia al terminar de leerlo. En primer lugar intenta cargar archivos incluidos en el paquete del adaptador. Si no los encuentra, intenta cargar un recurso creado durante la ejecución desde *DataAdapterEnvironment.openResourceForWriting(String)*. Los recursos de ejecución se pueden ver utilizando JMX (de la sonda y el servidor según corresponda).

Parámetro

- **resourceName:** El nombre del recurso que se va a recuperar. Este nombre debe ser único en todas las integraciones del mismo adaptador.

Valor de devolución

Devuelve una secuencia para leer.

Excepciones

- Este método genera la excepción *FileNotFoundException* si el tipo de recurso es **archivo** y el archivo no existe, si el recurso es un directorio en lugar de un archivo regular o si por algún otro motivo no se puede abrir el recurso para lectura.
- Este método genera la excepción *SecurityException* si existe un administrador de seguridad y su método *checkRead* deniega el acceso de lectura al archivo.

Properties openResourceAsProperties(String propertiesFile) throws IOException;

Este método abre un recurso con un nombre determinado y lo carga como una estructura de *propiedades*. Se utiliza para leer los datos persistentes para la integración. Este método debería utilizarse en lugar de intentar cargar los archivos **.properties** mediante métodos java. En primer lugar intenta cargar archivos incluidos en el paquete del adaptador. Si no los encuentra, intenta cargar un recurso creado durante la ejecución desde *DataAdapterEnvironment.openResourceForWriting(String)*. Los recursos de ejecución se pueden ver utilizando JMX (de la sonda y el servidor según corresponda).

Parámetro

- **propertiesFile**: El nombre del recurso que se va a recuperar. Este nombre debe ser único en todas las integraciones del mismo adaptador.

Valor de devolución

Devuelve el contenido del archivo representado en Propiedades.

Excepciones

- Este método genera la excepción *FileNotFoundException* si el tipo de recurso es **archivo** y el archivo no existe, si el recurso es un directorio en lugar de un archivo regular o si por algún otro motivo no se puede abrir el recurso para lectura.
- Este método genera la excepción *SecurityException* si existe un administrador de seguridad y su método *checkRead* deniega el acceso de lectura al archivo.
- Este método produce la excepción *IOException* si el archivo de propiedades no ha podido convertir el objeto *Properties*.

String openResourceAsString(String resourceName) throws IOException;

Este método abre un recurso con un nombre determinado y lo carga como una cadena. Se utiliza para leer los datos persistentes para la integración. Debería utilizarse este método en lugar de tratar de cargar archivos usando métodos java.

En primer lugar intenta cargar archivos incluidos en el paquete del adaptador. Si no los encuentra, intenta cargar un recurso creado durante la ejecución desde *DataAdapterEnvironment.openResourceForWriting(String)*. Los recursos de ejecución se pueden ver utilizando JMX (de la sonda y el servidor según corresponda).

Parámetro

- **resourceName:** El nombre de recurso que se va a recuperar. Este nombre debe ser único en todas las integraciones del mismo adaptador.

Valor de devolución

Devuelve el contenido del archivo representado en formato de cadena.

Excepciones

- Este método genera la excepción *FileNotFoundException* si el tipo de recurso es **archivo** y el archivo no existe, si el recurso es un directorio en lugar de un archivo regular o si por algún otro motivo no se puede abrir el recurso para lectura.
- Este método genera la excepción *SecurityException* si existe un administrador de seguridad y su método *checkRead* deniega el acceso de lectura al archivo.
- Este método genera la excepción *IOException* si se produce un error de E/S.

public void saveResourceFromString(String relativeFileName, String value) throws IOException;

Este método recibe una cadena y la guarda como un recurso. Se utiliza para guardar los datos persistentes para la integración. Debería utilizarse este método en lugar de tratar de guardar archivos usando métodos java. Este método convierte la cadena en una secuencia y la guarda para el recurso. Crea un recurso, pero no puede sobrescribir los archivos incluidos en el paquete del adaptador). Los recursos de ejecución se pueden ver utilizando JMX (de la sonda y el servidor según corresponda).

Parámetro

- **relativeFileName:** El nombre de recurso que se va a recuperar. Este nombre debe ser único en todas las integraciones del mismo adaptador.
- **value:** La cadena que se va a guardar como un recurso

Excepciones

Este método genera la excepción *IOException* si se produce un error de E/S.

boolean resourceExists(String resourceName);

Este método comprueba si el nombre de recurso dado existe. Busca los archivos incluidos en el paquete del adaptador y los recursos creados durante la ejecución de *DataAdapterEnvironment.openResourceForWriting(String)*.

Parámetro

- **resourceName:** El nombre del recurso que se va a recuperar. Este nombre debe ser único en todas las integraciones del mismo adaptador.

Valor de devolución

Devuelve **True** si *resourceName* existe.

boolean deleteResource(String resourceName);

Este método elimina el recurso dado de los datos persistentes. Elimina un recurso de tiempo de ejecución y no puede eliminar archivos incluidos en el paquete del adaptador. Los recursos de ejecución se pueden ver utilizando JMX (para la sonda y el servidor según corresponda).

Parámetro

- **resourceName:** El nombre del recurso que se va a eliminar. Este nombre debe ser único en todas las integraciones del mismo adaptador.

Valor de devolución

Devuelve **True** si el recurso se elimina correctamente.

Collection<Cadena> listResourcesInPath(String path);

Este método recupera una lista de los recursos en la ruta de recursos dada. Busca los archivos incluidos en el paquete del adaptador y para los recursos creados durante la ejecución de *DataAdapterEnvironment.openResourceForWriting(String)*. Los recursos de ejecución se pueden ver utilizando JMX (para la sonda y el servidor según corresponda).

Parámetro

- **path:** La ruta de acceso del recurso. Por ejemplo, "META-INF/myfiles/"

Valor de devolución

Devolver una lista de recursos en la ruta de acceso.

DataAdapterLogger getLogger();

Recupera el registrador que utilizará el adaptador. Este registrador se utiliza para eventos de registro en el adaptador.

Valor de devolución

Devuelve el registrador utilizado por DataAdapter.

DestinationConfig getDestinationConfig();

Este método recupera la configuración de destino de la integración. Esta configuración contiene toda la configuración de conexión y ejecución para la integración.

Valor de devolución

Devuelve el DestinationConfig del adaptador.

Int getChunkSize();

Este método recupera el tamaño del fragmento de relleno solicitado para esta integración.

Valor de devolución

Devuelve el tamaño del fragmento de relleno.

int getPushChunkSize();

Este método recupera el tamaño del fragmento de inserción solicitado para esta integración.

Valor de devolución

Devuelve el tamaño del fragmento de inserción.

ClassModel getLocalClassModel();

Este método recupera un modelo de clase para consultar información sobre el modelo de clase de la instancia local de UCMDB. Este método da como resultado un ClassModel actualizado. Una vez que se ha devuelto el objeto ClassModel, este no se actualizará para ningún cambio del modelo de clase. Para recuperar un modelo de clase actualizado, use este método de nuevo para recuperarlo.

Valor de devolución

Devuelve el modelo de clase de UCMDB.

CustomerInformation getLocalCustomerInformation();

Este método recupera información de cliente para el cliente que está ejecutando el adaptador.

Valor de devolución

Devuelve información de cliente para el cliente que está ejecutando el adaptador.

Object getSettingValue(String name);

Este método recupera un ajuste específico del adaptador.

Parámetro

name: El nombre del ajuste.

Valor de devolución

Devuelve el valor de ajuste del objeto.

Map<String, Object> getAllSettings();

Este método recupera toda la configuración del adaptador.

Valor de devolución

Devuelve la configuración del adaptador.

boolean isMTEnabled();

Este método comprueba si el entorno del servidor admite arrendamiento múltiple (MT).

Valor de devolución

Devuelve **true** si el entorno del servidor admite MT, de lo contrario devuelve **false**.

String getUcldbServerHostName();

Este método devuelve el nombre del host del servidor UCMDB local.

Valor de devolución

Devuelve el nombre del host del servidor UCMDB local.

Capítulo 6: Desarrollo de los adaptadores de inserción

Este capítulo incluye:

Desarrollo e implementación de los adaptadores de inserción	190
Construcción de un paquete de adaptadores	191
Crear asignaciones	195
Escritura de secuencias de comandos Jython	199
Admisión de la sincronización diferencial	203
Consultas de SQL del adaptador de inserción XML genérico	205
Adaptador genérico de inserción de servicio web	205
Referencia de archivos de asignación	225
Esquema del archivo de asignación	227
Esquema de resultados de asignación	239
Personalización	243

Desarrollo e implementación de los adaptadores de inserción

Los adaptadores genéricos de inserción ofrecen una plataforma común que permite el desarrollo rápido de integraciones que insertan datos de UCMDb en repositorios de datos externos (bases de datos y aplicaciones de terceros). Los adaptadores genéricos de inserción se categorizan según el protocolo que se utiliza para insertar los datos. Para obtener más información sobre la inserción a través de XML, mediante el adaptador de inserción XML genérico, consulte ["Consultas de SQL del adaptador de inserción XML genérico" en la página 205](#). Para obtener más información sobre la inserción a través del servicio web, mediante el adaptador de inserción de servicio web genérico, consulte ["Adaptador genérico de inserción de servicio web" en la página 205](#).

El desarrollo de una integración personalizada basada en el adaptador genérico de inserción requiere:

- Generar un nuevo paquete de adaptador apropiado de los archivos de plantilla del adaptador de inserción genérico. Para obtener más información, consulte ["Construcción de un paquete de adaptadores" en la página siguiente](#).
- Asignaciones entre los tipos de vínculos de CI de UCMDb y los elementos de datos externos. Las asignaciones se almacenan como XML y se personalizan para cada repositorio de datos externo. Para obtener más información, consulte ["Crear asignaciones" en la página 195](#).

- una secuencia de comandos Jython para insertar los elementos de datos en el repositorio de datos externo. Para obtener más información, consulte "[Escritura de secuencias de comandos Jython](#)" en la página 199.
- Pasos adicionales específicos del adaptador. Por ejemplo, elegir la ruta del archivo que se debe escribir para el adaptador de inserción XML, o crear un receptor de datos para el adaptador de inserción de servicio web.

Construcción de un paquete de adaptadores

Para crear un nuevo adaptador de inserción específico de MDR, debe hacer una copia del adaptador genérico y, a continuación, editarlo para personalizarlo como un adaptador para un destino de inserción específico.

Los paquetes de adaptadores genéricos pueden encontrarse en una de las dos ubicaciones siguientes:

- XML genérico - adaptador de inserción: **hp\UCMDB\UCMDBServer\content\adapters\push-adapter.zip**
- Adaptador de servicio web genérico: **hp\UCMDB\UCMDBServer\content\adapters\web-service-push-adapter.zip**

Para crear un nuevo adaptador de inserción a partir del adaptador genérico de inserción:

1. Extraiga el contenido del archivo zip de paquete seleccionado a una carpeta de trabajo.
2. Revise los siguientes directorios para preparar la fase de cambio de nombre y sustitución:
 - **adapterCode:** Contiene el directorio que se despliega en el directorio **C:\hp\UCMDB\UCMDBServer\runtime\fcmdb\CodeBase**. Los archivos jar desplegados aquí no reinician la sonda automáticamente y no aparecen automáticamente en el CLASSPATH de la sonda.
 - **discoveryConfigFiles:** Contiene las definiciones de asignaciones del adaptador y apunta a la secuencia de comandos correcta de Jython (**push.properties**)
 - **discoveryPatterns:** Contiene la definición XML del adaptador que se despliega en el servidor de UCMDB
 - **discoveryScripts:** Contiene las secuencias de comandos Jython del adaptador a través de las cuales se realiza la conexión al almacén de datos de terceros y se insertan los datos
 - **discoveryResources:** Contiene el **UCMDBDataReceiver.jar** que contiene las clases de integración de Java para el servicio web.

Nota: Al desplegar este paquete, se reinicia la sonda para incluir este .jar en la

CLASSPATH de la sonda. No se requiere ninguna acción aparte de desplegar el paquete.

3. Realice los siguientes cambios dentro de la estructura de directorios del adaptador descomprimido:
 - a. **discoveryConfigFiles\<Nombre del adaptador de inserción>**: cambie el nombre del directorio "PushAdapter" o "XMLtoWebService" al nombre del nuevo adaptador de inserción (por ejemplo, "myPushAdapter").
 - b. **discoveryConfigFiles\<Nombre del adaptador de inserción>\push.properties**: En el archivo **push.properties**, haga lo siguiente:
 - Actualice el nombre de **jythonScript.name** al nombre de la secuencia de comandos de Jython que el nuevo adaptador de inserción utilizará (por ejemplo, **pushToMyService.py**).
 - Actualice el nombre del archivo de asignaciones que el nuevo adaptador de inserción utilizará (por ejemplo, **myPushAdapter_mappings**). No agregue la extensión **.xml**, se rellena automáticamente.
 - c. **discoveryPatterns\<nombre de adaptador de inserción>.xml**: Cambie el nombre de este archivo al nombre del archivo XML de definición del nuevo adaptador (por ejemplo, **my_push_adapter.xml**).
 - d. **discoveryPatterns\<adaptador de inserción>.xml**: Actualice este archivo como se indica a continuación:
 - Para el elemento de XML **<pattern>**: establezca los atributos **id** y **description** como corresponda. Por ejemplo:

```
<pattern id="PushAdapter"
xsi:noNamespaceSchemaLocation="../../Patterns.xsd"
description="Discovery Pattern Description" schemaVersion="9.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

se ha cambiado a:

```
<pattern id="MyPushAdapter" displayLabel="My Push Adapter"
xsi:noNamespaceSchemaLocation="../../Patterns.xsd"
description="Discovery Pattern Description" schemaVersion="9.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

- Para el elemento de XML **<parameters>**: actualice los elementos secundarios en función de las necesidades de su adaptador. De forma predeterminada, los siguientes elementos secundarios se utilizan para definir un adaptador de inserción. Estos valores se asignan cuando se define el punto de integración en Integration Studio después de configurar el adaptador. Actualice la lista de parámetros de manera que la lista de

parámetros refleje los atributos de conexión requeridos. No suprima el atributo **probeName**.

- **host**: el nombre del servidor que aloja el servicio web
 - **port**: puerto del servicio de receptor de datos de UCMDB que escucha
 - **Adaptador de inserción de servicio web: uri**: el resto de la dirección URL para formar la dirección de punto final de servicio del receptor de datos.
 - **probeName**: define en qué sonda de flujo de datos se ejecuta el trabajo de inserción
- o Para el elemento de XML **<integration>**: actualice el valor del elemento secundario **<category>** a algo distinto a Genérico. De forma predeterminada, los adaptadores de integración que pertenecen a la categoría Genérico no se muestran en Integration Studio. Si está realizando una integración con un almacén de datos de terceros, establezca este valor en "Terceros". Si está integrando con un producto HP BTO, establezca este valor en "Productos HP BTO".
- e. **adapterCode\PushAdapter**: cambie el nombre de esta carpeta con el Id. de adaptador utilizado en el paso anterior (por ejemplo, **adapterCode\MyPushAdapter**).
- f. **discoveryScripts\<secuencia de comandos de inserción de Jython>.py**: cree un archivo con el mismo nombre que el definido en la propiedad **push.properties jythonScript.name**. En el archivo **discoveryScript** hay una secuencia de comandos que inserta los CI y vincula a una base de datos Oracle externa. Sustituya **discoveryScripts\pushScript.py** con la secuencia de comandos que ha escrito (para obtener más información, consulte ["Escritura de secuencias de comandos Jython" en la página 199](#)). Si cambia el nombre de la secuencia de comandos, la propiedad **jythonScript.name** de **adapterCode\<Id. de adaptador>\push.properties** debe actualizarse como corresponda.
- Adaptador de inserción XML: **pushScript.py**
 - Adaptador de inserción de servicio web: **XMLtoWebService.py**
- g. **tql\<TQL de integración>**: al igual que un paquete normal, coloque la definición XML de TQL de sus TQL de integración en este directorio. Todos los TQL de esta carpeta se despliegan al desplegar el paquete del adaptador.
- h. **discoveryConfigFiles\<Nombre del adaptador de inserción>\mappings**: cree un archivo de asignación XML por TQL que desee usar en la integración. Tenga en cuenta que el adaptador de inserción aplica las transformaciones del archivo de asignación a los resultados de los TQL de integración y, a continuación, envía esos datos en tres parámetros (**addResult**, **updateResult** y **deleteResult**) de una tarea ad hoc a la sonda del flujo de datos.
- i. **adapterCode\<Id. de adaptador>\mappings**: sustituya el archivo **mappings.xml** con los archivos de asignación que haya preparado (para obtener más información, consulte

"[Crear asignaciones](#)" en la página siguiente).

Adaptador de inserción XML: Este ejemplo de asignación corresponde al ejemplo de las tablas creadas en ORACLE en el archivo `sql_queries`.

Para usar un archivo de asignación para cada método de TQL, asigne el nombre del TQL correspondiente a cada archivo XML, seguido por `.xml`. En este caso, el archivo **mappings.xml** se utiliza de forma predeterminado, si no se encuentra ningún archivo de asignación específico para el nombre de TQL actual. El nombre del archivo de asignación predeterminado se puede modificar cambiando la propiedad `mappingFile.default` en `adapterCode\<Id. de adaptador>\push.properties`.

4. Después de realizar todos los cambios anteriores, cree un archivo `.zip` seleccionando las carpetas y los archivos especificados en el paso 3 (por ejemplo, `my_Push_Adapter.zip`).
5. Despliegue el archivo `.zip` recién creado en el servidor UCMDB mediante el Administrador de paquetes (vaya a **Administración > Administrador de paquetes**).
6. Cree un punto de integración en **Administración de Data Flow > Integration Studio** y defina los TQL de integración que el punto de integración utiliza. Establezca una planificación para la inserción automática de datos.

Solución de problemas

El procedimiento de creación de un nuevo adaptador de inserción requiere un cambio de nombre y sustitución completos y correctos. Cualquier error afectará probablemente al adaptador. El paquete debe descomprimirse y volver a comprimirse correctamente para actuar como un paquete de UCMDB. Consulte los paquetes de serie como ejemplos. Entre los errores comunes se encuentran los siguientes:

- Incluir otro directorio encima de los directorios de paquete en el archivo ZIP.
Solución: Comprimir el paquete como ZIP en el mismo directorio que los directorios del paquete, como **discoveryResources**, **adapterCode**, etc. No incluya otro nivel de directorio encima de este en el archivo ZIP.
- Omitir un cambio de nombre crítico de un directorio, un archivo, o una cadena de un archivo.
Solución: seguir las instrucciones en esta sección cuidadosamente.
- Deletrear erróneamente un cambio de nombre crítico de un directorio, un archivo, o una cadena de un archivo.
Solución: No cambie su convención de denominación a mitad de secuencia una vez que comience el procedimiento de cambio de nombre. Si se da cuenta de que tiene que cambiar el nombre, empiece de nuevo por completo en vez de intentarlo de forma retroactiva corrigiendo el nombre, ya que hay un riesgo elevado de error. Además, utilice la función de buscar y reemplazar en lugar de sustituir manualmente las cadenas para reducir el riesgo de errores.
- Desplegar los adaptadores con los mismos nombres de archivo que otros adaptadores, especialmente en los directorios **discoveryResources** y **adapterCode**.
Solución: Puede estar utilizando una versión de UCMDB con un problema conocido que impide

que los archivos de asignación tengan el mismo nombre que cualquier otro adaptador del mismo entorno de UCMDB. Si intenta desplegar un paquete con nombres duplicados, se producirá un error en el despliegue del paquete. Este problema puede ocurrir incluso si estos archivos se encuentran en distintos directorios. Además, este problema puede producirse independientemente de si los duplicados están dentro del paquete o con otros paquetes desplegados anteriormente.

En este punto puede crear un nuevo trabajo de adaptador de inserción en Integration Studio mediante el nuevo adaptador que acaba de desplegar.

Prácticas recomendadas de TQL para los adaptadores de inserción

1. Cree una estructura de carpetas en los árboles de TQL y Vista, y mantenga allí todos los nuevos TQL y vistas. Usar una convención de denominación.
2. A menos que el TQL sea pequeño, copie primero el TQL más similar.
3. Realice un cambio a la vez. Guarde, pruebe y obtenga una vista previa después de cada cambio. Repita el proceso hasta que los resultados cumplan con sus requisitos.

Crear asignaciones

Los datos de resultado del TQL sin procesar están en el formato de esquema de modelo de clase UCMDB. Es probable que el consumidor utilice un modelo de datos diferente. El adaptador de inserción proporciona un mecanismo de asignación para transformar los datos en un formato más apto para el consumo. Las asignaciones realizan transformaciones tanto directas como complejas, desde la conversión directa de tipo de nombre, a la agregación primaria/secundaria y funciones de referencia.

La especificación de asignación puede encontrarse en la sección "[Referencia de archivos de asignación](#)" en la [página 225](#). Use la referencia para crear un archivo de asignación.

Nota: El archivo de propiedades de adaptador se refiere al nombre del archivo de asignación. En los archivos de configuración del adaptador, el adaptador implementa una estructura de carpetas utilizando el nombre del adaptador. Cambie el nombre de esta carpeta al implementar un adaptador para mantener su singularidad según lo requiera el Administrador de paquetes.

Compilar un archivo de asignación

Para compilar un archivo de asignación:

1. Empiece por el archivo de asignación predeterminado.
2. Despliegue el adaptador y ejecútelo una vez.
3. Observe los resultados.

4. Identifique y tome nota de lo que se deba cambiar.
5. Realice los cambios identificados en el paso anterior. La siguiente lista puede servir como guía para ordenar los cambios.
 - a. Empiece por la parte superior, que no es transformadora. Asegúrese de que el adaptador se ejecute después de cada cambio.
 - b. Cambie la sección de CI de origen a los nombres de UCMDB en el resultado de TQL.
 - c. En primer lugar, asigne las claves.
 - d. A continuación, agregue todas las asignaciones directas.
 - e. Añada las asignaciones complejas.
 - f. Añada las asignaciones de vínculo.

Repita los pasos 2-5 hasta que los datos asignados sean los adecuados para el consumo. Seleccione el paquete de adaptador genérico apropiado a partir del que creará el nuevo adaptador de inserción.

Los archivos de asignación funcionan de la misma manera para todos los tipos de adaptadores de inserción. El adaptador de inserción XML genérico escribe los resultados asignados en un archivo. El adaptador de inserción de servicio web genérico envía los resultados de XML a un receptor de datos. Para obtener más información, consulte ["Adaptador genérico de inserción de servicio web" en la página 205](#).

Preparación de los archivos de asignación

Nota: Para poder recuperar todos los CI y relaciones tal como son en CMDB sin asignarlos, no debe crear el archivo **mappings.xml**. Esta acción devolverá todos los CI y relaciones con todos sus atributos.

Hay dos formas diferentes de preparar los archivos de asignación:

- Puede preparar un único archivo de asignación global.

Todas las asignaciones se colocan en un único archivo llamado **mappings.xml**.

- Puede preparar un archivo independiente para cada consulta de inserción.

Cada archivo de asignación se denomina **<nombre de consulta>.xml**.

Para obtener más información, consulte ["Esquema del archivo de asignación" en la página 227](#).

Esta tarea incluye los siguientes pasos:

- ["Cree un archivo mappings.xml" abajo](#)
- ["Asignación de CI" abajo](#)
- ["Asignación de vínculos" en la página siguiente](#)

1. Cree un archivo mappings.xml

La estructura del archivo de asignación se crea de la forma siguiente (use un archivo existente como una plantilla):

```
<?xml version="1.0" encoding="UTF-8"?>
<integración>
  <info>
    <source name="UCMDB" versions="9.x" vendor="HP" >
      <!-- for example: -->
      <target name="Oracle" versions="11g" vendor="Oracle" >
    </info>
  <targetcis>
    <!-- CI Mappings --->
  </targetcis>
  <targetrelations>
    <!-- Link Mappings --->
  </ targetrelations>
</integration>
```

2. Asignación de CI

Hay dos maneras de asignar tipos de CI de CMDB:

- Asigne un tipo de CI de manera que los CI de ese tipo y todos los tipos heredados se asignen de la misma manera:

```
<source_ci_type_tree name="node" mode="update_else_insert">
  <apioutputseq>1</apioutputseq>
  <target_ci_type name="host">
    <targetprimarykey>
      <pkey>name</pkey>
    </targetprimarykey>
    <target_attribute name=" name" datatype="STRING">
      <map type="direct" source_attribute="name" >
    </target_attribute>
    <!-- more target attributes --->
  </target_ci_type>
</source_ci_type_tree>
```

- Asigne un tipo de CI de manera que solo se procesen los CI de ese tipo. Los CI de tipos heredados no se procesan, a menos que también se asigne su tipo (de una de dos maneras posibles):

```
<source_ci_type name="node" mode="update_else_insert">
  <apioutputseq>1</apioutputseq>
  <target_ci_type name="host">
    <targetprimarykey>
      <pkey>name</pkey>
    </targetprimarykey>
    <target_attribute name=" name" datatype="STRING">
      <map type="direct" source_attribute="name" >
    </target_attribute>
    <!-- more target attributes --->
  </target_ci_type>
</source_ci_type>
```

Un tipo de CI que se asigne indirectamente (uno de sus antecesores se asigna mediante **source_ci_type_tree**), también puede sustituir la asignación del principal haciendo que aparezca en su propio **source_ci_type_tree** o **source_ci_type**.

Se recomienda utilizar **source_ci_type_tree** siempre que sea posible. De lo contrario, los CI resultantes de un tipo de CI que no aparece en los archivos de asignación no se transferirán a la secuencia de comandos de Jython.

3. Asignación de vínculos

Hay dos maneras de asignar vínculos:

- Asigne un vínculo de manera que los vínculos de ese tipo y todos los vínculos heredados se asignen de la misma manera:

```
<source_link_type_tree name="dependency" target_link_type="dependency"
mode="update_else_insert" source_ci_type_end1="webservice" source_ci_t
ype_end2="sap_gateway">
  <target_ci_type_end1 name="webservice" >
  <target_ci_type_end2 name="sap_gateway" >
    <target_attribute name="name" datatype="STRING">
      <map type="direct" source_attribute="name" >
    </target_attribute>
  </source_link_type_tree>
```

- Asigne un vínculo de manera que solo se procesen los vínculos de ese tipo. Los vínculos de tipos heredados no se procesan, a menos que también se asigne su tipo (de una de dos maneras posibles):

```
<link source_link_type="dependency" target_link_type="dependency" mode
="update_else_insert" source_ci_type_end1="webservice" source_ci_type_
end2="sap_gateway">
  <target_ci_type_end1 name="webservice" >
  <target_ci_type_end2 name="sap_gateway" >
  <target_attribute name="name" datatype="STRING">
    <map type="direct" source_attribute="name" >
  </target_attribute>
</link>
```

Escritura de secuencias de comandos Jython

La secuencia de comandos de asignación es una secuencia de comandos Jython regular y debería seguir las reglas de las secuencias de comandos Jython. Para obtener más información, consulte ["Desarrollo de los adaptadores Jython" en la página 39](#).

La secuencia de comandos debería contener la función **DiscoveryMain**, que puede devolver un **OSHVResult** vacío o una instancia **DataPushResults** en caso de resultado satisfactorio.

Para informar de algún fallo, la secuencia de comandos debería generar una excepción, por ejemplo:

```
raise Exception('Failed to insert to remote UCMDB using TopologyUpdateService. S
ee log of the remote UCMDB')
```

En la función **DiscoveryMain**, los elementos de datos que se van a insertar o a eliminar de la aplicación externa se pueden obtener de la forma siguiente:

```
# get add/update/delete result objects (in XML format) from the Framework
addResult = Framework.getTriggerCIData('addResult')
updateResult = Framework.getTriggerCIData('updateResult')
deleteResult = Framework.getTriggerCIData('deleteResult')
```

El objeto cliente para la aplicación externa se puede obtener de la forma siguiente:

```
oracleClient = Framework.createClient()
```

Este objeto cliente usa automáticamente el ID de credenciales, el nombre de host y el número de puerto que pasó el adaptador a través de **Framework**.

Si tiene que usar los parámetros de conexión definidos para el adaptador (para obtener más información, consulte el paso sobre cómo editar el archivo **discoveryPatterns\push_adapter.xml** en ["Construcción de un paquete de adaptadores" en la página 191](#)), utilice el código siguiente:

```
propValue = str(Framework.getDestinationAttribute('<Connection Property Name'))
```

Por ejemplo:

```
serverName = Framework.getDestinationAttribute('ip_address')
```

Esta sección también incluye:

- ["Trabajo con los resultados de la asignación" abajo](#)
- ["Manejo de la conexión de prueba en la secuencia de comandos" en la página 203](#)

Trabajo con los resultados de la asignación

El adaptador genérico de inserción crea cadenas XML que describen los datos que se van a agregar, actualizar o eliminar del sistema de destino. La secuencia de comandos Jython tiene que analizar este XML y, a continuación, realizar la operación de agregar, actualizar o eliminar en el destino.

