

IBM Security Directory Integrator
Versión 7.2.0.1

Guía del usuario



IBM Security Directory Integrator
Versión 7.2.0.1

Guía del usuario



Nota

Antes de utilizar esta información y el producto al que sirve de complemento, lea la información general contenida en la sección "Avisos" en la página 251.

Nota de edición

Nota: Esta edición es aplicable a la versión 7.2.0.1 del programa bajo licencia *IBM Security Directory Integrator* (5724-K74) y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

© Copyright IBM Corporation 2003, 2014.

Contenido

Acerca de esta publicación vii

Acceso a publicaciones y terminología	vii
Accesibilidad	ix
Formación técnica	ix
Información de soporte	x
Declaración de buenas prácticas de seguridad	x

Capítulo 1. Conceptos generales 1

Línea de ensamblaje..	1
Conectores	4
Modalidades del conector	6
modalidad Iterador	7
Modalidad Buscar	9
Modalidad Sólo adición	10
Modalidad Actualizar	10
modalidad Suprimir	12
Modalidad CallReply	13
modalidad Servidor	14
Modalidad Delta	17
Criterios de enlace	21
Funciones	23
Componentes de script	24
Correlaciones de atributos	25
Comportamiento ante un valor nulo	26
Componentes de rama.	29
Salida de una rama (o un bucle o el flujo de la línea de ensamblaje)	31
Analizadores	32
Conversión de la codificación de caracteres.	33
Acceso a sus propias clases Java	33
Creación de instancias de clases con el Editor de configuración.	34
Creación de instancias de las clases en tiempo de ejecución	34
Flujo y complementos de software de una línea de ensamblaje	34
Manejo de la finalización y la limpieza de los errores graves	39
Control del flujo de una línea de ensamblaje	39
Expresiones	40
Expresiones en parámetros de componente	43
Expresiones en los criterios de enlace.	45
Expresiones en ramas, bucles y conmutador/caso	45
Creación de scripts con expresiones	45
Objeto de entrada	46

Capítulo 2. Scripts en IBM Security Directory Integrator 49

Modelo de datos interno: entradas, atributos y valores	50
Trabajo con objetos de entrada jerárquicos	52
Integración de los scripts en su solución.	63
Control de la ejecución con scripts.	64
Utilización de variables	65
Utilización de propiedades	65

Puntos de control para scripts	67
Scripts en una línea de ensamblaje.	67
Componente de script	68
Complementos de software de línea de ensamblaje	68
Ganchos de servidor	68
Cómo llamar a complementos de software de servidor desde un script	71
Acceso a los componentes de la línea de ensamblaje que se encuentran dentro de la misma	71
Pase de parámetros de la línea de ensamblaje	71
TCB (bloque de llamada de tareas)	72
Uso básico.	72
Inicio de una línea de ensamblaje con operaciones	72
Utilización de un acumulador	73
Inhabilitación de los componentes de la línea de ensamblaje.	74
Suministro de una IWE (entrada work inicial)	74
Scripts en un conector	75
Definición de los parámetros internos mediante scripts	75
Scripts en un analizador	76
Java + Script ≠ JavaScript.	76
Representación de datos	76
Llamadas a función ambiguas	77
Datos Char/String en Java frente a series JavaScript	78
Ámbito y nombre de las variables	79
Creación de instancias de una clase de Java	80
Utilización de valores binarios en scripts	80
Utilización de valores de fecha en scripts	81
Utilización de valores de coma flotante en scripts	81

Capítulo 3. Editor de configuración . . . 83

Modelo de proyecto	83
Vista Servidores de IBM Security Directory Integrator	84
El proyecto de IBM Security Directory Integrator	85
Archivos de configuración	86
Creador de proyectos	87
Propiedades y sustitución	88
Modelo de interfaz de usuario	89
Interfaz de usuario	90
Ventana Aplicación	90
Vista Servidores	93
Editor de expresiones	95
Editor de líneas de ensamblaje	97
Opciones de línea de ensamblaje	100
Paneles de componente	107
Vista de documentación del usuario	112
Ventana Ejecutar línea de ensamblaje	114
Correlación de atributos y esquema	115
Correlación de atributos de entrada	120

Correlación de atributos de salida	121
Editor de conectores	122
Creación de un conector	122
Correlaciones de atributos de entrada y salida	123
Complementos de software	124
Conexión	125
Analizador	125
Criterios de enlace	126
Errores de conexión	128
Delta	130
Agrupación	132
Herencia del conector	132
Editor de servidores	133
Editor de esquemas	134
Explorador de datos	135
Explorador de datos genérico	136
Explorador de datos de secuencia	138
Explorador de datos JDBC	138
Explorador de datos LDAP	140
Editor de formularios	142
Asistentes	147
Asistente Importar configuración	147
Nuevo asistente de componente	150
Características del formulario de configuración del conector	155
Ejecución y depuración de líneas de ensamblaje	157
Informes de la línea de ensamblaje	157
Ejecución de la línea de ensamblaje	159
El repetidor y el depurador	162
Depuración de servidor	168
Opciones de ejecución	169
Selección del servidor	170
Soporte de equipo	172
Compartimiento de un proyecto	173
Utilización de un proyecto compartido	175
Vista Problemas	176
Mejoras en JavaScript	177
Terminación de código	177
Coloración de sintaxis	179
Comprobación de la sintaxis	180
Evaluación local	180
Editores externos	181
Registro de soluciones y valores	182
Valores de almacén del sistema	182
Registro cronológico	184
Lápidas	184
Bibliotecas Java	185
Inicio automático	185
Valores de interfaz de la solución	185
Propiedades de servidor	188
Herencia	189
Acciones y enlaces de teclas	190

Capítulo 4. Características de depuración de IBM Security Directory Integrator 193

Recinto de seguridad	193
Registro de entradas de la línea de ensamblaje	194

Reproducción del recinto de seguridad de los registros de la línea de ensamblaje	194
Modalidad de simulación de línea de ensamblaje	195
Flujo de trabajo de línea de ensamblaje de proxy	200
Flujo de trabajo de script de simulación	201

Capítulo 5. EasyETL 203

Capítulo 6. Almacén del sistema 211

Almacén de propiedades de usuario	212
Almacén delta	212
Métodos de la fábrica de almacén	213
Métodos del almacén de propiedades	214
Métodos de UserFunctions (objeto system)	215

Capítulo 7. Deltas 217

Funciones Delta	217
Entrada Delta	218
Producción de entradas Delta	220
Función Delta para modalidad Iterador	220
Conectores de detección de cambios	226
Consumo de entradas Delta	227
Conectores de modalidad Delta	228
Modalidad de actualización y entradas Delta	229
Ejemplos	229

Capítulo 8. Panel de control de IBM Security Directory Integrator 233

Acceso a la aplicación Panel de control	234
Apertura desde un navegador	234
Apertura desde el menú Inicio de Windows	234
Configuración de Internet Explorer para el acceso remoto	235
Carga de una solución de integración de datos	236
Creación de una solución de integración de datos	236
Configuración de soluciones	237
Añadir una descripción de la solución	238
Configuración de una planificación de línea de ensamblaje	238
Creación de una planificación	238
Eliminación de una planificación	239
Ejecución y detención del planificador de Panel de control	239
Configuración de un conector	239
Modificación de los detalles de conexión	239
Modificación de una correlación de atributos	240
EasyETL de Panel de control	240
Configuración de soluciones EasyETL	240
Configuración de servidor	242
Configuración de valores de registro	242
Configuración de las lápidas	242
Configuración de los valores de seguridad de Panel de control	243
Visualización de los componentes instalados	243
Visualización de los datos de almacenamiento del sistema	244
Informes de ejecución de Panel de control	244
Creación de informes de ejecución	244
Creación y planificación de un informe de ejecución	244

Eliminación de una planificación	246
Ejecución y detención del planificador de informes de ejecución	246
Configuración y examen de los datos del conector	246
Supervisor de soluciones	247
Inicio y detención de las líneas de ensamblaje	247
Visualización del historial de ejecución de la línea de ensamblaje	247

Visualización de registros de lápida	248
Visualización de archivos de registro	248

Avisos	251
-------------------------	------------

Índice.	255
------------------------	------------

Acerca de esta publicación

Esta publicación contiene la información que necesita para desarrollar soluciones utilizando componentes que forman parte de IBM® Security Directory Integrator.

Los componentes de IBM Security Directory Integrator están diseñados para los administradores de red responsables de mantener directorios de usuario y otros recursos. Se da por supuesto que el usuario tiene experiencia práctica en la instalación y utilización de IBM Security Directory Integrator y IBM Security Directory Server.

La información también está destinada a los usuarios que están encargados de desarrollar, instalar y administrar soluciones utilizando IBM Security Directory Integrator. Es necesario que el lector conozca los conceptos y la administración de los sistemas a los que se aplicará la solución desarrollada. Dependiendo de la solución, estos sistemas pueden incluir uno o más de los productos, sistemas y conceptos siguientes, pero sin limitarse a ellos:

- IBM Security Directory Server
- IBM Security Identity Manager
- IBM Java™ Runtime Environment (JRE) u Oracle Java Runtime Environment
- Microsoft Active Directory
- Sistemas operativos Windows y UNIX
- Gestión de la seguridad
- Protocolos de Internet, incluidos HTTP (HyperText Transfer Protocol), HTTPS (HyperText Transfer Protocol Secure) y TCP/IP (Transmission Control Protocol/Internet Protocol)
- Protocolo LDAP (Lightweight Directory Access Protocol) y servicios de directorio
- Un registro de usuarios soportado
- Conceptos de autenticación y autorización
- Servidor de aplicaciones SAP ABAP

Acceso a publicaciones y terminología

Lea las descripciones de la biblioteca de IBM Security Directory Integrator Versión 7.2.0.1 y las publicaciones relacionadas a las que puede acceder en línea.

Esta sección proporciona:

- Una lista de publicaciones contenidas en la “Biblioteca de IBM Security Directory Integrator”.
- Enlaces a “Publicaciones en línea” en la página viii.
- Un enlace al “Sitio web de terminología de IBM” en la página ix.

Biblioteca de IBM Security Directory Integrator

Los documentos siguientes están disponibles en la biblioteca de IBM Security Directory Integrator:

- *IBM Security Directory Integrator Versión 7.2.0.1 Federated Directory Server, Guía de administración*

Contiene información sobre la utilización de la consola de Federated Directory Server para diseñar, implementar y administrar soluciones de integración de datos. También contiene información sobre cómo utilizar el protocolo SCIM (System Cross-Domain Identity Management) y la interfaz de gestión de identidades.

- *IBM Security Directory Integrator Versión 7.2.0.1 Guía de inicio*
Contiene una breve guía de aprendizaje y una introducción a IBM Security Directory Integrator. Incluye ejemplos para crear interacción y obtener aprendizaje práctico sobre IBM Security Directory Integrator.
- *IBM Security Directory Integrator Versión 7.2.0.1: Guía del usuario*
Contiene información sobre cómo utilizar IBM Security Directory Integrator. Contiene instrucciones para diseñar soluciones utilizando la herramienta de diseño de Security Directory Integrator (el Editor de configuración) o para ejecutar soluciones ya preparadas desde la línea de mandatos. Asimismo, proporciona información acerca de las interfaces, los conceptos y la creación y gestión de la línea de ensamblaje.
- *IBM Security Directory Integrator Versión 7.2.0.1: Guía de instalación y del administrador*
Incluye información sobre la instalación, la migración desde una versión anterior, la configuración de la funcionalidad de inicio de sesión y el modelo de seguridad subyacente de la API del servidor remoto de IBM Security Directory Integrator. Contiene información sobre cómo desplegar y gestionar soluciones.
- *IBM Security Directory Integrator Versión 7.2.0.1 Guía de referencia*
Contiene información detallada acerca de los componentes individuales de IBM Security Directory Integrator: conectores, manejadores de sucesos, analizadores, objetos, etc., es decir, los componentes básicos de la línea de ensamblaje.
- *IBM Security Directory Integrator Versión 7.2.0.1 Guía de determinación de problemas*
Proporciona información acerca de herramientas, recursos y técnicas de IBM Security Directory Integrator que pueden resultar de ayuda en la identificación y resolución de problemas.
- *IBM Security Directory Integrator Versión 7.2.0.1 Guía de mensajes*
Proporciona una lista de todos los mensajes informativos, de aviso y de error asociados con IBM Security Directory Integrator.
- *IBM Security Directory Integrator Versión 7.2.0.1 Guía de conectores de sincronización de contraseñas*
Incluye información completa para instalar y configurar cada uno de los cinco plug-ins de sincronización de contraseñas de IBM: Windows Password Synchronizer, Sun Directory Server Password Synchronizer, IBM Security Directory Server Password Synchronizer, Domino Password Synchronizer y Password Synchronizer para UNIX y Linux. También proporciona instrucciones de configuración para los almacenes de contraseñas de LDAP y JMS.
- *Notas del release de IBM Security Directory Integrator Versión 7.2.0.1*
Describe características nuevas e información de última hora sobre IBM Security Directory Integrator que no se ha podido incluir en la documentación.

Publicaciones en línea

IBM publica publicaciones sobre un producto en las ubicaciones siguientes cuando se comercializa el producto y cuando se actualizan las publicaciones:

Biblioteca de IBM Security Directory Integrator

El sitio de documentación del producto (<http://www-01.ibm.com/>)

support/knowledgecenter/SSCQGF/welcome) muestra la página de bienvenida y de navegación para esta biblioteca.

IBM Security Systems Documentation Central

IBM Security Systems Documentation Central proporciona una lista alfabética de todas las bibliotecas de productos de IBM Security Systems y enlaces a la documentación en línea correspondiente a versiones específicas de cada producto.

IBM Publications Center

El sitio web de IBM Publications Center (<http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss>) ofrece funciones de búsqueda personalizada para ayudarle a encontrar todas las publicaciones de IBM que necesite.

Información relacionada

Las ubicaciones siguientes contienen información relacionada con IBM Security Directory Integrator:

- IBM Security Directory Integrator utiliza el cliente JNDI de Oracle. Para obtener información sobre el cliente JNDI, consulte la página *Java Naming and Directory Interface™ Specification* en <http://download.oracle.com/javase/7/docs/technotes/guides/jndi/index.html>.
- En el sitio web siguiente puede encontrar información para ayudarle a responder sus preguntas referentes a IBM Security Directory Integrator: https://www-947.ibm.com/support/entry/myportal/over-accesspubsview/software/security_systems/tivoli_directory_integrator.

Sitio web de terminología de IBM

El sitio web de terminología de IBM agrupa la terminología de bibliotecas de productos en un solo lugar. Puede acceder al sitio web de terminología en <http://www.ibm.com/software/globalization/terminology>.

Accesibilidad

Las funciones de accesibilidad ayudan a los usuarios con alguna discapacidad física, tal como una movilidad o visión restringida, a utilizar satisfactoriamente los productos de software. Con este producto, puede utilizar tecnologías de asistencia a discapacitados para oír o navegar por la interfaz. También puede utilizar el teclado en lugar del ratón para ejecutar todas las funciones de la interfaz gráfica de usuario.

Para obtener información adicional, consulte el apéndice sobre accesibilidad de la publicación *Configuración de Directory Integrator*.

Formación técnica

Para obtener información sobre formación técnica, consulte el siguiente sitio web de formación de IBM: <http://www.ibm.com/software/tivoli/education>.

Información de soporte

Soporte de IBM proporciona asistencia para problemas relacionados con código fuente y preguntas habituales de corta duración sobre instalación o uso. Puede acceder directamente al sitio de soporte de software de IBM situado en <http://www.ibm.com/software/support/probsub.html>.

La publicación *Resolución de problemas* proporciona detalles sobre:

- Qué información recopilar antes de ponerse en contacto con Soporte de IBM.
- Los distintos métodos para ponerse en contacto con Soporte de IBM.
- Cómo utilizar IBM Support Assistant.
- Instrucciones y recursos de determinación de problemas para identificar y corregir problemas sin ayuda exterior.

Declaración de buenas prácticas de seguridad

La seguridad de los sistemas de tecnología de la información implica proteger los sistemas y la información mediante la prevención, detección y respuesta al acceso no autorizado desde dentro y fuera de su empresa. El acceso no autorizado puede dar como resultado la alteración, destrucción, apropiación indebida o mal uso de la información, y también daños en los sistemas o mal uso de ellos, incluida su utilización para atacar a otros sistemas. Ningún producto o sistema de la tecnología de la información se debe considerar completamente seguro y ningún producto, servicio o medida de seguridad puede ser completamente efectivo para prevenir la utilización o acceso indebidos. Los sistemas, productos y servicios de IBM están diseñados para formar parte de un sistema de seguridad completo, que necesariamente incluye procedimientos operativos adicionales y puede necesitar otros sistemas, productos o servicios para lograr la máxima efectividad. IBM NO GARANTIZA QUE NINGÚN SISTEMA, PRODUCTO O SERVICIO SEA INMUNE, O HAGAN A SU EMPRESA INMUNE, A LA CONDUCTA MALINTENCIONADA O ILEGAL DE CUALQUIERA DE LAS PARTES.

Capítulo 1. Conceptos generales

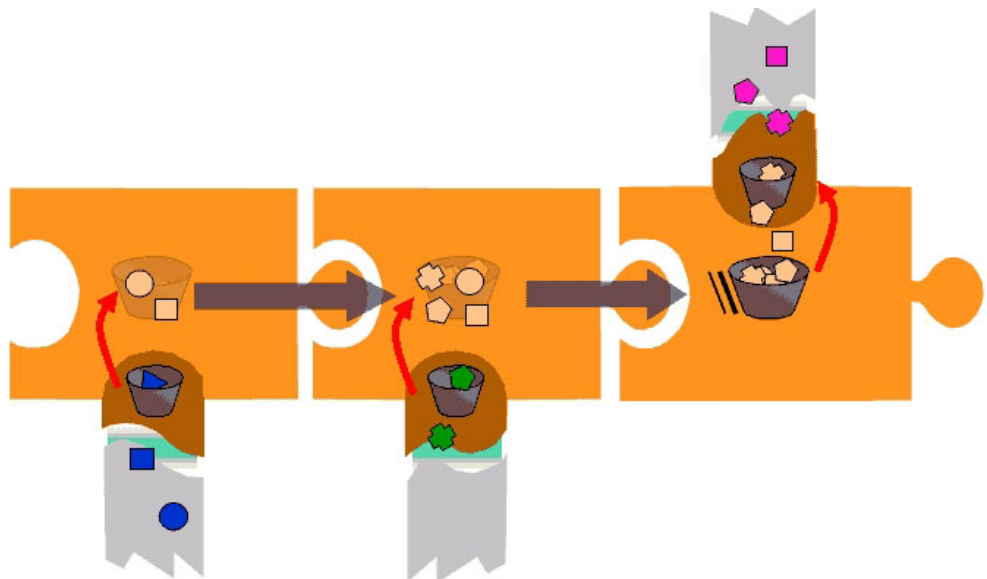
Esta sección introduce algunos de los conceptos básicos de IBM Security Directory Integrator, además de los elementos de la arquitectura que permiten crear soluciones, sus características y comportamientos.