En el XML de la operación de adición que recibe la secuencia de comandos Jython, el atributo `mamId` para los objetos y vínculos siempre es el identificador UCMDB del objeto o vínculo original antes de que su tipo, atributo o cualquier otra información se modifique al esquema del sistema remoto.

En el XML de las operaciones de actualización o supresión, el atributo `mamId` de cada objeto o vínculo contiene la representación de cadena del mismo `ExternalId` que se devolvió de la secuencia de comandos Jython de la anterior sincronización.

En el XML, el atributo `id` de un CI contiene `cmdbId` como `Id. externo` o el `ExternalId` de ese CI si el CI obtuvo un `ExternalId` cuando se envió el CI a la secuencia de comandos. Los campos `end1Id` y `end2Id` del vínculo contienen para cada extremo del vínculo el `cmdbId` como `Id. externo` o el `ExternalId` de ese extremo del vínculo si el CI que está en el extremo del vínculo obtuvo un `ExternalId` cuando se envió a la secuencia de comandos.

Al procesar los CI en la secuencia de comandos Jython, el valor devuelto de la secuencia de comandos es una asignación entre el `Id.` de CMDB del CI y el `Id.` proporcionado (el `Id.` dado a cada CI de la secuencia de comandos). Si se inserta un CI por primera vez, el `Id.` que está en el XML de ese CI es el `Id.` de CMDB. Si no se inserta el CI por primera vez, el `Id.` del CI es el mismo `Id.` que se dio a ese CI en la secuencia de comandos cuando se insertó por primera vez.

El `Id.` se recupera de la secuencia de comandos XML de CI de la manera siguiente:

1. En el CI Element del XML, recupere el `Id.` del atributo `id`. Por ejemplo: `id = objectElement.getAttributeValue('id')`.
2. Tras recuperar el `Id.` del XML, restaure el `Id.` del atributo (cadena). Por ejemplo: `objectId = CmdbObjectID.Factory.restoreObjectID(id)`.
3. Compruebe si el `objectId` recibido en el paso anterior es el `Id.` de CMDB. Para hacer esto, puede comprobar si el `objectId` tiene el nuevo `Id.` que le proporciona la secuencia de comandos. En caso afirmativo, el `Id.` devuelto no es el `Id.` de CMDB. Por ejemplo:
`newId = objectId.getPropertyValue(<el nombre del atributo id proporcionado por la secuencia de comandos>)`.

Si `newId` es nulo, el `Id.` devuelto en el XML es un `Id.` de CMDB.

4. Si el `Id.` es un `Id.` de CMDB (es decir, `newId` es nulo), realice lo siguiente (si el `Id.` no es un `Id.` de CMDB, vaya al paso 5):

- a. Cree una propiedad para ese CI que contiene el nuevo Id. Por ejemplo: `propArray = [TypesFactory.createProperty('<el nombre del atributo id proporcionado por la secuencia de comandos>', '<nuevo Id.>')]`.
 - b. Cree un `externalId` para ese CI. Por ejemplo:

```
cmdbId = extI.getPropertyValue('internal_id')
className = extI.getType()
externalId = ExternalIdFactory.createExternalCiId(className, propArray)
```
 - c. Asigne el Id. de CMDB al `externalId` recién creado (y en el próximo paso, devuelva esa asignación al adaptador). Por ejemplo: `objectMappings.put(cmdbId, externalId)`
 - d. Cuando se asignan todos los CI y los vínculos:

```
updateResult = DataPushResultsFactory.createDataPushResults
(objectMappings, linkMappings);
return updateResult
```
5. Si el Id. es el nuevo Id. (es decir, `newId` no es nulo), el `externalId` es el `newId`.

También es posible informar sobre el estado de inserción para cada CI y vínculo de la forma siguiente:

1. `updateStatus = ReplicationActionDataFactory.createUpdateStatus();`
donde `updateStatus` es una instancia de la clase `UpdateStatus` que contiene los estados de los CI y los vínculos.
2. Agregue un estado a `updateStatus` llamando al método `reportCIStatus` o `reportRelationStatus`.

Por ejemplo:

```
status = ReplicationActionDataFactory.createStatus(Severity.FAILURE, 'Failed',
ERROR_CODE_CI, errorParams, Action.ADD);
updateStatus.reportCIStatus(externalId, status);
```

Donde `ERROR_CODE_CI` es el número de los mensajes de error tal como aparecen en el archivo de adaptador **properties.errors** (para más información sobre el archivo **properties.errors**, consulte ["Convenciones para la escritura de errores" en la página 68](#)) y `errorParams` contiene los parámetros que se van a pasar al mensaje. Consulte el javadoc **ReplicationActionDataFactory** para obtener más información.

3. Cree un resultado de inserción con los estados como se indica a continuación:

```
updateResult = DataPushResultsFactory.createDataPushResults(objectMappings,
linkMappings, updateStatus);
return updateResult
```

Ejemplo de resultado XML

```
<root>
  <data>
    <objects>
      <Object mode="update_else_insert" name="UCMDB_UNIX" operation="add" mamId="0c82f591bc3a584121b0b85efd90b174" id="HiddenRmiDataSource%0Aunix%0A1%0Ainternal_id%3DSTRING%3D0c82f591bc3a584121b0b85efd90b174%0A">
        <field name="NAME" key="false" datatype="char" length="255">UNIX5</field>
        <field name="DATA_NOTE" key="false" datatype="char" length="255"></field>
      </Object>
    </objects>
    <links>
      <link targetRelationshipClass="TALK" targetParent="unix" targetChild="unix" operation="add" mode="update_else_insert" mamId="265e985c6ec51a8543f461b30fa58f81" id="end1id%5BHiddenRmiDataSource%0Aunix%0A1%0Ainternal_id%3DSTRING%3D41372a1cbcaba27b214b84a2ec9eb535%0A%5D%0Aend2id%5BHiddenRmiDataSource%0Aunix%0A1%0Ainternal_id%3DSTRING%3D0c82f591bc3a584121b0b85efd90b174%0A%5D%0AHiddenRmiDataSource%0Atalk%0A1%0Ainternal_id%3DSTRING%3D265e985c6ec51a8543f461b30fa58f81%0A">
        <field name="DiscoveryID1">41372a1cbcaba27b214b84a2ec9eb535</field>
        <field name="DiscoveryID2">0c82f591bc3a584121b0b85efd90b174</field>
        <field name="end1Id">HiddenRmiDataSource%0Aunix%0A1%0Ainternal_id%3DSTRING%3D41372a1cbcaba27b214b84a2ec9eb535%0A</field>
        <field name="end2Id">HiddenRmiDataSource%0Aunix%0A1%0Ainternal_id%3DSTRING%3D0c82f591bc3a584121b0b85efd90b174%0A</field>
        <field name="NAME" key="false" datatype="char" length="255">TALK4</field>
        <field name="DATA_NOTE" key="false" datatype="char" length="255"></field>
      </link>
    </links>
  </data>
</root>
```

Nota: Si datatype="BYTE", el valor del resultado devuelto es una **cadena** que se genera como: new String([the byte array attribute]). El objeto byte[] puede reconstruirse mediante: <la cadena recibida>.getBytes(). Si hay diferencias en la configuración regional predeterminada entre el servidor y la sonda, la reconstrucción se realiza de acuerdo a

la configuración regional predeterminada del servidor.

Manejo de la conexión de prueba en la secuencia de comandos

Se puede invocar una secuencia de comandos Jython para probar la conexión con una aplicación externa. En este caso, el atributo de destino `testConnection` será `true`. Este atributo se puede obtener de `Framework` de la forma siguiente:

```
testConnection = Framework.getTriggerCIData('testConnection')
```

Al ejecutarse en modo de conexión de prueba, la secuencia de comandos generaría una excepción si no se puede establecer una conexión a la aplicación externa. En caso contrario, si la conexión se realiza correctamente, la función **DiscoveryMain** debería devolver un **OSHVResult** vacío.

Admisión de la sincronización diferencial

Para que el adaptador de inserción admita la sincronización diferencial, la función **DiscoveryMain** debe devolver un objeto que implemente la interfaz **DataPushResults**, que contiene las asignaciones entre los Id. que recibe la secuencia de comandos Jython desde el XML y los Id. que crea la secuencia de comandos Jython en el equipo remoto. Los últimos Id. son de tipo **ExternalId**.

El comando **ExternalIdUtil.restoreExternal**, que recibe el Id. del CI en CMDB como parámetro, restaura el Id. externo del Id. del CI en CMDB. Por ejemplo, este comando puede utilizarse al realizar una sincronización diferencial y se recibe un vínculo en que uno de sus extremos no está en la operación masiva (ya se ha sincronizado).

Si el método **DiscoveryMain** de la secuencia de comandos Jython en la que se base el adaptador de inserción devuelve una instancia **ObjectStateHolderVector** vacía, el adaptador no admitirá la sincronización diferencial. Esto significa que aunque se esté ejecutando un trabajo de sincronización diferencial, en realidad se está realizando una sincronización completa. Por tanto, no se puede actualizar ni suprimir ningún dato en el sistema remoto, ya que todos los datos se añaden a CMDB durante cada sincronización.

Importante: Si está implementando la sincronización diferencial en un adaptador existente que se creó en la versión 9.00 o 0.01, debe usar el archivo `push-adapter.zip` de la versión 9.02 o posterior para recrear su paquete de adaptador. Para obtener más información, consulte ["Construcción de un paquete de adaptadores" en la página 191](#).

Esta tarea permite que el adaptador de inserción realice la sincronización diferencial.

La secuencia de comandos Jython devuelve el objeto **DataPushResults** que contiene dos asignaciones Java: una para las asignaciones de ID de objetos (las claves y valores son objetos de tipo `ExternalCild`) y otra para los ID de vínculos (las claves y valores son objetos de tipos `ExternalRelationId`).

- Añada las siguientes instrucciones **from** a la secuencia de comandos Jython:

```
from com.hp.ucmdb.federationspi.data.query.types import ExternalIdFactory
```

```
from com.hp.ucmdb.adapters.push import DataPushResults

from com.hp.ucmdb.adapters.push import DataPushResultsFactory

from com.mercury.topaz.cmdb.server.fcldb.spi.data.query.types import ExternalIDUtil
```

- Utilice la clase de fábrica **DataPushResultsFactory** para obtener el objeto **DataPushResults** de la función **DiscoveryMain**.

```
# Create the UpdateResult object

updateResult = DataPushResultsFactory.createDataPushResults(objectMappings, linkMappings);
```

- Utilice los comandos siguientes para crear asignaciones Java para el objeto **DataPushResults**:

```
# Prepare the maps to store the mappings if IDs

objectMappings = HashMap()

linkMappings = HashMap()
```

- Utilice la clase **ExternalIdFactory** para crear los siguientes ID de ExternalId:
 - ExternalId para objetos o vínculos que se originan en un CMDB (por ejemplo, todos los CI de una operación de adición proceden de CMDB):

```
externaCIId = ExternalIdFactory.createExternalCmdbCiId(ciType, ciIDAsString)

externalRelationId = ExternalIdFactory.createExternalCmdbRelationId(linkType, end1ExternalCIId, end2ExternalCIId, linkIDAsString)
```

- ExternalId para objetos o vínculos que no se originan en un CMDB (por lo general, cada operación de actualización o supresión contiene dichos objetos):

```
myIDField = TypesFactory.createProperty("systemID", "1")

myExternalId = ExternalIdFactory.createExternalCiId(type, myIDField)
```

Nota: Si la secuencia de comandos Jython ha actualizado la información existente y el ID del objeto (o vínculo) cambia, debe devolver una asignación entre el ID externo anterior y el nuevo.

- Utilice los métodos **restoreCmdbCiIDString** o **restoreCmdbRelationIDString** de la clase **ExternalIdFactory** para recuperar la cadena UCMDB ID de un ID externo de un objeto o vínculo

que se originó en UCMDB.

- Utilice los métodos **restoreExternalCiid** y **restoreExternalRelationId** de la clase **ExternalIdUtil** para restaurar el objeto **ExternalId** desde el valor del atributo **mamId** del XML de las operaciones de actualización o supresión.

Nota: Los objetos **ExternalId** son en realidad una matriz de propiedades. Esto significa que puede usar un objeto **ExternalId** para almacenar toda la información que puede necesitar que identificará los datos en el sistema remoto.

Consultas de SQL del adaptador de inserción XML genérico

En el paquete de adaptador, el archivo **sql_queries** ubicado en **adapterCode > PushAdapter > sqlTablesCreation** contiene las consultas necesarias para crear tablas en un esquema nuevo en Oracle para probar el adaptador. Las tablas corresponden al archivo **adapterCode\<Id. de adaptador>\mappings\mappings.xml**.

Nota: El archivo **sql_queries** no es necesario para el adaptador. Solo es un ejemplo.

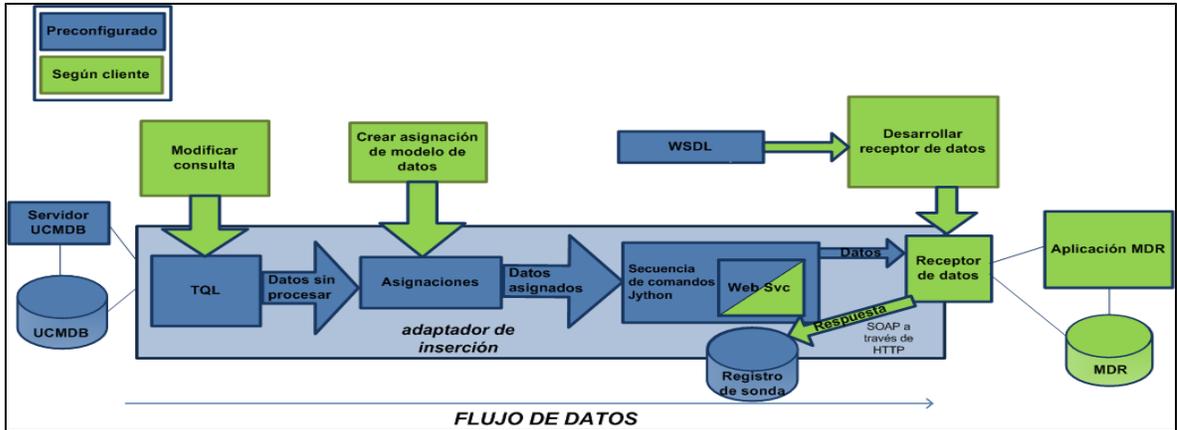
Adaptador genérico de inserción de servicio web

El adaptador genérico de inserción de servicio web proporciona una inserción iniciada por UCMDB de mensajes SOAP que contienen datos de consulta a un receptor de datos de servicio web. Los resultados asignados se envían en los mensajes SOAP estándar a través del protocolo HTTP POST al receptor de datos. El receptor de datos debe comprender los mensajes SOAP producidos por el adaptador de inserción. Para facilitar el desarrollo del receptor de datos adecuado, se proporciona un WSDL con este adaptador de inserción.

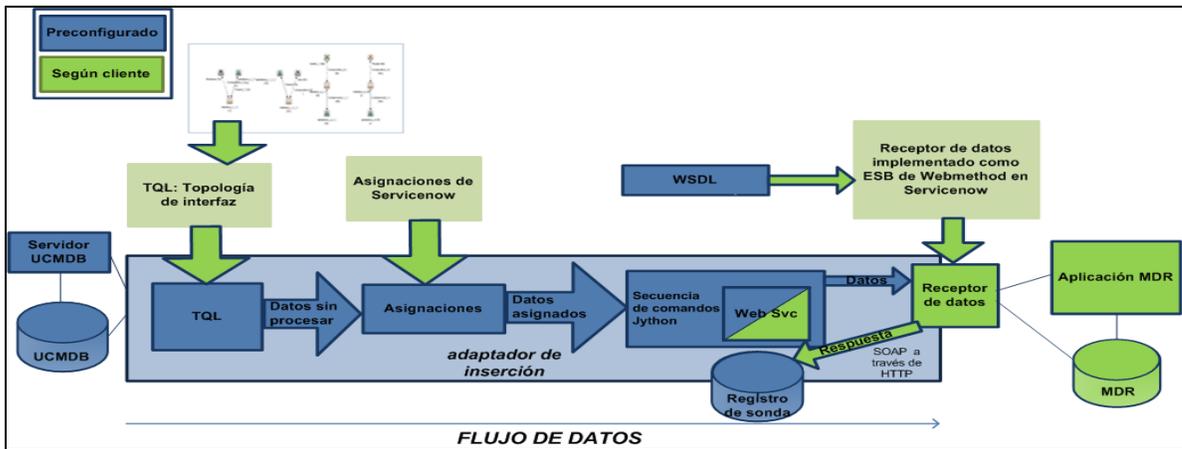
El proceso personalizado del XML de respuesta al mensaje SOAP es posible en la secuencia de comandos Jython.

Para comprender el formato de los datos asignados entrantes, el programador del receptor de datos debe comunicarse con el desarrollador del archivo de asignaciones. No se proporciona un **.xsd** con esta versión del adaptador de inserción de servicio web, por lo que los datos deben ser procesados de una manera que refleje los datos entrantes, que son una combinación del TQL original y las asignaciones aplicadas.

Las funciones del adaptador de inserción de servicio web para insertar datos al cliente se muestran a continuación. Los elementos de color verde se han personalizado o proporcionado por el cliente para implementar el adaptador para un destino de inserción específico. Los elementos en azul son los componentes de serie.

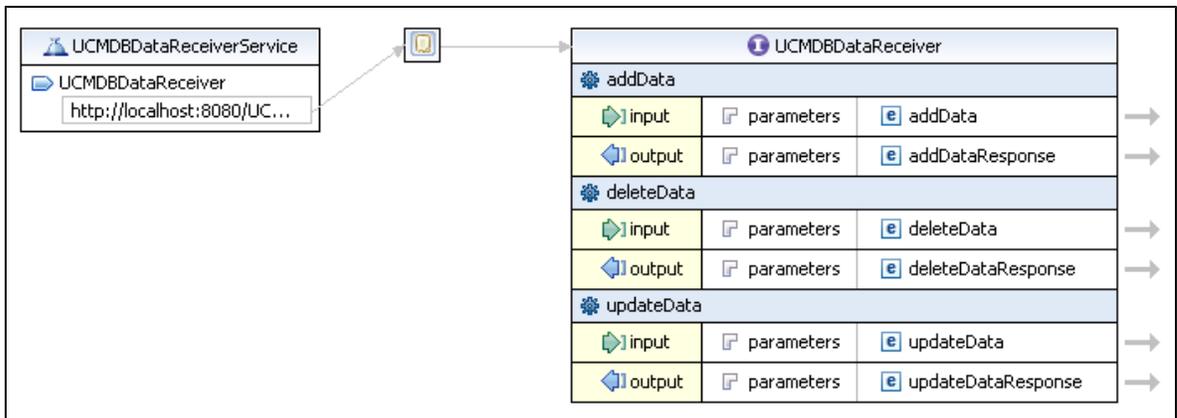


Aquí se muestra un ejemplo de una implementación del adaptador genérico de inserción de servicio web a un adaptador de inserción específico de MDR mediante un ESB (Enterprise Service Bus):



WSDL

Se proporciona un WSDL al programador del cliente para crear un receptor de datos capaz de comunicarse con el adaptador de inserción de UCMDB a través de un servicio web. El **UCMDBDataReceiver.wsdl** describe los mensajes SOAP que se utilizan para comunicar datos desde el UCMDB al receptor de datos. Aquí se muestra el diagrama de diseño del WSDL:



El receptor de datos (que en la práctica es un servidor o "punto final de servicio" en la terminología SOAP) deben implementar tres métodos: **addData**, **deleteData** y **updateData**, correspondientes a los conjuntos de datos que inserta UCMDB. Los encabezados HTTP contienen la palabra clave **SoapAction** correcta que indica el tipo de datos que se está enviando. El receptor de datos es responsable de implementar la lógica empresarial y procesar los datos.

La dirección URL de WSDL predeterminada es:

- <http://localhost:8080/UCMDBDataReceiver/services/UCMDBDataReceiver?wsdl>

Tal como la implementa el receptor de datos, la dirección URL podría parecerse a la siguiente:

- <http://testWSPAserver:4444/MyCo.IT.SvcMgt.ws.us:provider/UCMDBDataReceiver?wsdl>

La dirección URL del servicio web es la misma que la dirección URL de WSDL sin "?wsdl" al final.

El origen del WSDL se incluye a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://ucmdb.hp.com" xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://ucmdb.hp.com" xmlns:intf="http://ucmdb.hp.com" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4 Built on Apr 22, 2006 (06:55:48 PDT)-->
  <wsdl:types>
    <schema elementFormDefault="qualified" targetNamespace="http://ucmdb.hp.com" xmlns="http://www.w3.org/2001/XMLSchema">
      <element name="addData">
        <complexType>
          <sequence>
            <element name="xmlAdded" type="xsd:string"/>
          </sequence>
        </complexType>
      </element>
      <element name="addDataResponse">
        <complexType/>
      </element>
      <element name="deleteData">
```

```
        <complexType>
            <sequence>
                <element name="xmlDeleted" type="xsd:string"/>
            </sequence>
        </complexType>
    </element>
    <element name="deleteDataResponse">
        <complexType/>
    </element>
    <element name="updateData">
        <complexType>
            <sequence>
                <element name="xmlUpdate" type="xsd:string"/>
            </sequence>
        </complexType>
    </element>
    <element name="updateDataResponse">
        <complexType/>
    </element>
</schema>
</wsdl:types>

<wsdl:message name="addDataRequest">
    <wsdl:part element="impl:addData" name="parameters">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="deleteDataResponse">
    <wsdl:part element="impl:deleteDataResponse" name="parameters">
    </wsdl:part>
</wsdl:message>
<wsdl:message name="updateDataResponse">
```

```
<wsdl:part element="impl:updateDataResponse" name="parameters">
  </wsdl:part>
</wsdl:message>
<wsdl:message name="deleteDataRequest">
  <wsdl:part element="impl:deleteData" name="parameters">
    </wsdl:part>
  </wsdl:message>
<wsdl:message name="addDataResponse">
  <wsdl:part element="impl:addDataResponse" name="parameters">
    </wsdl:part>
  </wsdl:message>
<wsdl:message name="updateDataRequest">
  <wsdl:part element="impl:updateData" name="parameters">
    </wsdl:part>
  </wsdl:message>
<wsdl:portType name="UCMDBDataReceiver">
  <wsdl:operation name="addData">
    <wsdlsoap:operation soapAction="addDataRequest"/>
    <wsdl:input message="impl:addDataRequest" name="addDataRequest">
      </wsdl:input>
    <wsdl:output message="impl:addDataResponse" name="addDataResponse">
      </wsdl:output>
    </wsdl:operation>
  <wsdl:operation name="deleteData">
    <wsdlsoap:operation soapAction="deleteDataRequest"/>
    <wsdl:input message="impl:deleteDataRequest" name="deleteDataRequest">
      </wsdl:input>
    <wsdl:output message="impl:deleteDataResponse" name="deleteDataResponse">
      </wsdl:output>
  </wsdl:operation>
</wsdl:portType>
</wsdl:binding>
</wsdl:service>
```

```
</wsdl:operation>
<wsdl:operation name="updateData">
  <wsdlsoap:operation soapAction="updateDataRequest"/>
  <wsdl:input message="impl:updateDataRequest" name="updateDataRequest">
  </wsdl:input>
  <wsdl:output message="impl:updateDataResponse" name="updateDataResponse">
  </wsdl:output>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="UCMDBDataReceiverSoapBinding" type="impl:UCMDBDataReceiver">
  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="addData">
    <wsdl:input name="addDataRequest">
      <wsdlsoap:body use="literal" />
    </wsdl:input>
    <wsdl:output name="addDataResponse">
      <wsdlsoap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="deleteData">
    <wsdl:input name="deleteDataRequest">
      <wsdlsoap:body use="literal" />
    </wsdl:input>
    <wsdl:output name="deleteDataResponse">
      <wsdlsoap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="updateData">
    <wsdl:input name="updateDataRequest">
```

```
        <wsdlsoap:body use="literal" />
    </wsdl:input>
    <wsdl:output name="updateDataResponse">
        <wsdlsoap:body use="literal" />
    </wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="UCMDBDataReceiverService">
    <wsdl:port binding="impl:UCMDBDataReceiverSoapBinding" name="UCMDBDataReceiver">
        <wsdlsoap:address location="http://localhost:8080/UCMDBDataReceiver/services/UCMDBDataReceiver"/>
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Manejo de respuestas

El receptor de datos debe devolver una cadena en las estructuras **addDataResponse**, **deleteDataResponse** o **updateDataResponse**. El adaptador pasa los datos de respuesta no procesados al **probeMgr-adaptersDebug.log** de la sonda. El receptor puede devolver datos de cadena y las respuestas se encapsulan en XML compatible con SOAP. En la secuencia de comandos Jython puede usar **SOAPMessage** y las clases Java relacionadas para analizar los mensajes de respuesta. A continuación se muestra un ejemplo de un mensaje de respuesta del receptor de datos:

```
<2012-03-16 15:47:38,080> [INFO ] [Thread-110] - XMLtoWebService.py:addData
received response:
<soapenv:Body xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<intf:addDataResponse xmlns:intf="http://ucmdb.hp.com">
    <xml>&lt;result&gt;&lt;status&gt;error&lt;/status&gt;
    &lt;message&gt;Error publishing config item changes&lt;/message&gt;
    &lt;/result&gt;</xml>
</intf:addDataResponse>
</soapenv:Body>
```

El mensaje que se muestra es un mensaje de error **<Error al publicar cambios de elemento de configuración.>**, pero el contenido puede ser cualquier elemento con el que el receptor de datos

está diseñado para responder. La respuesta es un mensaje de error simplemente porque es la intención, porque el diseñador dice que se trata de un mensaje de error y el adaptador de inserción espera que la respuesta sea alguna indicación de éxito o fallo. El contenido puede ser los Id. de conciliación de todos los CI añadidos correctamente, o mensajes de error para unos CI concretos. Personalizar GWSPA podría incluir el análisis del mensaje de respuesta y la adopción de acciones tales como reenviar determinados CI o realizar otro registro.