Línea de ensamblaje.

Una línea de ensamblaje (LE) es un conjunto de componentes enlazados entre sí para mover y transformar datos; describe la "ruta" por la que pasarán los datos. Los datos que se manejan a través de esta ruta se representan como un objeto de *entrada*. La línea de ensamblaje funciona con una sola entrada a la vez en cada ciclo de la línea de ensamblaje. Es la unidad de trabajo en IBM Security Directory Integrator y generalmente representa un flujo de información entre uno o más orígenes de datos y uno o más destinos.

Visión general

Algunos de los componentes que componen la línea de ensamblaje recuperan datos de uno o más *sistemas conectados*—los datos que se obtienen de este modo se dice que "envían datos" a la línea de ensamblaje. Los datos que deben procesarse se pasan a la línea de ensamblaje de una en una *entrada*, y estas entradas contienen atributos con *valores* que provienen de entradas de directorio, filas de base de datos, mensajes de correo electrónico, documentos de *Lotus Notes*, registros u objetos de datos similares. Cada entrada contiene *atributos* que incluyen los valores de datos leídos de los campos o las columnas del sistema de origen. Estos atributos se renombran, se cambia su formato o se calculan de nuevo a medida que el proceso fluye de un componente al siguiente en la línea de ensamblaje. Puede "unirse" nueva información de otros orígenes, y todos los datos transformados o parte de ellos pueden escribirse en almacenes de destino o enviarse a sistemas de destino, según se desee. Esto se puede ilustrar del modo siguiente:



En este diagrama, imagine la colección de piezas grandes de rompecabezas como la línea de ensamblaje, los puntos y cuadrados azules situados más a la izquierda

de la corriente gris que entra desde abajo como datos sin formato de una corriente de entrada y las piezas púrpura de la parte superior como salida de datos de una corriente de salida. El elemento naranja más oscuro que intersecta una pieza de rompecabezas que contiene el cubo indica un *analizador*, que convierte los datos sin formato en datos estructurados, que a continuación pueden empezar a viajar por la línea de ensamblaje (como elementos de color más claro de un cubo). La pieza del rompecabezas del medio representa un conector que lee los datos ya estructurados, por ejemplo, de una base de datos.

Los datos entran en la línea de ensamblaje desde los sistemas conectados utilizando "Conectores" en la página 4 en algún tipo de *modalidad* de entrada y más adelante se envían como salida a uno o más sistemas conectados utilizando conectores en alguna modalidad de salida.

Los datos se pueden leer de sistemas orientados a registros como una base de datos o una cola de mensajes: en este caso, las distintas columnas de la entrada se correlacionan sin ningún problema en atributos en la entrada *work* resultante, que se representa como un "cubo" en la pieza de rompecabezas de la izquierda. O se pueden leer los datos de una corriente de datos, como un archivo de texto de un sistema de archivos, una conexión de red, etc. En este caso, se puede anteponer un analizador al conector, para hacer que la corriente de entrada tenga sentido y dividirla en fragmentos para que después se pueda asignar a atributos de la entrada *work*.

Cuando el primer conector ha realizado su trabajo, el cubo de información (la "entrada *work*", llamada apropiadamente *work*) pasa por la línea de ensamblaje hasta el siguiente componente - en la ilustración, otro conector. Puesto que los datos del primer conector están disponibles, ahora se pueden utilizar como información clave para recuperar o buscar datos en el segundo sistema conectado. Cuando se encuentran los datos pertinentes, se pueden fusionar en *work*, complementando los datos que todavía existen del primer conector.

Finalmente, los datos fusionados pasan por la línea de ensamblaje hasta la tercera pieza de rompecabezas o conector esta vez en alguna modalidad de salida, que se encarga de enviar los datos de salida al sistema conectado. Si el sistema conectado está orientado a registros, los distintos atributos de *work* tan solo se correlacionan con las columnas del registro; si el sistema conectado está orientado a corrientes, un analizador puede llevar a cabo el formateo necesario.

Otros componentes, como "Componentes de script" en la página 24 y "Funciones" en la página 23, se pueden insertar si se desea en la línea de ensamblaje para realizar operaciones en los datos de *work*.

Es importante recordar que la línea de ensamblaje se ha diseñado y optimizado para trabajar con un elemento cada vez. No obstante, si desea realizar varias actualizaciones o varias supresiones (por ejemplo, si desea procesar más de un elemento cada vez) entonces deberá escribir scripts de línea de ensamblaje para hacerlo. Si es necesario, este tipo de proceso puede implementarse utilizando bibliotecas de JavaScript y Java, y funciones estándar de IBM Security Directory Integrator, como agrupar los datos en un almacén de datos ordenado, por ejemplo con el conector JDBC, y luego leerlos de nuevo y procesarlos con una segunda línea de ensamblaje.

Las líneas de ensamblaje se crean, configuran y prueban utilizando el Editor de configuración (CE) de IBM Security Directory Integrator; consulte el apartado Capítulo 3, "Editor de configuración", en la página 83 para obtener más

información. La línea de ensamblaje tiene una pestaña denominada Flujo de datos del Editor de configuración. Aquí es donde se mantiene la lista de los componentes que forman esta línea de ensamblaje.

Todos los componentes de una línea de ensamblaje se registran automáticamente como variables de script. Por consiguiente, si tiene un conector denominado `ReadHRdump`, puede acceder al mismo y a sus métodos directamente desde un script utilizando la variable `ReadHRdump`. Como consecuencia de ello, es conveniente asignar a los componentes de la línea de ensamblaje los mismos nombres que se asignarían a las variables del script: deben utilizarse únicamente caracteres alfanuméricos, un nombre no debe empezar con un número y no deben utilizarse caracteres nacionales especiales (como `ã`, `ä`), separadores (excepto el guión bajo `'_'`), espacios en blanco, etc.

Siempre existe un método alternativo para acceder a un componente de línea de ensamblaje (por ejemplo, la función `task.getConnector()`), pero es recomendable utilizar un convenio de denominación consciente.

Iniciar una línea de ensamblaje en IBM Security Directory Integrator es una operación bastante costosa, ya que implica la creación de una nueva hebra Java y generalmente establece conexiones con uno o más orígenes de datos. Al diseñar una solución, debe considerar la posibilidad de que ésta funcione con un número reducido de líneas de ensamblaje (en lugar de utilizar muchas), y que cada una de las líneas de ensamblaje realice más de una función; por ejemplo, utilizando ramas o conmutadores para definir varias operaciones gestionadas por una única línea de ensamblaje. Debe tener en cuenta que cada operación puede implementarse como una línea de ensamblaje independiente, pero que estas líneas pueden integrarse y estar listas para su uso en una única línea de ensamblaje que proporcione trabajo a todas las líneas de ensamblaje a través del conector de la línea de ensamblaje o de la función de la línea de ensamblaje. Esto también permite utilizar funciones tales como las agrupaciones de conectores globales para gestionar el uso de recursos e incrementar el rendimiento y la escalabilidad.

Componentes

Las líneas de ensamblaje pueden incluir los siguientes componentes:

- “Conectores” en la página 4
- “Funciones” en la página 23
- “Componentes de script” en la página 24
- “Correlaciones de atributos” en la página 25
- “Componentes de rama” en la página 29

Adicionalmente, los conectores pueden tener “Analizadores” en la página 32 configurados; además en el nivel de sistema, configuración, línea de ensamblaje, correlación de atributos y atributo existen opciones para configurar el “Comportamiento ante un valor nulo” en la página 26.

Acceso a los componentes de la línea de ensamblaje que se encuentran dentro de la misma

Cada uno de los componentes de la línea de ensamblaje está disponible como una variable de script previamente registrada con el nombre que el usuario selecciona para el componente.

Tenga en cuenta que puede cargar dinámicamente los componentes con llamadas de script a funciones de tipo `system.getConnector()`, si bien los usuarios con poca experiencia no deberían hacerlo.¹

Pase de parámetros de la línea de ensamblaje

Hay tres formas de pasar datos a una línea de ensamblaje:

- Generar su propia entrada inicial en la línea de ensamblaje; por ejemplo, en un script de prólogo.
- Enviar datos de uno o más iteradores².
- Iniciar la línea de ensamblaje con los parámetros de otra línea de ensamblaje utilizando el conector o el componente de función de la línea de ensamblaje, o utilizando una llamada de API.

Si desea iniciar una línea de ensamblaje con parámetros de otra línea de ensamblaje, tiene dos opciones:

- Utilizar el método *TCB* (bloque de llamada de tareas), que es el preferido. Consulte el apartado “TCB (bloque de llamada de tareas)” en la página 72 para obtener más información. En este apartado, también se tratan las técnicas para inhabilitar y habilitar componentes de línea de ensamblaje de forma dinámica.
- Proporcionar directamente una entrada work inicial; consulte el apartado “Suministro de una IWE (entrada work inicial)” en la página 8 para obtener más detalles.

Nota: Estas opciones se proporcionan por motivos de compatibilidad con versiones anteriores.

Conectores

Los conectores se utilizan para acceder y actualizar los orígenes de información. El trabajo de un conector es nivelar el campo de juego de modo que el usuario no deba ocuparse de detalles técnicos del trabajo con distintos orígenes de datos, sistemas, servicios o transportes. De este modo, cada tipo de conector se ha diseñado para que utilice un protocolo específico o API, que maneje los detalles del acceso al origen de datos, de modo que el usuario pueda concentrarse en los procesos de manipulación y relación de datos y también en los procesos personalizados como, por ejemplo, el filtrado y el control de coherencia.

Visión general

Los conectores se utilizan para extraer los detalles de algún sistema o almacén, ofreciendo el mismo conjunto de funciones de acceso. Esto permite trabajar con una amplia gama de tecnologías y formatos distintos de forma coherente y previsible. Una línea de ensamblaje (LE) típica tiene un conector que proporciona entrada de datos y como mínimo un conector que escribe datos de salida.

Existen dos categorías de conectores:

1. El objeto de conector que se obtiene de esta llamada es un objeto de *interfaz de conector*, y es la parte específica del origen de datos de un conector de línea de ensamblaje. Cuando cambia el tipo de un conector, en realidad está cambiando la inteligencia de su origen de datos (la interfaz de conector) que proporciona la funcionalidad para acceder a los datos de un sistema, un servicio o un almacén de datos específico. La mayor parte de la funcionalidad de un conector de línea de ensamblaje, incluyendo las correlaciones de atributos, los criterios de enlace y los complementos de software, la proporciona el kernel y se mantiene intacta cuando se conmutan los tipos de conector.

2. Un *iterador* es una notación abreviada de un conector en modalidad Iterador.

- La primera categoría es aquella en la que el conector conoce tanto el transporte como la estructura del contenido de los datos; es decir, se puede detectar o consultar el esquema del origen de datos utilizando una API conocida como, por ejemplo, JDBC o LDAP.
- La segunda categoría es aquella en la que se conoce el mecanismo de transporte, pero no la estructura del contenido - que normalmente se parece a una corriente de datos. Esta categoría requiere un *analizador* (consulte la sección “Analizadores” en la página 32 y el capítulo “Parsers” en la publicación *Referencia*) para interpretar o generar la estructura del contenido de modo que la línea de ensamblaje funcione correctamente.

Una biblioteca de conectores completa es una de las cualidades de IBM Security Directory Integrator. La publicación *Referencia* contiene la lista de todos los conectores que se incluyen con IBM Security Directory Integrator. Pero también puede escribir su propio conector en JavaScript o incluso en Java; consulte “Implementing your own Components” en la publicación *Referencia*.

Cada conector está diseñado para un protocolo, una API o un transporte determinado y gestiona la clasificación de datos entre el tipo nativo del sistema conectado y los objetos Java. A diferencia de otros componentes, los conectores tienen un valor de modalidad que determina el modo en que el conector accede a su sistema conectado; consulte el apartado “Modalidades del conector” en la página 6 para obtener más información. Cada conector da soporte solamente a un subconjunto de modalidades que son adecuadas para su sistema conectado. Por ejemplo, el conector del sistema de archivos solamente da soporte a una modalidad de salida, Sólo adición, y no da soporte a las modalidades Actualizar, Suprimir o CallReply. Cuando utilice un conector, en primer lugar debe consultar la documentación de este componente para obtener una lista de las modalidades soportadas.

Nota: Las líneas de ensamblaje pueden constar de tantos, o tan pocos, conectores (y otros componentes) como sea necesario para implementar el flujo de datos específico. No hay un límite en el sistema. No obstante, lo recomendable es mantener una línea de ensamblaje del modo más simple posible para garantizar al máximo su mantenimiento.

Cuando selecciona un conector para la línea de ensamblaje, se muestra un recuadro de diálogo que le permite seleccionar el tipo de conector del que desea *heredar*. La herencia es un concepto importante cuando se trabaja con IBM Security Directory Integrator ya que todos los componentes que se incluyen en las soluciones heredan algunas de sus características de otro componente — de uno de los tipos básicos o de la biblioteca de componentes preconfigurados: conectores, analizadores, funciones, etc., de la sección Recursos del espacio de trabajo.

Cuando se utilizan en una línea de ensamblaje, los conectores ofrecen la opción Inicializar para controlar cuándo se configura el componente; por ejemplo, cuándo se establecen las conexiones, cuándo se enlazan los recursos, etc. De forma predeterminada, todos los conectores se inicializan cuando se inicia la línea de ensamblaje: *fase de arranque de la línea de ensamblaje*.

Los conectores en modalidad Servidor o Iterador proporcionan datos a la línea de ensamblaje y son responsables de proporcionar a la línea de ensamblaje una nueva entrada work para cada uno de los ciclo que cree la línea de ensamblaje. La entrada work se pasa de componente a componente en la sección Flujo, siguiendo la lógica de ramificación que se haya establecido, hasta alcanzar el final del flujo.

En este punto empieza el comportamiento de fin del ciclo, en que el iterador obtiene la siguiente entrada de su origen y la pasa a la sección Flujo para un nuevo ciclo.

Mientras que un iterador de la sección Entradas de datos en realidad controlará el flujo, un iterador de la sección Flujo simplemente obtendrá la entrada siguiente y ofrecerá sus atributos de datos para la correlación de entradas en la entrada work.

Nota: Puede colocar un conector en modalidad Iterador dentro de la sección Flujo. De este modo, el iterador funciona como lo haría en Entrada de datos: se inicializa, incluyendo la creación de su conjunto de resultados con la llamada `selectEntries`, durante el inicio de la línea de ensamblaje y recupera una entrada (`getNextEntry`) en cada ciclo de la línea de ensamblaje. Sin embargo, un iterador en la sección Flujo no controla la línea de ensamblaje en sí, como lo haría en Entrada de datos.

El comportamiento de la sección Entrada de datos es distinto para las modalidades Servidor e Iterador: un iterador espera ser el primer componente que se ejecuta en la línea de ensamblaje y sólo se leerá su entrada siguiente si la entrada work no existe todavía. Si se pasa una entrada work inicial a la línea de ensamblaje, los iteradores no leen ningún dato para este primer ciclo. Esto también significa que los iteradores se ejecutan en una serie, en la que el segundo iterador empieza a devolver entradas una vez que el primero ha alcanzado el final de los datos y no ha devuelto nada (null).

En la modalidad Servidor, en cambio, el conector inicia una hebra de *escucha* del servidor, por ejemplo, para un puerto IP o una llamada de notificación de suceso, y luego pasa el control al siguiente conector de Entradas de datos.

Cuando se llama a una línea de ensamblaje mediante el componente de función de la línea de ensamblaje (AL FC), cada vez que el componente de función ejecuta la llamada sólo se utilizan los componentes de la sección Flujo si se utiliza la modalidad de ciclo manual.

Existe una carpeta **Conectores** en el espacio de trabajo del navegador de IBM Security Directory Integrator donde puede mantener su biblioteca de conectores configurados. Es aquí también donde se definen las agrupaciones de conectores.

Modalidades del conector

La modalidad de un conector de línea de ensamblaje define el rol que el conector juega en el flujo de datos y controla el modo en que el comportamiento automatizado de la línea de ensamblaje dirige al componente. Determina si se leerá de una fuente de entrada, se grabará en ella o ambas cosas.

Los conectores se pueden establecer en una de las siguientes modalidades estándar:

- Iterador
- Búsqueda
- Sólo adición
- Actualizar
- Suprimir
- CallReply
- Servidor
- Delta

Estas modalidades se describen en los apartados siguientes. Para obtener una descripción detallada del comportamiento de la modalidad del conector y también para obtener información general sobre la línea de ensamblaje, consulte "AssemblyLine and Connector mode flowcharts" en la publicación *Referencia*.

modalidad Iterador:

Los conectores que están en modalidad Iterador se utilizan para explorar un origen de datos y extraer sus datos. El conector en modalidad Iterador en realidad se itera a través de las entradas de origen de datos, lee sus valores de atributo y entrega cada una de las entradas a los componentes de la sección Flujo de la línea de ensamblaje. Un conector en modalidad Iterador se conoce normalmente como conector iterador o sólo como iterador.

Generalmente, las líneas de ensamblaje (salvo las que se invocan con una IWE; consulte "Suministro de una IWE (entrada work inicial)" en la página 8) contienen como mínimo un conector en la modalidad Iterador. Los iteradores suministran datos a la línea de ensamblaje creando entradas work y pasando esas entradas a la sección Flujo de la línea de ensamblaje.

Los componentes de la sección Flujo se activan en orden, empezando por la parte superior de la lista de flujo. Una vez que ha finalizado el proceso del flujo, el control se devuelve al iterador para recuperar la entrada siguiente.

Varios iteradores en una línea de ensamblaje: Si tiene más de un conector en modalidad Iterador, estos conectores se apilan por el orden en el que aparecen en la configuración (y en la lista de conectores del Editor de configuración, en la sección Entradas de datos) y se procesan uno cada vez. Por lo tanto, si está utilizando dos iteradores, el primero lee en el origen de datos, y pasa la entrada work resultante al primer conector que no sea iterador, hasta que alcanza el final del conjunto de datos. Cuando el primer iterador ha agotado su origen de entrada, el segundo iterador comienza a leer datos.

Una entrada work inicial se trata como si procediese de un iterador invisible y se procesa antes que cualquier otro iterador. Esto significa que se pasa una entrada work inicial (IWE) al primer conector que no es iterador de la línea de ensamblaje, y se pasan por alto todos los iteradores durante el primer ciclo. Este comportamiento puede verse en la página del flujo de la línea de ensamblaje de "AssemblyLine and Connector mode flowcharts" en la publicación *Referencia*.