Prueba de WSDL

El complemento SOAPUI de Eclipse se usa para probar capas de servicio web durante el desarrollo. Puede usar SOAPUI como ayuda en la personalización de un servicio web. SOAPUI ofrece un entorno de desarrollo integrado (IDE) para probar la compilación, envío y recepción de mensajes SOAP. En la perspectiva de SOAPUI, el WSDL en las páginas [207-211](#) ha generado el siguiente mensaje de muestra:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ucm="http://ucmdb.hp.com">
  <soapenv:Header/>
  <soapenv:Body>
    <ucm:addData>
      <ucm:xmlAdded>?</ucm:xmlAdded>
    <ucm:addData>
  </soapenv:Body>
</soapenv:Envelope>
```

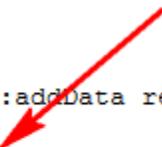
El elemento "?" en el elemento **xmlAdded** anterior es la ubicación de los datos, proporcionada por la integración del adaptador de inserción de servicio web.

Observación de los resultados

Cuando el adaptador de inserción está funcionando normalmente, en modo de no depuración, los datos no se escriben nunca en un archivo hasta que se escribe el resultado final (los resultados intermedios del TQL y los resultados de datos asignados normalmente no son visibles en ningún archivo de registro). Sin embargo, los resultados se pueden escribir en el archivo de depuración de la sonda eliminando los comentarios de las instrucciones **logger.debug** (elimine el carácter "#") en la sección DiscoveryMain como se muestra aquí:

```
# uncomment out the logger statements to see the data
empty = isEmpty(addResult, "addResult")
if not empty:
    addResult = cleanUp(addResult)
    # send to ESB web service
    logger.info(SCRIPT_NAME+":addData received response:"+resp)
    #logger.debug(addResult)
empty = isEmpty(updateResult, "updateResult")
```

Suprima # para activar el registro de depuración de datos



Asegúrese de que la instrucción del registrador comienza en la misma columna que las otras líneas anteriores y siguientes. Jython es sensible a la sangría y la secuencia de comandos dará error si la sangría de todas las líneas no es correcta.

El archivo del registro de depuración **probeMgr-adaptersDebug.log** en la sonda muestra el contenido de la salida:

```
<2011-12-07 14:02:23,019> [INFO ] [Thread-273] - XMLtoWebService.py started
<2011-12-07 14:02:23,019> [DEBUG] [Thread-273] - ESB Push parameters:
<2011-12-07 14:02:23,019> [DEBUG] [Thread-273] - Wshost=harpy.trtc.com
<2011-12-07 14:02:23,019> [DEBUG] [Thread-273] - WShostport=5555
<2011-12-07 14:02:23,019> [DEBUG] [Thread-273] - WSuri=ws/DtITServiceManagement.esla.v1.ws.provider:UMDBDataReceiver
<2011-12-07 14:02:23,019> [INFO ] [Thread-273] - URL is http://harpy.trtc.com:5555/ws/DtITServiceManagement.esla.v1.ws.provider:UMDBDataReceiver
<2011-12-07 14:02:23,035> [DEBUG] [Thread-273] - Connected to http://harpy.trtc.com:5555/ws/DtITServiceManagement.esla.v1.ws.provider:UMDBDataReceiver
<2011-12-07 14:02:23,035> [ERROR] [Thread-273] - sending results
<2011-12-07 14:02:23,035> [DEBUG] [Thread-273] - <?xml version="1.0" encoding="UTF-8"?>
<root>
  <data>
    <objects>
      <Object mode="" name="u_imp_ip_switch" operation="add" mamId="9e8c2f6bdfe4b7d0864c79e70833902c">
```

```
<field name="Correlation ID" key="true" datatype="char" length="">9e8c2f6bdfc4b7d0864c79e70833902c</field>

<field name="name" key="false" datatype="char" length="">nm
a_09sw</field>

<field name="location" key="false" datatype="char" length="" />

<field name="u_chassis_vendor_type" key="false" datatype="char" length="">ciscoCat2960-24TT</field>

<field name="serial_number" key="false" datatype="char" length="" />

<field name="ram" key="false" datatype="char" length="" />

<field name="os_version" key="false" datatype="char" length="" />

</Object>
```

Modificación de la secuencia de comandos Jython

XMLtoWebService.py

La secuencia de comandos Jython utilizada por el adaptador de inserción de servicio web es muy similar al adaptador de inserción XML. La secuencia de comandos utiliza **UCMDBDataReceiver.jar**, que se incluye con el adaptador. La secuencia de comandos implementa el método **SendDataToReceiver()**. **SendDataToReceiver()** utiliza tres parámetros:

1. Acción (agregar, actualizar o eliminar)
2. La dirección URL del receptor de datos
3. Los datos

Por ejemplo, el bloque de agregar es similar a lo siguiente: **SendDataToReceiver("add", URL, addResult)**

Se ajustan todas las capas de servicio web y SOAP. La dirección URL es la dirección de punto final de servicio del receptor de datos de UCMDB. Se trata de la misma dirección URL utilizada para obtener el archivo wsdl a través del sufijo "?wsdl".

El origen de la secuencia de comandos Jython se muestra a continuación. Las líneas de contenedor de la integración de servicio web se **resaltan en verde**.

```
#####
# script: XMLtoWebService.py
#####
```

```
# This jython script accepts TQL data results (adds, updates, and deletes) from the Integration adapter.
# and sends it to a web service. The web service is called UCMDBDataReceiver.
# A web service client of this name must be addressable at the URL provided by the parameters.
# The SendDataToReceiver.jar exposes the SendDataToReceiver function, as well as the service locator.
# examples of the service locator are in the testconnection section.
# regular expressions
import re
# logging
import logger
# web service interface
from com.hp.ucmdb import SendDataToReceiver
from com.hp.ucmdb.SendDataToReceiver import locateService
from com.hp.ucmdb.SendDataToReceiver import SendData
#####
#####          VARIABLES          #####
#####
SCRIPT_NAME = "XMLtoWebService.py"
logger.info(SCRIPT_NAME+" started")
def cleanUp(str):
    # replace mode=""
    str = re.sub("mode=\"\w+\\"s+", "", str)
    # replace mamId with id
    str = re.sub("\smamId=\"", " id=\"", str)
    # replace empty attributes
    str = re.sub("[\n|\\s|\\r]*<field name=\"\w+\" datatype=\"\w+\" />", "", str)
```

```
# replace targetRelationshipClass with name
str = re.sub("\stargetRelationshipClass=\"", " name=\"", str)

# replace Object with object with name
str = re.sub("<Object mode=\"", "<object mode=\"", str)
str = re.sub("<Object operation=\"", "<object operation=\"", str)
str = re.sub("<Object name=\"", "<object name=\"", str)
str = re.sub("</Object>", "</object>", str)

# replace field to attribute
str = re.sub("<field name=\"", "<attribute name=\"", str)
str = re.sub("</field>", "</attribute>", str)

#logger.debug("String = %s" % str)
#logger.debug("cleaned up")

return str
def isEmpty(xml, type = ""):
    objectsEmpty = 0
    linksEmpty = 0

    m = re.findall("<objects />", xml)
    if m:
        #logger.warn("\t[%s] No objects found" % type)
        objectsEmpty = 1

    m = re.findall("<links />", xml)
    if m:
        #logger.warn("\t[%s] No links found" % type)
        linksEmpty = 1
```

```
if objectsEmpty and linksEmpty:
    return 1
return 0

#####
#####      MAIN      #####
#####

def DiscoveryMain(Framework):
    #fix this for web service export
    errMsg = "UCMDBDataReceiver Service not found."
    testConnection = Framework.getTriggerCIData("testConnection")
    # Get Web Service Push variables
    WShostName = Framework.getTriggerCIData("Host Name")
    WShostport = Framework.getTriggerCIData("Protocol Port")
    WSuri = Framework.getTriggerCIData("URI")

    logger.info(SCRIPT_NAME+":ESB Push parameters:")
    logger.info("Host Name="+WShostName)
    logger.info("Protocol Port="+WShostport)
    logger.info("URI="+WSuri)
    URL = "http://" + WShostName + ":" + WShostport + "/" + WSuri
    logger.info("URL="+URL)
    if testConnection == 'true':
        # locate the service
        test_receiver = SendDataToReceiver()
        locator = test_receiver.locateService(URL)
        #locator = locateService(URL)
        if(locator):
            logger.info(SCRIPT_NAME+":Test connection was successful")
            return
        else:
            raise Exception, errMsg
```

```
        return

# do same thing here if not just a test connection -
receiver = SendDataToReceiver()
locator = receiver.locateService(URL)
if(locator):
    logger.info(SCRIPIT_NAME+":Connected to "+URL)
else:
    logger.error(SCRIPIT_NAME+":no locator")
    raise Exception, errMsg
    return

# get add/update/delete result objects from the Framework
addResult = Framework.getTriggerCIData('addResult')
updateResult = Framework.getTriggerCIData('updateResult')
deleteResult = Framework.getTriggerCIData('deleteResult')
logger.debug(deleteResult)

# get referenced data - unused in this adapter implementation
#addRefResult = Framework.getTriggerCIData('referencedAddResult')
#updateRefResult = Framework.getTriggerCIData('referencedUpdateResult')
#deleteRefResult = Framework.getTriggerCIData('referencedDeleteResult')
# uncomment out the logger statements to see the data
empty = isEmpty(addResult, "addResult")
if not empty:
    addResult = cleanUp(addResult)
    # send to ESB web service
    logger.info(SCRIPIT_NAME+":sending addData Result")
    rcvr = SendDataToReceiver()
    resp = rcvr.SendData("add", URL, addResult)
    logger.info(SCRIPIT_NAME+":addData received response:"+resp)
    #logger.debug(addResult)
```

```
empty = isEmpty(updateResult, "updateResult")
if not empty:
    updateResult = cleanUp(updateResult)
    # send to ESB web service
    #logger.debug(updateResult)
    logger.info(SCRIPY_NAME+":sending updateData Result")
    rcvr = SendDataToReceiver()
    resp = rcvr.SendData("update", URL, updateResult)
    logger.info(SCRIPY_NAME+":received response:"+resp)

empty = isEmpty(deleteResult, "deleteResult")
if not empty:
    deleteResult = cleanUp(deleteResult)
    # send to ESB web service
    #logger.debug(deleteResult)
    logger.info(SCRIPY_NAME+":sending deleteData Result")
    rcvr = SendDataToReceiver()
    resp = rcvr.SendData("delete", URL, deleteResult)
    logger.info(SCRIPY_NAME+":received response:"+resp)
logger.info(SCRIPY_NAME+" ended")
```

Personalización del proceso de mensajes de respuesta

El receptor de datos debe devolver una cadena que contenga cualquier respuesta o estado deseado. El adaptador de inserción de servicio web pasa la respuesta de forma predeterminada al registro de nivel de información de la sonda. El mensaje de respuesta es XML con formato SOAP que contiene la(s) cadena(s) devuelta(s) en su interior. Cualquier dato puede ser devuelto por el receptor como agrupado o mensajes individuales de error o éxito. Si se desea proceso adicional, la respuesta puede ser procesada por la secuencia de comandos Jython del adaptador. No es necesaria ninguna planificación en Java.

Un ejemplo de mensaje de respuesta de devolución, enviado utilizando lo siguiente:

```
// stub example for building your own UCMDBDataReceiver
public class UCMDBDataReceiver {
```

```
public String addData (String xmlAdd){
    System.out.println(xmlAdd); // do something with the data
    // send back a response message based on what you did
    String tr = new String("a test response from addData!");
    return tr;
}
```

se muestra aquí:

```
<soapenv:Body xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <addDataResponse xmlns="http://ucmdb.hp.com">
    <addDataReturn>a test response from addData!</addDataReturn>
  </addDataResponse>
</soapenv:Body>
```

Modificación del receptor de datos

Un cliente Java puede implementar las clases contenidas en **UCMDBDataReceiver.jar** y llamar al servicio web de la misma manera que Jython. Además, también se puede llamar a los métodos sin ajustar. Existe un javadoc para las clases **UCMDBDataReceiver.jar**. El código fuente siguiente muestra cómo utilizar estos métodos esenciales para ajustar los datos en un mensaje SOAP y enviarlos al receptor a través de HTTP.

El proceso consiste en crear un objeto **UCMDBDataReceiverServiceLocator** y, a continuación, asignar **UCMDBDataReceiverEndPointAddress** a la dirección URL del receptor de datos.

Para enviar datos, se llama al método **getUCMDBDataReceiver** del localizador para crear un objeto **UCMDBDataReceiver**. El objeto **UCMDBDataReceiver** implementa los métodos para enviar realmente los datos de agregar/cambiar/eliminar. Hay tres bloques de código idénticos para procesar cada tipo de solicitud.

El código fuente para la clase **SendDataToReceiver** se muestra a continuación. Los objetos y métodos resaltados son los elementos esenciales de uso.

```
/**
 * Test SendData for the UCMDB Data Receiver for the UCMDB Web Service Push A
 dapter
 */
package com.hp.ucmdb;
import com.hp.ucmdb.SendDataToReceiver;
/**
```

```
* TestSendData can be used to verify the SOAP classes are working.
* TestSendData creates a SendDataToReceiver class and invokes its SendData method.
* a response String is returned.
* The test URL is typically appended with "?wsdl" to get the WSDL of the service.
*/
public class TestSendData {
    /**
     * @param args - test SOAP message.
     * optional arguments [0] a test string [1] a service endpoint URL of a Data Receiver.
     * the default URL is sent the incoming argument as a test message.
     * the default URL is "http://localhost:8080/UCMDBDataReceiver/services/UCMDBDataReceiver".
     * If any errors are encountered, TestClient will attempt to throw exceptions.
     */
    public static void main(String[] args) {
        // use test message if supplied, otherwise supply a default test string
        String teststring = new String("Test SOAP message from UCMDBDataReceiver TestSendData.");
        if(args.length > 0) {
            teststring = args[0];
        }
        // use test URL if supplied, otherwise supply the default URL
        String URL = new String("");
        if(args.length > 1) {
            URL = args[1];
        }
        // return response
        String response = new String("");
        // perform the tests
```

```
try{
    if(URL.equals("")) {
        UCMDBDataReceiverServiceLocator locator = new UCMDBDataReceiverServiceLocator();
        UCMDBDataReceiver receiver = locator.getUCMDBDataReceiver();
        URL = locator.getUCMDBDataReceiverAddress();
        System.out.println("TestClient: tested URL="+locator.getUCMDBDataReceiverAddress());
        System.out.println("TestClient: receiver="+receiver.toString());
    }
    SendDataToReceiver sdtr = new SendDataToReceiver();
    // this sends a test push and gets a response message
    response = sdtr.SendData("add", URL, args[0]);
    System.out.println("Response received was:"+response);
} catch(Exception e){
    System.out.println("TestClient: Remote Error:");
    e.printStackTrace();
}
}
```

El código fuente también se incluye en el archivo **UCMDBDataReceiver**. para las demás clases:

- TestClient.java
- UCMDBDataReceiver.java
- UCMDBDataReceiverProxy.java
- UCMDBDataReceiverService.java
- UCMDBDataReceiverServiceLocator.java
- UCMDBDataReceiverSoapBindingStub.java

El origen se generó en el IDE de Eclipse y luego fue modificado. Tenga precaución al modificar el código de UCMDB, ya que gran parte del mismo se ha generado automáticamente para coincidir con la especificación SOAP y el receptor de datos de UCMDB.

Javadoc

Se proporciona un **javadoc** completamente comentado con el adaptador genérico de inserción de servicio web. El **javadoc** está incluido en la carpeta de documentos **javadoc**. Empiece por **index.html**. La página de información general proporciona acceso a la documentación de todas las clases y métodos en el SDK.

Todas las clases

- **SendDataToReceiver**: API para el contenedor de servicio web
- **TestClient**: cliente de prueba para verificar la conectividad con un punto final de servicio
- **UCMDBDataReceiver**: contenedor de servicio web

El resto se genera automáticamente por el compilador de servicio web:

- UCMDBDataReceiverProxy
- UCMDBDataReceiverService
- UCMDBDataReceiverServiceLocator
- UCMDBDataReceiverSoapBindingStub

Información general

El uso básico del SDK, incluidos ejemplos de código fuente, se explica en la documentación del paquete. Este **javadoc** es para el adaptador de inserción de servicio web de UCMDB. La API se puede llamar desde Jython o Java.

El SDK proporciona dos muestras de origen, **TestClient** y **SendDataToReceiver**. **TestClient** proporciona una prueba muy limitada del cliente local que responde. **SendDataToReceiver** es la clase principal que se utiliza para enviar datos a un servicio web.

En primer lugar, utilice este SDK (principalmente el WSDL incluido) para implementar un receptor de datos de UCMDB para comunicarse con este servicio web. A continuación, use este SDK para crear un adaptador de inserción en UCMDB para insertar los datos de resultados del TQL de UCMDB en el receptor de datos. El uso básico de esta API se describe a continuación, con implementaciones de Jython y Java.

Implementación de SendDataToReceiver()

SendDataToReceiver() ajusta todas las funciones con un solo método:

- Jython: `SendDataToReceiver("add",yourURL,"Hello!")`
- Java: `SendDataToReceiver("add",yourURL,"Hello!");`

O bien, cree un objeto **SendDataToReceiver** (por ejemplo, para manipular otros ajustes) y, a continuación, llame al método **SendData** por separado, tal como se muestra aquí:

- **Jython:**

```
rcvr = SendDataToReceiver()  
responseMsg = rcvr.SendData("add", yourURL, "Hello!")
```

- **Java:**

```
SendDataToReceiver rcvr = new SendDataToReceiver();  
String responseMsg = rcvr.SendData("add", yourURL, "Hello!");
```

O bien, si tiene que hacerlo paso a paso, puede hacer lo siguiente:

1. Cree un nuevo objeto **UCMDBDataReceiverServiceLocator()** x y luego establezca la dirección de punto final del objeto, como se muestra aquí:

- **Jython:**

```
x = UCMDBDataReceiverServiceLocator()  
x.setUCMDBDataReceiverEndPointAddress(URL)
```

- **Java:**

```
UCMDBDataReceiverServiceLocator x = new UCMDBDataReceiverServiceLocator();  
x.setUCMDBDataReceiverEndPointAddress(URL);
```

2. A continuación, cree un **UCMDBDataReceiver** con

- **Jython:** y = x.getUCMDBDataReceiver()

- **Java:** UCMDBDataReceiver y = x.getUCMDBDataReceiver();

3. A continuación, envíe los datos a través del servicio web SOAP así:

- **Jython:**

- y.addData(yourData)
- o y.updateData(yourData)
- o y.deleteData(yourData)

- **Java:**

- y.addData(yourData);
- o y.updateData(yourData);
- o y.deleteData(yourData);

4. Quizá sea necesario probar la conectividad y, si es correcta, volver a utilizar el mismo objeto localizador para devolver **UCMDBDataReceiver** para usarlo para la transferencia de datos.

Las clases no contienen destructores y no realizan administración de memoria.

Referencia de archivos de asignación

Utilización de las asignaciones

Debe crearse una asignación para cada atributo de destino de la salida XML transformada. Las asignaciones especifican dónde y cómo obtener los datos. Si los datos se encuentran en otro atributo correspondiente en UC MDB, se utiliza una asignación directa.

Para extraer datos de varios atributos, o atributos de los atributos del CI primario o secundario de UC MDB, puede que sean necesarias otras asignaciones complejas. El esquema de asignación siguiente muestra todas las asignaciones posibles.

El archivo de asignación es un archivo XML que define qué tipos de CI/relación de UC MDB se asignan a qué tipos de CI/relación del almacén de datos de destino. El formato se explica con detalle a continuación. El archivo de asignación controla qué tipos de CI y relaciones se insertan, así como controlar exactamente qué atributos se insertan.

Existe una entrada de asignación para cada atributo que debe insertarse en el MDR de destino. Cada entrada de asignación puede consistir en uno o más atributos en los datos de inserción sin procesar de UC MDB. Las entradas de asignación permiten un control completamente granular de la estructura y nomenclatura final de los datos que se van a insertar en el MDR de destino.

Asignaciones directas

Las asignaciones transforman un modelo de datos en otro (en este caso, UC MDB al MDR de destino de inserción). Las transformaciones pueden ser sencillas, en el caso de una relación 1:1 entre el atributo de UC MDB y el destino, solo se diferencian en el nombre y tal vez el tipo.

La mayoría de las asignaciones de atributo son directas. Por ejemplo, el nombre del servidor "ServerX", puede representarse en UC MDB como un CI de tipo **unix** con un nombre de atributo de **primary_server_name**, de tipo **string** con una longitud máxima de 50. El modelo de datos del MDR de destino puede especificar la misma entidad lógica con un CI de tipo **linux**, con un nombre de atributo de **host** con un tipo de **char** con una longitud máxima de 250. Las asignaciones directas pueden realizar todos estos tipos citados de tareas de conversión.

A continuación se muestra un ejemplo de una asignación directa:

```
<target_attribute name="dns_domain" datatype="char">  
<map type="direct" source_attribute="domain_name" />  
</target_attribute>
```

Esta asignación directa asigna el atributo **dns_domain** de UC MDB al atributo **domain_name** en el modelo de datos de destino.

Utilice el tipo de datos **char** independientemente del tipo de datos real, salvo que sea necesario para usar el tipo de datos real.

Asignaciones complejas

Más asignaciones complejas habilitan transformaciones adicionales:

- Para asignar valores de atributo de varios CI a un CI de destino.
- Para asignar atributos de CI secundarios (los que tienen una relación **container_f** o contenida) al CI primario del almacén de datos de destino. Por ejemplo, establecer un valor denominado **Número de CPU** en un CI Host de destino. Otro ejemplo podría ser establecer el valor de **Memoria total** (añadiendo los valores de tamaño de memoria de todos los CI de memoria de un CI de host en UCMDB) en un CI Host de destino.
- Para asignar atributos de CI primarios (los que tienen una relación **container_f** o contenida) en el CI del almacén de datos de destino. Por ejemplo, establecer un valor denominado **Servidor de contenedor** en un atributo de destino denominado CI de **Software instalado**, obteniendo el valor del host que contiene el CI de software en UCMDB.

A continuación se muestra un ejemplo de una asignación compleja, utilizando dos atributos de origen separados por un carácter de coma, para crear el atributo de destino **os**:

```
<target_attribute name="os" datatype="char">  
  <map type="compoundstring">  
    <source_attribute name="discovered_os_name" />  
    <constant value="," />  
    <source_attribute name="host_osinstalltype" />  
  </map>  
</target_attribute>
```

Inversión de direcciones de vínculo

Es posible que UCMDB contenga datos que difieren en estructura de un origen a otro. Por ejemplo, la relación entre un CI IpAddress y un CI Interface puede ser **parent**, como puede ocurrir con la integración de HP Network Node Manager. O puede ser un vínculo **containment** como Universal Discovery suele crearlo. Además, los sentidos de estos vínculos son opuestos entre sí.

Actualmente no es posible invertir la dirección de los vínculos en el archivo de asignaciones. La inversión de las variables **_end1** y **_end2** conmuta el orden de los datos en el XML transformado, o falta el vínculo en los datos de origen.

Una solución posible a este problema es definir una regla de enriquecimiento como se indica a continuación:

1. La parte de TQL del enriquecimiento es un subconjunto de un TQL utilizado por el adaptador de inserción. Este TQL en particular selecciona todos los vínculos que están en la dirección opuesta de lo deseado en el xml transformado.
2. La parte del enriquecimiento define un nuevo vínculo de la dirección correcta y el tipo deseado.
3. El enriquecimiento se activa y luego crea los vínculos correctos.

4. El TQL del trabajo de integración se refiere ahora al vínculo enriquecido, en lugar de hacerlo al vínculo original.
5. Las asignaciones de <vínculo> en el adaptador de inserción, por tanto, se refieren al vínculo enriquecido y producen un conjunto de vínculos coherentes en tipo y dirección.

Esquema del archivo de asignación

Nombre y ruta del elemento	Descripción	Atributos
integración	Define el contenido de asignación del archivo. Debe ser el bloque más exterior del archivo excepto para la línea del comienzo y cualquier comentario.	
info (integration)	Define la información sobre los repositorios de datos que se están integrando.	
source (integration > info)	Define la información sobre el repositorio de datos de origen.	<ol style="list-style-type: none"> 1. Nombre: type Descripción: Nombre del repositorio de datos de origen. ¿Es necesario?: necesario Tipo: Cadena 2. Nombre: versions Descripción: Versiones de los repositorios de datos de origen. ¿Es necesario?: necesario Tipo: Cadena 3. Nombre: vendor Descripción: Proveedor del repositorio de datos de origen. ¿Es necesario?: necesario Tipo: Cadena

Nombre y ruta del elemento	Descripción	Atributos
target (integration > info)	Define la información sobre el repositorio de datos de destino.	<ol style="list-style-type: none"> 1. Nombre: type Descripción: Nombre del repositorio de datos de origen. ¿Es necesario?: necesario Tipo: Cadena 2. Nombre: versions Descripción: Versiones del repositorio de datos de origen. ¿Es necesario?: necesario Tipo: Cadena 3. Nombre: vendor Descripción: Proveedor del repositorio de datos de origen. ¿Es necesario?: necesario Tipo: Cadena
targetcis (integration)	Elemento contenedor para todas las asignaciones de CIT.	
source_ci_type_tree (integration > targetcis)	Define un CIT de origen y todos los tipos de CI que heredan del mismo.	<ol style="list-style-type: none"> 1. Nombre: name Descripción: Nombre del CIT de origen. ¿Es necesario?: necesario Tipo: Cadena 2. Nombre: mode Descripción: El tipo de actualización requerido para el tipo de CI actual. ¿Es necesario?: necesario Tipo: Una de las cadenas siguientes: <ol style="list-style-type: none"> a. insert: Utilízcela solo si el CI todavía no existe. b. update: Utilízcela solo si se sabe que el CI existe. c. update_else_insert: Si el CI existe, actualízcelo; en caso contrario, cree un nuevo CI. d. ignore: No haga nada con este tipo de CI.

Nombre y ruta del elemento	Descripción	Atributos
source_ci_type (integration > targetcis)	Define un CIT de origen sin los tipos de CI que heredan del mismo.	<ol style="list-style-type: none"> <li data-bbox="868 325 1383 457"> 1. Nombre: name Descripción: Nombre del CIT de origen. ¿Es necesario?: necesario Tipo: Cadena <li data-bbox="868 489 1383 961"> 2. Nombre: mode Descripción: El tipo de actualización requerido para el tipo de CI actual. ¿Es necesario?: necesario Tipo: Una de las cadenas siguientes: <ol style="list-style-type: none"> <li data-bbox="909 657 1383 720">a. insert: Utilízcela solo si el CI todavía no existe. <li data-bbox="909 730 1383 793">b. update: Utilízcela solo si se sabe que el CI existe. <li data-bbox="909 804 1383 888">c. update_else_insert: Si el CI existe, actualízcelo; en caso contrario, cree un nuevo CI. <li data-bbox="909 898 1383 961">d. ignore: No haga nada con este tipo de CI.
target_ci_type (integration > targetcis > source_ci_type -O BIEN- integration > targetcis > source_ci_type_ tree)	Define un CIT de destino.	<ol style="list-style-type: none"> <li data-bbox="868 987 1383 1161"> 1. Nombre: name Descripción: Nombre del tipo de CI de destino. ¿Es necesario?: necesario Tipo: Cadena <li data-bbox="868 1192 1383 1392"> 2. Nombre: schema Descripción: El nombre del esquema que se usará para almacenar este tipo de CI en el destino. ¿Es necesario?: no necesario Tipo: Cadena <li data-bbox="868 1423 1383 1591"> 3. Nombre: namespace Descripción: Indica el espacio de nombres de este tipo de CI en el destino. ¿Es necesario?: no necesario Tipo: Cadena

Nombre y ruta del elemento	Descripción	Atributos
<p>targetprimarykey (integration > targetcis > source_ci_type)</p> <p>-O BIEN-</p> <p>(integration > targetcis > source_ci_type_tree)</p> <p>-O BIEN-</p> <p>(integration > targetrelations > link)</p> <p>-O BIEN-</p> <p>(integration > targetrelations > source_link_type_tree)</p>	<p>Identifica los atributos de clave principal del CIT de destino.</p>	
<p>pkey (integration > targetcis> source_ci_type > targetprimarykey)</p> <p>-O BIEN-</p> <p>integration > targetcis > source_ci_type_tree > targetprimarykey)</p> <p>-O BIEN-</p> <p>(integration > targetrelations > link > targetprimarykey)</p> <p>-O BIEN-</p> <p>integration > targetrelations > source_link_type_tree > targetprimarykey)</p>	<p>Identifica un atributo de clave principal.</p> <p>Necesario solo si el modo es update o insert_else_update o insert_else_update.</p>	

Nombre y ruta del elemento	Descripción	Atributos
<pre>target_attribute (integration > targetcis> source_ci_ type -O BIEN- integration > targetcis > source_ci_type_tree -O BIEN- integration > targetrelations > link -O BIEN- integration > targetrelations > source_link_type_tree)</pre>	<p>Define el atributo del CIT de destino.</p>	<ol style="list-style-type: none"> Nombre: name Descripción: Nombre del atributo del CIT de destino. ¿Es necesario?: necesario Tipo: Cadena Nombre: datatype Descripción: Tipo de datos del atributo del CIT de destino. ¿Es necesario?: necesario Tipo: Cadena Nombre: length Descripción: Para tipos de datos de cadena/carácter, tamaño de entero del atributo de destino. ¿Es necesario?: no necesario Tipo: Entero Nombre: option Descripción: La función de conversión que se va a aplicar al valor. Se requiere: False Tipo: Una de las cadenas siguientes: <ol style="list-style-type: none"> uppercase: convierte a mayúsculas lowercase: convierte a minúsculas <p>Si este atributo está vacío, no se aplicará ninguna función de conversión.</p>

Nombre y ruta del elemento	Descripción	Atributos
<p>map</p> <p>(integration > targetcis > source_ci_ type > target_attribute</p> <p>-O BIEN-</p> <p>integration > targetcis > source_ci_type_tree > target_attribute)</p> <p>-O BIEN-</p> <p>(integration > targetrelations > link > target_attribute</p> <p>-O BIEN-</p> <p>integration > targetrelations > source_link_type_tree > target_attribute)</p>	<p>Determina cómo obtener el valor del atributo del CIT de origen</p>	<ol style="list-style-type: none"> <p>Nombre: type Descripción: El tipo de asignación entre los valores de origen y destino. Se requiere: necesario Tipo: Una de las cadenas siguientes: a. direct: especifica una asignación 1-a-1 del valor del atributo de origen al valor del atributo de destino. b. compoundstring: los elementos secundarios se unen en una única cadena y se establece el valor del atributo de destino. c. childattr: los elementos secundarios son uno o más atributos de los CIT secundarios. Los CIT secundarios se definen como los de la relación composition o containment. d. constant: cadena estática.</p> <p>Nombre: value Descripción: Cadena constante para type=constant Se requiere: Solo se requiere cuando type=constant Tipo: Cadena</p> <p>Nombre: attr Descripción: Nombre de atributo de origen para type=direct Se requiere: Solo se requiere cuando type=direct Tipo: Cadena</p>

Nombre y ruta del elemento	Descripción	Atributos
<p>aggregation</p> <p>(integration > targetcis > source_ci_type > target_attribute > map</p> <p>-O BIEN-</p> <p>integration > targetcis > source_ci_type_tree > target_attribute > map</p> <p>-O BIEN-</p> <p>(integration > targetrelations > link > target_attribute > map</p> <p>-O BIEN-</p> <p>integration > targetrelations > source_link_type_tree > target_attribute > map)</p> <p>Solo es válido cuando el tipo de asignación es childattr)</p>	<p>Determina cómo los valores de atributo de los CI secundarios de los CI de origen se combinan en un valor único que se asignará al atributo del CI de destino. Opcional.</p>	<p>Nombre: type Descripción: El tipo de la función de agregación ¿Es necesario?: necesario Tipo: Una de las cadenas siguientes:</p> <ul style="list-style-type: none"> • csv: concatena todos los valores incluidos en una lista (numérica o cadena/carácter) separada por comas. • count: devuelve un recuento numérico de todos los valores incluidos. • sum: devuelve la suma de todos los valores numéricos incluidos. • average: devuelve un promedio numérico de todos los valores incluidos. • min: devuelve el valor numérico/carácter más bajo incluido. • max: devuelve el valor numérico/carácter más alto incluido.

Nombre y ruta del elemento	Descripción	Atributos
<p>source_child_ci_type (integration > targetcis> source_ci_type > target_attribute > map</p> <p>-O BIEN-</p> <p>integration > targetcis > source_ci_type_tree > target_attribute > map</p> <p>-O BIEN-</p> <p>(integration > targetrelations > link > target_attribute > map</p> <p>-O BIEN-</p> <p>integration > targetrelations > source_link_type_tree > target_attribute > map)</p> <p>Solo es válido cuando el tipo de asignación es childattr.</p>	<p>Determina de qué CI conectado se toma el atributo secundario.</p>	<ol style="list-style-type: none"> 1. Nombre: name Descripción: El tipo de CI secundario Se requiere: necesario Tipo: Cadena 2. Nombre. source_attribute Descripción: El atributo del CI secundario que se asigna. Se requiere: Se requiere solo si el tipo de agregación childAttr (que está en la misma ruta) no es =count. Tipo: Cadena

Nombre y ruta del elemento	Descripción	Atributos
<p>validation (integration > targetcis > source_ci_type > target_attribute > map -O BIEN- integration > targetcis > source_ci_type_tree > target_attribute > map -O BIEN- (integration > targetrelations > link > target_attribute > map -O BIEN- integration > targetrelations > source_link_type_tree > target_attribute > map)</p> <p>Solo es válido cuando el tipo de asignación es childattr.</p>	<p>Permite la exclusión del filtrado de los CI secundarios del CI de origen basándose en valores de atributos. Se utiliza con el subelemento de agregación para conseguir granularidad de los atributos secundarios que se asignan al valor de atributo de los CIT de destino. Opcional.</p>	<ol style="list-style-type: none"> 1. Nombre: minlength Descripción: Excluye cadenas más cortas que el valor dado. ¿Es necesario?: no necesario Tipo: Entero 2. Nombre: maxlength Descripción: Excluye cadenas más largas que el valor dado. ¿Es necesario?: no necesario Tipo: Entero 3. Nombre: minvalue Descripción: Excluye números más pequeños que el valor especificado. ¿Es necesario?: no necesario Tipo: Numérico 4. Nombre: maxvalue Descripción. Excluye números más grandes que el valor especificado. ¿Es necesario?: no necesario Tipo: Numérico
<p>targetrelations (integration)</p>	<p>Elemento contenedor para todas las asignaciones de relaciones. Opcional.</p>	

Nombre y ruta del elemento	Descripción	Atributos
source_link_type_tree (integration > targetrelations)	Asigna a una relación de destino un tipo de relación de origen sin los tipos que heredan del mismo. Solo obligatorio si targetrelation está presente.	<ol style="list-style-type: none"> 1. Nombre: name Descripción: Nombre de la relación de origen. ¿Es necesario?: necesario Tipo: Cadena 2. Nombre: target_link_type Descripción: Nombre de la relación de destino. ¿Es necesario?: necesario Tipo: Cadena 3. Nombre: nameSpace Descripción: El espacio de nombres para el vínculo se creará en el destino. ¿Es necesario?: no necesario Tipo: Cadena 4. Nombre: mode Descripción: El tipo de actualización requerido para el vínculo actual. ¿Es necesario?: necesario Tipo: Una de las cadenas siguientes: <ul style="list-style-type: none"> ■ insert: utilízela solo si el CI todavía no existe. ■ update: utilízela solo si el CI se sabe que existe. ■ update_else_insert: si el CI existe, actualízelo; en caso contrario, cree un nuevo CI. ■ ignore: no hacer nada con este tipo de CI. 5. Nombre: source_ci_type_end1 Descripción: Tipo de CI End1 de la relación de origen. ¿Es necesario?: necesario Tipo: Cadena 6. Nombre: source_ci_type_end2 Descripción: Tipo de CI End2 de la relación de origen. ¿Es necesario?: necesario

Nombre y ruta del elemento	Descripción	Atributos
		Tipo: Cadena

Nombre y ruta del elemento	Descripción	Atributos
link (integration > targetrelations)	Asigna una relación de origen a una relación de destino. Solo obligatorio si targetrelation está presente.	<ol style="list-style-type: none"> 1. Nombre: source_link_type Descripción: Nombre de la relación de origen. ¿Es necesario?: necesario Tipo: Cadena 2. Nombre: target_link_type Descripción: Nombre de la relación de destino. ¿Es necesario?: necesario Tipo: Cadena 3. Nombre: nameSpace Descripción: El espacio de nombres para el vínculo se creará en el destino. ¿Es necesario?: no necesario Tipo: Cadena 4. Nombre: mode Descripción: El tipo de actualización requerido para el vínculo actual. ¿Es necesario?: necesario Tipo: Una de las cadenas siguientes: <ul style="list-style-type: none"> ■ insert: utilízcela solo si el CI todavía no existe. ■ update: utilízcela solo si el CI se sabe que existe. ■ update_else_insert: si el CI existe, actualízcelo; en caso contrario, cree un nuevo CI. ■ ignore: no hacer nada con este tipo de CI. 5. Nombre: source_ci_type_end1 Descripción: Tipo de CI End1 de la relación de origen. ¿Es necesario?: necesario Tipo: Cadena 6. Nombre: source_ci_type_end2 Descripción: Tipo de CI End2 de la relación de origen. ¿Es necesario?: necesario

Nombre y ruta del elemento	Descripción	Atributos
		Tipo: Cadena
target_ci_type_end1 (integration > targetrelations > link -O BIEN- integration > targetrelations > source_link_type_tree)	Tipo de CI End1 de la relación de destino.	1. Nombre: name Descripción: Nombre del tipo de CI End1 de la relación de destino. ¿Es necesario?: necesario Tipo: Cadena 2. Nombre: superclass Descripción: Nombre de la super clase del tipo de CI End1. ¿Es necesario?: no necesario Tipo: Cadena
target_ci_type_end2 (integration > targetrelations > link -O BIEN- integration > targetrelations > source_link_type_tree)	Tipo de CI End2 de la relación de destino.	1. Nombre: name Descripción: Nombre del tipo de CI End2 de la relación de destino. ¿Es necesario?: necesario Tipo: Cadena 2. Nombre: superclass Descripción: Nombre de la super clase del tipo de CI End2. ¿Es necesario?: no necesario Tipo: Cadena

Esquema de resultados de asignación

Nombre y ruta del elemento	Descripción	Atributos
root	La raíz del documento de resultados.	
data (root)	La raíz de los datos propiamente dichos.	
objects (root > data)	El elemento raíz de los objetos que se van a actualizar.	

Nombre y ruta del elemento	Descripción	Atributos
Objeto (root > data > objects)	Describe la operación de actualización para un objeto y todos sus atributos.	<ol style="list-style-type: none"> <li data-bbox="868 325 1383 457">1. Nombre: name Descripción: Nombre del tipo de CI ¿Es necesario?: Requerido Tipo: Cadena <li data-bbox="868 493 1383 961">2. Nombre: mode Descripción: El tipo de actualización requerido para el tipo de CI actual. ¿Es necesario?: Requerido Tipo: Una de las cadenas siguientes: <ol style="list-style-type: none"> <li data-bbox="909 661 1383 724">a. insert: utilízela solo si el CI todavía no existe. <li data-bbox="909 730 1383 793">b. update: utilízela solo si se sabe que el CI existe. <li data-bbox="909 800 1383 898">c. update_else_insert: si el CI existe, actualízelo; en caso contrario, cree un nuevo CI. <li data-bbox="909 905 1383 961">d. ignore: no haga nada con este tipo de CI. <li data-bbox="868 997 1383 1333">3. Nombre: operation Descripción: La operación que se va a realizar este CI. ¿Es necesario?: Requerido Tipo: Una de las cadenas siguientes: <ol style="list-style-type: none"> <li data-bbox="909 1165 1383 1192">a. add: el CI debe agregarse <li data-bbox="909 1199 1383 1226">b. update: el CI debe actualizarse <li data-bbox="909 1232 1383 1260">c. delete: el CI debe eliminarse Si no se establece ningún valor, se usa el valor predeterminado add. <li data-bbox="868 1369 1383 1528">4. Nombre: mamId Descripción: El Id. del objeto en el CMDB de origen. ¿Es necesario?: Requerido Tipo: Cadena

Nombre y ruta del elemento	Descripción	Atributos
field (root > data > objects> Object -O BIEN- root > data > links > link)	Describe el valor de un único campo para un objeto. El texto del campo es la nuevo valor del campo y, si si el campo contiene un vínculo, el valor es el Id. de uno de los extremos. Cada Id. de extremo aparece como un objeto (en <objects>).	<ol style="list-style-type: none"> 1. Nombre: name Descripción: Nombre del campo. ¿Es necesario?: Requerido Tipo: Cadena 2. Nombre: key Descripción: Especifica si este campo es una clave para el objeto. ¿Es necesario?: Requerido Tipo: Booleano 3. Nombre: datatype Descripción: El tipo del campo. ¿Es necesario?: Requerido Tipo: Cadena 4. Nombre: length Descripción: Para tipos de datos de cadena/carácter, es el tamaño de entero del atributo de destino. ¿Es necesario?: no necesario Tipo: Entero

Nombre y ruta del elemento	Descripción	Atributos
links (root > data)	El elemento raíz de los vínculos que se van a actualizar.	<ol style="list-style-type: none"> 1. Nombre: targetRelationshipClass Descripción: El nombre de la relación (vínculo) del sistema destino. ¿Es necesario?: Requerido Tipo: Cadena 2. Nombre: targetParent Descripción: El tipo del primer extremo del vínculo (elemento de primer nivel). ¿Es necesario?: Requerido Tipo: Cadena 3. Nombre: targetChild Descripción: El tipo del segundo extremo del vínculo (elemento de segundo nivel). ¿Es necesario?: Requerido Tipo: Cadena 4. Nombre: mode Descripción: El tipo de actualización requerido para el tipo de CI actual. ¿Es necesario?: Requerido Tipo: Una de las cadenas siguientes: <ol style="list-style-type: none"> a. insert: utilícela solo si el CI todavía no existe. b. update: utilícela solo si se sabe que el CI existe. c. update_else_insert: si el CI existe, actualízcelo; en caso contrario, cree un nuevo CI. d. ignore: no haga nada con este tipo de CI. 5. Nombre: operation Descripción: La operación que se va a realizar este CI. ¿Es necesario?: Requerido Tipo: Una de las cadenas siguientes: <ol style="list-style-type: none"> a. add: el CI debe agregarse b. update: el CI debe actualizarse c. delete: el CI debe eliminarse Si no se establece ningún valor, se usa el valor predeterminado add. 6. Nombre: mamId

Nombre y ruta del elemento	Descripción	Atributos
		Descripción: El Id. del objeto en el CMDB de origen. ¿Es necesario?: Requerido Tipo: Cadena

Personalización

Esta sección explica algunos de los procedimientos básicos para los tipos comunes de personalización para los adaptadores de inserción.

Adición de un atributo

1. Asegúrese de que el atributo está incluido en el resultado de TQL.
2. Agregue la asignación de atributo al archivo de asignaciones en la sección de asignación de CI correcta.
3. Asegúrese de que el receptor de datos está preparado para recibir al atributo adicional en los datos.

Eliminación de un atributo

Para eliminar un atributo, elimine el atributo del archivo de asignación. También debe eliminar el atributo del TQL si ya no se usa en el resultado o como un nodo condicional.

Adición de un tipo de CI

1. Agregue el tipo de CI al TQL.
2. Asegúrese de que el tipo de CI y sus datos de atributo aparecen en el resultado de TQL (use calcular y vista previa).
3. Agregue la asignación del tipo de CI en el archivo de asignaciones. Copie las asignaciones de otro tipo de CI para crear rápidamente un nuevo tipo de CI.
4. Modifique el nombre y las asignaciones de atributos del XML copiado para que se correspondan con el nuevo tipo de CI y sus atributos. Consulte "[Referencia de archivos de asignación](#)" en la página 225 para ver los tipos de asignaciones disponibles.

Eliminación de un tipo de CI

1. Elimine el tipo de CI del TQL
2. Elimine la sección de asignación de dicho tipo de CI en el archivo de asignaciones.

Adición de enlaces

1. Asegúrese de que los dos CI finales están presentes en los datos.
2. Asegúrese de que el vínculo que necesita agregar es realmente un vínculo válido (compruébelo en el administrador de tipos de CI).
3. Agregue los elementos del vínculo en la sección de relaciones del archivo xml de asignaciones.

Eliminación de vínculos

1. Elimine la sección de vínculos del vínculo que desea eliminar del archivo de asignaciones.
2. Si es posible, elimine el vínculo del TQL (a menos que afecte a la eficiencia o la función del TQL).

Capítulo 7: Desarrollo de adaptadores de inserción genéricos mejorados

Este capítulo incluye:

Información general	245
El archivo de asignación	245
Groovy Traveler	248
Escritura de secuencias de comandos Groovy	251
Implementar la interfaz de PushConnector	252
Construcción de un paquete de adaptadores	253
Esquema del archivo de asignación	254

Información general

Un adaptador de inserción mejorado es una estructura de datos que representa el resultado de la consulta TQL. Cada adaptador creado sobre el adaptador de inserción mejorado gestionará esta estructura de datos y lo insertará en su destino requerido.

La estructura de datos se denomina **ResultTreeNode (RTN)**. El RTN se crea según el archivo de asignación del adaptador y los resultados de la consulta TQL. Las consultas utilizadas para el adaptador de inserción mejorado deben estar basadas en la raíz, es decir, la consulta debe contener un nodo de consulta con el nombre de elemento **root**, o uno o más elementos de relación que empiezan por el prefijo **root**. Este CI o relación sirve como elemento raíz de la consulta. Para obtener más información, consulte "Inserción de datos" en *HP Universal CMDB – Guía de Administración de Data Flow*.

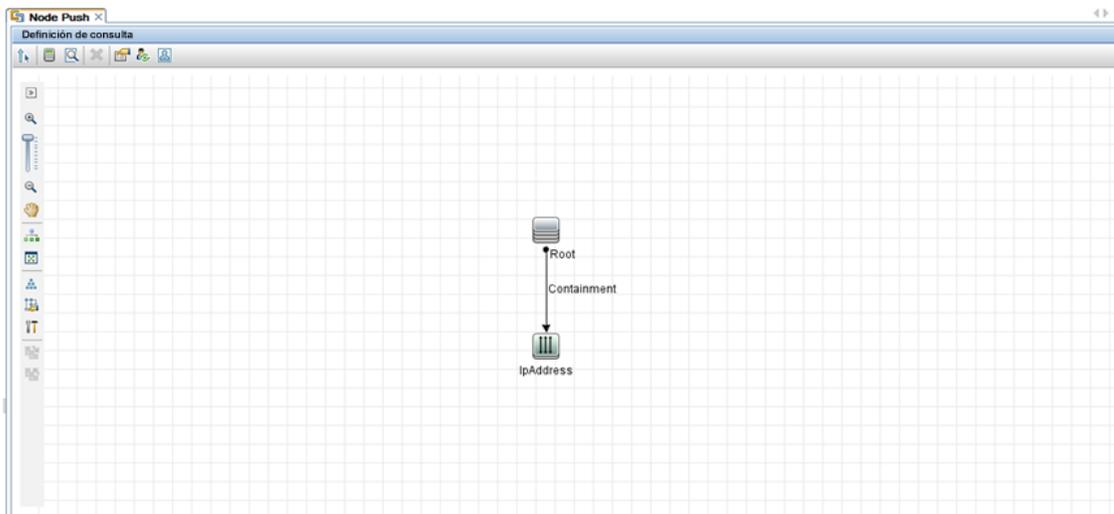
Hay dos pasos básicos implicados en el desarrollo de adaptadores de inserción mejorados:

1. Implementación de la interfaz de PushConnector: esta interfaz recibe los datos que se deben agregar, actualizar y eliminar como una Lista de RTN y para realizar la inserción en el destino.
2. Creación del archivo de asignación: el archivo de asignación determina la creación de la estructura del RTN, al asignar los CI y los atributos del resultado del TQL.

El archivo de asignación

El ejemplo siguiente muestra cómo crear el archivo de asignación.

En este ejemplo, simularemos una inserción de nodo y dirección IP. Crearemos una consulta TQL denominada: **Node Push**, de la manera siguiente:

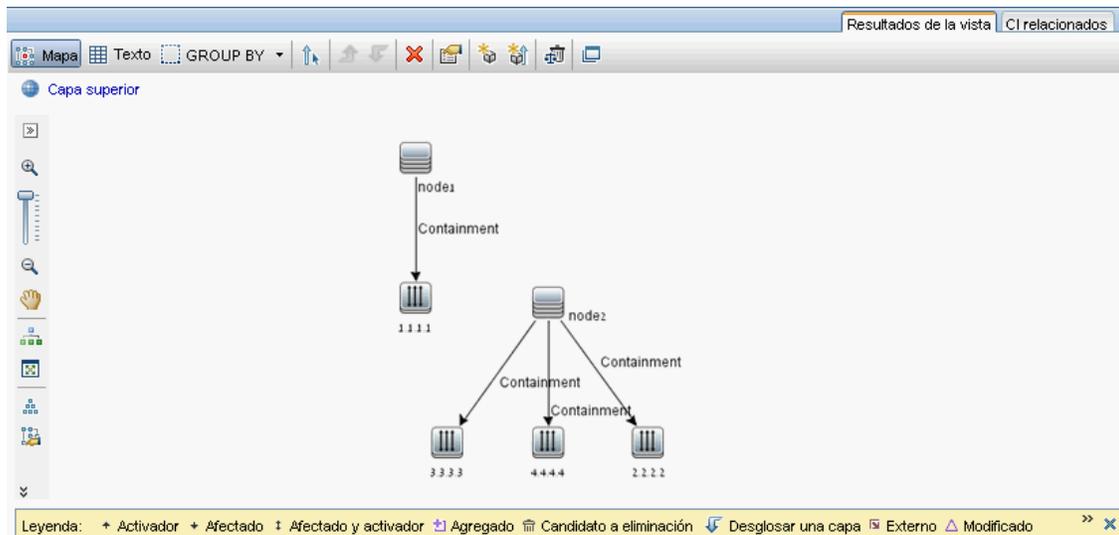


En el archivo de asignación creamos dos tipos de CI de destino: **Computer** e **IP**. Computer tiene una variable y dos atributos. IP tiene un atributo.

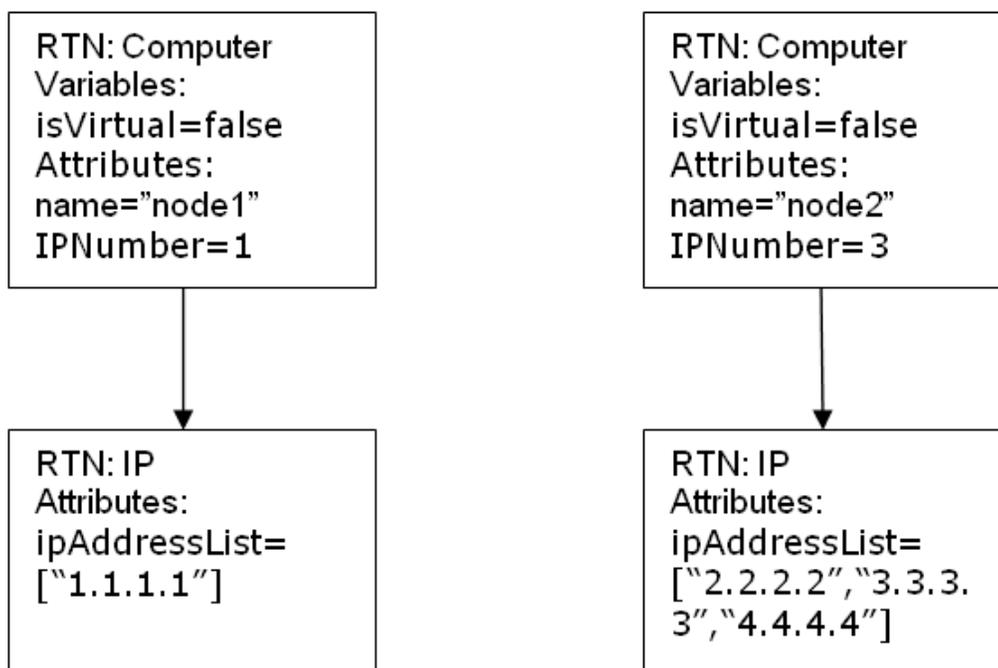
El siguiente es el archivo XML de asignación:

```
<integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://schemas.hp.com/ucmdb/1/pushAdap
ter">
  <info>
    <source name="UCMDB" versions="10.0" vendor="HP" />
    <target name="PushProduct" versions="9.3" vendor="HP" />
  </info>
  <import>
    <!--Imports the Groovy script file. -->
    <scriptFile path="mappings.scripts.PushFunctions" />
  </import>
  <targetcis>
    <source_instance_type query-name="Node Push" root-element-name="Root">
      <target_ci_type name="Computer" is-
        valid="(Root['root_iscandidatefordeletion']==null) ? true :
        !Root['root_iscandidatefordeletion']">
        <variable name="isVirtual" datatype="BOOLEAN"
          value="PushFunctions.isVirtual(Root['root_class'])" />
        <target_mapping name="name" datatype="String" value="Root['name']" />
        <target_mapping name="ipNumber" datatype="INTEGER"
          value="Root.IpAddress.size()" />
        <target_mapping name="description" datatype="STRING" value="PushFunctions
          .getDescription(isVirtual)" />
        <target_ci_type name="IP">
          <target_mapping name="IpAddressList" datatype="STRING_LIST"
            value="Root.IpAddress*.getAt('name')" />
        </target_ci_type>
      </target_ci_type>
    </targetcis>
  </integration>
```

Los resultados de la consulta aparecen de la manera siguiente:



Esta es la lista de RTN creada según este archivo de asignación:



Cada instancia raíz se asigna por separado mediante el archivo de asignación. Por lo tanto, en este ejemplo, PushConnector recibe una lista de dos raíces RTN.

Nota: El adaptador de inserción anterior tenía la capacidad de crear una asignación general para un tipo de CI. La nueva asignación de adaptador de inserción se realiza por consulta TQL. Mientras se ejecuta un trabajo de inserción que utiliza una consulta denominada *x*, el adaptador busca el archivo de asignación relevante (el que tiene el atributo: query-name=*x*).

Puede calcular los valores en el archivo de asignación mediante el lenguaje de secuencia de comandos groovy. Para obtener más información, consulte "[Groovy Traveler](#)" [abajo](#).

Groovy Traveler

Acceda a los resultados de la consulta TQL de la manera siguiente:

- **Root[*attr*]** devuelve el atributo ***attr*** del elemento Root.
- **Root.Query_Element_Name** devuelve una lista de instancias de CI indicadas en Query_Element_Name del TQL y están vinculadas al CI raíz actual.
- **Root.Query_Element_Name[2][*attr*]** devuelve el atributo ***attr*** del tercer Query_Element_Name que está vinculado al CI raíz actual.
- **Root.Query_Element_Name*.getAt(*attr*)** devuelve una lista de los atributos ***attr*** de las instancias de CI indicadas como Query_Element_Name en el TQL y están vinculadas al CI raíz actual.

Hay atributos adicionales a los que Groovy Traveler puede acceder:

- **cmdb_id** – devuelve el Id. de UCMDB del CI o la relación como una cadena.
- **external_cmdb_id** – devuelve el Id. externo de UCMDB del CI o la relación como una cadena.
- **Element_type** – devuelve el tipo de elemento del CI o la relación como una cadena.

La etiqueta import:

```
<import>
<scriptFile path="mappings.scripts.PushFunctions"/>
</import>
```

Significa que se declara una importación para todas las secuencias de comandos groovy en el archivo de asignación. En este ejemplo, **PushFunctions** es un archivo de secuencia de comandos groovy que contiene algunas funciones estáticas y a las que podemos acceder durante la asignación (por ejemplo, value="PushFunctions.foo()")

source_instance_type

La asignación se realiza por TQL y el valor de query-name es el TQL relacionado de la asignación actual. '*' significa que este archivo de asignación está asociado con todas las consultas de TQL que empiezan por el prefijo: **Node Push**.

```
<source_instance_type query-name="Node Push*" root-element-name="Root">
```

La etiqueta source_instance_type designa el elemento Root que estamos asignando.

root-element-name debe ser exactamente el mismo que el nombre del elemento raíz en el TQL.

target_ci_type

Esta etiqueta se utiliza para la creación del RTN.

El atributo `name` representa el nombre de `target_ci_type`: `name=Computer`

El atributo **`is-valid`** es un valor booleano que se calcula durante la asignación y determina si el `target_ci` actual es válido. Los `target_ci_types` no válidos no se agregan al RTN. En este ejemplo, no queremos crear una instancia de `target_ci_type` para la que el atributo **`root_iscandidatefordeletion`** de UC MDB sea `true`.

El `target_ci_type` puede tener variables que se calculan durante la asignación:

```
<variable name="vSerialNo" datatype="STRING" value="Root['serial_number']"/>
```

La variable **`vSerialNo`** obtiene el valor de **`serial_number`** de la raíz actual.

La etiqueta **`target_mapping`** crea el atributo del RTN. El resultado de la ejecución de la secuencia de comandos groovy en el campo **`value`** se asigna al atributo RTN.