Supongamos que tiene una línea de ensamblaje con dos Iteradores, **a** que precede a **b**. El primer iterador, **a**, se utiliza hasta que **a** deja de devolver entradas. A continuación, la línea de ensamblaje conmuta a **b** y pasa por alto **a**. Si se pasa una entrada work inicial (IWE) a esta línea de ensamblaje, se pasan por alto ambos iteradores para el primer ciclo, después del cual la línea de ensamblaje empieza a llamar a **a**.

Algunas veces la entrada work inicial, IWE, se utiliza para pasar parámetros de configuración a una línea de ensamblaje, pero no para pasar datos. Pero la presencia de una IWE hace que se pasen por los iteradores de la línea de ensamblaje durante el primer ciclo. Si no desea que esto suceda, debe vaciar el objeto de entrada work llamando a la función `task.setWork(null)` en un script de prólogo. Esto hace que el primer iterador funcione con normalidad.

Utilización de la modalidad Iterador:

Es muy común que un Iterador controle la línea de ensamblaje.

El patrón más común para utilizar un conector en la modalidad Iterador es:

1. Mediante el Editor de configuración, añadir un conector en la modalidad Iterador al espacio de trabajo. Consulte el apartado "Creación de un conector" en la página 122.
2. Establecer la modalidad (Iterador) y otros parámetros de conexión para este conector en la pestaña **Conexión**; los parámetros necesarios se indican con un asterisco (*). Algunos conectores requieren que también se configure un analizador en la pestaña **Analizador**.
3. Configurar la correlación de atributos; consulte el apartado "Correlación de atributos de entrada" en la página 120.

Estos atributos correlacionados se recuperan del origen de datos, en la entrada *work*, y se pasan a los conectores de la sección Flujo de la línea de ensamblaje.

Si no ha creado el conector directamente en una línea de ensamblaje, para utilizar este conector en una línea de ensamblaje, arrastre el conector de su ubicación en <espaciotrabajo>/Resources/Connectors a la sección **Entrada de datos** de una línea de ensamblaje.

Suministro de una IWE (entrada work inicial): Se trata de un método alternativo de pasar parámetros utilizando un TCB y se da soporte al mismo para la compatibilidad con versiones anteriores.

Cuando se inicia una línea de ensamblaje con la llamada `system.startAL()` desde un script, se pueden pasar parámetros a la línea de ensamblaje definiendo valores de atributo o de propiedad en la entrada *work* inicial, a la que se accede a través de la variable *work*. Por tanto, es tarea suya aplicar dichos valores para definir parámetros de conector; por ejemplo, en el complemento de software **Prólogo – Init** de la línea de ensamblaje utilizando la función `connectorName.setParam()`.

Nota: Debe borrar la entrada *Work* con la llamada `task.setWork(null)`, de lo contrario los iteradores de la línea de ensamblaje pasan a través del primer ciclo.

Puede examinar el resultado de la línea de ensamblaje, que es la **entrada Work** cuando la línea de ensamblaje se detiene, utilizando la función `getResult()`. Consulte también "Runtime provided Connector" en la publicación *Referencia*.

El siguiente es un ejemplo de cómo pasar un valor de parámetro del conector con una IWE:

```
var entry = system.newEntry();
entry.setAttribute ("userNameForLookup", "John Doe");

// Aquí comienza la línea de ensamblaje
var al = main.startAL ( "EmailLookupAL", entry );

// esperar a que finalice la línea de ensamblaje
al.join();

var result = al.getResult();

// se supone que la línea de ensamblaje establece el atributo mail en su entrada de trabajo
task.logmsg ("Correo electrónico devuelto = " + result.getString("mail"));
```


Modalidad Buscar:

La modalidad Buscar le permite unir datos de diferentes orígenes de datos utilizando la relación entre los atributos de estos sistemas. Generalmente, decimos que un conector en modalidad Buscar es un conector de búsqueda.

Para poder configurar un conector de búsqueda en el Editor de configuración, debe indicar al conector cómo se define la comparación entre los datos que ya están incluidos en la línea de ensamblaje y los que se encuentran en el sistema conectado. Esto se conoce como *Criterios de enlace* del conector, y cada conector de búsqueda tiene una pestaña **Criterios de enlace** asociada en la que se definen las reglas para localizar entradas coincidentes. Consulte el apartado "Criterios de enlace" en la página 21 para obtener más información.

Utilización de la modalidad Buscar: el patrón más común para utilizar un conector en modalidad Buscar es:

1. Utilizando el Editor de configuración, añadir un conector en modalidad Buscar al espacio de trabajo. Consulte el apartado "Creación de un conector" en la página 122.
2. Establecer la modalidad (Buscar) y otros parámetros de conexión para este conector en la pestaña **Conexión**; los parámetros necesarios se indican con un asterisco (*). Algunos conectores requieren que también se configure un analizador en la pestaña **Analizador**.
3. Configurar la correlación de atributos; consulte el apartado "Correlación de atributos de entrada" en la página 120.
4. Abra la pestaña **Criterio de enlace** de la ventana de configuración del conector y establezca las normas para la comparación de atributos. El resultado de este proceso determinará qué entradas se recuperarán del sistema conectado y para ello existen dos opciones:
 - a. Pulse **Añadir** para añadir un nuevo criterio de enlace y seleccione un atributo del sistema conectado, el operador de coincidencia (por ejemplo, Equivale a, Empieza por, etc.) y después el atributo de la entrada work con el que debe coincidir. Cuando el conector realiza la búsqueda, crea la sintaxis de la API o del protocolo subyacente basada en el criterio de enlace que usted ha especificado, lo cual permite que la solución siga siendo independiente del tipo de sistema utilizado. Puede añadir varios criterios de enlace, que se conectan mediante el operador booleano AND para crear la llamada de búsqueda.
 - b. También puede seleccionar **Crear criterios con script personalizado**, que abre la ventana del editor de scripts en la cual puede crear su propia serie de búsqueda, que se devuelve al conector utilizando el objeto `ret.filter`. Por ejemplo:

```
ret.filter = "uid=" + work.getString("uid");
```

Recuerde que las expresiones también pueden utilizarse para especificar de forma dinámica el atributo o el valor que debe utilizarse para cualquier criterio de enlace. Consulte "Expresiones" en la página 40 para obtener información. Consulte también "Criterios de enlace" en la página 21 para obtener más detalles sobre los criterios de enlace.

Los atributos que lea (y calcule) en la Correlación de entrada están disponibles para los conectores que vienen a continuación y para la lógica de scripts mediante el objeto de entrada work.

Si no ha creado el conector directamente en una línea de ensamblaje, para utilizar el conector en una línea de ensamblaje, arrástrelo de su ubicación en <espaciotrabajo>/Resources/Connectors a la sección **Flujo** de la línea de ensamblaje.

Modalidad Sólo adición:

Los conectores en modalidad Sólo adición, comúnmente referidos como conectores Sólo adición, se utilizan para añadir entradas de datos nuevas a un origen de datos.

Esta modalidad de conector prácticamente no requiere configuración. Establezca los parámetros de conexión y, a continuación, seleccione (correlacione) los atributos que deben escribirse desde la Entrada de trabajo.

Utilización de la modalidad Sólo adición: el patrón más común y sencillo para utilizar un conector en modalidad Sólo adición es el siguiente:

1. Utilizando el Editor de configuración, añadir un conector en modalidad Sólo adición al espacio de trabajo. Consulte el apartado "Creación de un conector" en la página 122.
2. Establecer la modalidad (Sólo adición) y otros parámetros de conexión para este conector en la pestaña **Conexión**; los parámetros necesarios se indican con un asterisco (*). Algunos conectores requieren que también se configure un analizador en la pestaña **Analizador**.
3. Configurar la correlación de atributos; consulte el apartado "Correlación de atributos de salida" en la página 121.

Si no ha creado el conector directamente en una línea de ensamblaje, para utilizar el conector en una línea de ensamblaje, arrástrelo de su ubicación en <espaciotrabajo>/Resources/Connectors a la sección **Flujo** de la línea de ensamblaje.

Los atributos de la entrada work que ha correlacionado se pasan como salida al sistema conectado cuando la línea de ensamblaje llama al conector.

Modalidad Actualizar:

Los conectores en modalidad Actualizar, a los que normalmente se hace referencia como conectores Actualizar, se utilizan para añadir y modificar datos de un origen de datos. Para cada una de las entradas que se pasa de la línea de ensamblaje, el conector Actualizar intenta localizar una entrada coincidente del origen de datos para modificarla con los valores de los atributos de entrada recibidos. Si no se encuentra ninguna coincidencia, el conector de modalidad Actualizar añadirá una entrada nueva.

Al igual que con los conectores Búsqueda, debe indicar al conector cómo se define la comparación entre los datos que ya están incluidos en la línea de ensamblaje y los que se encuentran en el sistema conectado. Esto se conoce como criterio de enlace del conector, y cada conector Actualizar tiene una pestaña Criterio de enlace asociada (consulte el apartado "Criterios de enlace" en la página 21) en la que se definen las reglas para localizar entradas coincidentes. Si no se encuentra una entrada de este tipo, se añade una entrada nueva al origen de datos. No obstante, si se encuentra una entrada coincidente, ésta se modifica. Si hay más de una entrada que coincida con el criterio de enlace, se llama al complemento de

software En caso de varias entradas. Asimismo, se puede configurar la correlación de salida para que especifique qué atributos se han de utilizar durante una operación de adición o modificación.

Cuando realice una operación de modificación, sólo se modificarán en el origen de datos los atributos que estén marcados como modificar (Mod) en la correlación de salida. Si la entrada que se pasa desde la línea de ensamblaje no tiene un valor para un atributo, el Comportamiento ante un valor nulo de dicho atributo pasa a ser importante. Si éste se establece en Suprimir, el atributo no existe en la entrada de modificación, por lo que el atributo no se puede modificar en el origen de datos. Si se establece en *Nulo*, el atributo existe en la entrada de modificación pero tiene un valor nulo, lo que significa que el atributo se suprime del origen de datos.

Una función importante de los conectores Actualizar es la opción Calcular cambios. Cuando está activada, el conector primero compara los nuevos valores con los antiguos y solamente realiza la actualización donde sea necesario y si es necesario. Por consiguiente, puede omitir las actualizaciones innecesarias, lo que puede ser muy favorable si la operación de actualización tiene una carga considerable para el origen de datos en particular que está actualizando.

Algunos conectores Actualizar ofrecen la opción de pasar omitir búsquedas innecesarias cuando se realizan actualizaciones. Si el conector lo permite, verá un recuadro de selección **Omitir búsqueda** junto al recuadro de selección **Calcular cambios**. Cuando se selecciona, modifica el comportamiento del conector para que no se realice ninguna búsqueda para buscar una entrada que corresponda al criterio de enlace. Por este motivo, no se invoca ningún complemento de software Antes/Después de búsqueda. El complemento de software En caso de varias entradas tampoco se puede invocar. Con esta opción activada sólo se crea el criterio de búsqueda y la modificación se llama directamente. Esto difiere del comportamiento del conector predeterminado en modalidad Actualizar ya que si no se encuentra ninguna entrada con el criterio de enlace, no se añade como nuevo.

Utilización de la modalidad Actualizar: el patrón más común y sencillo para utilizar un conector en modalidad Actualizar es:

1. Utilizando el Editor de configuración, añadir un conector en modalidad Actualizar al espacio de trabajo. Consulte el apartado “Creación de un conector” en la página 122.
2. Establecer la modalidad (Actualizar) y otros parámetros de conexión para este conector en la pestaña **Conexión**; los parámetros necesarios se indican con un asterisco (*). Algunos conectores requieren que también se configure un analizador en la pestaña **Analizador**.
3. Configurar la correlación de atributos; consulte el apartado “Correlación de atributos de salida” en la página 121.
4. Abra la pestaña **Criterio de enlace** de la ventana de configuración del conector y establezca las normas para la comparación de atributos. Las siguientes son dos de las opciones por las que puede optar:
 - a. Pulse **Añadir** para añadir un nuevo criterio de enlace y seleccione un atributo del sistema conectado, el operador de coincidencia (por ejemplo, Equivale a, Empieza por, etc.) y después el atributo de la entrada work con el que debe coincidir. Cuando el conector realiza la búsqueda, crea la sintaxis de la API o del protocolo subyacente basada en el criterio de enlace que usted ha especificado, lo cual permite que la solución siga siendo

independiente del tipo de sistema utilizado. Puede añadir varios criterios de enlace, que se conectan mediante el operador booleano AND para crear la llamada de búsqueda.

- b. También puede seleccionar **Crear criterios con script personalizado**, que abre la ventana del editor de scripts en la cual puede crear su propia serie de búsqueda, que se devuelve al conector utilizando el objeto `ret.filter`. Por ejemplo:

```
ret.filter = "uid=" + work.getString("uid");
```

Recuerde que las expresiones también pueden utilizarse para especificar de forma dinámica el atributo o el valor que debe utilizarse para cualquier criterio de enlace. Consulte “Expresiones” en la página 40 para obtener información. Consulte también “Criterios de enlace” en la página 21 para obtener más detalles sobre los criterios de enlace.

Las entradas con los atributos que haya seleccionado para correlacionar en la entrada se añaden al origen de datos durante la ejecución de la línea de ensamblaje.

Para utilizar el conector en una línea de ensamblaje, arrástrelo de su ubicación en la <espaciotrabajo>/Resources/Connectors a la sección **Flujo** de una línea de ensamblaje. Ahora puede correlacionar los atributos de la entrada `work` con la salida arrastrando los atributos que se hayan correlacionado anteriormente, al conector Actualizar de la ventana **Correlaciones de atributos** de la línea de ensamblaje.

También puede crear atributos completamente nuevos pulsando con el botón derecho del ratón en el conector en esta ventana y seleccionando **Añadir elemento de correlación de atributos**.

Nota: en la modalidad Actualizar se pueden actualizar varias entradas. Consulte "AssemblyLine and Connector mode flowcharts" en la publicación *Referencia*.

modalidad Suprimir:

Los conectores en modalidad Suprimir, a menudo referidos como conectores Suprimir, se utilizan para suprimir datos de un origen de datos.

Para cada entrada que se pasa al conector Suprimir, ésta intenta localizar los datos coincidentes en el sistema conectado. Si se encuentra una entrada que coincida se suprime, de lo contrario se llama al complemento de software 'En caso de ninguna coincidencia' o al complemento de software 'En caso de varias entradas' si se encuentra más de una coincidencia. Al igual que con las modalidades Buscar y Actualizar, la modalidad Suprimir requiere que defina las normas para buscar la entrada coincidente que se ha de suprimir. Esto se configura en la pestaña Criterio de enlace del conector.

Algunos conectores Suprimir ofrecen la opción de pasar omitir búsquedas innecesarias cuando se realizan supresiones. Si el conector lo permite, verá un recuadro de selección Omitir búsqueda junto al recuadro de selección Calcular cambios. Cuando se selecciona, modifica el comportamiento del conector para que no se realice ninguna búsqueda para buscar una entrada que corresponda al criterio de enlace. Por este motivo, no se invoca ningún complemento de software Antes/Después de búsqueda. El complemento de software En caso de varias entradas tampoco se puede invocar. Con esta opción activada sólo se crean criterios de búsqueda y la supresión se llama directamente.

Utilización de la modalidad Suprimir: el patrón más común y sencillo para utilizar un conector en modalidad Suprimir es el siguiente:

1. Utilizando el Editor de configuración, añadir un conector en modalidad Suprimir al espacio de trabajo. Consulte el apartado “Creación de un conector” en la página 122.
2. Establecer la modalidad (Suprimir) y otros parámetros de conexión para este conector en la pestaña **Conexión**; los parámetros necesarios se indican con un asterisco (*). Algunos conectores requieren que también se configure un analizador en la pestaña **Analizador**.
3. Configurar la correlación de atributos; consulte el apartado “Correlación de atributos de entrada” en la página 120.
4. En la pestaña Correlación de atributos de entrada puede seleccionar los atributos de la lista de Atributos del conector y, a continuación, arrastrarlos a la correlación de entrada. También puede añadir o eliminar atributos manualmente.

Nota: en la modalidad Suprimir, la correlación de entrada se utiliza para leer la entrada coincidente del origen de datos en el objeto de entrada *conn*, el cual se puede utilizar a continuación en los scripts para, por ejemplo, determinar si realmente la entrada se ha de suprimir.

5. Abra la pestaña **Criterio de enlace** de la ventana de configuración del conector y establezca las normas para la comparación de atributos. El resultado de este proceso determinará qué entradas se recuperarán del sistema conectado y para ello existen dos opciones:
 - a. Pulse **Añadir** para añadir un nuevo criterio de enlace y seleccione un atributo del sistema conectado, el operador de coincidencia (por ejemplo, Equivale a, Empieza por, etc.) y después el atributo de la entrada work con el que debe coincidir. Cuando el conector realiza la búsqueda, crea la sintaxis de la API o del protocolo subyacente basada en el criterio de enlace que usted ha especificado, lo cual permite que la solución siga siendo independiente del tipo de sistema utilizado. Puede añadir varios criterios de enlace, que se conectan mediante el operador booleano AND para crear la llamada de búsqueda.
 - b. También puede seleccionar **Crear criterios con script personalizado**, que abre la ventana del editor de scripts en la cual puede crear su propia serie de búsqueda, que se devuelve al conector utilizando el objeto `ret.filter`. Por ejemplo:

```
ret.filter = "uid=" + work.getString("uid");
```

Consulte el apartado “Criterios de enlace” en la página 21 para obtener más información acerca del criterio de enlace.

Si no ha creado el conector directamente en una línea de ensamblaje, para utilizar el conector en una línea de ensamblaje, arrástrelo de su ubicación en `<espaciotrabajo>/Resources/Connectors` a la sección **Flujo** de la línea de ensamblaje.

Modalidad CallReply:

La modalidad CallReply se utiliza para realizar peticiones a los servicios de origen datos, como por ejemplo servicios web, que requieren que se envíen parámetros de entrada y que se reciba una respuesta con valores de retorno.

A diferencia de las demás modalidades, la modalidad CallReply proporciona acceso a correlaciones tanto de atributos de entrada como de salida.

El conector primero realiza una operación de correlación de salida y al hacerlo llama a un sistema externo, con los parámetros proporcionados por la operación de correlación de salida. A continuación se realiza una operación de correlación de entrada, de manera que obtiene la respuesta del sistema externo.