```
<target_mapping name="SerialNo" datatype="STRING" value="vSerialNo"/>
```

`SerialNo` asigna el valor de la variable **`vSerialNo`**.

Es posible definir `target_ci_type` como subordinado de otro `target_ci_type` de la forma siguiente:

```
<target_ci_type name="Portfolio">
  <variable name="vSerialNo" datatype="STRING" value="Root['global_id']"/>
  <target_mapping name="CMDBId" datatype="STRING" value="globalId"/>
  <target_ci_type name="Asset">
    <target_mapping name="SerialNo" datatype="STRING" value="vSerialNo"/>
  </target_ci_type>
</target_ci_type>
```

El RTN **`Portfolio`** tendrá el RTN subordinado denominado **`Asset`**.

for-each-source-ci

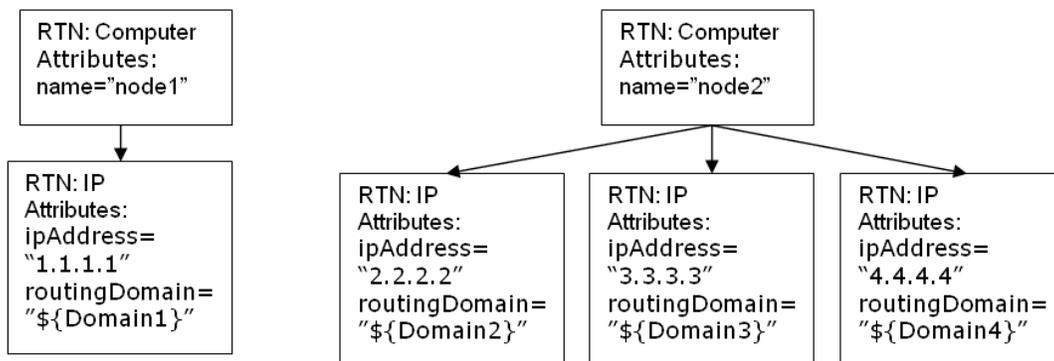
Esta etiqueta genera una lista de los CI específicos de la instancia raíz. Tiene los campos siguientes:

- `source-cis=""` – la lista de los CI para los que se crea un CI de destino. Groovy Traveler define esta lista en el campo **`Root.IpAddress`**.
- `count-index=""` – variable que contiene el índice del CI en la iteración actual del bucle.
- `var-name=""` – nombre del CI en la iteración actual del bucle.

Vamos a modificar nuestro archivo de asignación de ejemplo:

```
<target_ci_type name="Computer">  
  <target_mapping name="name" datatype="String" value="Root['name']" />  
  <for-each-source-ci count-index="i" var-name="currIP" source-cis="Root.IpAddress">  
    <target_ci_type name="IP">  
      <target_mapping name="ipAddress" datatype="STRING"  
        value="Root.IpAddress[i]['name']"/>  
      <target_mapping name="routingDomain" datatype="STRING"  
        value="currIP['routing_domain']"/>  
    </target_ci_type>  
  </for-each-source-ci>  
</target_ci_type>
```

La lista de RTN que construirá según este archivo de asignación tendrá un aspecto como el siguiente:



dynamic_mapping

Esta etiqueta agrega la capacidad de crear una asignación de datos del almacén de datos de destino durante la creación de la estructura de RTN.

Ejemplo: Suponga que el destino es una base de datos con una tabla denominada **Computer** que tiene una columna **id** y una columna **name** que está correlacionada con **Node.name** en UCMDB. Ambas columnas son únicas. Además, la base de datos tiene una tabla denominada **IP** que tiene una clave referenciada con **parentID** en la tabla **Computer**. 'dynamic_mapping' puede crear una asignación que almacene el nombre y el Id. como <name,id>. Según este mapa, el adaptador puede emparejar los Id. con equipos y puede insertar el valor correcto en el atributo **parentID** de la tabla de IP. Puede utilizar este mapa para asignar un valor al atributo **parentID** mientras se crea el RTN.

map_property determina la asignación. **dynamic_mapping** se ejecuta una sola vez por cada fragmento.

```
<dynamic_mapping name="IdByName " keys-unique="true">
```

El atributo **name** representa el nombre del mapa. El atributo **keys-unique** indica si las claves son únicas (cada clave se asigna a un valor o a un conjunto de valores).

El nombre del mapa en este ejemplo es **IdByName** y tiene unas claves exclusivas. Para acceder al mapa en la secuencia de comandos, ejecute el comando siguiente:

```
DynamicMapHolder.getMap('IdByName')
```

Devuelve una referencia a ese mapa.

La etiqueta **map_property** crea la propiedad en la que se basa la asignación.

Ejemplo:

```
<map_property property-name="SQLQuery" datatype="STRING"  
property-value="SELECT name, id FROM Computer"/>
```

En este ejemplo, el nombre de la propiedad es **SQLQuery** y su valor es una sentencia SQL que crea el mapa. La implementación de los métodos **retrieveUniqueMapping** y **retrieveNonUniqueMapping** para que la interfaz de PushConnector determine el contenido real del mapa devuelto.

Variables globales

Las variables globales siguientes son accesibles para la secuencia de comandos groovy del archivo de asignación:

- **Topology** – Tipo: Topología. Una instancia de la topología del fragmento actual.
- **QueryDefinition** - Tipo: QueryDefinition. Una instancia de la definición de consulta del TQL actual.
- **OutputCI** – Tipo: ResultTreeNode. El RTN del elemento raíz de la asignación de árbol actual.
- **ClassModel** – Tipo: ClassModel. Una instancia del modelo de clase.
- **CustomerInformation** – Tipo: CustomerInformation. Información sobre el cliente que ejecuta el trabajo.
- **Logger** – Tipo: DataAdapterLogger. Este registrador está disponible en el adaptador para escribir registros en el marco de registros de UCMDDB.

Escritura de secuencias de comandos Groovy

En esta sección crearemos el archivo **PushFunctions.groovy**. Este archivo contendrá funciones estáticas que se utilicen durante la asignación de la instancia raíz.

```
package mappings.scripts  
  
public class PushFunctions {  
  
    public static boolean isVirtual(def nodeRole){  
        return isListContainsOne(def list, "MY_VM", "MY_SIMULATOR");  
    }  
}
```

```
    }

    public static String getDescription(boolean isVirtual){
        if(isVirtual){
            return "Esta es una máquina virtual";
        }
        else{
            return "Esta es una máquina física";
        }
    }

    private static boolean isListContainsOne(def list, ...stringList){
        //devuelve true si la lista contiene uno de los valores.
    }
}
```

Implementar la interfaz de PushConnector

La implementación debe admitir los pasos básicos siguientes:

```
public class PushExampleAdapter implements PushAdapterConnector
{

    public UpdateResult pushTreeNodes(PushConnectorInput input) throws DataAccess
    sException{

        // 1. crear una instancia de UpdateResult: UpdateResult se utiliza para devo
        lver asignaciones entre los Id. enviados a los Id. reales que introdujeron el
        almacén de datos.
        // También tiene un estado de actualización que permite pasar el estado de l
        os datos que realmente se insertaron, informes de estado detallados sobre los
        Id. erróneos y las acciones realizadas en los Id. correctos.
        // 2. manejar los datos:
        // a. manejar los datos que se agregarán. Pueden recuperarse mediante: input
        .getResultTreeNodes.getDataToAdd();
        // b. manejar datos que actualizar.
        // c. manejar datos que eliminar.
        // 3. Devolver el resultado de la actualización.
    }

    public void start(PushDataAdapterEnvironment env) throws DataAccessException{
        // se llama a este método al crear el punto de integración
    }
}
```

```
o cuando se recarga el adaptador
    //(es decir, después de cambiar uno de los archivos de asignación
    // y pulsar 'guardar').

}

public void testConnection(PushDataAdapterEnvironment env) throws DataAccess
Exception {
    // se llama a este método al pulsar el botón 'probar
conexión' durante
    //la creación del punto de integración.
    // Por ejemplo, si insertamos datos en RDBMS, este método
puede crear una conexión
    //a la base de datos y ejecutará una instrucción SQL ficticia.
    // Si da error, escribe un mensaje de error en el registro
y genera una excepción.
    }

Map<Object, Object> retrieveUniqueMapping(MappingQuery mappingQuery){
//Este método creará el mapa según un mappingQuery dado. Se llamará en la
// fase de asignación de la ejecución del adaptador, antes del método 'Updat
eResult pushTreeNodees'.
// Se llama a este método cuando el atributo 'keys-unique' de la etiqueta 'd
ynamic_mapping' es true.
}

Map<Object, Set<Object>> retrieveNonUniqueMapping(MappingQuery mappingQuery){
// Se llama a este método cuando el atributo 'keys-unique' de la etiqueta '
dynamic_mapping' es false.
// En este caso, una clave se puede asignar a varios valores.
}
}
```

Construcción de un paquete de adaptadores

Asegúrese de que el paquete del adaptador contiene las siguientes carpetas:

- **adapterCode**. En esta carpeta, cree una carpeta llamada **PushExampleAdapter**, que contendrá el archivo .jar que creamos a partir de PushExampleAdapter.java. También contendrá una carpeta llamada **mappings**, donde podrá colocar el archivo de asignación creado anteriormente, **computerIPMapping.xml**. También debe contener otra carpeta llamada **scripts** que contiene el archivo **PushFunctions.groovy**.

- **discoveryConfigFiles.** Para contener archivos de configuración como, por ejemplo, los códigos de error utilizados al notificar un error mediante UpdateResult. En este ejemplo, la carpeta está vacía.
- **discoveryPatterns.** Para contener **push_example_adapter.xml**.
- **tql.** Para contener la consulta TQL creada para el ejemplo. Esta carpeta es opcional, pero cuando se despliega el paquete, el TQL se crea automáticamente.

Esquema del archivo de asignación

Nombre y ruta del elemento	Descripción	Atributos
Integración	Define el contenido de la asignación del archivo. Debe ser el bloque más exterior del archivo excepto para la línea de comienzo y cualquier comentario.	
info (integration)	Define la información sobre los repositorios Repositorios que se están integrando.	

Nombre y ruta del elemento	Descripción	Atributos
source (info)	El producto de origen	<p>Nombre: name</p> <p>Descripción: el nombre del producto</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: vendor</p> <p>Descripción: el proveedor del producto</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: versions</p> <p>Descripción: la versión del producto</p> <p>¿Es necesario?: necesario</p> <p>Tipo: decimal</p>

Nombre y ruta del elemento	Descripción	Atributos
target (info)	El producto de destino	<p>Nombre: name</p> <p>Descripción: el nombre del producto</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: vendor</p> <p>Descripción: el proveedor del producto</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: versions</p> <p>Descripción: la versión del producto</p> <p>¿Es necesario?: necesario</p> <p>Tipo: decimal</p>
Importar (integration)	Un elemento contenedor para archivos de secuencia de comandos importados.	
scriptFile (integration>import)	Define el archivo de secuencia de comandos groovy que se van a importar.	<p>Nombre: ruta de acceso</p> <p>Descripción: la ruta del archivo de secuencia de comandos</p> <p>¿Es necesario?: necesario</p> <p>Tipo: cadena</p>
Targetcis (integration)	Elemento contenedor de los tipos de CI.	

Nombre y ruta del elemento	Descripción	Atributos
Source_instance_type (integration > targetcis)	Define el tipo de instancia de CI de origen. Debe ser el elemento raíz tal como está definido en el TQL.	<p>Nombre: query-name.</p> <p>Descripción: El nombre del TQL relevante</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: name</p> <p>Descripción: El elemento raíz tal como está definido en el TQL.</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p>
dynamic_mapping (integration > targetcis > source_instance_type)	Define un mapa que se crea una vez por fragmento.	<p>Nombre: name</p> <p>Descripción: el nombre del mapa</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: keys-unique</p> <p>Descripción: Indica si las claves son únicas o no.</p> <p>¿Es necesario?: necesario</p> <p>Tipo: booleano</p>

Nombre y ruta del elemento	Descripción	Atributos
target_ci_type (integration > targetcis > source_instance_type) -O BIEN- (integration > targetcis > source_instance_type > for-each-source-ci)	Define el tipo de CI de destino que se agregará al RTN.	<p>Nombre: name</p> <p>Descripción: el nombre del tipo de CI de destino</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: is-valid</p> <p>Descripción: comprueba si el CI de destino actual según la secuencia de comandos dada.</p> <p>¿Es necesario?: no necesario</p> <p>Tipo: Cadena (secuencia de comandos groovy)</p>

Nombre y ruta del elemento	Descripción	Atributos
Variable (target_ci_type)	Define una variable para el tipo de CI de destino.	<p>Nombre: name</p> <p>Descripción: el nombre de la variable</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: datatype</p> <p>Descripción: el tipo de datos de la variable.</p> <p>¿Es necesario?: necesario</p> <p>Tipo: type-enum (puede ser uno de los siguientes: INTEGER, LONG, FLOAT, DOUBLE, STRING, BYTES, XML, BOOLEAN, DATE, INTEGER_LIST, STRING_LIST)</p> <hr/> <p>Nombre: value</p> <p>Descripción: El valor que se asignará a la variable</p> <p>¿Es necesario?: necesario</p> <p>Tipo: secuencia de comandos groovy</p>

Nombre y ruta del elemento	Descripción	Atributos
target_mapping (target_ci_type)	Define un atributo para el tipo de CI de destino.	<p>Nombre: name</p> <p>Descripción: el nombre del atributo</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: datatype</p> <p>Descripción: el tipo de datos de la variable.</p> <p>¿Es necesario?: necesario</p> <p>Tipo: type-enum (puede ser uno de los siguientes: INTEGER, LONG, FLOAT, DOUBLE, STRING, BYTES, XML, BOOLEAN, DATE, INTEGER_LIST, STRING_LIST)</p> <hr/> <p>Nombre: value</p> <p>Descripción: El valor que se asignará a la variable</p> <p>¿Es necesario?: necesario</p> <p>Tipo: secuencia de comandos groovy</p>

Nombre y ruta del elemento	Descripción	Atributos
		<p>Nombre: ignore-on-null</p> <p>Descripción: Si el valor de asignación de destino es nulo y este atributo es TRUE, omite el atributo</p> <p>¿Es necesario?: no necesario</p> <p>Tipo: Booleano (secuencia de comandos groovy)</p>
before-mapping (target_ci_type)	Una secuencia de comandos groovy que se ejecuta antes de la asignación del tipo de CI de destino.	
after-mapping (target_ci_type)	Una secuencia de comandos groovy que se ejecuta después de la asignación del tipo de CI de destino.	

Nombre y ruta del elemento	Descripción	Atributos
for-each-source-ci (target_ci_type)	Define una iteración de los CI específicos de la instancia raíz.	<p>Nombre: count-index</p> <p>Descripción: el índice del CI iterado actual</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: var-name</p> <p>Descripción: la variable que hace referencia al CI iterado actual.</p> <p>¿Es necesario?: no necesario</p> <p>Tipo: Cadena</p> <hr/> <p>Nombre: source-cis</p> <p>Descripción: El nombre de CI de la consulta que debe iterarse</p> <p>¿Es necesario?: necesario</p> <p>Tipo: Cadena (secuencia de comandos groovy)</p>

Utilización de las API

Capítulo 8: Introducción a las API

Este capítulo incluye:

Información general de las API	264
--------------------------------------	-----

Información general de las API

Las API se incluyen con HP Universal CMDB:

- **API de Java de UCMDB.** Explica cómo las herramientas de terceros o personalizadas pueden usar la API de Java para extraer datos y cálculos y para escribir datos en UCMDB (Universal Configuration Management database). Para obtener más información, consulte "[API de HP Universal CMDB](#)" en la página 265.
- **API del servicio web de UCMDB.** Permite escribir definiciones de elementos de configuración y relaciones topológicas en UCMDB y consultar la información con TQL y consultas ad hoc. Para obtener más información, consulte "[HP Universal CMDB API del servicio web](#)" en la página 274.
- **API de Java de Administración de Data Flow.** Permite la administración de sondas, trabajos, activadores y credenciales para Data Flow Management. Para obtener más información, consulte "[API de Java de Administración de Data Flow](#)" en la página 309.
- **API de servicio web de Data Flow Management.** Permite la administración de sondas, trabajos, activadores y credenciales para Data Flow Management. Para obtener más información, consulte "[API de servicio web de Administración de Data Flow](#)" en la página 311.

Nota: Para obtener todo el valor de la documentación de API, es recomendable acceder a la documentación en línea. La versión PDF no tiene los vínculos a la documentación de API que se ha generado en formato HTML.

Capítulo 9: API de HP Universal CMDB

Este capítulo incluye:

Convenciones	265
Utilización de las API de HP Universal CMDB	265
Estructura general de una aplicación	266
Colocar el archivo Jar de la API en la ruta de clase	269
Creación de un usuario de integración	269
Casos de uso de la API de UCMDB	271
Ejemplos	272

Convenciones

Este capítulo utiliza las convenciones siguientes:

- UCMDB se refiere a la misma base de datos de Universal Configuration Management. HP Universal CMDB hace referencia a la aplicación.
- Los elementos de UCMDB y argumentos del método se deletrean en caso de que sean específicos en las interfaces.

Para consultar la documentación completa de las API disponibles, consulte HP UCMDB API Reference.

Estos archivos están ubicados en la carpeta siguiente:

```
\\<UCMDB root directory>\hp\UCMDB\UCMDBServer\deploy\ucmdb-docs\docs\eng\APIs\UCMDB_JavaAPI\index.html
```

Utilización de las API de HP Universal CMDB

Nota: Utilice este capítulo junto con el documento de las API Java, disponible en la biblioteca de documentación en línea.

La API de HP Universal CMDB se utiliza para integrar aplicaciones con Universal CMDB (CMDB). La API proporciona métodos para:

- agregar, eliminar y actualizar CI y relaciones de CMDB
- recuperar información sobre el modelo de clase
- recuperar información sobre el historial de UCMDB

- ejecutar escenarios what-if
- recuperar información sobre elementos de configuración y relaciones

Los métodos para recuperar información sobre elementos de configuración y relaciones suelen usar el Lenguaje de consulta de topología (TQL). Para obtener más información, consulte "Topology Query Language" en *HP Universal CMDB – Guía de modelado*.

Los usuarios de la API HP Universal CMDB deberían estar familiarizados con:

- El lenguaje de planificación Java
- HP Universal CMDB

Esta sección incluye los siguientes temas:

- ["Usos de la API" abajo](#)
- ["Permisos" abajo](#)

Usos de la API

La API se usa para cumplir un número de requisitos empresariales. Por ejemplo, un sistema de terceros puede consultar el modelo de clase en busca de información sobre los elementos de configuración disponibles (CI). Para obtener información sobre los casos de uso, consulte "[Casos de uso de la API de UCMDB](#)" en la página 271.

Permisos

El administrador proporciona credenciales de inicio de sesión para conectarse con la API. El cliente de la API necesita el nombre de usuario y la contraseña de un usuario de integración definido en CMDB. Estos usuarios no representan usuarios humanos de CMDB, sino más bien aplicaciones que se conectan a CMDB.

Además, el usuario debe tener el permiso de acción general **Acceso a SDK** para poder iniciar sesión.

Precaución: El cliente de la API puede trabajar también con usuarios normales mientras estos tengan permiso de autenticación de API. Sin embargo, no se recomienda esta opción.

Para obtener más información, consulte "[Creación de un usuario de integración](#)" en la página 269.

Estructura general de una aplicación

Solo hay una fábrica estática, `UcmdbServiceFactory`. Esta fábrica es el punto de entrada para una aplicación. `UcmdbServiceFactory` expone los métodos `getServiceProvider`. Estos métodos devuelven una instancia de la interfaz **`UcmdbServiceProvider`**.

El cliente crea otros objetos usando métodos de interfaz. Por ejemplo, para crear una nueva definición de consulta, el cliente:

1. Obtiene el servicio de consulta del objeto de servicio de CMDB principal.
2. Obtiene un objeto de fábrica de consulta del objeto de servicio.
3. Obtiene una nueva definición de consulta de la fábrica.

```

UcldbServiceProvider provider =
    UcldbServiceFactory.getServiceProvider(HOST_NAME, PORT);
UcldbService ucldbService =
    provider.connect(provider.createCredentials(USERNAME,
        PASSWORD), provider.createClientContext("Test"));
TopologyQueryService queryService = ucldbService.getTopologyQueryService(
);
TopologyQueryFactory factory = queryService.getFactory();
QueryDefinition queryDefinition = factory.createQueryDefinition("Test Que
ry");
queryDefinition.addNode("Node").ofType("host");
Topology topology = queryService.executeQuery(queryDefinition);
System.out.println("There are " + topology.getAllCIs().size() + " hosts i
n uCMDB");
    
```

Los servicios disponibles de **UcldbService** son:

Métodos de servicio	Uso
getAuthorizationModelService	Realizar operaciones de autorización (crear usuarios y grupos de usuarios, asignar funciones a usuarios y grupos, etc.).
getClassModelService	Información sobre tipos de CI y relaciones
getConfigurationService	Administración de configuración de infraestructura, para la configuración de servidor
getDataStoreMgmtService	Información del almacén de datos de consulta que incluye los CI y atributos que se van a federar.
getDDMConfigurationService	Configurar el sistema de Data Flow Management
getDDMManagementService	Analizar y ver el progreso, resultados y errores del sistema Data Flow Management
getDDMZoneService	Importa y exporta zonas de administración (con sus actividades).
getHistoryService	Información sobre el historial de los CI supervisados (cambios, eliminaciones, etc.)

Métodos de servicio	Uso
getImpactAnalysisService	Ejecutar un escenario de análisis de impacto (también conocido como correlación).
getLicensingService	Consulta información sobre las licencias instaladas en el sistema.
getMultipleCMDBService	Convierte entre los identificadores globales y de UCMDDB.
getMultiTenancyService	Crea, lee, actualiza y elimina arrendatarios.
getPersistencyService	Conserva datos binarios en pares clave-valor.
getQueryManagementService	Gestionar el acceso a las consultas: guardar, eliminar, salir de la lista. También proporciona validación de consultas y detección de dependencias de consultas.
getReconciliationService	Proporciona prestaciones de identificación y fusión.
getResourceBundleManagementService	Etiquetado de recursos (servicios de "agrupación"). Permite la creación explícita de nuevas etiquetas y la eliminación de etiquetas de todos los recursos etiquetados.
getResourceManagementService	Desplegar paquetes de recursos (de consultas de TQL, vistas, usuarios, etc.) en el sistema.
getSecurityService	Comprobar si las credenciales son válidas.
getServerService	Consulta información genérica sobre el sistema.
getSnapshotService	Proporciona servicios para administrar instantáneas (obtener, guardar, comparar, etc.)
getSoftwareSignatureService	Definir los elementos de software que va a detectar el sistema de Administración de Data Flow
getStateService	Proporciona servicios para administrar estados (listar, agregar, quitar, etc.)
getSystemHealthService	Proporciona servicios de estado del sistema (indicadores de rendimiento básico del sistema, medidas de capacidad y disponibilidad)
getTopologyQueryService	Obtener información sobre el universo de TI
getTopologyUpdateService	Cambiar la información en el universo de TI
getUcmdbVersion	Consulta UCMDDB, versiones de paquete de contenido e información de compilación.

Métodos de servicio	Uso
getViewArchiveService	Ver servicios de archivado de resultados. Permite el guardado del resultado de la vista actual y la recuperación de los resultados previamente guardados.
getViewService	Ver el servicio de ejecución (definición de ejecución, ejecución guardada) y el servicio de gestión (guardar, eliminar, lista existente). También proporciona una validación de vistas y detección de dependencias.

El cliente se comunica con el servidor mediante HTTP(S).

Colocar el archivo Jar de la API en la ruta de clase

El uso de este conjunto de API requiere el archivo **ucmdb-api.jar**. Puede descargar el archivo introduciendo `http://<localhost>:8080` en un navegador web, donde `localhost` es el equipo donde se ha instalado UCMB, y haciendo clic en el vínculo **API Client Download link**.

Coloque el archivo **.jar** en la ruta de clase antes de compilar o ejecutar la aplicación.

Nota: El uso del archivo jar de la API de Java de UCMB requiere tener instalada la versión 6 o posterior de JRE.

Creación de un usuario de integración

Puede crear un usuario dedicado para integraciones entre otros productos y UCMB. Este usuario habilita un producto que usa el SDK del cliente UCMB para que se autentique en el SDK del servidor y ejecute las API. Las aplicaciones escritas con este conjunto de API deben iniciar sesión con las credenciales del usuario de integración.

Precaución: También es posible conectar con un usuario normal de UCMB (por ejemplo, admin). Sin embargo, no se recomienda esta opción. Para conectar con un usuario de UCMB, debe concederle el permiso de autenticación de API.

Para crear un usuario de integración:

1. Inicie el explorador web y especifique la dirección del servidor, del siguiente modo:

`http://localhost:8080/jmx-console.`

Es posible que tenga que iniciar sesión con un nombre de usuario y una contraseña.

2. En UCMB, haga clic en **service=UCMDB Authorization Services**.
3. Localice la operación **createUser**. Este método acepta los parámetros siguientes:

- **customerId**. El Id. del cliente.
- **username**. El nombre de usuario de integración.
- **userDisplayName**. El nombre de visualización del usuario de integración.
- **userLoginName**. El nombre de inicio de sesión del usuario de integración.
- **password**. La contraseña de usuario de integración.

4. Haga clic en **Invoke**.

5. En un entorno de un solo arrendatario, localice el método **setRolesForUser** e introduzca los parámetros siguientes:

- **userName**. El nombre de usuario de integración.
- **roles**. SuperAdmin.

Haga clic en **Invoke**.

6. En un entorno de múltiples arrendatarios, localice el método **grantRolesToUserForAllTenants** e introduzca los parámetros siguientes para asignar la función en conexión con todos los arrendatarios:

- **userName**. El nombre de usuario de integración.
- **roles**. SuperAdmin.

Haga clic en **Invoke**.

Como método alternativo, para asignar la función en conexión con arrendatarios específicos, invoque el método **grantRolesToUserForTenants** utilizando los mismos valores de los parámetros **userName** y **roles**. En el parámetro **tenantNames**, escriba los arrendatarios requeridos.

7. Cree más usuarios o cierre la consola JMX.

8. Inicie sesión en UCMDB como administrador.

9. En la ficha **Administración**, ejecute **Administrador de paquetes**.

10. Haga clic en el icono **Crear paquete personalizado**.

11. Introduzca un nombre para el nuevo paquete y haga clic en **Siguiente**.

12. En la ficha Selección de un recurso, en **Configuración**, haga clic en **Usuarios**.

13. Seleccione a un usuario o usuarios que haya creado usando la consola JMX.

14. Haga clic en **Siguiente** y, a continuación, en **Finalizar**. Su nuevo paquete aparece en la lista Nombre del paquete en Administrador de paquetes.
15. Implante el paquete a los usuarios que ejecutarán las aplicaciones API.

Para obtener más información, consulte la sección "Cómo desplegar un paquete" en la *HP Universal CMDB – Guía de administración*.

Nota:

El usuario de integración es por cliente. Para crear un usuario de integración más fuerte para un uso entre clientes, utilice un **systemUser** con el indicador **isSuperIntegrationUser** establecido en **true**. Utilice los métodos **systemUser** (**removeUser**, **resetPassword**, **UserAuthenticate**, etc.).

Existen dos usuarios del sistema listos para usar. Se recomienda cambiar sus contraseñas tras la instalación usando el método **resetPassword**.

- **sysadmin/sysadmin**.
- **UISysadmin/UISysadmin** (este usuario también es **SuperIntegrationUser**).

Si cambia la contraseña de UISysadmin mediante **resetPassword**, debe realizar lo siguiente:

- i. En la consola JMX, localice el servicio **UCMDB-UI:name=UCMDB Integration**.
- ii. Ejecute **setCMDBSuperIntegrationUser** con el nombre de usuario y la nueva contraseña del usuario de integración.

Casos de uso de la API de UCMDB

Los casos de uso enumerados en esta sección dan por supuestos dos sistemas:

- Servidor de HP Universal CMDB
- Un sistema de terceros que contiene un repositorio de elementos de configuración

Esta sección incluye los siguientes temas:

- ["Rellenado de CMDB" en la página siguiente](#)
- ["Consultas a CMDB " en la página siguiente](#)
- ["Consulta del modelo de clase" en la página siguiente](#)
- ["Análisis del impacto de cambio " en la página siguiente](#)

Rellenado de CMDB

Casos de uso:

- Una gestión de activos de terceros actualiza CMDB con información disponible solo en la gestión de activos
- Un número de sistemas de terceros rellenan CMDB para crear un CMDB central que puede realizar el seguimiento de los cambios y realizar análisis de impacto
- Un sistema de terceros crea elementos de configuración y relaciones de acuerdo con la lógica empresarial de terceros, para aprovechar las capacidades de consulta de UCMDB

Consultas a CMDB

Casos de uso:

- Un sistema de terceros obtiene los elementos de configuración y las relaciones que representan el sistema SAP recuperando los resultados de SAP TQL
- Un sistema de terceros obtiene la lista de servidores Oracle que se han agregado o modificado en las últimas cinco horas
- Un sistema de terceros obtiene la lista de servidores cuyo nombre de host contiene la subcadena **lab**
- Un sistema de terceros encuentra los elementos relacionados con un CI determinado obteniendo sus vecinos

Consulta del modelo de clase

Casos de uso:

- Un sistema de terceros permite a los usuarios especificar el conjunto de datos que se va a recuperar desde CMDB. Una interfaz de usuario se puede construir en el modelo de clase para mostrar a los usuarios las posibles propiedades y solicitarles los datos requeridos. El usuario puede elegir después la información que se va a recuperar.
- Un sistema de terceros explora el modelo de clase cuando el usuario no puede acceder a la interfaz de usuario de UCMDB.

Análisis del impacto de cambio

Caso de uso:

- Un sistema de terceros produce una lista de los servicios empresariales que podrían verse impactados por un cambio en un host especificado.

Ejemplos

Vea los ejemplos de código siguientes:

- Create a Connection
- Create and Execute an Ad Hoc Query
- Create and Execute a View
- Add and Delete Data
- Execute an Impact Analysis
- Query the Class Model
- Query a History Sample

Estos archivos están ubicados en el siguiente directorio:

\\<directorio raíz de UCMDB>\hp\UCMDB\UCMDBServer\deploy\ucmdb-docs\docs\eng\APIs\JavaSDK_Samples

Capítulo 10: HP Universal CMDB API del servicio web

Este capítulo incluye:

Convenciones	274
Información general de la API de servicio web de HP Universal CMDB	275
Llamada al servicio web	278
Consulta de CMDB	278
Actualización de UCMDB	282
Importación del modelo de clase de UCMDB	283
Consulta de análisis de impacto	285
Parámetros generales de UCMDB	285
Parámetros de salida de UCMDB	288
Métodos de consulta de UCMDB	289
Métodos de actualización de UCMDB	301
Métodos de análisis de impacto de UCMDB	303
API del servicio web de estado real	305
Casos de uso de la API de los servicios web de UCMDB	307
Ejemplos	308

Convenciones

Este capítulo utiliza las convenciones siguientes:

- UCMDB se refiere a la misma base de datos de Universal Configuration Management. HP Universal CMDB hace referencia a la aplicación.
- Los elementos de UCMDB y argumentos del método se deletrean en caso de que se especifiquen en el esquema. Los elementos o argumentos a un método no se escriben en mayúsculas. Por ejemplo, `relation` es un elemento del tipo `Relation` pasado a un método.

Para obtener la documentación completa en las estructuras de solicitud y respuesta, consulte HP UCMDB Web Service API Reference. Estos archivos están ubicados en la carpeta siguiente:

```
<directorio raíz de UCMDB>\UCMDBServer\deploy\ucmdb-docs\docs\eng\APIs\CMDB_Schema\webframe.html
```

Información general de la API de servicio web de HP Universal CMDB

Nota: Utilice este capítulo junto con la documentación del esquema de UCMDB, disponible en la biblioteca de documentación en línea.

HP Universal CMDBLa API del servicio web de HP Universal CMDB (UCMDB).. La API proporciona métodos para:

- agregar, eliminar y actualizar CI y relaciones de CMDB
- recuperar información sobre el modelo de clase
- recuperar los análisis de impacto
- recuperar información sobre elementos de configuración y relaciones
- administrar credenciales: ver, agregar, actualizar y suprimir
- administrar trabajos: ver estado, activar y desactivar
- Administrar intervalos de sondas: ver, agregar y actualizar
- administrar activadores: agregar o suprimir un CI de activación, y agregar, suprimir o deshabilitar un TQL de activación
- ver datos generales en dominios y sondas

Los métodos para recuperar información sobre elementos de configuración y relaciones suelen usar el Lenguaje de consulta de topología (TQL). Para obtener más información, consulte "Topology Query Language" en *HP Universal CMDB – Guía de modelado*.

Los usuarios de la API del servicio web de HP Universal CMDB deberían estar familiarizados con:

- La especificación SOAP
- Un lenguaje de planificación orientado a objetos, como C++, C# o Java
- HP Universal CMDB
- Administración de Data Flow

Esta sección incluye los siguientes temas:

- ["Usos de la API" en la página siguiente](#)
- ["Permisos" en la página siguiente](#)

Usos de la API

La API de servicios web de UCMDB se usa para cumplir un número de requisitos empresariales. Por ejemplo:

- Un sistema de terceros puede consultar el modelo de clase en busca de información sobre los elementos de configuración disponibles (CI).
- Una herramienta de gestión de activos de terceros puede actualizar CMDB con información disponible solo a dicha herramienta, unificando por tanto sus datos con los recopilados por las aplicaciones de HP.
- Un número de sistemas de terceros pueden rellenar CMDB para crear un CMDB central que puede realizar el seguimiento de los cambios y realizar el análisis de impacto.
- Un sistema de terceros puede crear entidades y relaciones de acuerdo con su lógica empresarial y, después, escribir los datos en CMDB para sacar partido a las capacidades de consulta de CMDB.
- Otros sistemas, como Release Control (CCM), puede usar los métodos de análisis de impacto para el análisis de cambios.

Permisos

Para acceder al archivo WSDL para el servicio web, visite:

<http://localhost:8080/axis2/services/UcmdbService?wsdl>. Tendrá que proporcionar credenciales de usuario administrador de servidor para ver el archivo WSDL.

El usuario debe tener el permiso de acción general **Ejecutar API heredada** para poder iniciar la sesión.

La tabla siguiente muestra los permisos necesarios adicionales para cada comando de la API de servicio web:

Comando de la API de servicio web	Permisos obligatorios
addCIsAndRelations deleteCIsAndRelations updateCIsAndRelations	Acción general: Actualización de datos
executeTopologyQueryByName(AdHoc) executeTopologyQueryByNameWithParameters(AdHoc) executeTopologyQueryWithParameters(AdHoc)	Acción general: Ejecutar consulta por definición Para cada consulta: Ver permiso

Comando de la API de servicio web	Permisos obligatorios
getTopologyQueryExistingResultByName getTopologyQueryResultCountByName releaseChunks pullTopologyMapChunks getCI Neighbours getFilteredCIsByType getCIsById getCIsByType getRelationsById	Acción general: Ver los CI Para cada consulta: Ver permiso
getQueryNameOfView	Acción general: Ver los CI Para cada vista: Ver permiso
getChangedCIs	Acción general: Ver historial, Ver los CI
calculateImpact getImpactPath getImpactRulesByGroupName getImpactRulesByNamePrefix	Acción general: Ejecutar análisis de impacto
getAllClassesHierarchy getClassAncestors getCmdbClassDefinition	Ninguno

Nota: Cuando se cambie el contexto raíz en UCMDB, siga estos pasos para habilitar el acceso a la API del servicio web:

1. Abra el archivo de configuración **\UCMDB\UCMDBServer\deploy\axis2WEB-INF\web.xml** y busque la siguiente sección:

```
<servlet-class>
org.apache.axis2.transport.http.AxisServlet
</servlet-class>
```

Agregue las siguientes líneas después de esa sección:

```
<init-param>
<param-name>axis2.find.context</param-name>
<param-value>>false</param-value>
</init-param>
```

2. Abra el archivo de configuración **\UCMDB\UCMDBServer\deploy\axis2WEB-INF\conf\axis2.xml** y busque la siguiente línea:

```
<parameter name="enableSwA" locked="false">>false</parameter>
```

Agregue lo siguiente después de esa línea:

```
<parameter name="contextRoot" locked="false">test1/setup1/axis2
</parameter>
```

donde **test1/setup1** es el contexto raíz.

(Para quitar el contexto raíz, elimine el texto agregado a la ruta).

3. Reinicie el servidor de UCMDB.

Llamada al servicio web

Utilice las técnicas de planificación SOAP estándar en la API de servicios web de HP Universal CMDB para habilitar la llamada a los métodos del servidor. Si la instrucción no se puede analizar o si hay un problema al invocar al método, los métodos de la API lanzan una excepción `SoapFault`. Cuando se lanza una excepción `SoapFault`, UCMDB rellena uno o varios de los campos de mensaje de error, código de error y mensaje de excepción. Si no hay ningún error, se devuelven los resultados de la invocación.

Los programadores de SOAP pueden acceder a WSDL en:

`http://<servidor>[:port]/axis2/services/UcmdbService?wsdl`

La especificación del puerto solo es necesaria para las instalaciones no estándares. Pregunte al administrador del sistema el número de puerto correcto.

La dirección URL para llamar al servicio es:

`http://<servidor>[:port]/axis2/services/UcmdbService`

Para obtener ejemplos de conexión a CMDB, consulte "[Casos de uso de la API de los servicios web de UCMDB](#)" en la página 307.

Consulta de CMDB

CMDB es consultado mediante las API descritas en "[Métodos de consulta de UCMDB](#)" en la [página 289](#). Las consultas y los elementos CMDB devueltos siempre contienen unos Id. de UCMDB reales. Para obtener ejemplos del uso de métodos de consulta, consulte Query Example.

Esta sección incluye los siguientes temas:

- "[Cálculo de la respuesta JIT \(Justo a tiempo\)](#)" en la página siguiente
- "[Procesamiento de respuestas largas](#)" en la página siguiente
- "[Especificación de propiedades para devolver](#)" en la página siguiente
- "[Propiedades concretas](#)" en la página 280

- ["Propiedades derivadas" en la página siguiente](#)
- ["Propiedades de nomenclatura" en la página 281](#)
- ["Otros elementos de especificación de propiedades" en la página 281](#)

Cálculo de la respuesta JIT (Justo a tiempo)

Para todos los métodos de consulta, el servidor de UCMDB calcula los valores solicitados por el método de consulta cuando se recibe la petición, y devuelve los resultados basados en los datos más recientes. El resultados siempre se calcula en el momento en que se recibe la petición, aunque la consulta TQL esté activa y haya un resultado anteriormente calculado. Por tanto, los resultados de ejecutar una consulta devuelta a la aplicación cliente pueden ser diferentes de los resultados de la misma consulta mostrada en la interfaz de usuario.

Sugerencia: Si la aplicación usa los resultados de una consulta dada más de una vez y no está previsto que los datos cambien de manera significativa entre los usos de los datos del resultado, puede mejorar el rendimiento haciendo que la aplicación cliente almacene los datos en lugar de ejecutar repetidamente la consulta.

Procesamiento de respuestas largas

La respuesta a una consulta siempre incluye las estructuras de los datos solicitados por el método de consulta, aunque no se esté transmitiendo ningún dato real. Para los métodos cuyos datos sean una colección o asignación, la respuesta también incluye la estructura `ChunkInfo`, compuesta de `chunksKey` y `numberOfChunks`. El campo `numberOfChunks` indica el número de fragmentos que contienen datos que deben recuperarse.

El administrador del sistema establece el tamaño máximo de transmisión de datos. Si los datos devueltos de la consulta son mayores que el tamaño máximo, las estructuras de datos de la primera respuesta no contendrán información significativa y el valor del campo `numberOfChunks` será 2 o mayor. Si los datos no son mayores que el tamaño máximo, el campo `numberOfChunks` será 0 (cero) y los datos se transmitirán en la primera respuesta. Por consiguiente, cuando procese una respuesta, compruebe primero el valor de `numberOfChunks`. Si es mayor que 1, descarte los datos de la transmisión y solicite los fragmentos de datos. En caso contrario, utilice los datos en la respuesta.

Para obtener información sobre cómo gestionar datos fragmentados, consulte ["pullTopologyMapChunks" en la página 299](#) y ["releaseChunks" en la página 300](#).

Especificación de propiedades para devolver

Los CI y las relaciones por lo general tienen varias propiedades. Algunos métodos que devuelven colecciones o gráficos de estos elementos aceptan parámetros de entrada que especifican qué valores de propiedades se devuelven con cada elemento que coincida con la consulta. CMDB no devuelve propiedades vacías. Por tanto, la respuesta a una consulta puede tener menos propiedades que las solicitadas en la consulta.

En esta sección se describen los tipos de conjuntos usados para especificar las propiedades que se van a devolver.

Las propiedades se pueden hacer referencia en dos formas:

- Por sus nombres
- Usando nombres de reglas de propiedades predefinidas. Las reglas de propiedades predefinidas las usa CMDB para crear una lista de nombres de propiedades reales.

Cuando una aplicación hace referencia a las propiedades por el nombre, pasa un elemento `PropertiesList`.

Sugerencia: Siempre que sea posible, utilice `PropertiesList` para especificar los nombres de las propiedades en las que está interesado, en lugar de un conjunto basado en reglas. El uso de reglas de propiedades predefinidas casi siempre da como resultado la devolución de más propiedades de las necesarias, lo que afecta al rendimiento.

Hay dos tipos de propiedades predefinidas: propiedades de calificador y propiedades simples

- **Propiedades de calificador.** Utilícelas cuando la aplicación cliente deba pasar un elemento `QualifierProperties` (una lista de calificadores que se pueden aplicar a las propiedades). CMDB convierte la lista de calificadores pasados por la aplicación cliente a la lista de propiedades a la que se aplica al menos uno de los calificadores. Los valores de estas propiedades se devuelven con los elementos `CI` o `Relation`.
- **Propiedades simples.** Para usar propiedades basadas en reglas simples, la aplicación cliente pasa un elemento `SimplePredefinedProperty` o `SimpleTypedPredefinedProperty`. Estos elementos contienen el nombre de la regla por la cual CMDB genera la lista de propiedades que se van a devolver. Las reglas que se pueden especificar en un elemento `SimplePredefinedProperty` o `SimpleTypedPredefinedProperty` son `CONCRETE`, `DERIVED` y `NAMING`.

Propiedades concretas

Las propiedades concretas son el conjunto de propiedades definidas por el CIT especificado. Las propiedades agregadas por las clases derivadas no se devuelven por instancias de esas clases derivadas.

Una colección de instancias devueltas por un método puede consistir en instancias de un CIT especificado en la invocación del método e instancias de los CIT que heredan de dicho CIT. Los CIT derivados heredan las propiedades del CIT especificado. Además, los CIT derivados amplían el CIT principal agregando propiedades.

Ejemplo de propiedades concretas:

El CIT T1 tiene las propiedades P1 y P2. El CIT T11 hereda de T1 y amplía T1 con las propiedades P21 y P22.

La colección de los CI de tipo T1 incluye las instancias de T1 y T11. Las propiedades concretas de todas las instancias de esta colección son P1 y P2.

Propiedades derivadas

Las propiedades derivadas son el conjunto de propiedades definidas para el CIT especificado y,

para cada CIT derivado, las propiedades añadidas por el CIT derivado.

Ejemplo de propiedades derivadas:

Siguiendo con el ejemplo de las propiedades concretas, las propiedades derivadas de las instancias de T1 son P1 y P2. Las propiedades derivadas de las instancias de T11 son P1, P2, P21 y P22.

Propiedades de nomenclatura

Las propiedades de nomenclatura son `display_label` y `data_name`.

Otros elementos de especificación de propiedades

- **PredefinedProperties**

`PredefinedProperties` puede contener un elemento `QualifierProperties` y un elemento `SimplePredefinedProperty` para cada una de las otras reglas posibles. Un conjunto de `PredefinedProperties` no contiene necesariamente todos los tipos de listas.

- **PredefinedTypedProperties**

`PredefinedTypedProperties` se usa para aplicar un conjunto diferente de propiedades para cada CIT. `PredefinedTypedProperties` puede contener un elemento `QualifierProperties` y un elemento `SimpleTypedPredefinedProperty` para cada una de las otras reglas aplicables. Como `PredefinedTypedProperties` se aplica a cada CIT individualmente, no son relevantes las propiedades derivadas. Un conjunto de `PredefinedProperties` no contiene necesariamente todos los tipos de listas aplicables.

- **CustomProperties**

`CustomProperties` puede contener cualquier combinación de la `PropertiesList` básica y las listas de propiedades basadas en reglas. El filtro de propiedades es la unión de todas las propiedades devueltas por todas las listas.

- **CustomTypedProperties**

`CustomTypedProperties` puede contener cualquier combinación de la `PropertiesList` básica y las listas de propiedades basadas en reglas aplicables. El filtro de propiedades es la unión de todas las propiedades devueltas por todas las listas.

- **TypedProperties**

`TypedProperties` se usa para pasar un conjunto diferente de propiedades para cada CIT. `TypedProperties` es una colección de pares compuestos de nombres de tipo y conjuntos de propiedades de todos los tipos. Cada conjunto de propiedades se aplica solo al tipo correspondiente.

Actualización de UCMDB

Actualice CMDB con las API de actualización. Para obtener más información sobre los métodos de API, consulte ["Métodos de actualización de UCMDB" en la página 301](#).

Esta tarea incluye los siguientes pasos:

- ["Parámetros de actualización de UCMDB" abajo](#)
- ["Utilización de tipos de Id. con métodos de actualización" abajo](#)

Parámetros de actualización de UCMDB

En este tema se describen los parámetros usados solo por los métodos de actualización del servicio.

- **CIsAndRelationsUpdates**

El tipo `CIsAndRelationsUpdates` consta de `CIsForUpdate`, `relationsForUpdate`, `referencedRelations` y `referencedCIs`. Una instancia de `CIsAndRelationsUpdates` no incluye necesariamente los tres elementos.

`CIsForUpdate` es una colección de `CI`. `relationsForUpdate` es una colección de elementos `Relation`. Los elementos `CI` y `relation` en las colecciones tienen un elemento `props`. Al crear un `CI` o una relación, las propiedades que tienen el atributo `required` o el atributo `key` en la definición del tipo de `CI` deben estar rellenos con valores. Los elementos de estas colecciones se actualizan o se crean por el método.

`referencedCIs` y `referencedRelations` son colecciones de `CI` que ya están definidas en CMDB. Los elementos de la colección se identifican con un `Id. temporal` junto con todas las propiedades clave. Estos elementos se usan para resolver las identidades de los `CI` y las relaciones para su actualización. Nunca se crean ni se actualizan por el método.

Cada uno de los elementos `CI` y `relation` de estas colecciones tienen una colección de propiedades. Los nuevos elementos se crean con los valores de las propiedades de estas colecciones.

Utilización de tipos de Id. con métodos de actualización

Lo siguiente describe los `CIT` de `Id.`, así como los `CI` y relaciones. Cuando el `Id.` no es un `Id.` de CMDB auténtico, se requieren el tipo y los atributos clave.

- **Eliminación o actualización de elementos de configuración**

El cliente puede usar un `Id. temporal` o vacío al llamar a un método para eliminar o actualizar un elemento. En este caso, deben establecerse el tipo de `CI` y los ["Atributos clave"](#) que identifican el `CI`.

- **Eliminación o actualización de relaciones**

Al eliminar o actualizar relaciones, el Id. de relación puede estar vacío, ser temporal o real.

Si el Id. de un CI es temporal, el CI debe pasarse en la colección `referencedCIs` y se deben especificar sus atributos clave. Para más información, consulte `referencedCIs` en ["CIsAndRelationsUpdates"](#) en la página precedente.

- **Inserción de nuevos elementos de configuración en CMDB**

Es posible usar un Id. vacío o un Id. temporal para insertar un nuevo CI. Sin embargo, si el Id. está vacío, el servidor no puede devolver el Id. de CMDB en la estructura `createIDsMap` porque no hay ningún `clientID`. Para obtener más información, consulte ["addCIsAndRelations"](#) en la página 301 y ["Métodos de consulta de UCMDB"](#) en la página 289.

- **Inserción de nuevas relaciones en CMDB**

El Id. de relación puede ser temporal o estar vacío. Sin embargo, si la relación es nueva pero los elementos de configuración de cada extremo de la relación ya están definidos en CMDB, en ese caso los CI que ya existen deben ser identificados por un Id. de CMDB real o especificarse en una colección de `referencedCIs`.

Importación del modelo de clase de UCMDB

Los métodos del modelo de clase devuelven información sobre los CIT y relaciones. El modelo de clase se configura usando el Administrador de tipos de CI. Para obtener más información, consulte `CI Type Manager` en *HP Universal CMDB – Guía de modelado*.

En esta sección se proporciona información sobre los siguientes métodos que devuelven información sobre CIT y relaciones:

- ["getClassAncestors"](#) abajo
- ["getAllClassesHierarchy"](#) en la página siguiente
- ["getCmdbClassDefinition"](#) en la página siguiente

getClassAncestors

El método `getClassAncestors` recupera la ruta entre el CIT dado y su raíz, incluyendo la raíz.

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte "CmdbContext" en la página 285.
<code>className</code>	El nombre del tipo. Para obtener más información, consulte "Nombre de tipo" en la página 287.

Salida

Parámetro	Comentario
classHierarchy	Una colección de pares de nombres de clase y nombre de clase principal.
comments	Solo para uso interno.

getAllClassesHierarchy

El método `getAllClassesHierarchy` recupera todo el árbol del modelo de clase.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página siguiente.

Salida

Parámetro	Comentario
classesHierarchy	Una colección de pares de nombre de clase y nombre de clase principal.
comments	Solo para uso interno.

getCmdbClassDefinition

El método `getCmdbClassDefinition` recupera información sobre la clase especificada.

Si usa `getCmdbClassDefinition` para recuperar los atributos clave, también debe consultar las clases principales hasta la clase base. `getCmdbClassDefinition` identifica como atributos clave solo aquellos atributos con el `ID_ATTRIBUTE` establecido en la definición de clase especificada por `className`. Los atributos clave heredados no se reconocen como atributos clave de la clase especificada. Por consiguiente, la lista completa de atributos clave para la clase especificada es la unión de todas las claves de la clase y de todos sus elementos principales, hasta la raíz.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página siguiente.
className	El nombre del tipo. Para obtener más información, consulte "Parámetros generales de UCMDB" en la página siguiente.

Salida

Parámetro	Comentario
cmdbclass	La definición de la clase, que consiste en name, classType, displayLabel, description, parentName, calificadores y atributos.
comments	Solo para uso interno.

Consulta de análisis de impacto

El identificador en los métodos de análisis de impacto apunta a los datos de respuesta del servicio. Es único para la respuesta actual y se descarta de la caché de la memoria del servidor si pasan 10 minutos sin utilizarse.

Para ver ejemplos del uso de los métodos de análisis de impacto, consulte [Impact Analysis Example](#).

Parámetros generales de UCMDB

Esta sección describe los parámetros más comunes de los métodos del servicio.

Esta sección incluye los siguientes temas:

- ["CmdbContext" abajo](#)
- ["ID" abajo](#)
- ["Atributos clave" en la página siguiente](#)
- ["Tipos de Id." en la página siguiente](#)
- ["CIProperties" en la página siguiente](#)
- ["Nombre de tipo" en la página 287](#)
- ["Elemento de configuración \(CI\)" en la página 287](#)
- ["Relation" en la página 287](#)

CmdbContext

Todas las invocaciones a los servicios de la API del servicio web de UCMDB requieren un argumento CmdbContext. CmdbContext es una cadena de callerApplication que identifica la aplicación que invoca al servicio. CmdbContext se utiliza para el registro y la solución de problemas.

ID

Cada CI y Relation tiene un campo ID. Consiste en una cadena de Id. que distingue mayúsculas de minúsculas y un indicador temp opcional, que indica si el Id. es temporal.

Atributos clave

Para identificar un `CI` o `Relation` en algunos contextos, se pueden usar los atributos clave en lugar de un `Id.` de CMDB. Los atributos clave son aquellos atributos con el conjunto `ID_ATTRIBUTE` en la definición de clase.

En la interfaz de usuario, los atributos clave tiene un icono de clave junto a ellos en la lista de atributos de tipos de elementos de configuración en la interfaz de usuario. Para obtener más información, consulte "Cuadro de diálogo Agregar/Editar atributo" en HP Universal CMDB – Guía de modelado. Para obtener información sobre cómo identificar los atributos clave desde dentro de la aplicación cliente de API, consulte ["getCmdbClassDefinition" en la página 284](#).

Tipos de Id.

Un elemento `ID` puede contener un `Id.` real o un `Id.` temporal.

Un `Id.` real es una cadena asignada por CMDB que identifica una entidad en la base de datos. Un `Id.` temporal puede ser cualquier cadena que sea única en la petición actual.

El cliente puede asignar un `Id.` temporal y con frecuencia representa el `Id.` del `CI` tal como fue almacenado por el cliente. No representa necesariamente una entidad ya creada en CMDB. Cuando el cliente pasa un `Id.` temporal, si CMDB puede identificar un elemento de configuración de datos existente usando las propiedades clave del `CI`, dicho `CI` se usa como resultado apropiado para el contexto, como si hubiera sido identificado con un `Id.` real.

CIProperties

El elemento `CIProperties` está compuesto de colecciones, conteniendo cada una de ellas una secuencia de elementos nombre-valor que especifican propiedades del tipo indicado por el nombre de la colección. No se requiere ninguna de las colecciones, por tanto el elemento `CIProperties` puede contener cualquier combinación de colecciones.

`CIProperties` los usan los elementos `CI` y `Relation`. Para obtener más información, consulte ["Elemento de configuración \(CI\)" en la página siguiente](#) y ["Relation" en la página siguiente](#).

Las colecciones de propiedades son las siguientes:

- `dateProps`: colección de elementos `DateProp`
- `doubleProps`: colección de elementos `DoubleProp`
- `floatProps`: colección de elementos `FloatProp`
- `intListProps`: colección de elementos `intListProp`
- `intProps`: colección de elementos `IntProp`
- `strProps`: colección de elementos `StrProp`
- `strListProps`: colección de elementos `StrListProp`
- `longProps`: colección de elementos `LongProp`

- bytesProps: colección de elementos BytesProp
- xmlProps: colección de elementos XmlProp

Nombre de tipo

El nombre de tipo es el nombre de la clase de un tipo de elemento de configuración o tipo de relación. El nombre de tipo se usa en el código para referirse a la clase. No debería confundirse con el nombre de visualización, que se ve en la interfaz de usuario donde se menciona la clase, pero que carece de sentido en el código.

Elemento de configuración (CI)

Un elemento CI está compuesto de un ID, type y una colección de props.

Si se utiliza "[Métodos de actualización de UCMDB](#)" para actualizar un CI, el elemento ID puede contener un Id. de CMDB real o un Id. temporal asignado por el cliente. Si se usa un Id. temporal, establezca el indicador temp en true. Al eliminar un elemento, el ID puede estar vacío. Los "[Métodos de consulta de UCMDB](#)" reconocen los ID reales como parámetros de entrada y devuelven ID reales en los resultados de las consultas.

type puede ser cualquier nombre de tipo definido en el Administrador de tipos de CI. Para obtener más información, consulte "Administrador de tipos de CI" en *HP Universal CMDB – Guía de modelado*.

El elemento props es una colección de CIProperties. Para más información, consulte "[Parámetros generales de UCMDB](#)" en la [página 285](#).

Relation

Relation es una entidad que enlaza dos elementos de configuración. El elemento Relation está compuesto de un ID, un type, los identificadores de los dos elementos que se están enlazando (end1ID y end2ID) y una colección de props.

Al utilizar los "[Métodos de actualización de UCMDB](#)" para actualizar un elemento Relation, el valor de Id. de Relation puede ser un Id. CMDB real o un Id. temporal. Al eliminar un elemento, el ID puede estar vacío. Los "[Métodos de consulta de UCMDB](#)" reconocen los ID reales como parámetros de entrada y devuelven ID reales en los resultados de las consultas.

El tipo de relación es el nombre de tipo de la clase de UCMDB desde la que se instancia la relación. El tipo puede ser cualquier tipo de relación definido en CMDB. Para obtener más información sobre las clases o tipos, consulte "[Importación del modelo de clase de UCMDB](#)" en la [página 283](#).

Para obtener más información, consulte "Administrador de tipos de CI" en *HP Universal CMDB – Guía de modelado*.

Los dos Id. de extremo de la relación no deben ser ID vacíos porque se utilizan para crear el Id. de la relación actual. Sin embargo, ambos pueden tener unos Id. temporales asignados a ellos por el cliente.

El elemento props es una colección de CIProperties. Para más información, consulte "[CIProperties](#)" en la [página precedente](#).

Parámetros de salida de UCMDB

Esta sección describe los parámetros de salida más comunes de los métodos del servicio. Para obtener más información, consulte la [online schema documentation](#).

Esta sección incluye los siguientes temas:

- ["CI" abajo](#)
- ["ShallowRelation" abajo](#)
- ["Topology" abajo](#)
- ["CINode" abajo](#)
- ["RelationNode" abajo](#)
- ["TopologyMap" abajo](#)
- ["ChunkInfo" en la página siguiente](#)

CI

CI es una colección de elementos de CI.

ShallowRelation

ShallowRelation es una entidad que enlaza dos elementos de configuración, compuestos de un ID, un tipo y de los identificadores de los dos elementos que se están enlazando (`end1ID` y `end2ID`). El tipo de relación es el nombre de tipo de la clase de CMDB desde la que se instancia la relación. El tipo puede ser cualquier tipo de relación definido en CMDB.

Topology

Topology es un grafo de elementos y relaciones de CI. Un elemento Topology consta de una colección de CIs y una colección de Relations que contienen uno o varios elementos Relation.

CINode

CINode está compuesto de una colección de CIs con un elemento `label`. El elemento `label` en CINode es la etiqueta definida en el nodo del TQL usado en la consulta.

RelationNode

RelationNode es un conjunto de colecciones de elementos Relation con un elemento `label`. El elemento `label` en RelationNode es la etiqueta definida en el nodo del TQL usado en la consulta.

TopologyMap

TopologyMap es la salida de un cálculo de consulta que coincide con una consulta TQL. El elemento `label` en TopologyMap son las etiquetas de nodo definidas en el TQL usado en la consulta.

Los datos de TopologyMap se devuelven de la forma siguiente:

- `CINodes`. Es uno o varios `CINode` (consulte ["CINode" en la página precedente](#)).
- `relationNodes`. Es uno o varios `RelationNode` (consulte ["RelationNode" en la página precedente](#)).

Los elementos `label` de estas dos estructuras ordenan las listas de elementos de configuración y relaciones.

ChunkInfo

Cuando una consulta devuelve una gran cantidad de datos, el servidor almacena los datos, divididos en segmentos llamados fragmentos. La información que usa el cliente para recuperar los datos fragmentados se localiza en la estructura `ChunkInfo` devuelta por la consulta. `ChunkInfo` está compuesta de `numberOfChunks` que debe recuperarse y por `chunksKey`. `chunksKey` es un identificador exclusivo de los datos en el servidor para esta invocación de consulta específica.

Para obtener más información, consulte ["Procesamiento de respuestas largas" en la página 279](#).

Métodos de consulta de UCMDB

En esta sección se proporciona información sobre los siguientes métodos:

- ["executeTopologyQueryByNameWithParameters" abajo](#)
- ["executeTopologyQueryWithParameters" en la página siguiente](#)
- ["getChangedCIs" en la página 291](#)
- ["getCINeighbours" en la página 292](#)
- ["getCIsByID" en la página 292](#)
- ["getCIsByType" en la página 293](#)
- ["getFilteredCIsByType" en la página 293](#)
- ["getQueryNameOfView" en la página 297](#)
- ["getTopologyQueryExistingResultByName" en la página 298](#)
- ["getTopologyQueryResultCountByName" en la página 298](#)
- ["pullTopologyMapChunks" en la página 299](#)
- ["releaseChunks" en la página 300](#)

executeTopologyQueryByNameWithParameters

El método `executeTopologyQueryByNameWithParameters` recupera un elemento `topologyMap` que coincide con la consulta parametrizada especificada.

Los valores de los parámetros de la consulta se pasan en el argumento `parameterizedNodes`. El TQL especificado debe tener etiquetas únicas definidas para cada `CINode` y para cada `relationNode` o la invocación del método producirá un error.

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte "CmdbContext" en la página 285 .
<code>queryName</code>	El nombre del TQL parametrizado en CMDB para el que se obtiene el mapa.
<code>parameterizedNodeList</code>	Las condiciones que cada nodo debe cumplir para ser incluido en los resultados de la consulta.
<code>queryTypedProperties</code>	Una colección de conjuntos de propiedades para recuperar elementos de un tipo de elemento de configuración específico.

Salida

Parámetro	Comentario
<code>topologyMap</code>	Para obtener más información, consulte "TopologyMap" en la página 288 .
<code>chunkInfo</code>	Para obtener más información, consulte "ChunkInfo" en la página precedente y "Procesamiento de respuestas largas" en la página 279 .

executeTopologyQueryWithParameters

El método `executeTopologyQueryWithParameters` recupera un elemento `topologyMap` que coincide con la consulta parametrizada.

La consulta se pasa en el argumento `queryXML`. Los valores de los parámetros de la consulta se pasan en el argumento `parameterizedNodeList`. El TQL debe tener etiquetas únicas definidas para cada `CINode` y para cada `relationNode`.

El método `executeTopologyQueryWithParameters` se usa para pasar consultas ad hoc, en lugar de acceder a una consulta definida en CMDB. Puede usar este método cuando no tiene acceso a la interfaz de usuario de UCMDB para definir una consulta, o cuando no desea guardar la consulta en la base de datos.

Para utilizar un TQL exportado como entrada de este método, realice lo siguiente:

1. Inicie el navegador web e introduzca la siguiente dirección:
`http://localhost:8080/jmx-console`.

Es posible que tenga que iniciar sesión con un nombre de usuario y una contraseña.

2. Haga clic en **UCMDB:service=TQL Services**.

3. Localice la operación **exportTql**.
 - En el cuadro del parámetro **customerId**, escriba **1** (el valor predeterminado).
 - En el cuadro de parámetros **patternName**, introduzca un nombre de TQL válido.
4. Haga clic en **Invoke**.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285 .
queryXML	Una cadena XML que representa un TQL sin etiquetas de recursos.
parameterizedNodeList	Las condiciones que cada nodo debe cumplir para ser incluido en los resultados de la consulta.

Salida

Parámetro	Comentario
topologyMap	Para obtener más información, consulte " TopologyMap " en la página 288 .
chunkInfo	Para obtener más información, consulte " ChunkInfo " en la página 289 y " Procesamiento de respuestas largas " en la página 279 .

getChangedCIs

El método `getChangedCIs` devuelve los datos de cambio para todos los CI relacionados con los CI especificados.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285 .
ids	La lista de ID de los CI raíz en cuyos CI relacionados se están comprobando los cambios. Solo los Id. reales de CMDB son válidos en esta colección
fromDate	El comienzo del periodo en donde comprobar si los CI han cambiado.
toDate	El fin del periodo en donde comprobar si los CI han cambiado.

Salida

Parámetro	Comentario
getChangedCIsResponseList	Cero o más colecciones de elementos ChangedDataInfo.

getCI Neighbours

El método `getCI Neighbours` devuelve los vecinos inmediatos del CI especificado.

Por ejemplo, si la consulta es sobre los vecinos de CI A y el CI A contiene el CI B que usa el CI C, se devuelve el CI B, pero no el CI C. Es decir, solo se devuelven los vecinos del tipo especificado.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
ID	El Id. del CI para el que se van a recuperar los vecinos. Debe ser un Id. de CMDB real.
neighbourType	El nombre de CIT de los vecinos que se van a recuperar. Se devuelven los vecinos del tipo especificado y de los tipos procedentes de ese tipo. Para obtener más información, consulte " Nombre de tipo " en la página 287.
CIProperties	Los datos que se van a devolver en cada elemento de configuración, conocidos como diseño de consulta en la interfaz de usuario. Para obtener más información, consulte " TypedProperties " en la página 281.
relationProperties	Los datos que se van a devolver en cada relación (conocidos como diseño de consulta en la interfaz de usuario). Para obtener más información, consulte " TypedProperties " en la página 281.

Salida

Parámetro	Comentario
topology	Para obtener más información, consulte " Topology " en la página 288.
comments	Solo para uso interno.

getCIsByID

El método `getCIsByID` recupera los elementos de configuración por sus Id. de CMDB.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285 .
CIsTypedProperties	Una colección de propiedades con tipo. Para obtener más información, consulte "Otros elementos de especificación de propiedades" en la página 281 .
IDs	Solo los Id. reales de CMDB son válidos en esta colección

Salida

Parámetro	Comentario
CI	Una colección de elementos de CI.
chunkInfo	Para obtener más información, consulte "ChunkInfo" en la página 289 y "Procesamiento de respuestas largas" en la página 279 .

getCIsByType

El método `getCIsByType` devuelve la colección de elementos de configuración del tipo especificado y de todos los tipo que heredan del tipo especificado.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285 .
type	El nombre de la clase. Para obtener más información, consulte "Nombre de tipo" en la página 287 .
properties	Los datos que se van a devolver en cada elemento de configuración. Para obtener más información, consulte "CustomProperties" en la página 281 .

Salida

Parámetro	Comentario
CI	Una colección de elementos de CI.
chunkInfo	Para obtener más información, consulte: "ChunkInfo" en la página 289 y "Procesamiento de respuestas largas" en la página 279 .

getFilteredCIsByType

El método `getFilteredCIsByType` recupera los CI del tipo especificado que cumplen las condiciones usadas por el método. Una condición consta de:

- un campo de nombre que contiene el nombre de una propiedad
- un campo de operador que contiene un operador de comparación
- un campo de valor opcional que contiene un valor o lista de valores

Juntos, forman una expresión booleana:

```
<elemento>.property.value [operator] <condición>.value
```

Por ejemplo, si el nombre de la condición es `root_actualdeletionperiod`, el valor de la condición es `40` y el operador es `Igual`, la instrucción booleana es:

```
<elemento>.root_actualdeletionperiod.value = = 40
```

La consulta devuelve todos los elementos cuyo `root_actualdeletionperiod` es `40`, asumiendo que no hay otras condiciones.

Si el argumento `conditionsLogicalOperator` es `AND`, la consulta devuelve los elementos que cumplen todas las condiciones de la colección `conditions`. Si `conditionsLogicalOperator` es `OR`, la consulta devuelve los elementos que cumplen al menos una de las condiciones de la colección `conditions`.

La siguiente tabla muestra los operadores de comparación:

Operador	Tipo de condición/comentarios
ChangedDuring	<p>Fecha</p> <p>Esta es una comprobación del intervalo. El valor de la condición se especifica en horas. Si el valor de la propiedad de fecha se encuentra en el intervalo del tiempo en que se invoca el método más o menos el valor de la condición, la condición es true.</p> <p>Por ejemplo, si el valor de la condición es 24, la condición es true si el valor de la propiedad de fecha se encuentra entre ayer a esta hora y mañana a esta hora.</p> <p>Nota: El nombre <code>ChangedDuring</code> se mantiene para conservar la compatibilidad con versiones anteriores. En versiones anteriores, el operador se ha usado solo con las propiedades de tiempo de creación y modificación.</p>
Igual	Cadena y numérica
EqualIgnoreCase	Cadena
Mayor que	Numérica
GreaterEqual	Numérica

Operador	Tipo de condición/comentarios
En	<p>Cadena, numérica y lista</p> <p>El valor de la condición es una lista. La condición es true si el valor de la propiedad es uno de los valores de la lista.</p>
InList	<p>Lista</p> <p>El valor de la condición y el valor de la propiedad son listas.</p> <p>La condición es true si todos los valores de la lista de condiciones también aparece en la lista de propiedades del elemento. Puede haber más valores de propiedad que los especificados en la condición sin que afecte a la verdad de la condición.</p>
IsNull	<p>Cadena, numérica y lista</p> <p>La propiedad del elemento no tiene ningún valor. Cuando se usa el operador <code>IsNull</code>, el valor de la condición se ignora y en algunos casos puede ser nulo.</p>
Menos	Numérica
LessEqual	Numérica
Como	<p>Cadena</p> <p>El valor de la condición es una subcadena del valor del valor de la propiedad. El valor de la condición debe ir rodeado con signos de porcentaje (%). Por ejemplo, <code>%Bi%</code> coincide con <code>Bismark</code> y <code>Bay of Biscay</code>, pero no con <code>biscuit</code>.</p>
LikeIgnoreCase	<p>Cadena</p> <p>Utilice el operador <code>LikeIgnoreCase</code> de la misma forma que usa el operador <code>Like</code>. Sin embargo, la coincidencia no distingue entre mayúsculas y minúsculas. Por tanto, <code>%Bi%</code> coincide con <code>biscuit</code>.</p>
NotEqual	Cadena y numérica

Operador	Tipo de condición/comentarios
UnchangedDuring	<p>Fecha</p> <p>Esta es una comprobación del intervalo. El valor de la condición se especifica en horas. Si el valor de la propiedad de fecha se encuentra en el intervalo del tiempo en que se invoca el método más o menos el valor de la condición, la condición es false. Si se encuentra fuera de ese intervalo, la condición es true.</p> <p>Por ejemplo, si el valor de la condición es 24, la condición es true si el valor de la propiedad de fecha se encuentra antes de ayer a esta hora y después de mañana a esta hora.</p> <p>Nota: El nombre <code>UnchangedDuring</code> se mantiene para conservar la compatibilidad con versiones anteriores. En versiones anteriores, el operador se ha usado solo con las propiedades de tiempo de creación y modificación.</p>

Ejemplo de configuración de una condición:

```
FloatCondition fc = new FloatCondition();
FloatProp fp = new FloatProp();
fp.setName("attr_name");
fp.setValue(11f);
fc.setCondition(fp);
fc.setFloatOperator(FloatCondition.FloatOperator.EQUAL);
```

Ejemplo de consulta para propiedades heredadas:

El CI de destino es `sample` que tiene dos atributos, `name` y `size`. `sampleII` amplía el CI con dos atributos, `level` y `grade`. Este ejemplo configura una consulta para las propiedades de `sampleII` que se heredaron de `sample` especificándolas por nombre.

```
GetFilteredCIsByType request = new GetFilteredCIsByType()
request.setCmdbContext(cmdbContext)
request.setType("sampleII");
CustomProperties customProperties = new CustomProperties();
PropertiesList propertiesList = new PropertiesList();
propertiesList.setPropertyNames(Arrays.asList("name", "size"));
customProperties.setPropertiesList(propertiesList);
request.setProperties(customProperties);
```

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285 .
type	El nombre de la clase. Para obtener más información, consulte "Nombre de tipo" en la página 287 . El tipo puede ser cualquiera de los tipos definidos usando el Administrador de tipos de CI. Para obtener más información, consulte "Administrador de tipos de CI" en HP Universal CMDB – Guía de modelado .
properties	Los datos que se van a devolver en cada CI (conocidos como diseño de consulta en la interfaz de usuario). Para obtener más información, consulte "CustomProperties" en la página 281 .
conditions	Una colección de pares nombre-valor y los operadores que están relacionados entre sí. Por ejemplo, host_hostname like QA.
conditionsLogicalOperator	<ul style="list-style-type: none">• AND. Todas las condiciones deben cumplirse.• OR. Al menos una de las condiciones debe cumplirse.

Salida

Parámetro	Comentario
CI	Colección de elementos de CI.
chunkInfo	Para obtener más información, consulte "ChunkInfo" en la página 289 y "Procesamiento de respuestas largas" en la página 279 .

getQueryNameOfView

El método `getQueryNameOfView` recupera el nombre del TQL en el que está basada la vista especificada.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285 .
viewName	El nombre de una vista, que es un subconjunto del modelo de clase de CMDB.

Salida

Parámetro	Comentario
queryName	El nombre de TQL en CMDB en el que está basada la vista.

getTopologyQueryExistingResultByName

El método `getTopologyQueryExistingResultByName` recupera el resultado más reciente de la ejecución del TQL especificado. La llamada no ejecuta el TQL. Si no hay resultados de una ejecución anterior, no se devuelve nada.

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte "CmdbContext" en la página 285 .
<code>queryName</code>	El nombre de un TQL.
<code>queryTypedProperties</code>	Una colección de conjuntos de propiedades para recuperar elementos de un tipo de elemento de configuración específico.

Salida

Parámetro	Comentario
<code>topologyMap</code>	Para obtener más información, consulte "TopologyMap" en la página 288 .
<code>chunkInfo</code>	Para obtener más información, consulte "ChunkInfo" en la página 289 y "Procesamiento de respuestas largas" en la página 279 .

getTopologyQueryResultCountByName

El método `getTopologyQueryResultCountByName` recupera el número de instancias de cada nodo que coincida con la consulta especificada.

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte "CmdbContext" en la página 285 .
<code>queryName</code>	El nombre de un TQL.
<code>countInvisible</code>	Si es true, la salida incluye los CI definidos como invisible en la consulta.

Salida

Parámetro	Comentario
<code>getTopologyQueryResultCountByNameResponse</code>	Número de instancias coincidentes con la consulta.

pullTopologyMapChunks

El método `pullTopologyMapChunks` recupera uno de los fragmentos que contienen la respuesta a un método.

Cada fragmento contiene un elemento `topologyMap` que forma parte de la respuesta. El primer fragmento se numera en 1, por tanto el contador del bucle de recuperación se repite de 1 a *<objeto de respuesta>.getChunkInfo().getNumberOfChunks()*.

Para obtener más información, consulte ["ChunkInfo" en la página 289](#) y ["Consulta de CMDB" en la página 278](#).

La aplicación cliente debe ser capaz de gestionar los mapas parciales.

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte "CmdbContext" en la página 285 .
<code>ChunkRequest</code>	El número del fragmento que se va a recuperar y el <code>ChunkInfo</code> devuelto por el método de consulta.
<code>queryTypedProperties</code>	Colección de conjuntos de propiedades para recuperar elementos de un tipo de CI determinado.

Salida

Parámetro	Comentario
<code>topologyMap</code>	Para obtener más información, consulte "TopologyMap" en la página 288 .
<code>comments</code>	Solo para uso interno.

Ejemplo del manejo de fragmentos:

```
GetCIsByType request =
    new GetCIsByType(cmdbContext, typeName, customProperties);
GetCIsByTypeResponse response =
    ucmdbService.getCIsByType(request);
ChunkRequest chunkRequest = new ChunkRequest();
chunkRequest.setChunkInfo(response.getChunkInfo());
for(int j=1; j<=response.getChunkInfo().getNumberOfChunks(); j++){
    chunkRequest.setChunkNumber(j);
    PullTopologyMapChunks req =new PullTopologyMapChunks(cmdbContext,chunkRe
    quest);
    PullTopologyMapChunksResponse res =
        ucmdbService.pullTopologyMapChunks(req);
    for(int m=0 ;
        m < res.getTopologyMap().getCINodes().sizeCINodeList() ;
```

```
        m++) {
        CIs cis =
        res.getTopologyMap().getCINodes().getCINode(m).getCIs();
        for(int i=0 ; i < cis.sizeCICollection() ; i++) {
            // your code to process the CIs
        }
    }
}

GetCIsByType request =
    new GetCIsByType(cmdbContext, typeName, customProperties);
GetCIsByTypeResponse response =
    ucmdbService.getCIsByType(request);
ChunkRequest chunkRequest = new ChunkRequest();
chunkRequest.setChunkInfo(response.getChunkInfo());
for(int j=1 ; j <= response.getChunkInfo().getNumberOfChunks() ; j++) {
    chunkRequest.setChunkNumber(j);
    PullTopologyMapChunks req = new PullTopologyMapChunks(cmdbContext, chunk
Request);
    PullTopologyMapChunksResponse res =
        ucmdbService.pullTopologyMapChunks(req);
    for(int m=0 ;
        m < res.getTopologyMap().getCINodes().getCINodes().size();
        m++) {
        CIs cis =
        res.getTopologyMap().getCINodes().getCINodes().get(m).getCIs();
        for(int i=0 ; i < cis.getCIs().size(); i++) {
            // your code to process the CIs
        }
    }
}
```

releaseChunks

El método `releaseChunks` libera la memoria de los fragmentos que contienen datos de la consulta.

Sugerencia: El servidor descarta los datos tras diez minutos. Llamar a este método para descartar los datos tan pronto como se han leído conserva los recursos del servidor.

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte "CmdbContext" en la página 285 .
<code>chunksKey</code>	El identificador de los datos en el servidor que se ha fragmentado. La clave es un elemento de <code>ChunkInfo</code> .

Métodos de actualización de UCMDB

En esta sección se proporciona información sobre los siguientes métodos:

- ["addCIsAndRelations" abajo](#)
- ["addCustomer" en la página siguiente](#)
- ["deleteCIsAndRelations" en la página siguiente](#)
- ["removeCustomer" en la página 303](#)
- ["updateCIsAndRelations" en la página 303](#)

addCIsAndRelations

El método `addCIsAndRelations` agrega o actualiza los CI y las relaciones.

En el caso de CI o relaciones que no existen en CMDB, se agregan y sus propiedades se establecen de acuerdo con el contenido del argumento `CIsAndRelationsUpdates`.

Si los CI o relaciones existen en CMDB, se actualizan con los nuevos datos, si `updateExisting` es **true**.

Si `updateExisting` es **false**, `CIsAndRelationsUpdates` no puede hacer referencia a elementos de configuración o relaciones existentes. Cualquier intento de hacer referencia a elementos existentes cuando `updateExisting` es falso da como resultado una excepción.

Si `updateExisting` es **true**, la operación de adición o de actualización se realiza sin validar los CI, con independencia del valor de `ignoreValidation`.

Si `updateExisting` es **false** e `ignoreValidation` es **true**, la operación de adición se realiza sin validar los CI.

Si `updateExisting` es **false** e `ignoreValidation` es **true**, la operación de adición se realiza sin validar los CI.

Las relaciones nunca se validan.

`CreatedIDsMap` es un mapa o diccionario de tipo `ClientIDToCmdbID` que conecta los Id. temporales del cliente con los Id. de CMDB reales correspondientes.

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte "CmdbContext" en la página 285 .
<code>updateExisting</code>	Establézcalo en true para actualizar los elementos que ya existen en CMDB. Establézcalo en false para lanzar una excepción si ya existe un elemento

Parámetro	Comentario
CIsAndRelationsUpdates	Los elementos que se van a actualizar o crear. Para obtener más información, consulte " CIsAndRelationsUpdates " en la página 282.
ignoreValidation	Si es true, no se realiza ninguna comprobación antes de actualizar CMDB.
dataStore.	Información del convertidor.

Salida

Parámetro	Comentario
createdIDsMapList	Lista de los Id. de cliente asignados a los Id. de CMDB. Para obtener más información, consulte la descripción anterior.
comments	Solo para uso interno.

addCustomer

El método `addCustomer` agrega un cliente.

Entrada

Parámetro	Comentario
CustomerID	El Id. numérico del cliente.

deleteCIsAndRelations

El método `deleteCIsAndRelations` elimina los elementos de configuración y relaciones específicos de CMDB.

Cuando se elimina un CI y este es un extremo de uno o varios elementos `Relation`, estos elementos `Relation` también se eliminan.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
CIsAndRelationsUpdates	Los elementos que se van a eliminar. Para obtener más información, consulte " CIsAndRelationsUpdates " en la página 282.
dataStore.	Información del convertidor.

removeCustomer

El método `removeCustomer` elimina un registro del cliente.

Entrada

Parámetro	Comentario
CustomerID	El Id. numérico del cliente.

updateCIsAndRelations

El método `updateCIsAndRelations` actualiza los CI y relaciones especificados.

La actualización usa los valores de propiedad desde el argumento `CIsAndRelationsUpdates`. Si alguna de los CI o relaciones no existe en CMDB, se lanza una excepción.

`CreatedIDsMap` es un mapa o diccionario de tipo `ClientIDToCmdbID` que conecta los Id. temporales del cliente con los Id. de CMDB reales correspondientes.

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte " CmdbContext " en la página 285 .
<code>CIsAndRelationsUpdates</code>	Los elementos que se van a actualizar. Para obtener más información, consulte " CIsAndRelationsUpdates " en la página 282 .
<code>ignoreValidation</code>	Si es true, no se realiza ninguna comprobación antes de actualizar CMDB.
<code>dataStore</code>	Información del convertidor.

Salida

Parámetro	Comentario
<code>createdIDsMapList</code>	Lista de los Id. de cliente asignados a los Id. de CMDB. Para obtener más información, consulte " addCIsAndRelations " en la página 301 .

Métodos de análisis de impacto de UCMDB

En esta sección se proporciona información sobre los siguientes métodos:

- "[calculateImpact](#)" en la [página siguiente](#)
- "[getImpactPath](#)" en la [página siguiente](#)

- ["getImpactRulesByNamePrefix" en la página siguiente](#)

calculateImpact

El método `calculateImpact` calcula los CI que están afectados por un CI determinado de acuerdo con las reglas definidas en CMDB.

Esto muestra el efecto de un evento que activa la regla. La salida `identifier` de `calculateImpact` se usa como entrada para ["getImpactPath" abajo](#).

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte "CmdbContext" en la página 285 .
<code>impactCategory</code>	El tipo de evento que podría activar la regla que se está simulando.
<code>IDs</code>	Una colección de elementos de <code>Id</code> .
<code>impactRulesNames</code>	Una colección de elementos de <code>ImpactRuleName</code> .
<code>severity</code>	La gravedad del evento de activación.

Salida

Parámetro	Comentario
<code>impactTopology</code>	Para obtener más información, consulte "Topology" en la página 288 .
<code>identifier</code>	La clave de la respuesta del servidor.

getImpactPath

El método `getImpactPath` recupera el gráfico de topología de la ruta entre el CI afectado y el CI que lo afecta.

La salida `identifier` de ["calculateImpact" arriba](#) se usa como argumento de entrada de `identifier` de `getImpactPath`.

Entrada

Parámetro	Comentario
<code>cmdbContext</code>	Para obtener más información, consulte "CmdbContext" en la página 285 .
<code>identifier</code>	La clave a la respuesta del servidor que devolvió <code>calculateImpact</code> .
<code>relation</code>	Relación basada en uno de los "ShallowRelation" devueltos por <code>calculateImpact</code> en el elemento <code>impactTopology</code> .

Salida

Parámetro	Comentario
impactPathTopology	Una colección de CIs y una colección de ImpactRelations.
comments	Solo para uso interno.

Un elemento ImpactRelations consta de un ID, type, end1ID, end2ID, rule y action.

getImpactRulesByNamePrefix

El método `getImpactRulesByNamePrefix` recupera las reglas usando un filtro de prefijo.

Este método se aplica a las reglas de impacto que se denominan con un prefijo que indica el contexto al que se aplican, por ejemplo, `SAP_myrule`, `ORA_myrule`, etc. Este método filtra todos los nombres de las reglas de impacto para los que comienzan con el prefijo especificado por el argumento `ruleNamePrefixFilter`.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
ruleNamePrefixFilter	Una cadena que contiene las primeras letras de los nombres de reglas que van a coincidir.

Salida

Parámetro	Comentario
impactRules	<code>impactRules</code> está compuesta por cero o varias <code>impactRule</code> . Una <code>impactRule</code> , que especifica el efecto de un cambio, está compuesta de <code>ruleName</code> , <code>description</code> , <code>queryName</code> e <code>isActive</code> .

API del servicio web de estado real

La API del servicio web de estado real la utiliza principalmente el Administrador de servicio para recuperar información de estado real para un Id. de CMDB o Id. global y un Id. de cliente específico. La API busca una consulta coincidente en la carpeta **Integration/SM Query** y ejecuta el TQL con el Id. de CMDB o el Id. global como condición, y devuelve la salida de la consulta.

URL de servicio web: `http://[nombre_equipo]:8080/axis2/services/ucmdbSMSservice`

Esquema de servicio web: `http://[nombre_equipo]:8080/axis2/services/ucmdbSMSservice?xsd=xsd0`

Flujo

Cuando se llama al método API, este intenta encontrar una consulta adecuada en la carpeta **Integration/SM Query**. Intenta primero que coincida el tipo del CMDBID/GlobalID con una de las

consultas de la última carpeta; para ello, busque un **QueryElement** con el nombre **Root**, y si no lo encuentra, intenta utilizar cualquier **QueryNode** del mismo tipo que el CMDBID/GlobalID solicitado. Una vez que se han encontrado un Query y QueryNode adecuados, pone CMDBID/GlobalID como condición en QueryNode y ejecuta la consulta. A continuación, el resultado es devuelto al llamante de la API.

Manipulación del resultado mediante transformaciones

En algunos casos, quizá quiera aplicar transformaciones adicionales al XML resultante (por ejemplo, para sumar los tamaños de todos los discos y agregar esa suma como un atributo adicional al CI). Para agregar transformaciones adicionales en los resultados de TQL, ponga un recurso denominado **[tql_name].xslt** en la configuración del adaptador como se indica a continuación: **Administración de adaptador > ServiceDeskAdapter7-1 > Archivos de configuración > [tql_name].xslt**.

Registros de la API del servicio web de estado real

La configuración de registro para UCMDB reside en: **UCMDBServer/Conf/log** en los diversos archivos ***.properties**.

Para ver registros del flujo de Estado actual de SM:

1. Abra el archivo **cmdb_soaapi.properties** y cambie el nivel de registro a DEBUG de la manera siguiente: **loglevel=DEBUG**.
2. Abra el archivo **fcmdb.properties** y cambie el nivel de registro a DEBUG de la manera siguiente: **loglevel=DEBUG**.
3. Espere 1 minuto a que el servidor recupere los cambios.
4. Ejecute el Estado actual de SM.
5. Vea los archivos de registro siguientes en **UCMDBServer/Runtime/log**:
 - **cmdb.soaapi.log**
 - **fcmdb.log**

Habilitación del estado real de los CI replicados después de cambiar el contexto raíz

Si ha cambiado el contexto raíz que se utiliza para acceder a UCMDB, debe realizar los cambios de configuración siguientes para habilitar el estado real de los CI replicados:

1. En **UCMDBServer\deploy\axis2\WEB-INF**, abra el archivo **web.xml**.
2. Agregue el parámetro **servlet init** siguiente a AxisServlet (pegue estas cuatro líneas después de la línea 28):

```
<init-param>  
<param-name>axis2.find.context</param-name>
```

```
<param-value>false</param-value>  
</init-param>
```

Esta configuración impide que Axis2 intente calcular la raíz de contexto y le indica que la busque explícitamente en **axis2.xml**.

3. En **UCMDBServer\deploy\axis2\WEB-INF\conf**, abra el archivo **axis2.xml**.
4. En la línea 58, elimine comentarios del parámetro **contextRoot** y edítelo de la manera siguiente:

```
<parameter name="contextRoot" locked="false">test/axis2</parameter>  
(donde test es el nuevo contexto raíz en cmdb.xml).
```

Nota: No hay ninguna barra al inicio de **test/axis2**.

Casos de uso de la API de los servicios web de UCMDB

Los casos de uso siguientes asumen dos sistemas:

- HP Universal CMDB
- Un sistema de terceros que contiene un repositorio de elementos de configuración

Esta sección incluye los siguientes temas:

- ["Rellenado de CMDB" abajo](#)
- ["Consultas en CMDB" en la página siguiente](#)
- ["Consulta del modelo de clase" en la página siguiente](#)
- ["Análisis del impacto de cambio" en la página siguiente](#)

Rellenado de CMDB

Casos de uso:

- Una gestión de activos de terceros actualiza CMDB con información disponible solo en la gestión de activos
- Un número de sistemas de terceros pueden rellenar CMDB para crear un CMDB central que puede realizar el seguimiento de los cambios y realizar el análisis de impacto.
- Un sistema de terceros crea elementos de configuración y relaciones de acuerdo con la lógica empresarial de un tercero, para aprovechar las capacidades de consulta de CMDB

Consultas en CMDB

Casos de uso:

- Un sistema de terceros obtiene los elementos de configuración y las relaciones que representan el sistema SAP obteniendo los resultados de SAP TQL
- Un sistema de terceros obtiene la lista de servidores Oracle que se han agregado o modificado en las últimas cinco horas
- Un sistema de terceros obtiene la lista de servidores cuyo nombre de host contiene la subcadena *lab*
- Un sistema de terceros encuentra los elementos relacionados con un CI determinado obteniendo sus vecinos

Consulta del modelo de clase

Casos de uso:

- Un sistema de terceros permite a los usuarios especificar el conjunto de datos que se va a recuperar desde CMDB. Una interfaz de usuario se puede construir en el modelo de clase para mostrar a los usuarios las posibles propiedades y solicitarles los datos requeridos. El usuario puede elegir después la información que se va a recuperar.
- Un sistema de terceros explora el modelo de clase cuando el usuario no puede acceder a la interfaz de usuario de UCMDB.

Análisis del impacto de cambio

Caso de uso:

Un sistema de terceros produce una lista de los servicios empresariales que podrían verse impactados por un cambio en un host especificado.

Ejemplos

Vea los ejemplos de código siguientes:

- The Example Base Class
- Query Example
- Update Example
- Class Model Example
- Impact Analysis Example

Estos archivos están ubicados en el siguiente directorio:

```
\\<UCMDB root directory>\hp\UCMDB\UCMDBServer\deploy\ucmdb-docs\docs\eng\APIs\WebServiceAPI_Samples\
```

Capítulo 11: API de Java de Administración de Data Flow

Este capítulo incluye:

Uso de la API de Java de Administración de Data Flow 309

Uso de la API de Java de Administración de Data Flow

Nota: Utilice este capítulo junto con el documento de las API de Java de DFM, que se encuentra disponible en la biblioteca de documentación en línea.

Este capítulo explica cómo las herramientas de terceros o personalizadas pueden utilizar las API de Java de la Administración de Data Flow de HP para administrar Data Flow. La API proporciona métodos para:

- **Administrar credenciales.** Ver, agregar, actualizar y quitar.
- **Administrar trabajos.** Ver estado, activar y desactivar.
- **Administrar intervalos de sondas.** Ver, agregar y actualizar.
- **Administrar activadores.** Agregar o quitar un CI de activación, y agregar, quitar o deshabilitar una consulta de activación
- **Datos generales de la vista.** Datos sobre dominios y sondas.

Los siguientes servicios están disponibles en el paquete Discovery Services:

- **DDMConfigurationService.** Servicios para configurar las Data Flow Probes, los clústeres, los intervalos de IP y las credenciales. El servidor Universal Discovery puede configurarse con un archivo XML o mediante Data Flow Probe.
- **DDMManagementService.** Servicios para analizar y ver el progreso, los resultados y los errores de la ejecución de Universal Discovery.
- **DDMSoftwareSignatureService.** Servicios para definir los elementos de software que deben detectar los componentes de Data Flow Probe. Las definiciones son para todo el sistema. Si se han definido varias Data Flow Probes, las definiciones se aplican a todas ellas.
- **DDMZoneService.** Servicios para gestionar la detección basada en zonas.

Además de estos servicios, hay API de cliente de la Administración de Data Flow que se utilizan para crear adaptadores de Jython. Para obtener más información, consulte "[Desarrollo de los adaptadores Jython](#)" en la página 39.

Permisos

El administrador proporciona credenciales de inicio de sesión para conectarse con la API. El cliente de la API necesita el nombre de usuario y la contraseña de un usuario de integración definido en CMDB. Estos usuarios no representan usuarios humanos de CMDB, sino más bien aplicaciones que se conectan a CMDB.

Además, el usuario debe tener el permiso de acción general **Acceso a SDK** para poder iniciar sesión.

Precaución: El cliente de la API puede trabajar también con usuarios normales mientras estos tengan permiso de autenticación de API. Sin embargo, no se recomienda esta opción.

Para obtener más información, consulte "[Creación de un usuario de integración](#)" en la página 269.

Capítulo 12: API de servicio web de Administración de Data Flow

Este capítulo incluye:

Información general de la API de servicio web de Administración de Data Flow	311
Convenciones	312
Llamar al servicio web	312
Métodos y estructuras de datos de Administración de Data Flow	312
Ejemplo de código	323
Ejemplo de adición de credenciales	326

Información general de la API de servicio web de Administración de Data Flow

Este capítulo explica cómo las herramientas de terceros o personalizadas pueden utilizar la API de servicio web de Administración de Data Flow de HP para administrar Data Flow.

La API de servicio web de Administración de Data Flow Management de HP se utiliza para integrar aplicaciones con HP Universal CMDB. La API proporciona métodos para:

- **Administrar credenciales.** Ver, agregar, actualizar y quitar.
- **Administrar trabajos.** Ver estado, activar y desactivar.
- **Administrar intervalos de sondas.** Ver, agregar y actualizar.
- **Administrar activadores.** Agregar o quitar un CI de activación, y agregar, quitar o deshabilitar una consulta de activación
- **Datos generales de la vista.** Datos sobre dominios y sondas.

Los usuarios de la API del servicio web de HP Data Flow Management deberían estar familiarizados con:

- La especificación SOAP
- Un lenguaje de planificación orientado a objetos, como C++, C# o Java
- HP Universal CMDB
- Administración de Data Flow

El usuario debe tener el permiso de acción general **Ejecutar API heredada** para poder iniciar la sesión.

Para obtener toda la información sobre las operaciones disponibles, consulte *HP Universal Discovery Schema Reference*. Estos archivos están ubicados en la carpeta siguiente:

`<directorio raíz de UC MDB>\UCMDBServer\deploy\lucmdb-docs\docs\eng\APIs\DDM_Schema\webframe.html`

Convenciones

Este capítulo utiliza las convenciones siguientes:

- Este estilo `Element` indica que un elemento es una entidad de la base de datos o un elemento definido en el esquema, incluidas las estructuras pasadas a, o devueltas por métodos. El texto sin formato indica que el elemento se analiza en un contexto general.
- Los elementos de Administración de Data Flow y los argumentos del método se deletrean en caso de que sean específicos en el esquema. Esto significa generalmente que un nombre de clase o una referencia genérica a una instancia de la clase se escribe con mayúscula inicial. Los elementos o argumentos a un método no se escriben en mayúsculas. Por ejemplo, `credential` es un elemento del tipo `Credential` pasado a un método.

Llamar al servicio web

La API de servicio web de Administración de Data Flow de HP permite llamar a métodos del servidor mediante técnicas de planificación SOAP estándar. Si la instrucción no se puede analizar o si hay un problema al invocar al método, los métodos de la API lanzan una excepción `SoapFault`. Cuando se lanza una excepción `SoapFault`, el servicio rellena uno o varios de los campos de mensaje de error, código de error y mensaje de excepción. Si no hay ningún error, se devuelven los resultados de la invocación.

Para llamar al servicio, utilice:

- Protocol: `http` o `https` (en función de la configuración del servidor)
- Dirección URL: `<Procesamiento de datos BSM>:8080/axis2/services/DiscoveryService`
- Contraseña predeterminada: `"admin"`
- Nombre de usuario predeterminado: `"admin"`

Los programadores de SOAP pueden acceder a WSDL en:

- `axis2/services/DiscoveryService?wsdl`

Métodos y estructuras de datos de Administración de Data Flow

Esta sección enumera los métodos y estructuras de datos de la API de servicio web de Administración de Data Flow y proporciona un breve resumen de sus usos. Para consultar la

documentación completa de la petición y la respuesta a cada operación, consulte *HP Discovery and Dependency Mapping Schema Reference*.

Esta sección incluye los siguientes temas:

- ["Estructuras de datos" abajo](#)
- ["Administración de los métodos de trabajo de detección" en la página siguiente](#)
- ["Gestión de métodos de activación" en la página 315](#)
- ["Métodos de datos de sondas y dominios" en la página 317](#)
- ["Métodos de datos de credenciales" en la página 319](#)
- ["Métodos de actualización de datos" en la página 321](#)

Estructuras de datos

Estas son algunas de las estructuras de datos que se utilizan en la API de servicio web de Data Flow Management.

CIProperties

CIProperties es una colección de colecciones. Cada colección contiene propiedades de un tipo de datos diferente. Por ejemplo, puede haber una colección dateProps, una colección strListProps, una colección xmlProps, etc.

Cada colección de tipos contiene propiedades individuales de ese tipo concreto. Los nombres de estos elementos de propiedades son los mismos que para el contenedor, pero en singular. Por ejemplo, dateProps contiene elementos de dateProp. Cada propiedad es un par de nombre-valor.

Consulte CIProperties en *HP Universal Discovery Schema Reference*.

IPList

Lista de elementos de IP, cada uno de los cuales contiene una dirección IPv4 o IPv6.

Consulte IPList en *HP Universal Discovery Schema Reference*.

IPRange

Un IPRange tiene dos elementos: Start y End. Cada uno de ellos contiene un elemento Address, que es una dirección IPv4 o IPv6.

Consulte IPRange en *HP Universal Discovery Schema Reference*.

Scope

Dos IPRanges. Exclude es una colección de IPRanges que se deben excluir del trabajo. Include es una colección de IPRanges que se deben incluir en el trabajo.

Consulte Scope en *HP Universal Discovery Schema Reference*.

Administración de los métodos de trabajo de detección

activateJob

Activa el trabajo especificado.

Consulte "[Ejemplo de código](#)" en la página 323.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	El nombre del trabajo.

deactivateJob

Desactiva el trabajo especificado.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	El nombre del trabajo.

dispatchAdHocJob

Distribuye un trabajo en la sonda ad hoc. El trabajo debe estar activo y contener el CI de activación especificado.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	El nombre del trabajo.
CIID	El Id. del CI de activación.
ProbeName	El nombre de la sonda.
Tiempo de espera	En milisegundos

getDiscoveryJobsNames

Devuelve la lista de nombres de trabajos.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.

Salida

Parámetro	Comentario
strList	La lista de nombres de trabajos.

isJobActive

Comprueba si el trabajo está activo.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	El nombre del trabajo que se va a comprobar.

Salida

Parámetro	Comentario
JobState	Es verdadero (true) si el trabajo está activo.

Gestión de métodos de activación

addTriggerCI

Agrega un nuevo CI de activación al trabajo especificado.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	El nombre del trabajo.
CIID	El Id. del CI de activación.

addTriggerTQL

Agrega un nuevo TQL de activación al trabajo especificado.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	El nombre del trabajo.
TqlName	El nombre del TQL que se va a agregar.

disableTriggerTQL

Evita que el TQL active el trabajo, pero no lo elimina de manera permanente de la lista de consultas que activan el trabajo.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	El nombre del trabajo.

removeTriggerCI

Suprime el CI especificado de la lista de CI que activan el trabajo.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	El nombre del trabajo.
CIID	El Id. del CI de activación.

removeTriggerTQL

Suprime el TQL especificado de la lista de consultas que activan el trabajo.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	Colección de nombres de trabajos que se van a comprobar.
CIID	El Id. del TQL que va a eliminarse.

setTriggerTQLProbesLimit

Restringe las sondas en las que TQL está activo en el trabajo a la lista especificada.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
JobName	El nombre del trabajo.
TqlName	El nombre del TQL.
probesLimit	La lista de las sondas para las que está activo el TQL.

Métodos de datos de sondas y dominios

getDomainType

Devuelve el tipo de dominio.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
domainName	El nombre del dominio.

Salida

Parámetro	Comentario
domainType	El tipo de dominio.

getDomainsNames

Devuelve los nombres de los dominios actuales.

Consulte "[Ejemplo de código](#)" en la página 323.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.

Salida

Parámetro	Comentario
domainNames	La lista de nombres de dominios.

getProbeIPs

Devuelve las direcciones IP de la sonda especificada.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285.
domainName	El dominio que se va a comprobar.
probeName	El nombre de la sonda usada en ese dominio.

Salida

Parámetro	Comentario
probelPs	La "IPList" de las direcciones de la sonda.

getProbesNames

Devuelve los nombres de las sondas en el dominio especificado.

Consulte ["Ejemplo de código" en la página 323.](#)

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285.
domainName	El dominio que se va a comprobar.

Salida

Parámetro	Comentario
probesName	La lista de sondas del dominio.

getProbeScope

Devuelve la definición del ámbito de la sonda especificada.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285.
domainName	El dominio que se va a comprobar.
probeName	El nombre de la sonda.

Salida

Parámetro	Comentario
probeScope	El "Scope" de la sonda.

isProbeConnected

Comprueba si la sonda especificada está conectada.

Consulte ["Ejemplo de código" en la página 323](#).

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285 .
domainName	El dominio que se va a comprobar.
probeName	La sonda que se va a comprobar.

Salida

Parámetro	Comentario
isConnected	True si la sonda está conectada.

updateProbeScope

Establece el ámbito de la sonda especificada, reemplazando el ámbito existente.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285 .
domainName	El dominio.
probeName	La sonda que se va a actualizar.
newScope	El "Scope" que se establecerá para la sonda.

Métodos de datos de credenciales

addCredentialsEntry

Agrega una entrada de credenciales al protocolo especificado para el dominio especificado.

Consulte ["Ejemplo de código" en la página 323](#).

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte "CmdbContext" en la página 285 .

Parámetro	Comentario
domainName	El dominio que se va a actualizar.
protocolName	El nombre del protocolo.
credentialsEntryParameters	La colección " CIProperties " de las nuevas credenciales.

Salida

Parámetro	Comentario
credentialsEntryID	El Id. de CI de la nueva entrada de credencial.

getCredentialsEntriesIDs

Devuelve los Id. de las credenciales definidas para el protocolo especificado.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
domainName	El dominio para el que se van a conseguir las credenciales.
protocolName	El nombre de un protocolo usado en ese dominio.

Salida

Parámetro	Comentario
credentialsEntryIDs	La lista de Id. de credenciales para el protocolo del dominio.

getCredentialsEntry

Devuelve las credenciales definidas para el protocolo especificado. Los atributos cifrados se devuelven vacíos.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
domainName	El dominio para el que se van a conseguir las credenciales.
protocolName	El nombre de un protocolo usado en ese dominio.
credentialsEntryID	El Id. de credencial que se va a conseguir.

Salida

Parámetro	Comentario
credentialsEntryParameters	La colección " CIProperties " de las credenciales.

removeCredentialsEntry

Elimina las credenciales especificadas del protocolo.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285 .
domainName	El dominio.
protocolName	El nombre de un protocolo usado en el dominio.
credentialsEntryID	El Id. de la credencial que va a eliminarse.

updateCredentialsEntry

Establece nuevos valores para las propiedades de la entrada de credenciales especificada.

Las propiedades existentes se eliminan y estas propiedades se establecen. Las propiedades cuyo valor no se ha establecido en esta llamada quedarán sin definir.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285 .
domainName	El dominio en el que se van a actualizar credenciales.
protocolName	El nombre de un protocolo usado en el dominio.
credentialsEntryID	El Id. de las credenciales que van a actualizarse.
credentialsEntryParameters	La colección " CIProperties " que se establecerá como las propiedades de las credenciales.

Métodos de actualización de datos

rediscoverCIs

Localiza los activadores que han detectado los objetos de CI especificados y vuelve a ejecutar dichos activadores. **rediscoverCIs** se ejecuta de forma asíncrona. Llame a **checkDiscoveryProgress** para determinar si la redetección está completa.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
CmdbIDs	Colección de los Id. de los objetos que se volverán a detectar.

Salida

Parámetro	Comentario
isSucceed	Es verdadero (true) si la nueva detección de los CI se ha realizado correctamente.

checkDiscoveryProgress

Devuelve el progreso de la llamada a **rediscoverCIs** más reciente en los Id. especificados. La respuesta es un valor de 0 a 1. Cuando la respuesta es 1, la llamada a **rediscoverCIs** se ha completado.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
CmdbIDs	Colección de los Id. de los objetos de la llamada para volver a detectar de los cuales se hará un seguimiento.

Salida

Parámetro	Comentario
progress	Un trabajo completado tiene un valor de progress igual a 1. Los trabajos que no se han completado tienen una fracción inferior a 1.

rediscoverViewCIs

Localiza los activadores que han creado los datos para rellenar la vista especificada y vuelve a ejecutar dichos activadores. **rediscoverViewCIs** se ejecuta de forma asíncrona. Llame a **checkViewDiscoveryProgress** para determinar si la redetección está completa.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
viewName	Las vistas que se van a comprobar.

Salida

Parámetro	Comentario
isSucceed	Es verdadero (true) si la nueva detección de los CI se ha realizado correctamente.

checkViewDiscoveryProgress

Devuelve el progreso de la llamada a **rediscoverViewCIs** más reciente en la vista especificada. La respuesta es un valor de 0 a 1. Cuando la respuesta es 1, la llamada a **rediscoverCIs** se completa.

Entrada

Parámetro	Comentario
cmdbContext	Para obtener más información, consulte " CmdbContext " en la página 285.
viewName	La colección de vistas que se van a comprobar.

Salida

Parámetro	Comentario
progress	Un trabajo completado tiene un valor de progress igual a 1. Los trabajos que no se han completado tienen una fracción inferior a 1.

Ejemplo de código

```
import java.net.URL;
import org.apache.axis2.transport.http.HTTPConstants;
import org.apache.axis2.transport.http.HttpTransportProperties;
import com.hp.ucmdb.generated.params.discovery.*;
import com.hp.ucmdb.generated.services.*;
import com.hp.ucmdb.generated.types.*;
public class test {
    static final String HOST_NAME = "<my_hostname>";
    static final int PORT = 8080;
    private static final String PROTOCOL = "http";
    private static final String FILE = "/axis2/services/DiscoveryService";

    private static final String PASSWORD = "<my_password>";
    private static final String USERNAME = "<my_username>";

    private static CmdbContext cmdbContext = new CmdbContext("ws tests");

    public static void main(String[] args) throws Exception {
        // Get the stub object
        DiscoveryService discoveryService = getDiscoveryService();

        // Activate Job
        discoveryService.activateJob(new ActivateJobRequest(
            "Range IPs by ICMP", cmdbContext));

        // Get domain & probes info
```

```
        getProbesInfo(discoveryService);
        // Add credentilas entry for ntcmd protocol
        addNTCMDCredentialsEntry();
    }

public static void addNTCMDCredentialsEntry() throws Exception {
    DiscoveryService discoveryService = getDiscoveryService();

    // Get domain name
    StrList domains =
        discoveryService.getDomainsNames(
            new GetDomainsNamesRequest(cmdbContext)).
            getDomainNames();
    if (domains.sizeStrValueList() == 0) {
        System.out.println("No domains were found, can't create credential
s");
        return;
    }
    String domainName = domains.getStrValue(0);
    // Create propeties with one byte param
    CIProperties newCredsProperties = new CIProperties();

    // Add password property - this is of type bytes
    newCredsProperties.setBytesProps(new BytesProps());
    setPasswordProperty(newCredsProperties);

    // Add user & domain properties - these are of type string
    newCredsProperties.setStrProps(new StrProps());
    setStringProperties("protocol_username", "test user", newCredsPropertie
s);
    setStringProperties("ntadminprotocol_ntdomain",
        "test doamin", newCredsProperties);

    // Add new credentials entry
    discoveryService.addCredentialsEntry(
        new AddCredentialsEntryRequest(domainName,
            "ntadminprotocol", newCredsProperties, cmdbContext));
    System.out.println("new credentials craeted for domain: " + domainName +
" in ntcmd protocol");
}

private static void setPasswordProperty(CIProperties newCredsProperties) {
    BytesProp bProp = new BytesProp();
    bProp.setName("protocol_password");
    bProp.setValue(new byte[] {101,103,102,104});
    newCredsProperties.getBytesProps().addBytesProp(bProp);
}
```

```
private static void setStringProperties(String propertyName, String value, C
IProperties newCredsProperties) {
    StrProp strProp = new StrProp();
    strProp.setName(propertyName);
    strProp.setValue(value);
    newCredsProperties.getStrProps().addStrProp(strProp);
}
```

```
private static void getProbesInfo(DiscoveryService discoveryService) throws
Exception {
    GetDomainsNamesResponse result = discoveryService.getDomainsNames(new Ge
tDomainsNamesRequest(cmdbContext ));
    // Go over all the domains
    if (result.getDomainNames().sizeStrValueList() > 0) {
        String domainName =
            result.getDomainNames().getStrValue(0);
        GetProbesNamesResponse probesResult =
            discoveryService.getProbesNames(
                new GetProbesNamesRequest(domainName, cmdbContext));
        // Go over all the probes
        for (int i=0; i<probesResult.getProbesNames().sizeStrValueList(); i+
+) {
            String probeName = probesResult.getProbesNames().getStrValue(i);
            // Check if connected
            IsProbeConnectedResponse connectedRequest =
                discoveryService.isProbeConnected(
                    new IsProbeConnectedRequest(
                        domainName, probeName, cmdbContext));
            Boolean isConnected = connectedRequest.getIsConnected();
            // Do something ...
            System.out.println("probe " + probeName + " isconnect=" + isConn
ected);
        }
    }
}
```

```
private static DiscoveryService getDiscoveryService() throws Exception {
    DiscoveryService discoveryService = null;
    try {
        // Create service
        URL url = new URL(PROTOCOL,HOST_NAME,PORT, FILE);
        DiscoveryServiceStub serviceStub =
            new DiscoveryServiceStub(url.toString());

        // Authenticate info
```

```
        HttpTransportProperties.Authenticator auth =
            new HttpTransportProperties.Authenticator();
        auth.setUsername(USERNAME);
        auth.setPassword(PASSWORD);
        serviceStub._getServiceClient().getOptions().setProperty(
            HTTPConstants.AUTHENTICATE,auth);

        discoveryService = serviceStub;
    } catch (Exception e) {
        throw new Exception("cannot create a connection to service ", e);
    }
    return discoveryService;
}
}
```

Ejemplo de adición de credenciales

```
import java.net.URL;
import org.apache.axis2.transport.http.HTTPConstants;
import org.apache.axis2.transport.http.HttpTransportProperties;
import com.hp.ucmdb.generated.params.discovery.*;
import com.hp.ucmdb.generated.services.DiscoveryService;
import com.hp.ucmdb.generated.services.DiscoveryServiceStub;
import com.hp.ucmdb.generated.types.BytesProp;
import com.hp.ucmdb.generated.types.BytesProps;
import com.hp.ucmdb.generated.types.CIProperties;
import com.hp.ucmdb.generated.types.CmdbContext;
import com.hp.ucmdb.generated.types.StrList;
import com.hp.ucmdb.generated.types.StrProp;
import com.hp.ucmdb.generated.types.StrProps;

public class test {
    static final String HOST_NAME = "hostname";
    static final int PORT = 8080;
    private static final String PROTOCOL = "http";
    private static final String FILE = "/axis2/services/DiscoveryService";

    private static final String PASSWORD = "admin";
    private static final String USERNAME = "admin";

    private static CmdbContext cmdbContext = new CmdbContext("ws tests");

    public static void main(String[] args) throws Exception {
        // Get the stub object
        DiscoveryService discoveryService = getDiscoveryService();

        // Activate Job
        discoveryService.activateJob(new ActivateJobRequest("Range IPs by ICMP",
            cmdbContext));
    }
}
```

```
        // Get domain & probes info
        getProbesInfo(discoveryService);
        // Add credentilas entry for ntcmd protocol
        addNTCMDCredentialsEntry();
    }

    public static void addNTCMDCredentialsEntry() throws Exception {
        DiscoveryService discoveryService = getDiscoveryService();

        // Get domain name
        StrList domains =
            discoveryService.getDomainsNames(new GetDomainsNamesRequest(cmdbCont
ext)).getDomainNames();
        if (domains.sizeStrValueList() == 0) {
            System.out.println("No domains were found, can't create credential
s");
            return;
        }
        String domainName = domains.getStrValue(0);
        // Create propeties with one byte param
        CIProperties newCredsProperties = new CIProperties();

        // Add password property - this is of type bytes
        newCredsProperties.setBytesProps(new BytesProps());
        setPasswordProperty(newCredsProperties);

        // Add user & domain properties - these are of type string
        newCredsProperties.setStrProps(new StrProps());
        setStringProperties("protocol_username", "test user", newCredsPropertie
s);
        setStringProperties("ntadminprotocol_ntdomain", "test doamin", newCredsP
roperties);

        // Add new credentials entry
        discoveryService.addCredentialsEntry(new AddCredentialsEntryRequest(doma
inName, "ntadminprotocol", newCredsProperties, cmdbContext));
        System.out.println("new credentials craeted for domain: " + domainName +
" in ntcmd protocol");
    }

    private static void setPasswordProperty(CIProperties newCredsProperties) {
        BytesProp bProp = new BytesProp();
        bProp.setName("protocol_password");
        bProp.setValue(new byte[] {101,103,102,104});
        newCredsProperties.getBytesProps().addBytesProp(bProp);
    }

    private static void setStringProperties(String propertyName, String value, C
IProperties newCredsProperties) {
        StrProp strProp = new StrProp();
```

```

        strProp.setName(propertyName);
        strProp.setValue(value);
        newCredsProperties.getStrProps().addStrProp(strProp);
    }

    private static void getProbesInfo(DiscoveryService discoveryService) throws
Exception {
        GetDomainsNamesResponse result = discoveryService.getDomainsNames(new Ge
tDomainsNamesRequest(cmdbContext ));
        // Go over all the domains
        if (result.getDomainNames().sizeStrValueList() > 0) {
            String domainName = result.getDomainNames().getStrValue(0);
            GetProbesNamesResponse probesResult =
                discoveryService.getProbesNames(new GetProbesNamesRequest(domain
Name, cmdbContext));
            // Go over all the probes
            for (int i=0; i<probesResult.getProbesNames().sizeStrValueList(); i+
+) {

                String probeName = probesResult.getProbesNames().getStrValue(i);
                // Check if connected
                IsProbeConnectedResponse connectedRequest =
                    discoveryService.isProbeConnected(new IsProbeConnectedReques
t(domainName, probeName, cmdbContext));
                Boolean isConnected = connectedRequest.getIsConnected();
                // Do something ...
                System.out.println("probe " + probeName + " isconnect=" + isConn
ected);
            }
        }
    }

    private static DiscoveryService getDiscoveryService() throws Exception {
        DiscoveryService discoveryService = null;
        try {
            // Create service
            URL url = new URL(PROTOCOL,HOST_NAME,PORT, FILE);
            DiscoveryServiceStub serviceStub = new DiscoveryServiceStub(url.toSt
ring());

            // Authenticate info
            HttpTransportProperties.Authenticator auth = new HttpTransportProper
ties.Authenticator();
            auth.setUsername(USERNAME);
            auth.setPassword(PASSWORD);
            serviceStub._getServiceClient().getOptions().setProperty(HTTPConstan
ts.AUTHENTICATE,auth);

            discoveryService = serviceStub;
        } catch (Exception e) {
            throw new Exception("cannot create a connection to service ", e);
        }
    }

```

```
        }  
        return discoveryService;  
    }  
} // End class
```

Agradecemos sus comentarios.

Si desea hacer algún comentario sobre este documento, puede ponerse en [contacto con el equipo de documentación](#) por correo electrónico. Si en este sistema está configurado un cliente de correo electrónico, haga clic en el vínculo anterior para abrir una ventana de correo electrónico con la información siguiente en la línea del asunto:

Comentarios sobre Guía de referencia para el desarrollador (Universal CMDB 10.10)

Solo añada sus comentarios al correo electrónico y haga clic en Enviar.

Si no hay disponible ningún cliente de correo electrónico, copie la información anterior en un nuevo mensaje de un cliente de correo web y envíe sus comentarios a sw-doc@hp.com.