Utilización de la modalidad CallReply: el patrón más común y sencillo para utilizar un conector en modalidad CallReply es el siguiente:

1. Utilizando el Editor de configuración, añadir un conector en modalidad CallReply al espacio de trabajo. Consulte el apartado "Creación de un conector" en la página 122. Muy pocos conectores dan soporte a esta modalidad.
2. Establecer la modalidad (CallReply) y otros parámetros de conexión para este conector en la pestaña **Conexión**; los parámetros necesarios se indican con un asterisco (*). Algunos conectores requieren que también se configure un analizador en la pestaña **Analizador**.
3. Configurar la correlación de atributos de salida; consulte el apartado "Correlación de atributos de salida" en la página 121.
4. Configurar la correlación de atributos de entrada; consulte el apartado "Correlación de atributos de entrada" en la página 120.

Tenga en cuenta que los atributos de la correlación de salida se proporcionan al sistema conectado como parámetros de entrada; y los atributos correlacionados mediante la correlación de atributos de entrada contienen la respuesta del sistema conectado.

modalidad Servidor:

La modalidad Servidor, disponible en un número seleccionado de conectores, está diseñado para proporcionar funciones de espera de un suceso entrante, generar una hebra que trate el suceso y enviar una respuesta al origen.

Actualmente, todos los conectores de modalidad Servidor están basados en conexión. Como resultado, una línea de ensamblaje (LE) que utiliza un conector en modalidad Servidor en la sección Entradas de datos inicializará y, a continuación, esperará una conexión entrante (mediante una conexión SNMP, Web Services, LDAP, HTTP, TCP); cuando se inicia una conexión, el conector en modalidad Servidor clona la línea de ensamblaje de la cual forma parte y reanuda la espera del suceso siguiente (es decir, un nuevo inicio de conexión). En la línea de ensamblaje de trabajo clonada, mientras tanto, el conector en modalidad Servidor se pone en modalidad Iterador, y empieza a leer datos de la conexión. Los datos obtenidos de la conexión se envían a resto de la línea de ensamblaje en la forma normal de iterador, que incluye seguir el flujo de complemento de software del iterador estándar, leer las entradas del suceso una a una y pasarlas a los otros componentes de flujo para procesarlas hasta que no hay más datos que leer. Al final de cada ciclo (a menudo sólo habrá uno) la línea de ensamblaje encabezada por el conector en modalidad Servidor envía una respuesta al cliente, a menos que decida saltarse la fase de respuesta con, por ejemplo, `system.skipEntry()`;

Una vez se completa la línea de ensamblaje a la que envía datos (es decir, se agota el origen de datos) esa hebra termina; en este momento se vacía la línea de ensamblaje de trabajo, y si es necesario, se informa al Gestor de agrupaciones de que esta instancia de línea de ensamblaje vuelve a estar disponible.

El conector en modalidad Servidor original sigue activamente a la escucha de otras iniciaciones de conexión.

Nota: Bajo ciertas condiciones poco frecuentes, cuando emite más de 5 solicitudes de cliente al servidor en paralelo, básicamente en el sistema operativo zOS, los clientes SNMP salen de la conexión, presentando excepciones de Protocol Data Unit (PDU). Sin embargo, en una situación real es muy raro que un agente, como el conector de servidor SNMP, sea consultado intensamente por varios gestores, como el conector SNMP.

El conector SNMP tiene un parámetro de configuración sin documentar llamado *snmpWalkTimeout*. Puede alterar temporalmente el valor predeterminado para este parámetro, que es 5000 ms. Este parámetro no es accesible utilizando el Editor de configuración. Puede establecer el nuevo valor para este parámetro con JavaScript. Establezca el valor deseado para el parámetro *snmpWalkTimeout* en el siguiente formato:

```
thisConnector.connector.setParam("snmpWalkTimeout", "100000");
```

La modalidad Servidor y la agrupación de líneas de ensamblaje (ALPool): El proceso de creación de una línea de ensamblaje clonada se puede optimizar utilizando la Agrupación de línea de ensamblaje (ALPool). Cuando se detecta un suceso, el conector en modalidad Servidor, o bien continúa con la sección Flujo de esta línea de ensamblaje, o, si se ha configurado una agrupación de líneas de ensamblaje (ALPool) para esta línea de ensamblaje, se pone en contacto con el proceso gestor de agrupaciones para solicitar una instancia de la línea de ensamblaje disponible para manejar este suceso.

Cuando una línea de ensamblaje con un conector en modalidad Servidor utiliza la agrupación de líneas de ensamblaje, dicha agrupación ejecuta todas las instancias de la línea de ensamblaje desde el principio hasta el final. Antes de que la instancia de la línea de ensamblaje de la agrupación de líneas de ensamblaje cierre los conectores de flujo, la agrupación de líneas de ensamblaje recupera estos conectores en un conector agrupado establecido que se reutilizará en la próxima instancia de línea de ensamblaje creada por la agrupación de líneas de ensamblaje. Básicamente, la agrupación de líneas de ensamblaje utiliza el método `tcb.setRuntimeConnector()`.

Hay dos propiedades del sistema que controlan el comportamiento de la agrupación de conectores:

com.ibm.di.server.connectorpooltimeout

Esta propiedad define el tiempo de espera en segundos antes de que se libere un conjunto de conectores agrupados.

Tabla 1. Tabla de valores de la propiedad de tiempo de espera de agrupación de conectores

Valor	Significado
< 0	Inhabilitar agrupación de conectores
0	Tiempo de espera inhabilitado; los conectores de la agrupación nunca superan el tiempo de espera
> 0	Número de segundos antes de que los conectores agrupados superen el tiempo de espera

com.ibm.di.server.connectorpoolexclude

Esta propiedad define los tipos de conector que se excluyen de la agrupación. Si aparece un nombre de clase de conector en esta lista separada por comas, no se incluye en el conjunto de agrupación de conectores.

Cuando la agrupación de líneas de ensamblaje crea una instancia de línea de ensamblaje nueva, busca un conjunto de conectores agrupados disponible que, si existe, se proporciona a la nueva instancia de línea de ensamblaje como conectores proporcionados en tiempo de ejecución. De este modo se garantiza el flujo correcto de la línea de ensamblaje en general en términos de ejecución de complementos de software, correlación de atributos, etc.

Los conectores nunca se comparten. Únicamente se asignan a una sola instancia de la línea de ensamblaje cuando se utilizan.

Utilización de la modalidad Servidor: el patrón más común para utilizar un conector en modalidad Servidor es:

1. Utilizando el Editor de configuración, añadir un conector en modalidad Servidor al espacio de trabajo. Consulte el apartado "Creación de un conector" en la página 122.
2. Establecer la modalidad (Servidor) y otros parámetros de conexión para este conector en la pestaña **Conexión**; los parámetros necesarios se indican con un asterisco (*). Algunos conectores requieren que también se configure un analizador en la pestaña **Analizador**.
3. Configurar la correlación de atributos; consulte el apartado "Correlación de atributos de entrada" en la página 120.

Estos atributos correlacionados se recuperan del origen de datos, en la entrada *work*, y se pasan a los conectores de la sección Flujo de la línea de ensamblaje.

Nota:

1. Los conectores en modalidad Servidor son especiales en el sentido de que suelen tener que devolver alguna información al cliente que se conecta a ellos. Debido a esta naturaleza, la configuración de correlaciones de atributos pulsando **Descubrir atributos** en muchos casos no dará como resultado ningún esquema significativo; por lo tanto, en la mayoría de casos deberá configurar las correlaciones de atributos completamente de forma manual seleccionando **Añadir nuevo atributo**; como alternativa, suprima las innecesarias seleccionando una correlación y suprimiéndola. Los datos correlacionados de esta forma se reenvían al cliente para cada ciclo de la línea de ensamblaje; aunque, como se ha mencionado anteriormente, normalmente en una solicitud o respuesta a menudo habrá sólo un ciclo.
2. Los conectores de modalidad Servidor basados en protocolos TCP disponen de un parámetro denominado registro cronológico de reserva de conexión que controla la longitud de cola para las conexiones de entrada.
3. En la lista de componentes de la línea de ensamblaje puede ver las secciones denominadas Entradas de datos y Flujo. Los conectores en modalidad Servidor cumplen una tercera fase en el proceso de la línea de ensamblaje, la fase de *respuesta*. Los complementos de software de correlación de salida y de respuesta forman parte del mismo conector en modalidad Servidor en la pantalla, pero la ejecución del comportamiento de respuesta se efectúa una vez finalizado el proceso de la sección Flujo.

Tenga en cuenta que una llamada `system.skipEntry()`; indicará a la línea de ensamblaje que omita el comportamiento de respuesta y, por lo tanto, el conector en modalidad Servidor no dará ninguna respuesta. Si prefiere omitir simplemente los componentes de la sección Flujo restantes y, sin embargo, enviar la respuesta, en su lugar utilice la llamada `system.exitBranch("Flow");`.

Si no ha creado el conector directamente en una línea de ensamblaje, para utilizar este conector en una línea de ensamblaje, arrastre el conector de su ubicación en <espaciotrabajo>/Resources/Connectors a la sección **Entrada de datos** de una línea de ensamblaje.

Modalidad Delta:

La modalidad Delta está diseñada para simplificar la aplicación de cambios en los datos y, para ello, proporciona modificaciones incrementales en el sistema conectado, en función de los códigos de operación delta.

Los códigos de operación delta se definen mediante la función de motor delta de iterador (pestaña **Delta** para iteradores) o los conectores de detección de cambios, tales como el IBM Security Directory Server, conectores de LDAP o Active Directory o los conectores para cambios de RDBMS y Lotus/Domino; o mediante el análisis de esta información delta con los analizadores de Lightweight Directory Interchange Format (LDIF) o Directory Services Markup Language (DSML).

En versiones anteriores de IBM Security Directory Integrator V6.1, las instantáneas que se escribían en el almacén delta, una función del almacén del sistema, durante el proceso del motor delta se comprometían de inmediato. Como resultado de ello, el motor delta consideraba una entrada modificada como manejada aunque el proceso de la sección Flujo de la línea de ensamblaje hubiera fallado. Esta limitación se solventa utilizando el parámetro *Comprometer* en la pestaña Delta del conector. El valor de este parámetro controla cuando el motor delta compromete las instantáneas que se toman de los datos entrantes en el almacén del sistema.

La modalidad delta sólo está disponible para conectores LDAP y JDBC.

Nota: Un conector en modalidad Delta debe estar emparejado con otro conector que proporcione información delta, de lo contrario la modalidad Delta no tiene códigos de operación delta con los que trabajar.

Las funciones delta de IBM Security Directory Integrator (consulte el apartado Capítulo 7, "Deltas", en la página 217) se han diseñado para facilitar soluciones de sincronización. Puede considerar las posibilidades delta del sistema como si estuvieran divididas en dos secciones: *Detección y Aplicación*.

Detección delta: IBM Security Directory Integrator proporciona varios mecanismos y herramientas de detección de cambios, delta:

Motor delta

Esta función está disponible para conectores en la modalidad Iterador. Si se habilita desde la pestaña Delta del iterador, la función Motor delta utiliza el almacén del sistema para tomar una instantánea de los datos que se iteran. A continuación, en sucesivas ejecuciones, cada entrada iterada se compara con la base de datos de la instantánea (almacén delta) para comprobar qué ha cambiado.

Conector de detección de cambios

Estos componentes aprovechan la información del sistema conectado para detectar cambios y se utilizan en modalidad Iterador o Servidor, en función del conector. Por ejemplo, la modalidad Iterador se utiliza para muchos conectores de detección de cambios, tales como conectores de LDAP, registro de cambios de IBM Security Directory Server, así como conectores de detección de cambios de RDBMS, Active Directory y Notes/Domino.

Los conectores de detección de cambios se han rediseñado para que funcionen del mismo modo y para que proporcionen a la vez las mismas etiquetas de parámetros para los valores comunes. Los conectores son:

- Registro de cambios de IBM Security Directory Server
- Detección de cambios de AD (Active Directory)
- Detección de cambios de Domino
- Detección de cambios de Sun Directory (openLDAP, SunOne, iPlanet, etc.)
- Detección de cambios de RDBMS DB2, Oracle, SQL Server, etc.)
- Registro de cambios LDAP de z/OS

Nota: El sistema operativo z/OS no está soportado en IBM Security Directory Integrator a partir de la versión 7.2.

Consulte la sección sobre "Conectores" en la publicación *Referencia* para obtener más información sobre estos conectores.

La función de motor delta informa de los cambios específicos hasta los valores individuales de atributos. Este grado preciso de detección de cambios también está disponible cuando se analizan archivos LDIF. Otros componentes se limitan a informar simplemente de si se añade, se modifica o se suprime una entrada completa.

Cuando se selecciona el recuadro de selección **Permitir claves Delta duplicadas** en la pestaña Delta del iterador, se indica que se permiten claves delta duplicadas, en casos de líneas de ensamblaje de ejecución larga que necesitan procesar las mismas entradas más de una vez. Esto significa que se pueden manejar entradas duplicadas para líneas de ensamblaje que utilizan conectores de detección de cambios o de registro de cambios, conectores de modalidad Delta y almacenamientos de modalidad Delta, cuando una entrada ya se ha actualizado.

Atención: Existe la posibilidad de tener, por ejemplo, una línea de ensamblaje con varios conectores de modalidad Delta y de registro de cambios. En este caso, si el conector de modalidad Delta apunta al mismo sistema subyacente que el conector de registro de cambios, la operación delta podría volver a desencadenar el registro de cambios. Dado que no se puede diferenciar entre los cambios que se acaban de recibir y los que ha desencadenado el motor delta, debe tener consideración con atención el escenario para no entrar en un bucle infinito.

La información delta calculada por el motor delta se almacena en el objeto de entrada de trabajo y en función del componente o la función de detección de cambios que se utiliza puede almacenarse como un *código de operación de nivel de entrada*, en el *nivel de atributo* o incluso en el *nivel de valor de atributo*.

Por ejemplo, configure un conector del sistema de archivos con la función de motor delta habilitada. Póngalo en iteración sobre un documento XML sencillo que pueda modificar fácilmente en un editor de texto. Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocRoot>
  <Entry>
    <Telephone>
      <ValueTag>111-1111</ValueTag>
      <ValueTag>222-2222</ValueTag>
      <ValueTag>333-3333</ValueTag>
    </Telephone>
    <Birthdate>1958-12-24</Birthdate>
```

```

        <Title>Full-Time SDI Specialist</Title>
        <uid>jdoe</uid>
        <FullName>John Doe</FullName>
    </Entry>
</DocRoot>

```

Asegúrese de utilizar el carácter de correlación de atributos especial de correlación de todo, el asterisco (*). Este es el único atributo necesario en la correlación para asegurar que todos los atributos devueltos se correlacionan en el objeto de entrada de trabajo.

Ahora añada un componente de script con el código siguiente:

```

// Obtener los nombres de todos los atributos de trabajo como una matriz de series
var attName = work.getAttributeNames();
// Imprimir el código de operación delta de nivel de entrada
task.logmsg(" Entry ( " +
    work.getString( "FullName" ) + " ) : " +
    work.getOperation() );
// Repetir en bucle a través de todos los atributos del trabajo
for ( i = 0; i < attName.length; i++) {
    // Tomar un atributo e imprimir el código de operación de nivel de atributo
    att = work.getAttribute( attName[ i ] );
    task.logmsg("    Att ( " + attName[i] + " ) : " + att.getOperation() );
    // Ahora repetir un bucle por todos los valores de atributo e imprimir sus códigos
    // de operación
    for ( j = 0; j < att.size(); j++) {
        task.logmsg( "        Val ( " +
            att.getValue( j ) + " ) : " +
            att.getValueOperation( j ) );
    }
}

```

La primera vez que ejecuta esta línea de ensamblaje, el código del componente de script creará esta salida de registro:

```

12:46:31  Entry (John Doe) : add
12:46:31    Att ( Telephone) : replace
12:46:31      Val (111-1111) :
12:46:31      Val (222-2222) :
12:46:31      Val (333-3333) :
12:46:31    Att ( Birthdate) : replace
12:46:31      Val (1958-12-24) :
12:46:31    Att ( Title) : replace
12:46:31      Val (Full-Time SDI Specialist) :
12:46:31    Att ( uid) : replace
12:46:31      Val (jdoe) :
12:46:31    Att ( FullName) : replace
12:46:31      Val (John Doe) :

```

Dado que esta entrada no se ha encontrado en el almacén delta anterior vacío, se marca a nivel de entrada como *new* (nueva). Además, cada uno de sus atributos tiene un código *replace* (sustituir), que significa que todos los valores han cambiado (lo que tiene sentido puesto que la función delta nos indica que estos datos son nuevos).

Efectúe los siguientes cambios en su archivo XML:

1. Cambie el valor del último número de teléfono, Telephone, a 333-3334.
2. Suprima Birthdate (fecha de nacimiento).
3. Añada un nuevo atributo Address (dirección).

La configuración resultante debería ser:

```

<?xml version="1.0" encoding="UTF-8"?>
<DocRoot>
  <Entry>
    <Telephone>
      <ValueTag>111-1111</ValueTag>
      <ValueTag>222-2222</ValueTag>
      <ValueTag>333-3334</ValueTag>
    </Telephone>
    <Title>Full-Time SDI Specialist</Title>
    <uid>jdoe</uid>
    <FullName>John Doe</FullName>
    <Address>123 Willowby Lane</Address>
  </Entry>
</DocRoot>

```

Ejecute de nuevo la línea de ensamblaje. Esta vez la salida del registro se parecerá a esta:

```

13:53:22 Entry (John Doe) : modify
13:53:22   Att ( Telephone) : modify
13:53:22     Val (111-1111) : unchanged
13:53:22     Val (222-2222) : unchanged
13:53:22     Val (333-3334) : add
13:53:22     Val (333-3333) : delete
13:53:22   Att ( Birthdate) : delete
13:53:22     Val (1958-12-24) : delete
13:53:22   Att ( uid) : unchanged
13:53:22     Val (jdoe) : unchanged
13:53:22   Att ( Title) : unchanged
13:53:22     Val (Full-Time SDI Specialist) : unchanged
13:53:22   Att ( Address) : add
13:53:22     Val (123 Willowby Lane) : add
13:53:22   Att ( FullName) : unchanged
13:53:22     Val (John Doe) : unchanged

```

Ahora la entrada está marcada como *modify* (modificar) y los atributos reflejan las modificaciones en cada uno de ellos. Como puede ver, el atributo Birthdate (Fecha de nacimiento) está marcado como *delete* (suprimir) y Address (Dirección) como *add* (añadir). Éste es el motivo por el que se ha utilizado el carácter especial de correlación de todo para la correlación de entrada. Si sólo hubiese correlacionado los atributos que existían en la primera versión de este documento XML, no habríamos podido recuperar la dirección cuando apareció en la entrada.

Observe especialmente las dos últimas entradas de valor bajo el atributo Telephone (Número de teléfono), marcado como *modify* (modificar). El cambio en uno de estos valores del atributo ha producido dos elementos delta: un valor *delete* y otro *add*.

Para crear una línea de ensamblaje de sincronización de datos en versiones anteriores de IBM Security Directory Integrator, tenía que crear un script para gestionar el control de flujo. Aunque podía recibir *adiciones*, *modificaciones* y *supresiones* del componente o la función de cambio, un conector sólo podía establecerse en una de las dos modalidades de salida necesarias: Actualizar o Suprimir. Por consiguiente, o bien tenía dos conectores que apuntaban al mismo sistema de destino y colocaba el script en el complemento de software Antes de Ejecute de cada uno para pasar por alto la entrada si el código de operación no coincidía con la modalidad del componente, o bien podía tener un único conector (en modalidad Actualizar o Suprimir) en el estado *pasivo* y controlar su ejecución desde el código de script en el que comprobaba el código de la operación. Y esto implicaba que, aun cuando supiera qué había cambiado en el caso de una entrada modificada, el conector en modalidad Actualizar leía los datos originales antes de escribir los cambios de nuevo en el origen de datos. Esto puede conllevar un

tráfico de red o de origen de datos no deseable cuando sólo está cambiando un único valor en un grupo de varios valores relacionado con atributos que contienen miles de valores.

Entre la modalidad *delta* del conector.

Aplicación delta (modalidad Delta del conector): La modalidad Delta está diseñada para simplificar la aplicación de información delta; es decir, hacer los cambios reales de varios modos.

En primer lugar, la modalidad delta maneja todos los tipos de deltas: adiciones, modificaciones y supresiones. Esto reduce el número de líneas de ensamblaje de sincronización de datos a dos conectores: un conector de detección delta en la sección Entradas de datos para seleccionar los cambios y un segundo en modalidad Delta para aplicar estos cambios a un sistema de destino.

Además, la modalidad Delta aplicará la información delta al nivel más inferior soportado por el propio sistema de destino. Esto se lleva a cabo comprobando primero la interfaz del conector para ver a qué nivel de modificación incremental da soporte el origen de datos³. Si trabaja con un directorio LDAP, la modalidad delta realizará las adiciones y supresiones del valor de atributo. En el contexto de un RDBMS (JDBC) tradicional, realizar una supresión y luego una adición de un valor de columna no tiene sentido, por lo tanto esto se maneja como una sustitución de un valor de ese atributo.

Esto se realiza automáticamente mediante la modalidad Delta para los orígenes de datos que admiten esta funcionalidad⁴. Si el origen de datos ofrece llamadas optimizadas para manejar modificaciones incrementales y éstas las admite la interfaz del conector, entonces la modalidad delta las utilizará. Por otro lado, si el sistema conectado no ofrece mecanismos de actualización delta "inteligentes", la modalidad delta los simulará en la medida en que sea posible, realizando búsquedas de actualización previa (como la modalidad Actualizar), cálculos de cambios y la aplicación subsiguiente de los cambios detectados.

Criterios de enlace:

Los criterios de enlace se utilizan para indicar a un conector en modalidad Actualizar, Buscar y Suprimir cómo define la coincidencia entre atributos de datos de la línea de ensamblaje y los que se encuentran en el sistema conectado.

Se puede acceder a los criterios de enlace desde el Editor de configuración mediante la pestaña **Criterios de enlace**, que sólo está habilitado para las modalidades de conector Actualizar, Buscar y Suprimir.

Existen dos tipos de criterios de enlace, **Simple** y **Avanzado**.

Criterio de enlace simple: Para cada criterio de enlace simple, especifique el Atributo del conector (los atributos que se definen en el esquema del conector), el Operador que se ha de utilizar (por ejemplo, Contiene, Equivale a, etc.) y el Valor que se ha de utilizar con la operación. El valor que utilice se puede especificar directamente o puede hacer referencia al valor de un atributo de la entrada work que está

3. Recuerde que los únicos conectores que dan soporte a las modificaciones incrementales son los conectores LDAP y JDBC, ya que los directorios LDAP proporcionan esta funcionalidad.

4. Además, puede controlar estos comportamientos incorporados mediante parámetros de configuración y código de complemento de software.

disponible en este punto del flujo de la línea de ensamblaje. Cuando el conector efectúa la operación de búsqueda (para las modalidades Buscar, Actualizar y Suprimir) convierte el criterio de enlace en la llamada específica del origen de datos, lo que permite mantener la solución independiente de la tecnología subyacente.

Si desea crear un criterio de enlace utilizando el valor de un atributo de la entrada work, simplemente utilice el nombre del atributo en el campo Valor del criterio de enlace, precedido del símbolo del dólar (\$). Por lo tanto, si desea comparar el atributo llamado *cn* con un atributo de la entrada work llamado *FullName*, el criterio de enlace se especifica del modo siguiente:

```
cn EQUALS $FullName
```

Si desea buscar directamente a una persona específica, establezca el criterio de enlace con una constante literal:

```
cn EQUALS Joe Smith
```

Nota: el símbolo del dólar (\$) compara solamente el primer valor de un atributo de varios valores. Si desea comparar un atributo del origen de datos con cualquiera de los diferentes valores almacenados en un atributo de la entrada work, utilice el símbolo de *arroba* (@). Por ejemplo:

```
dn EQUALS @members
```

Este ejemplo intenta comparar el atributo dn del sistema conectado con cualquiera de los valores del atributo de varios valores de la entrada work llamada *members*.

Un conector puede tener definidos varios criterios de enlace que normalmente se conectan entre sí, mediante el operador booleano AND, para buscar la coincidencia.

No obstante, si selecciona en **Coincidir con cualquiera**, sólo tiene que coincidir uno de los criterios de enlace, el equivalente de una operación OR.

Tenga en cuenta que el nombre del atributo que debe coincidir puede especificarse como una expresión (consulte el apartado "Expresiones" en la página 40 para obtener más detalles). Los formatos posibles para el campo Valor de un criterio de enlace simple son:

Una serie de texto

Se correlaciona con una constante con dicho valor.

\$Name

Se corresponde con `work.getString("Name")`, es decir, el primer valor del atributo *Name*.

@Name

Se corresponde con uno de los valores del atributo de varios valores *Name*.

Una expresión de IBM Security Directory Integrator

Se describe detalladamente en el apartado "Expresiones" en la página 40.

Criterio de enlace avanzado: También puede crear su propio criterio de búsqueda personalizado activando el recuadro de selección **Crear criterio con script personalizado**. Esto le presentará un editor de scripts para que escriba su propia expresión de criterio de enlace. No todos los conectores dan soporte al criterio de enlace avanzado y es la documentación del conector la que indica si se da soporte al criterio de enlace avanzado. Consulte "Connectors" en la publicación *Referencia*.

La expresión de búsqueda que se cree debe coincidir con la sintaxis que el sistema subyacente espera. Para pasar la expresión de búsqueda al conector, debe rellenar el objeto *ret.filter* con la expresión de la serie.

Un ejemplo simple de JavaScript para un conector SQL es el siguiente:

```
ret.filter = " ID LIKE '" + work.getString("Name") + "'";
```

Este criterio de enlace personalizado presupone un ejemplo en el que el origen de datos tiene un atributo denominado *ID* (generalmente un nombre de columna) que deseamos comparar con el atributo *Name* de la entrada *Work*.

Nota:

1. La primera parte de la expresión SQL, `Select * from Table Where`, la proporciona IBM Security Directory Integrator.
2. Las comillas simples se han añadido debido a que `work.getString()` devuelve una serie, mientras que la sintaxis SQL exige comillas simples para en"cerrar"r constantes de serie.
3. La sintaxis especial con `$` y `@` no se utiliza aquí.

Errores de criterio de enlace

El error más común que se obtiene cuando se utiliza el criterio de enlace es:

```
ERROR>
La línea de ensamblaje x ha fallado porque
No se puede crear ningún criterio a partir de la entrada (no se ha especificado
ningún criterio de enlace).
```

Este error se produce cuando tiene un criterio de enlace que hace referencia a un atributo que no se puede encontrar durante la búsqueda. Por ejemplo, con el criterio de enlace siguiente:

```
uid equals $w_uid
```

La configuración del criterio de enlace falla si *w_uid* no está presente en la entrada *Work*. Esto puede ser debido a que no se ha leído en el origen de entrada (por ejemplo, no está en una correlación de entrada o falta en el origen de entrada) o a que se ha eliminado de la entrada *Work* en un script. Es decir, la llamada de la función `work.getAttribute("w_uid")` devuelve el valor `NULL`.

Para evitar esto puede escribir código en el complemento de software Antes de Execute del conector en modalidad Buscar, Suprimir o Actualizar que ignore esta operación cuando el criterio de enlace no se pueda resolver porque faltan los atributos. Por ejemplo:

```
if (work.getAttribute("w_uid") == null)
    system.ignoreEntry();
```

Es posible que las reglas de empresa requieran otro proceso como, por ejemplo, una llamada `skipEntry()` en lugar de una llamada `ignoreEntry()`, que haga que la línea de ensamblaje detenga el proceso de la entrada actual y comience desde la parte superior en una nueva iteración. La función `ignoreEntry()` simplemente ignora el conector actual y continúa con el resto de la línea de ensamblaje.

Funciones

Una *función*, a menudo referida como un componente de función (FC), es un componente muy parecido a un conector, salvo que no tiene un valor de modalidad. Mientras los conectores proporcionan verbos de acceso estándar para

los sistemas conectados (Buscar, Suprimir, Actualizar, etc.), las funciones sólo realizan una operación, como pasar datos a través del analizador, entregar trabajo a otra línea de ensamblaje o realizar una llamada de servicio web.

Visión general

Las funciones aparecen en cualquier lugar de la sección Flujo de una línea de ensamblaje. La carpeta de biblioteca Funciones del navegador de configuración se puede utilizar para gestionar la biblioteca de componentes de función.

Del mismo modo que los conectores, las funciones de las líneas de ensamblaje proporcionan una opción Inicializar para determinar en qué momento se inicia un componente. De forma predeterminada, las funciones se inicializan cuando se inicia la línea de ensamblaje.

Componentes de script

El componente de script (SC) es un bloque de código JavaScript definido por el usuario que puede soltarse en cualquier lugar de la lista de flujo de datos de la línea de ensamblaje, junto con los conectores y los componentes de función, lo que hace que se ejecute el código de script existente para cada ciclo en este punto del flujo de trabajo de la línea de ensamblaje.

Visión general

A diferencia de los complementos de software, los componentes de script se mueven rápidamente alrededor del flujo de la línea de ensamblaje, lo que les convierte en herramientas de grandes prestaciones para distintos tipos de operaciones, desde depurar a crear prototipos e implementar su propia lógica de flujo. Además, a diferencia de otros tipos de componentes de línea de ensamblaje, el script no tiene ningún comportamiento ni ninguna modalidad definida; puede implementar el comportamiento y la modalidad con el script. Una biblioteca de scripts puede almacenarse en la carpeta de biblioteca de scripts del navegador de configuración.

Utilización

Por ejemplo, si desea probar y depurar sólo una parte de una línea de ensamblaje, podría incluir el código siguiente en un componente de script para limitar y controlar el flujo de la línea de ensamblaje.

```
task.dumpEntry( work );  
system.skipEntry();
```

Cuando está situado en el punto adecuado del flujo de una línea de ensamblaje, este componente de script muestra el contenido del objeto work grabándolo en la salida del registro cronológico y, a continuación, omite el resto de la línea de ensamblaje para este ciclo. Subiendo o bajando el componente de script en la lista de componentes, controlará qué parte de la línea de ensamblaje se ejecuta realmente. Si cambia la llamada `system.skipEntry()` por `system.skipTo("NombreComponenteAL")`, pasa directamente el control a un componente específico de la línea de ensamblaje.

También puede utilizar los componentes de script para controlar otros componentes. Un caso típico cuando se realiza una sincronización del directorio o de la base de datos es el de tener que manejar la información actualizada y la suprimida. Dado que los conectores que funcionan con el flujo de trabajo de la

línea de ensamblaje incorporado sólo pueden operar en una modalidad a la vez (Actualizar o Suprimir) tendrá que ampliar un poco esta lógica con su propio código. Un método es añadir dos conectores, uno en modalidad Actualizar y otro en modalidad Suprimir y, a continuación, incluir el código en el complemento de software Antes de Execute de cada conector para indicarle que omita operaciones de cambio que no debe manejar. Por ejemplo, en el complemento de software Antes de Execute del conector en modalidad Actualizar debería escribir algo similar a esto:

```
// El registro cronológico de cambios LDAP contiene un atributo denominado "changeType"
if (work.getString("changeType").equals("delete"))
    system.ignoreEntry();
```

Esto provocará que el conector en modalidad Actualizar omita las entradas suprimidas. Podría tener un código complementario en el complemento de software Antes de Execute del conector en modalidad Suprimir, omitiendo todo excepto las supresiones.

No obstante, si va a sincronizar actualizaciones en varios destinos, se requerirán dos conectores por cada origen de datos. Otro procedimiento es tener un solo conector en estado Pasivo que se active desde el script. A modo de ejemplo, supongamos que tiene un conector de línea de ensamblaje pasivo denominado *synchIDS*. A continuación, puede añadir un componente de script con el siguiente código para ejecutarlo:⁵

```
if (work.getString("changeType").equals("delete"))
    synchIDS.deleteEntry( work )
else
    synchIDS.update( work );
```

Si etiqueta claramente el componente de script, indicando que ejecuta los conectores pasivos, el resultado serán líneas de ensamblaje más cortas y más sencillas de leer y mantener. Este es un ejemplo de tener que elegir uno de los dos métodos más recomendados: mantener corta la longitud de la línea de ensamblaje y utilizar la lógica incorporada frente a scripts personalizados. No obstante, en este caso los objetivos de legibilidad y simplicidad se cumplen mejor escribiendo un pequeño script.

El componente de script también es muy útil si desea probar la lógica y el código de script. Basta con crear una línea de ensamblaje con un solo componente de script en la lista del flujo de datos, incluirla en el código y ejecutarla.

Consulte también

Capítulo 2, “Scripts en IBM Security Directory Integrator”, en la página 49.

Correlaciones de atributos

Las correlaciones de atributos son rutas por las que los datos deben entrar o salir de la línea de ensamblaje. Las correlaciones de atributos aparecen en los conectores y las funciones como correlaciones de entrada y de salida y también están disponibles como componentes autónomos de la línea de ensamblaje.

5. Dado que los conectores pasivos no los ejecuta la lógica de línea de ensamblaje, no importa dónde aparezcan en la lista del flujo de datos.

Visión general

El diagrama que hay al principio del apartado titulado “Línea de ensamblaje.” en la página 1 describe tres correlaciones de atributos como flechas curvadas: dos correlaciones de entrada que proporcionan datos a la línea de ensamblaje, además de una correlación de salida que pasa datos a la memoria caché del conector (la entrada *conn*) para que puedan grabarse.

Cada correlación de atributos contiene una lista reglas que crean atributos en la entrada *work* o la entrada *conn*. Una regla de correlación especifica dos cosas:

1. El nombre del atributo que debe crearse (o sobregrabarse) en la entrada de destino. Es la entrada *work*, en el caso de correlaciones de entrada y componentes de correlación de atributos libres, mientras las correlaciones de salida tienen como destino la entrada *conn*.
2. La asignación que se utiliza para llenar el atributo con uno o más valores. Y la asignación puede ser un script de asignación como, por ejemplo:

```
work.Title
```

o puede ser un texto literal (que incluya caracteres de nueva línea) con sustitución de señal opcional:

```
<html>
  <header>
    <title>{work.Title}</title>
  </header>
  <body>
    <p>Busque el valor del atributo "Title" en el título de esta página</p>
  </body>
</html>
```

Las correlaciones de atributos se crean utilizando el Editor de configuración; consulte el apartado “Correlación de atributos y esquema” en la página 115.

Las correlaciones de atributos dan soporte a la herencia, en el nivel de correlación y para reglas de correlación individuales. Tenga en cuenta que puede arrastrar un script a una correlación de atributos para configurar una regla de correlación de JavaScript heredada.

Las correlaciones de atributos también proporcionan una funcionalidad llamada “Comportamiento ante un valor nulo”, que se utiliza para controlar cómo se manejan los datos que faltan.

Consulte el apartado “Modelo de datos interno: entradas, atributos y valores” en la página 50 para obtener más información sobre correlaciones de atributos.

Comportamiento ante un valor nulo

Ocasionalmente, el sistema intenta correlacionar un atributo que falta. Por ejemplo, si falta un número de teléfono opcional en el origen de datos de entrada o si se suprime un atributo de la correlación de salida de la entrada *work*. Otras veces, aunque esté presente un atributo, no tiene valores - como una columna anulable de una tabla de base de datos.

Distintos orígenes de datos tratan los valores que faltan de formas diferentes (valor nulo, serie vacía) y la función que se describe en esta sección proporciona una forma de personalizar cómo se tratan los atributos (o valores de atributos) que faltan. Esta función se conoce como *comportamiento ante un valor nulo* y, con ella, puede definir tanto un "valor nulo" como el modo en que debe manejarse.

Nota: El conector JDBC tiene el valor del parámetro `jdbcExposeNullValues` que le permite correlacionar los valores nulos con los atributos que faltan (consulte "JDBC Connector" en la publicación *Referencia*).

El comportamiento ante un valor nulo puede especificarse en distintos niveles: sistema, configuración, línea de ensamblaje, correlación de atributos y atributo. No obstante, puesto que el comportamiento ante un valor nulo es, en gran medida, específico del origen de datos, tiene más sentido definir esta propiedad del sistema en el nivel de correlación de atributos (por ejemplo, para todos los atributos gestionados por las correlaciones de entrada o salida de un conector). A continuación se describen los niveles posibles indicados:

Nivel de sistema

El comportamiento ante un valor nulo de nivel de sistema se especifica en el archivo `global.properties` estableciendo las propiedades `rsadmin.attribute.nullBehavior` y `rsadmin.attribute.nullDefinition`, uno de los valores que se indican más adelante en este apartado.

Nivel de configuración

Este nivel sobrescribe el comportamiento ante un valor nulo de nivel de sistema, y se configura estableciendo las propiedades `rsadmin.attribute.nullBehavior` o `rsadmin.attribute.nullDefinition` de un almacén de propiedades en uno de los valores que se indican más adelante en este apartado.

Nivel de línea de ensamblaje

Para especificar el comportamiento ante un valor nulo de línea de ensamblaje debe pulsar el botón **Opciones...** en la barra de herramientas de la línea de ensamblaje, seleccionar **Valores de línea de ensamblaje** y pulsar **Comportamiento ante un valor nulo** en el recuadro de diálogo que aparece.

Nivel de correlación de atributos

Para definir el comportamiento ante un valor nulo para todos los atributos de una correlación, debe pulsar el botón **Más...** situado encima de la lista de correlación de atributos y seleccionar **Comportamiento ante un valor nulo**.

Nivel de atributo

El comportamiento ante un valor nulo se puede configurar para un atributo específico pulsando el botón derecho del ratón sobre él en una correlación de atributos y seleccionando **Comportamiento ante un valor nulo**. Cuando esto se define para un atributo, esto se indica mediante la presencia de un símbolo de viñeta azul en el elemento correlacionado.

El comportamiento ante un valor nulo admite cinco valores distintos para definir un valor nulo, como se muestra más adelante (Observe que el texto entre paréntesis para cada valor es el valor que se utiliza para establecer la definición del comportamiento ante un valor nulo en el nivel del sistema y generalmente se define en el almacén de propiedades global o de la solución). Cada valor muestra el valor de la propiedad real entre paréntesis. Estas definiciones se listan en orden inclusivo, por lo que el segundo caso también incluye el primero, el tercero incluye los dos primeros, y así sucesivamente:

Falta un atributo (AbsentAttribute)

El atributo al que se hace referencia como origen de los valores en una correlación de atributos no existe.

Atributo sin valores (EmptyAttribute)

El atributo que se utiliza como origen de los valores de un atributo se encuentra, pero no tiene valores. El caso anterior también se comprueba.

El atributo contiene un valor de serie vacía (EmptyString)

El atributo se ha encontrado, pero sólo tiene un valor de serie vacía.

Valor (value)

El atributo contiene el valor especificado. Para la definición de un valor nulo a nivel de línea de ensamblaje, correlación de atributos y atributo, este valor se establece en el campo **Valor** del diálogo Comportamiento ante un valor nulo. Aquí puede especificar varios valores de atributo, si lo desea, colocándolos en líneas diferentes. Si utiliza la propiedad `rsadmin.attribute.nullDefinition` para definir el valor en el nivel de sistema y configuración, también debe establecer la propiedad `rsadmin.attribute.nullDefinitionValue`.

Nota: Se han realizado varias mejoras en el conector de servidor HTTP. Los componentes basados en TCP, como el conector de servidor HTTP, disponen de un conmutador en sus pantallas de configuración para devolver cabeceras TCP como valores de atributos. Cuando este distintivo se deselecciona, las cabeceras TCP se almacenan como propiedades en el objeto de entrada devuelto.

Comportamiento predeterminado (Default Behavior)

La definición del valor nulo debe heredarse de un nivel superior. Por ejemplo, un atributo hereda su definición de valor nulo del valor de correlación de atributos, que a su vez lo hereda de la línea de ensamblaje.

Nota: el comportamiento ante un valor nulo a nivel de configuración sobrescribe cualquier valor a nivel de sistema. Asimismo, el comportamiento predeterminado a nivel de sistema es igual que especificar **delete**, mientras que en el nivel de configuración es equivalente a **value**.

La función Comportamiento ante un valor nulo también le permite definir la acción que debe llevarse a cabo si se detecta un valor nulo:

Serie vacía (empty string)

Los atributos que faltan se correlacionan con un solo valor que tiene un valor de serie vacía ("").

Valor nulo (null)

Los atributos que faltan no se correlacionan con ningún valor, lo que significa que la llamada `att.getValue()` devuelve **null**.

Suprimir (delete)

El atributo se suprime de la correlación.

Valor (value)

Los atributos que faltan se correlacionan con un valor especificado. Para el comportamiento ante un valor nulo de nivel de línea de ensamblaje, correlación de atributos y atributo, los valores se establecen en el campo **Valor** del diálogo Comportamiento ante un valor nulo. Aquí puede especificar varios valores de atributo, si lo desea, colocándolos en líneas diferentes. Si utiliza la propiedad `rsadmin.attribute.nullBehavior` para definir el valor en el nivel del sistema y configuración, también debe establecer la propiedad `rsadmin.attribute.nullBehaviorValue`.

Comportamiento predeterminado (Default Behavior)

El comportamiento ante un valor nulo debe heredarse de un nivel superior.

Por ejemplo, el nivel de atributo hereda de la correlación de atributos que, a su vez, hereda del valor de línea de ensamblaje.

Nota: el comportamiento ante un valor nulo a nivel de configuración sobrescribe cualquier valor a nivel de sistema. Asimismo, el comportamiento ante un valor nulo a nivel de sistema es igual que especificar **delete**, mientras que en el nivel de configuración es equivalente a **value**.

Componentes de rama

Los componentes de rama afectan el orden en el que se ejecutan los demás componentes como, por ejemplo, conectores, scripts, funciones, correlaciones de atributos y otros componentes de rama en el flujo de la línea de ensamblaje.

Visión general

Los componentes de rama pueden ser de tres tipos:

- Simples (también denominados únicamente Ramas)
- Bucles (ramas que forman un bucle)
- Conmutadores (ramas que comparten la misma expresión)

Las ramas pueden aparecer en cualquier lugar de la sección Flujo, pero no existe ninguna carpeta de biblioteca para éstas en la sección Recursos del espacio de trabajo.

Una rama no tiene que ejecutarse hasta completarse; se utiliza la misma llamada de script para salir mediante programación de cualquier tipo de rama: `system.exitBranch()`. Consulte el apartado "Salida de una rama (o un bucle o el flujo de la línea de ensamblaje)" en la página 31 para obtener más información.

Los tres tipos de componente de rama son:

Rama

Cada tipo de rama le permite definir rutas alternativas para el proceso de las líneas de ensamblaje; este tipo, que es el más simple, determina la acción para el caso en que: "Si se produce esta situación, debe realizarse esta acción." Debe definir qué significa "se produce esta situación" estableciendo las condiciones que deben cumplirse; por ejemplo, comparando valores de datos o verificando el resultado de alguna operación. Si las condiciones se cumplen, se ejecutan los componentes conectados a esta rama.

La rama le permite definir condiciones basadas en cualquier dato del servidor de IBM Security Directory Integrator: valores de atributo, valores de parámetros, propiedades a las que se puede acceder externamente y cualquier otro tipo de información disponible a través de JavaScript, como llamadas del sistema operativo para el uso del disco o de la memoria. En muchas de las condiciones se utiliza el operador AND u OR, en función del valor del recuadro de selección **Coincidir con cualquiera**.

Esta forma más simple de componente de rama también da soporte a tres valores de subtipo, IF, ELSE-IF y ELSE, que puede seleccionar.

IF Puede aparecer en cualquier lugar dentro del flujo de la línea de ensamblaje; la rama IF proporciona una vía alternativa que el proceso debe seguir si se cumplen las condiciones indicadas. Una

vez que se han ejecutado los componentes de la rama, el control pasa al primer componente que se encuentra *después* de esta rama. Si no desea que esto ocurra, debe añadir una rama ELSE o ELSE IF, o salir de la rama con una llamada de script a `system.exitBranch()`.

ELSE-IF

Esta rama es idéntica a la rama IF, pero sólo puede aparecer inmediatamente después de una rama IF o ELSE-IF.

ELSE Sólo puede aparecer inmediatamente después de una rama IF o ELSE-IF y no tiene condiciones. Sus componentes sólo se procesan si no se cumple ninguna de las ramas IF o ELSE-IF precedentes. Además, la instancia de ELSE siempre se evalúa como true, de modo que no se evalúan condiciones durante el ciclo.

Como se ha indicado anteriormente, puede salir de forma prematura de una rama mediante un script, utilizando `system.exitBranch()`.

Bucle

El componente de bucle proporciona funciones para añadir la lógica de ciclo dentro de una línea de ensamblaje. Los bucles se pueden configurar para tres modalidades de funcionamiento; se pueden basar en condiciones, se pueden basar en un conector o se pueden basar en los valores de un atributo:

Condicional

Del mismo modo que con una rama simple, puede definir condiciones que controlen el comportamiento del bucle. El bucle continuará con el ciclo mientras se vayan cumpliendo las condiciones y se detendrá tan pronto como dejen de cumplirse. La ventana de detalles para los bucles es la misma que la ventana de las ramas simples descrita en la sección anterior.

Conector

Este método permite configurar un conector en modalidad Iterador o Buscar y recorrerá el flujo del bucle para cada una de las entradas devueltas. La ventana de detalles de este tipo de bucle contiene las pestañas de conector necesarias para configurarlo, para conectar y descubrir atributos y para configurar la correlación de entrada.

Recuerde que tiene un parámetro llamado **Opciones de inicialización** mediante el cual puede indicar a la línea de ensamblaje que puede:

- **No hacer nada** significa que el conector no se preparará de ningún modo entre ciclos de línea de ensamblaje.
- **Inicializar y seleccionar/buscar** provoca que se reinicialice el conector para cada ciclo de la línea de ensamblaje.
- **Sólo seleccionar/buscar**, que mantiene inicializado el conector, pero rehace la selección del iterador o la búsqueda, en función del valor de la modalidad.

Recuerde también que existe una pestaña **Parámetros del conector** que funciona de forma parecida a una correlación de salida en el sentido que puede seleccionar los parámetros de conector que deben definirse a partir de los valores del atributo **work**.

Esto nos lleva a hablar de las diferencias entre los bucles en modalidad Iterador y los bucles en modalidad Buscar. Ambas opciones realizan *búsquedas* que crean un conjunto de resultados que se devuelve para el bucle. En modalidad Iterador, el conjunto de resultados se controla exclusivamente mediante los valores de los parámetros de este componente. La modalidad Buscar, en cambio, utiliza criterios de enlace para definir reglas de búsqueda o de coincidencia. Teniendo en cuenta que evita tener que codificar complementos de software del tipo En caso de ninguna coincidencia o En caso de múltiples coincidencias, éste es el método preferido para realizar búsquedas que es posible que no siempre devuelvan una única entrada coincidente.

Valor de atributo

Seleccionando cualquier atributo disponible en la entrada work, se ejecutará el flujo del bucle para cada uno de sus valores. Cada valor se pasa al bucle en un nuevo atributo de entrada work nombrado en el segundo parámetro. Esta opción permite trabajar fácilmente con atributos con varios valores, como las listas de miembros de grupos o el correo electrónico.

Puede salir de forma prematura de un bucle mediante un script, utilizando `system.exitBranch()`.

Conmutador

A diferencia de las expresiones que se utilizan en las condiciones de ramas y bucles, la expresión de conmutación puede dar lugar a más valores a parte de true o false. Por ejemplo, puede activar el valor de un atributo o la operación solicitada cuando se llamó esta línea de ensamblaje desde otra línea de ensamblaje o desde otro proceso. En el componente de conmutador se añade un caso para cada valor constante de la expresión de conmutador que se desea gestionar. Por consiguiente, si, por ejemplo, configura el conmutador para que utilice un código de operación delta en la entrada work, los casos serían para valores de tipo "add", "delete" y "modify".

En una construcción de caso de conmutador de una línea de ensamblaje, puede haber varios casos activos a la vez. IBM Security Directory Integrator comprueba cada caso, del mismo modo que haría con una serie de ramas IF estándar. A continuación se muestra un ejemplo de varios casos:

```
work.setAttribute("test","abc");

Switch work.test
  Case startsWith("a"): this is true
  Case contains ("bc"): this is true
  Case length=3: this is true
```

Las tres expresiones work.test del conmutador que son true desencadenarán la ejecución del conmutador.

Puede salir de forma prematura de un caso de conmutador mediante un script, utilizando `system.exitBranch()`;

Salida de una rama (o un bucle o el flujo de la línea de ensamblaje)

Si desea salir de una rama, un bucle o un conmutador, o incluso de ramas incorporadas como la sección Flujo de la línea de ensamblaje, utilice el método `system.exitBranch()` desde un lugar en el que pueda ejecutar un script como, por

ejemplo, un complemento de software o incluso un componente de script. La llamada a `system.exitBranch()` sin parámetros (o con una serie vacía) provocará que la rama que contiene salga y el flujo seguirá con el primer componente después de la rama.

También puede proporcionar al método un parámetro tipo string que puede contener:

Una de las palabras clave reservadas: Branch, Loop, Flow, Cycle o AssemblyLine (se tienen en cuenta las mayúsculas y minúsculas)

Esto interrumpirá la primera rama de este tipo, llevando un seguimiento hacia atrás hasta la línea de ensamblaje. Por lo tanto si el código de script está en una rama dentro de un bucle y ejecuta la llamada `system.exitBranch("Loop")`, saldrá de la rama y del bucle que lo contiene. Si se utiliza la palabra reservada Flow, el flujo sale de la sección Flujo de la línea de ensamblaje y continúa con el comportamiento de respuesta en el caso de un conector de modalidad Servidor, con un iterador activo para leer la entrada siguiente o con el cierre de la línea de ensamblaje (epílogos, ...). La palabra clave Cycle pasa el control al final del ciclo de la línea de ensamblaje actual y no invoca el comportamiento de respuesta en los conectores de modalidad Servidor, mientras que la palabra clave AssemblyLine hará que la línea de ensamblaje se detenga y se cierre.

Todos los demás valores utilizados en la llamada `system.exitBranch()` provocarán una interrupción de la rama o del bucle cuyo nombre se haya especificado. Así, por ejemplo, la llamada `system.exitBranch("IF_LookupOk")` envía el flujo después de la rama o del bucle denominado "IF_LookupOk". Observe que, a diferencia de `system.skipTo()`, que pasará el control a cualquier componente de la línea de ensamblaje indicada, `system.exitBranch()` hará que el proceso continúe después del bucle o la rama especificados.

El nombre de la rama o bucle (sensible a las mayúsculas y minúsculas)

Si pasa el nombre de una rama o bucle en el que la llamada al script está anidada, se pasará el control al componente a continuación de la línea de ensamblaje. Si no se encuentra ninguna rama o bucle con este nombre, el rastreo hacia atrás desde el punto de la llamada, se producirá un error.

También existe la función *continue* (continuar) en los componentes de bucle. Los métodos siguientes están disponibles en el objeto del sistema:

```
system.continueLoop();  
system.continueLoop(nombre);
```

donde *nombre* es una serie sensible a las mayúsculas y minúsculas, que indica un nombre de bucle. En caso de que se proporcione un nombre de bucle, el flujo del programa se transfiere al componente de bucle que tiene ese nombre.

Analizadores

Los analizadores se utilizan junto con un componente de corriente de bytes, por ejemplo, un conector del sistema de archivos, para interpretar o generar la estructura del contenido que se está leyendo o escribiendo.

Tenga en cuenta que cuando la corriente de bytes que se intenta analizar no está coordinada con el analizador seleccionado, se obtiene una excepción `sun.io.MalformedInputException`. Por ejemplo, se puede mostrar este mensaje de error cuando se utilice la pestaña **Correlación de entrada** para examinar un archivo.

El Editor de configuración proporciona dos ubicaciones en las que pueden seleccionarse analizadores:

1. En la pestaña **Analizador** de un conector de corriente de bytes.
2. Desde sus propios scripts; por ejemplo, complementos de software y componentes de script.

Para obtener más información acerca de los analizadores individuales, consulte "Parsers" en la publicación *Referencia*.

Conversión de la codificación de caracteres

Java2 utiliza Unicode como su codificación de caracteres interna. Unicode es un juego de caracteres de doble byte. Cuando trabaja con cadenas y caracteres en las líneas de ensamblaje y conectores, siempre se presupone que están en Unicode. La mayor parte de los conectores proporcionan algún medio de conversión de la codificación de caracteres. Cuando se leen archivos de texto del sistema local, Java2 ya tiene establecida una conversión de la codificación de caracteres que depende de la plataforma que esté ejecutando.

El servidor de IBM Security Directory Integrator tiene la opción de línea de mandatos **-n**, que especifica el juego de caracteres de los archivos de configuración que utilizará cuando escriba archivos nuevos; también incorpora este designador del juego de caracteres en el archivo, para que más tarde, cuando lea de nuevo el archivo, pueda interpretarlo correctamente.

No obstante, en algunas ocasiones leerá o escribirá datos de los archivos de texto o en ellos cuya información esté codificada en codificaciones distintas de caracteres. Por ejemplo, los conectores que requieren un analizador generalmente aceptan un parámetro **Character Set** en la configuración del analizador. Este parámetro se debe establecer con una de las tablas de conversión aceptadas según la especificación del registro del juego de caracteres IANA (<http://www.iana.org/assignments/character-sets>).

Algunos archivos, cuando tienen código UTF-8, UTF-16 o UTF-32, podrían contener un BOM (Byte Order Marker - Marcador de orden de bytes) al principio del archivo. Un BOM es la codificación de los caracteres 0xFEFF. Esto puede utilizarse como una firma del código utilizado. Pero el conector de archivos de IBM Security Directory Integrator no reconoce el BOM.

Si intenta leer un archivo con un BOM, debería añadir este código a, por ejemplo, el complemento de software Antes de selección del conector:

```
var bom = thisConnector.connector.getParser().getReader().read(); // omite el BOM = 65279
```

Este código leerá y omitirá el BOM, suponiendo que haya especificado el juego de caracteres correcto para el analizador.

Debe tener cuidado con el protocolo HTTP; consulte en la publicación *Referencia*, el apartado que trata sobre la codificación de juegos de caracteres en la descripción del analizador de HTTP para obtener más detalles.

Acceso a sus propias clases Java

Puede acceder a sus propias clases Java personalizadas desde la infraestructura de IBM Security Directory Integrator, siempre y cuando se trate de clases y métodos *públicos*. Debe empaquetar estas bibliotecas en un archivo .jar o .zip y, a continuación, colocarlas en el directorio TDI_install/jars, preferiblemente en su propio subdirectorio. También puede utilizar la variable CLASSPATH o la carpeta

de extensiones de entorno de tiempo de ejecución Java, aunque estos dos métodos no se recomiendan. Estos métodos le permiten llamar a clases desde sus propias clases en caso de que el cargador cargue las clases antes de cargar las suyas.

Si está ejecutando el servidor desde el Editor de configuración, debe reiniciar el Editor de configuración para que detecte las clases nuevas del directorio y subdirectorios de `TDI_install/jars`.

Después de colocar los archivos `.jar` en el subdirectorio `jars`, puede crear una instancia de la clase a la que hacer referencia desde IBM Security Directory Integrator. Observe que el componente de función Java le permite abrir archivos `.jar`, examinar objetos contenidos en estos archivos, así como sus métodos. Una vez que haya seleccionado la función que debe llamarse, el componente de función prepara el esquema de entrada y salida para establecer coincidencia con los parámetros que necesita la función Java.

Para obtener más información sobre cómo llamar a clases Java desde un script, consulte el apartado "Creación de instancias de una clase de Java" en la página 80.

Creación de instancias de clases con el Editor de configuración

Utilice la carpeta Bibliotecas de Java de la ventana **Registro de soluciones y valores** del Editor de configuración para declarar las clases. Esto sólo funciona si la clase tiene un constructor que no es de argumentos, generalmente, aunque no siempre, es el constructor predeterminado.

Cuando añada un objeto de clase, pulse **Añadir...** y especifique dos parámetros: el nombre del objeto de script, es decir, el nombre de la variable de script que es una instancia de su clase Java, y el nombre de la clase Java. Por ejemplo, puede tener un nombre de objeto de script `mycls` mientras que la clase Java puede ser `my.java.classname`. El objeto `mycls` estará disponible para cualquier línea de ensamblaje definida antes de que se ejecuten los prólogos globales.

Nota: Debe tener en cuenta que esto hace que se cree una instancia del objeto para cada una de las líneas de ensamblaje que se ejecute. Si este no es el comportamiento deseado y prefiere que se creen las instancias bajo demanda, consulte el apartado siguiente.

Creación de instancias de las clases en tiempo de ejecución

Si desea crear una instancia de su clase en un momento determinado de la ejecución o para clases que no tienen constructores que no son de argumentos, debe crear dicha instancia durante el tiempo de ejecución. Por ejemplo:

```
cryptoLib = new com.acme.myCryptoLib();
```

Flujo y complementos de software de una línea de ensamblaje

Las líneas de ensamblaje proporcionan un comportamiento automatizado incorporado que le ayuda a crear y desplegar rápidamente los flujos de datos. Este comportamiento automatizado se detalla en los diagramas de flujo en la publicación *Referencia*. Además, los conectores y las funciones tienen sus propios comportamientos, que también se muestran en los diagramas de flujo. Tenga en cuenta que el comportamiento del conector depende del valor de la modalidad.

A lo largo de estos flujos de lógica incorporados existen varios *puntos de paso* en los que puede añadir su propia lógica con scripts para ampliar el comportamiento incorporado o para sobrescribir dicho comportamiento por completo. Estos puntos de paso se denominan "complementos de software" y están disponibles para ser

personalizados en la pestaña Complementos de software de todos los conectores y todas las funciones, así como en la línea de ensamblaje propiamente dicha.

Puede habilitar o inhabilitar complementos de software según si cada complemento de software en particular es aplicable a la línea de ensamblaje que está ejecutando. Al inhabilitar un complemento de software, no se rompe su herencia en el conector del cual forma parte.

Cuando se inicia una línea de ensamblaje, ésta pasa por tres fases: arranque, flujo de datos y cierre. Durante el arranque, existen complementos de software de prólogo disponibles para configurar de nuevo los componentes antes de que se inicialicen. En la fase de flujo de datos, cada entrada work que se proporciona a la línea de ensamblaje se pasa a los componentes de flujo para su proceso.

Por último, durante el cierre, pueden utilizarse complementos de software de epílogo para llevar a cabo la fase final del trabajo, como verificar y notificar estados de error, u ordenar los datos sobre el estado para la próxima vez que se inicie la línea de ensamblaje.

Fase de arranque

En este punto, el servidor recibe instrucciones para cargar y ejecutar una línea de ensamblaje. El servidor utiliza el anteproyecto almacenado en el archivo de configuración para configurar la línea de ensamblaje. Si se pasa un TCB (bloque de llamadas de tareas) a la línea de ensamblaje, se evalúa su contenido (lo que puede conllevar cambios en los parámetros del componente de la línea de ensamblaje). En este punto, los flujos de los complementos de software de prólogo están iniciados.

Prólogos globales

En primer lugar, se evalúan los prólogos globales que se hayan definido. Los prólogos globales son scripts de la carpeta Recursos del proyecto que se ha incluido en la línea de ensamblaje (en la ventana **Registro de soluciones y valores**). Esto generalmente se lleva a cabo en la ventana Valores de línea de ensamblaje de la línea de ensamblaje, seleccionando los scripts que deben ejecutarse en la fase de arranque de la línea de ensamblaje. Una vez que se han finalizado todos los prólogos globales, se llama a los complementos de software de prólogo de la línea de ensamblaje.

Complementos de software de prólogo de línea de ensamblaje (Antes de la inicialización)

Primero se invoca el complemento de software Prólogo - Antes de Initialize de la línea de ensamblaje. A continuación, todos los conectores y las funciones que están configurados para inicializarse durante el "arranque" pasan por su fase de inicialización, en la que también se invocan sus complementos de software de prólogo, como se puede ver en el punto siguiente.

Inicialización de conectores/funciones

La secuencia de inicialización se lleva a cabo para cada uno de los conectores y las funciones cuyo valor de inicialización está establecido en "durante el inicio". Se inician uno tras otro según el orden establecido en la línea de ensamblaje. Para cada conector o función, el flujo es el siguiente:

1. Se llama al complemento de software **Prólogo – Antes de Initialize** del componente.

2. El componente se inicia; por ejemplo, estableciendo conexión con su origen de datos subyacente, su destino o API.
3. Si se trata de un conector en modalidad Iterador, se procesa el complemento de software **Prólogo – Antes de selección**, y el conector realiza la selección de entradas; desencadena una llamada específica del origen de datos, como ejecutar una sentencia SQL SELECT o realizar una búsqueda LDAP.
4. Para los iteradores, se evalúa el complemento de software **Prólogo – Después de selección**.
5. Se llama al complemento de software **Prólogo – Después de Initialize**, terminando así la secuencia.

Si falla la inicialización de un conector, entonces el flujo de la línea de ensamblaje pasa al complemento de software **Prólogo – En caso de error** donde puede tratar este error.

La función de reconexión permite configurar un conector para que intente automáticamente volver a establecer su conexión si se produce un error durante la configuración o el acceso a datos. Estos valores se encuentran en la pestaña **Error de conexión** del conector.

Nota: los conectores de script, es decir, los conectores que se implementan utilizando JavaScript, se evalúan en esta fase, de modo que se registran las funciones de conector necesarias y se ejecuta el código de inicialización.

Complementos de software de prólogo de la línea de ensamblaje (Después de Initialize)

Se ejecuta el complemento de software **Prólogo - Después de Initialize de la línea de ensamblaje**. La finalización de este complemento de software indica el final de la fase de arranque y el inicio de la fase de flujo de datos.

Fase de flujo de datos

Complemento de software Inicio del ciclo de la línea de ensamblaje

Este complemento de software se invoca al iniciar cada ciclo antes de los componentes de Entradas de datos o de Flujo.

Ciclo de la línea de ensamblaje

El control se pasa al primer componente del flujo, generalmente un conector en modalidad Servidor o Iterador de la sección Entradas de datos.

Si tiene uno o más iteradores en la línea de ensamblaje, el primero inicia el ciclo recuperando la siguiente entrada de su conjunto de resultados y correlacionando los atributos con la entrada work. La entrada work resultante se pasa a los componentes de la sección Flujo, empezando por el principio de la lista como se ha visto en el Editor de configuración (CE).

Para los conectores de modalidad Servidor, se inicia un proceso de escucha que espera la llegada de conexiones de cliente entrantes. Cuando se detecta una solicitud de conexión, el conector se clona a sí mismo, acepta la conexión y luego pasa automáticamente a la modalidad Iterador para poder pasar datos del cliente a la sección Flujo para su proceso. De cualquier modo, se obtiene un iterador que dirige las entradas work a los componentes de flujo.

(Entretanto, el conector original en modalidad Servidor espera la llegada de solicitudes de conexión entrantes adicionales.)

Si su línea de ensamblaje no tiene ningún conector en modalidad Iterador ni en modalidad Servidor, usted tiene una línea de ensamblaje de un sólo uso, utilizadas normalmente para procesar una entrada de trabajo inicial (IWE) enviada por otro proceso de llamada.

Fin de ciclo

Cuando se ejecuta el último componente de flujo, puede pasar una de las tres cosas siguientes:

- Si la entrada work actual provenía de un iterador, se devuelve el control al iterador para obtener la entrada siguiente de su origen.
- En el caso de un conector en modalidad Servidor, se da una respuesta al cliente.
- En el caso de una línea de ensamblaje que se ha llamado en modalidad de ciclo manual, la hebra se devuelve al llamante de modo que se pueda acceder a los resultados.

No existe ningún complemento de software específico en este punto, aunque puede añadirse a la línea de ensamblaje insertando un script al final.

Fin de los datos

Fin de los datos es un complemento de software de modalidad Iterador que se llama cuando se alcanza el final del conjunto de datos de entrada. En este punto, o bien se pasa el control al siguiente conector de entradas de datos o la línea de ensamblaje pasa a la fase de cierre.

Fase de cierre

En este punto, el proceso de la línea de ensamblaje ha finalizado con normalidad o dicho proceso se ha cancelado debido a un error.

Complemento de software Epílogo - Antes de Close de la línea de ensamblaje

Se procesa el complemento de software de la línea de ensamblaje denominado **Epílogo – Antes de Close**.

Flujo de cierre de los conectores/funciones

Los complementos de software de epílogo de cada conector y de cada función se llaman en el orden en el que aparecen en el Editor de configuración:

1. El complemento de software **Antes de Close**.
2. La operación de cierre se lleva a cabo; por ejemplo, cerrando una conexión o liberando una llamada de API.
3. El complemento de software **Después de Close**.

Complemento de software Epílogo - Después de Close de la línea de ensamblaje

Finalmente, se ejecuta el complemento de software de la línea de ensamblaje **Epílogo – Después de Close**.

Configuración de un conector en modalidad Servidor

Cuando se inicia un conector en modalidad Servidor, entra en modalidad de escucha de sucesos. Cuando se recibe un suceso, el conector clona la línea de ensamblaje y deja de esperar sucesos. Entretanto, en el clon, el

conector cambia a modalidad Iterador y pasa el control al siguiente componente de la lista de Entradas de datos. Este proceso permite tener varios conectores activos en modalidad Servidor y realizar la entrada de datos en el flujo a la vez — un ejemplo sería tener varios conectores del servidor de HTTP en la modalidad Servidor realizando la escucha de puertos distintos, pero que la entrada de datos sea a la misma línea de ensamblaje. Aunque los conectores en modalidad Servidor son parte de una configuración de línea de ensamblaje, se ejecutan como procesos aparte, hebras.

Hay un conjunto de complementos de software adicionales que se evalúan para los conectores en esta modalidad. Los complementos de software específicos de la función en modalidad Servidor que se encargan de las conexiones entrantes son:

Antes de Accepting connection

Se llama a este complemento de software antes de que el conector entre en la modalidad de escucha.

Después de Accepting connection

Cuando se recibe una conexión, se invoca a este complemento de software. Recuerde que en este momento no hay datos disponibles. Para examinar la información de sucesos entrantes, utilice los complementos de software de iterador como **Después de GetNext** o **GetNext satisfactorio**.

Error en Accepting connection

Este complemento de software se ejecuta si se produce un error en cualquiera de los complementos de software de la modalidad Servidor o se recibe del origen de datos durante la escucha del suceso.

- Como se ha mencionado anteriormente, si tiene más de un conector en modalidad Iterador (consulte el apartado “Varios iteradores en una línea de ensamblaje” en la página 7), estos conectores se apilan en el orden en que se muestran en la configuración, de arriba abajo. Por ejemplo, si tiene dos iteradores, **a** y **b**, entonces se llama a **a** hasta que no devuelva más entradas antes de que la línea de ensamblaje pase a **b**.
- Si no tiene ningún conector en modalidad Iterador y no se ha proporcionado una entrada work inicial (IWE) a la línea de ensamblaje cuando se inicia; por ejemplo, si se llama desde otra línea de ensamblaje y no se crea ninguna entrada work en el complemento de software Prólogo de la línea de ensamblaje o del conector, la línea de ensamblaje realizará sólo un paso.

Finalmente, hay un complemento de software **Solicitud de conclusión** en el que puede colocar código que se procese si la línea de ensamblaje se cierra correctamente debido a una solicitud externa de conclusión (a diferencia de una línea de ensamblaje que se detiene anormalmente) lo que permite realizar una conclusión con normalidad.

Dispone de funciones especiales del objeto *system* para ignorar o reintentar la entrada work actual, al igual que para ignorar un conector, etc. Consulte el apartado “Control del flujo de una línea de ensamblaje” en la página 39, para obtener información detallada.

Manejo de la finalización y la limpieza de los errores graves

Existen varios métodos para detectar y tratar errores internos de IBM Security Directory Integrator, así como los errores que se producen en conectores, analizadores y componentes de función de IBM Security Directory Integrator. Estos métodos son:

- Complementos de software de error, en los que se puede escribir código JavaScript para tratar un error. Este método es accesible a los usuarios de IBM Security Directory Integrator. Véase también “Control del flujo de una línea de ensamblaje”.
- Bloques Java de intento de detección final, que garantizan que una anomalía menor no interrumpa el servidor y que todos los errores se manejen de forma adecuada. Estos bloques ya se han colocado en las clases núcleo del servidor de IBM Security Directory Integrator.

La función de complemento de software de conclusión de la JVM mejora la fiabilidad del servidor. Los complementos de software de conclusión de Java permiten que una pieza de código realice parte del proceso después de pulsar Control-C o cuando la JVM está concluyendo por algún motivo, aunque sea System.exit.

Se puede especificar que se inicie un programa externo cuando la JVM está concluyendo. Este programa externo se inicia desde dentro del complemento de software de conclusión de JVM. Este programa externo se configura mediante una propiedad opcional en el archivo `global.properties` o `solution.properties`:

```
jvm.shutdown.hook=<ejecutable  
de aplicación externa>
```

Los scripts de shell y los archivos de proceso por lotes también se pueden especificar como valor de esta propiedad.

Cuando se realiza una llamada al complemento de software de conclusión de la JVM, no se puede hacer nada para impedir la finalización de la JVM. De todos modos, con la ejecución de un programa externo, se pueden realizar operaciones personalizables; por ejemplo, enviar un mensaje que indique que el servidor de IBM Security Directory Integrator ha finalizado, llevar a cabo operaciones de limpieza o incluso reiniciar un nuevo servidor, si así se desea.

Control del flujo de una línea de ensamblaje

Los complementos de software son puntos de paso programables del comportamiento integrado automatizado de IBM Security Directory Integrator en los que puede imponer su propia lógica.

Los complementos de software se encuentran en líneas de ensamblaje, conectores y componentes de función. Por ejemplo, si desea ignorar o reiniciar completamente partes de la línea de ensamblaje, normalmente lo hará desde un complemento de software de un conector:

Nota: las construcciones que se detallan a continuación pueden utilizarse para salir de un componente de rama o también de un bucle.

system.ignoreEntry()

Ignorar el conector actual y continuar procesando la entrada de datos existente con el conector siguiente.

system.skipEntry()

Ignorar (omitir) la entrada por completo, abortando el ciclo actual, devolver el control al principio de la línea de ensamblaje y obtener la entrada siguiente del iterador actual.

system.exitFlow()

Omitir el resto del proceso de la entrada actual, ejecutar la lógica de fin de ciclo; por ejemplo, guardar la Clave de estado de iterador (si el conector está configurado para hacerlo), devolver el control al principio de la línea de ensamblaje y obtener la entrada siguiente del iterador actual.

system.restartEntry()

Reiniciar desde el principio de la línea de ensamblaje, forzando al iterador actual para que vuelva a utilizar la entrada actual.

system.skipTo(String name)

Saltar al conector especificado.

system.abortAssemblyLine(String reason)

Abortar toda la línea de ensamblaje con el mensaje de error especificado.

Nota: si coloca código en un **complemento de software Error** y no finaliza la línea de ensamblaje actual o el manejador de sucesos, entonces el proceso continúa independientemente de cómo haya llegado al complemento de software Error. Esto significa que se ignoran incluso los errores de sintaxis del script. Por lo tanto, no olvide comprobar el objeto *error* si desea saber la causa del error.

Los métodos descritos en la lista anterior se pueden considerar como sentencias goto, puesto que se deja de ejecutar el resto del código del complemento de software. Por ejemplo:

```
system.skipEntry(); // Hace que se modifique el flujo
// La línea siguiente no se ejecuta nunca.
task.logmsg("Aquí no se llegará nunca");
```

Nota: Existe una diferencia entre un complemento de software de error que está ausente y un complemento de software vacío - aunque esto no siempre es fácil de detectar en el Editor de configuración. Un complemento de software de error *vacío* hace que el sistema restablezca la condición de error que ha causado la llamada del complemento de software, después de lo cual el servidor continúa el proceso, mientras que un complemento de software *ausente* o sin definir hace que el sistema lleva a cabo un manejo de errores predeterminado (normalmente con una terminación anormal de la línea de ensamblaje).

Expresiones

IBM Security Directory Integrator proporciona la característica Expresiones, que es compatible con la versión 6. Le permite calcular parámetros y otros valores en tiempo de ejecución, lo que permite configurar soluciones de forma dinámica. Esta característica amplía el manejo de propiedades existente en versiones anteriores.

Además de dar soporte a referencias de propiedades externas simples (totalmente compatible con versiones anteriores), la función Expresiones proporciona más prestaciones para manipular valores de configuración de las líneas de ensamblaje y de los componentes durante la inicialización y la ejecución de los mismos. La función Expresiones también puede utilizarse para las correlaciones de atributos, así como para las condiciones y los criterios de enlace, lo que reduce notablemente las tareas de creación de scripts que anteriormente debían llevarse a cabo para

crear soluciones que pudieran configurarse de forma dinámica. IBM Security Directory Integrator proporciona un Editor de expresiones para facilitar la creación de estas expresiones.

La característica Expresiones está basada en los servicios proporcionados por la clase estándar `java.text.MessageFormat` de Java. La clase `MessageFormat` proporciona potentes prestaciones de sustitución y formateo. A continuación se proporciona un enlace a una página en línea en la que se describe esta clase y sus características: <http://docs.oracle.com/javase/1.6.0/docs/api/java/text/MessageFormat.html>.

Nota: Las expresiones basadas en `MessageFormat` que se muestran en esta sección eran el elemento básico de la sustitución de parámetros en IBM Security Directory Integrator versión 6. En la versión 7 es recomendable utilizar expresiones avanzadas (JavaScript) en su lugar.

Además de las características descritas en la clase anterior, IBM Security Directory Integrator proporciona varios objetos de tiempo de ejecución que se pueden utilizar en expresiones, pero la disponibilidad de algunos objetos dependerá del estado de ejecución; por ejemplo, si está definido `conn` o `current` o la entrada `error`. La sintaxis de las expresiones proporciona una notación abreviada para acceder a la información de estos objetos, como los atributos del objeto de entrada determinado o un parámetro específico de un componente.

Tabla 2. Objetos de script, su utilización y disponibilidad.

Referencia de IBM Security Directory Integrator	Valor	Disponibilidad
<code>work.nombreatrib[.índice]</code>	<p>Entrada <code>work</code> de la línea de ensamblaje actual.</p> <p>El <code>índice</code> opcional hace referencia al valor <code>n</code> del atributo. De lo contrario, se utiliza el primer valor.</p> <p>Esta correlación de atributos avanzada:</p> <pre>ret.value = work.getString("givenName") + " " + work.getString("sn");</pre> <p>puede expresarse simplemente de este modo:</p> <pre>{work.givenName} {work.sn}</pre>	línea de ensamblaje
<code>conn.nombreatrib[.índice]</code>	<p>Entrada <code>conn</code> de la línea de ensamblaje actual.</p> <p>El <code>índice</code> opcional hace referencia al valor <code>n</code> del atributo. De lo contrario, se utiliza el primer valor.</p>	Línea de ensamblaje durante la correlación de atributos
<code>current.nombreatrib[.índice]</code>	<p>Entrada <code>current</code> de la línea de ensamblaje actual</p> <p>El <code>índice</code> opcional hace referencia al valor <code>n</code> del atributo. De lo contrario, se utiliza el primer valor.</p>	Línea de ensamblaje durante la correlación de atributos

Tabla 2. Objetos de script, su utilización y disponibilidad. (continuación)

Referencia de IBM Security Directory Integrator	Valor	Disponibilidad
config.parám	<p>Objeto de configuración de la línea de ensamblaje del componente actual. Además, si se utiliza <i>config</i> en el parámetro de un conector, de un analizador o de una función, hace referencia al objeto de configuración de <i>Interfaz</i> de dicho componente; por ejemplo, conector JDBC o analizador XML.</p> <p><i>parám</i> es el nombre del parámetro propiamente dicho, como si realizara una llamada a <code>getParam()</code> o a <code>setParam()</code>. Por ejemplo, para el Conector JDBC, podría realizar la referencia siguiente: <code>{config.jdbcSource}</code></p>	<p>Línea de ensamblaje Manej. de sucesos Conector Analizador Comp. de función</p>
alcomponent.nombre.parám	<p>Valor del parámetro de interfaz del componente de un componente de línea de ensamblaje determinado.</p> <p><i>nombre</i> es el nombre del componente de línea de ensamblaje.</p> <p><i>parám</i> es el nombre de parámetro del objeto <i>name</i>.</p> <p>Por consiguiente, la siguiente expresión: <code>{alcomponent.DB2conn.jdbcSource}</code></p> <p>es equivalente a la siguiente llamada de script: <code>DB2conn.connector.getParam("jdbcSource");</code></p>	<p>línea de ensamblaje</p>
property[:storename].name property[:storename/ bidi].name	<p>Referencia a <i>propiedades de TDI</i>.</p> <p>El nombre de almacén opcional se dirige a un almacén de propiedades específico. Si no se especifica ningún nombre de almacén, se utiliza el almacén por omisión.</p> <p><i>nombre</i> es el nombre de la propiedad.</p> <p>Si está presente, <i>bidi</i> hará que el valor del parámetro se defina de modo que se reenvíe la llamada al almacén de propiedades al que se hace referencia. Si <i>bidi</i> está presente no se permite el uso de ningún otro patrón de sustitución ni texto.</p>	<p>Siempre.</p>

Tabla 2. Objetos de script, su utilización y disponibilidad. (continuación)

Referencia de IBM Security Directory Integrator	Valor	Disponibilidad
<pre>JavaScript<<EOF script code ... // Debe contener "return" EOF</pre> <p>Nota: Sintaxis de la v.6; utilice la opción Avanzado (JavaScript) del editor de expresiones en lugar de ello</p>	<p>Se utiliza código de script <i>incorporado</i> para generar un valor para la expresión. Este script debe devolver un valor.</p> <p>El texto "EOF" que se utiliza aquí es una serie arbitraria que determina el fragmento de código JavaScript. Se recopila el JavaScript hasta que se encuentra una línea con la serie EOF o no hay ningún EOF definido - véase la nota siguiente.</p> <p>Observe que el JavaScript incorporado se evalúa utilizando la instancia del motor de scripts de la línea de ensamblaje, por lo que tiene acceso a todas las variables presentes para el script.</p> <p>Nota: Existe una forma abreviada de añadir JavaScript que funciona para los campos de entrada y que no da soporte a varias líneas (como los criterios de enlace o los de los atributos en las correlaciones) y que, por consiguiente, no tiene la línea EOF necesaria:</p> <pre>{JavaScript return work.givenName + " " + work.surName}</pre>	Siempre.

El JavaScript incorporado en expresiones tiene acceso a motor de script de la línea de ensamblaje. Como consecuencia de ello, es posible acceder a todas las variables de script aunque estén definidas en cualquier otro lugar de la línea de ensamblaje. Si hace referencia a una variable o a un objeto que no es uno de los específicamente listados en las tablas que se muestran en este apartado, el evaluador de expresiones comprobará con el motor de scripts de la línea de ensamblaje si dicha variable u objeto está definido allí.

Expresiones en parámetros de componente

Cuando se utilizan para un parámetro de componente, los objetos siguientes tienen especial interés:

Tabla 3. Objetos especiales que se pueden utilizar en expresiones

Objeto	Valor
config	Objeto de configuración de interfaz del componente.
mc	Objeto MetamergeConfig de la instancia de configuración (config.getMetamergeConfig()).
work	Entrada Work de la línea de ensamblaje.
task	Objeto de la línea de ensamblaje.

A modo de ejemplo, tome un conector JDBC con el parámetro Table Name establecido en "Accounts". A continuación, podría pulsar en la etiqueta de parámetro **SQL Select** o el botón **Abrir el diálogo del valor del parámetro** que está junto al campo, seleccionar **Texto con sustitución** y luego especificar lo siguiente en el campo de texto grande en la parte inferior del diálogo:

```
select * from {config.jdbcTable}
```

De este modo se tomará el parámetro Table Name y se creará la siguiente sentencia SQL Select:

```
select * from Accounts
```

O podría ir un poco más allá y probar lo siguiente para el parámetro SQL Select:

```
SELECT {JavaScript<<EOF  
  
    var str = new Array();  
    str[0] = "A";  
    str[1] = "B";  
    return str.join(",");  
EOF  
  
} FROM {property:mystore.tablename} WHERE A = '{work.uniqueID}'
```

El JavaScript incorporado devolverá el valor "A,B", que luego se utiliza para completar el resto de la expresión. Si tiene un almacén de propiedades denominado *mystore* con una propiedad *tablename* establecida en "Accounts" y existe un atributo *uniqueID* en la entrada Work con el valor "42", el resultado final será el siguiente:

```
SELECT A,B FROM Accounts WHERE A = '42'
```

Este resultado evaluado no se muestra en el CE. Especificar simplemente las llaves no dará lugar a que se evalúe la expresión para el valor del parámetro. En lugar de ello, tiene dos opciones para escribir expresiones para parámetros:

1. Pulsar el botón **Abrir el diálogo del valor del parámetro** que hay junto al campo (o pulsar la etiqueta Parámetro) y seleccionar **Texto con sustitución** para abrir el diálogo Expresiones mientras está en el campo de entrada de parámetros. Puede especificar la expresión en el campo de texto grande de este diálogo. Pulse **Aceptar** para especificar la expresión.
2. Especificar manualmente un preámbulo especial, @SUBSTITUTE, en el campo de entrada del parámetro seguido por la expresión. Por ejemplo:

```
@SUBSTITUTEhttp://{property.myProperties:HTTP.Host}/
```

Nota:

- No se recomienda este último método de entrar expresiones directamente; en lugar de ello utilice el editor de expresiones.
- Si está seleccionado **Texto con sustitución** para el valor de la vía de acceso del archivo del Conector de archivo, y se ha especificado {work.fullPath}, se producirá el siguiente error: "CTGDIC114E Parameter 'File Path' is required". Se espera este resultado porque se supone que el Editor de configuración mostrará el resultado de aplicar la sustitución en el texto que ha especificado. En este caso, no hay ningún objeto work porque el objeto work sólo está definido en un AssemblyLine en ejecución. Por lo tanto, el resultado en una serie vacía. Para este parámetro, una serie vacía es un error y, por ello, se mostrará el mensaje de error. Cuando se ejecute la AssemblyLine, puede que haya un objeto work, y la evaluación puede dar lugar a la serie de parámetros, según sea necesario.

Expresiones en los criterios de enlace

Las expresiones en los criterios de enlace proporcionan una lista parecida de objetos predefinidos. De nuevo, recuerde cuenta que también tiene acceso a cualquier otro objeto o variable que esté definido actualmente en el motor de scripts de la línea de ensamblaje.

Tabla 4. Objetos predefinidos para utilizar en expresiones en los criterios de enlace

Objeto	Significado
config	Objeto de configuración de interfaz del componente.
mc	Objeto MetamergeConfig de la instancia de configuración (config.getMetamergeConfig())
work	Entrada de trabajo de la línea de ensamblaje.
task	El mismo componente o un componente indicado.
alcomponent	Componente de conector o función.

Así, por ejemplo, supongamos que desea configurar criterios de enlace para un conector de modo que el atributo que deba utilizarse en la coincidencia se determine en tiempo de ejecución. Además de los atributos de datos estándar de la entrada Work, también existe un atributo *matchAtt* con el valor de serie "uid". En este caso, la expresión siguiente que se utiliza en los criterios de enlace:

```
{work.matchAtt} EQUALS {work.uid}
```

es equivalente a:

```
uid EQUALS $uid
```

Expresiones en ramas, bucles y conmutador/caso

La lista de objetos de expresión siguiente es parecida a la de los criterios de enlace:

Tabla 5. Objetos predefinidos para utilizar en expresiones en componentes de rama

Objeto	Significado
config	Objeto de configuración de interfaz del componente.
mc	Objeto MetamergeConfig de la instancia de configuración (config.getMetamergeConfig())
work	Entrada de trabajo de la línea de ensamblaje
task	Línea de ensamblaje.
alcomponent	Componente de conector o función.

Puede utilizar expresiones para el nombre de atributo y para el operando de una condición. También puede utilizar expresiones para configurar componentes de conmutador y de caso.

Creación de scripts con expresiones

También puede utilizar expresiones directamente desde el código JavaScript. A continuación se muestra un ejemplo que crea una expresión utilizando la nueva clase *ParameterSubstitution*:

```
var ps = new com.ibm.di.util.ParameterSubstitution("{work.FullName} -> {work.uid}");  
  
map = new java.util.HashMap();  
  
map.put("mc", main.getMetamergeConfig());  
map.put("work", work);  
  
task.logmsg(ps.substitute(map));
```


La expresión que resulta del código JavaScript emite los siguientes mensajes de registro cuando se ejecuta para varias iteraciones en la línea de ensamblaje:

```
14:35:29 Patty S Duggan -> duggan
14:35:29 Nicholas P Butler -> butler
14:35:29 Henri T Deutch -> deutch
14:35:29 Ivan L Rodriguez -> rodriguez
14:35:29 Akhbar S Kahn -> sahmad
14:35:29 Manoj M Gupta -> gupta
```

Objeto de entrada

Uno de los fundamentos para comprender el funcionamiento de IBM Security Directory Integrator es saber cómo se almacenan los datos y cómo se transportan dentro del sistema. Esto puede hacerse mediante un objeto denominado *de entrada*. El objeto de entrada puede considerarse como un "contenedor Java" que puede contener cualquier número de atributos: ninguno, uno o muchos.

En IBM Security Directory Integrator, los atributos también son como objetos de tipo contenedor. Cada atributo puede contener cero o más *valores* y éstos son los valores de datos reales que se leen de los sistemas conectados y que se escriben en ellos. Los valores de los atributos también son objetos Java; pueden ser series, enteros e indicaciones de fecha y hora, según sea necesario para corresponder con el tipo nativo de este valor de datos. Un único atributo puede contener sin ningún problema valores de tipos diferentes. No obstante, los valores de un único atributo tenderán a ser del mismo tipo en la mayor parte de los orígenes de datos.

Aunque este paradigma *entrada-atributo-valor* coincide con el concepto de las entradas de directorio Lightweight Directory Access Protocol (LDAP), también es el modo en el que se representan las filas de las bases de datos dentro de IBM Security Directory Integrator, como los registros en los archivos, los documentos de IBM Lotus Notes y las páginas HTTP recibidas a través de la red. Todos los datos, de cualquier origen con el que IBM Security Directory Integrator trabaje, se almacenan internamente como objetos de entrada con atributos y sus valores.

A diferencia de las versiones anteriores de IBM Security Directory Integrator, a partir de la v7.0 se permiten objetos de entrada jerárquicos en la línea de ensamblaje y algunos de los componentes que pueden formar parte de una línea de ensamblaje. El objeto de entrada se ha ampliado para proporcionar varios métodos convenientes para gestionar datos jerárquicos, aunque de forma predeterminada esto está oculto y sólo entra en acción cuando se habilita explícitamente o se utiliza con componentes que requieren características jerárquicas. También implementa `org.w3c.dom.Document`, que lo convierte en el nodo de nivel superior en la jerarquía. Para obtener más información, consulte el apartado "Trabajo con objetos de entrada jerárquicos" en la página 52.

IBM Security Directory Integrator crea y realiza el mantenimiento de unos cuantos objetos de entrada. La instancia más visible se denomina *entrada work* y sirve como portador de datos principal en una línea de ensamblaje (LE). Éste es el contenedor que se utiliza para transportar datos a la línea de ensamblaje, pasando de un componente al siguiente.

La instancia de la entrada *work* está disponible para ser utilizada al crear scripts mediante la variable *work* registrada previamente, lo que proporciona acceso directo a los atributos gestionados por una línea de ensamblaje (y sus valores). Además, todos los atributos de la entrada *Work* se muestran en el Editor de configuración, en la cabecera **Atributo Work** del área Correlaciones de atributos de la ventana Editor de líneas de ensamblaje de una línea de ensamblaje.

Tipos de entradas

Existen numerosos objetos de datos que residen en líneas de ensamblaje que siguen el modelo de datos de entrada. Son los siguientes:

Work Es la ya mencionada entrada que va de un componente a otro componente en la línea de ensamblaje y transporta datos entre ellos. El nombre de variable previamente registrado es *work* y está disponible para utilizarse en scripts casi en cualquier lugar⁶.

Conn Es el objeto de tipo entrada que un conector utiliza como intermediario entre el sistema conectado y la línea de ensamblaje, antes de crear los datos, o un subconjunto de ellos, disponible en la entrada Work. El proceso de mover datos entre Conn y Work se denomina correlación de atributos. Su nombre de variable previamente registrado es *conn* y está disponible para utilizarse en scripts en muchos de los complementos de software de los conectores y componentes de función.

Current

Este objeto de tipo entrada está disponible en determinados complementos de software de conectores en modalidad Actualizar y contiene los datos del sistema conectado antes de que se le hayan aplicado actualizaciones. Su nombre de variable previamente registrado es *current*.

Error Este objeto de tipo entrada sólo existe en determinados complementos de software de componentes cuando se ha producido una condición de error y se ha invocado el complemento de software de error pertinente. Contiene información sobre la excepción real que se ha generado, posiblemente con variables y datos adicionales, y le permite determinar qué ha causado exactamente el error. Su nombre de variable previamente registrado es *error*.

Los diagramas de flujo de los conectores de *Referencia* le mostrarán cuál de estos objetos está disponible según las circunstancias.

Consulte también

“Modelo de datos interno: entradas, atributos y valores” en la página 50.

6. Cuando la línea de trabajo con la que trabaja no se llama con una entrada Work inicial, el objeto *work* no está disponible hasta después de los complementos de software de prólogo. En los complementos de software de prólogo puede tener el código siguiente:

```
if (work != null) {
  // Se ha proporcionado una entrada work inicial, podemos obtener valores de allí
  ... some code
} else {
  // No se ha proporcionado una entrada work inicial
  ... some other code
}
```

Capítulo 2. Scripts en IBM Security Directory Integrator

IBM Security Directory Integrator proporciona a sus usuarios un motor muy flexible que puede personalizarse desde los controles de la interfaz de usuario del Editor de configuración o a través de scripts de lógica personalizada. Mientras que la interfaz de usuario controla un medio para controlar el flujo de datos a un nivel más alto, los scripts le permiten controlar prácticamente todos los aspectos del flujo de datos a cualquier nivel, incluida la alteración del proceso estándar de IBM Security Directory Integrator. El objeto *system* dispone de funciones especiales que permiten reiterar sobre una entrada de la línea de ensamblaje, ignorar un conector e iniciar nuevas líneas de ensamblaje. El lenguaje de script que se utiliza para implementar esta lógica personalizada es JavaScript.

IBM Security Directory Integrator está listo para usar y viene equipado con las herramientas necesarias para ensamblar rápidamente la infraestructura de una solución de integración. Sin embargo, para todos los trabajos de migración, exceptuando los más triviales, tendrá que personalizar y ampliar el comportamiento integrado del producto mediante código JavaScript.

IBM Security Directory Integrator es puro Java. Siempre que emita un mandato a IBM Security Directory Integrator, trabaje con componentes y objetos o manipule datos del flujo, estará trabajando con objetos Java. IBM Security Directory Integrator utiliza IBM Java versión 7.0.4.

Por otro lado, la personalización se realiza en JavaScript, y esta fusión de dos lenguajes de programación similares, pero básicamente distintos, justifica un examen más detallado.

Tener experiencia en el uso de JavaScript será muy útil. Los ejemplos proporcionados pueden ampliar su experiencia. Pero este manual no describe la creación de JavaScript, solamente su aplicación en IBM Security Directory Integrator. Tendrá que conseguir su material de referencia de JavaScript en algún otro lugar.

Hay varias guías de referencia comerciales disponibles de JavaScript, así como documentación, guías de aprendizaje y ejemplos en Internet. Recuerde, no obstante, que la mayoría del contenido de JavaScript en la web se utiliza para embellecer y automatizar el contenido HTML. Sólo deberá ocuparse del lenguaje base en sí, como se describe en el enlace siguiente: <http://devedge-temp.mozilla.org/library/manuals/2000/javascript/1.5/guide/index.html>

En este sitio también encontrará un enlace práctico para bajarse la referencia en formato HTML para su instalación local. Un excelente manual sobre JavaScript es *The Definitive JavaScript Guide*, 4ª edición de David Flanagan (O'Reilly).

También necesitará los Javadocs para Java, pues todos los objetos de IBM Security Directory Integrator, así como los valores de datos contenidos en la solución, están en forma de objetos Java. Estos documentos se encuentran en línea en la siguiente dirección URL: <http://docs.oracle.com/javase/1.6.0/docs/api/index.html>

La documentación de J2SE en sí se puede encontrar aquí: <http://docs.oracle.com/javase/1.6.0/docs/index.html>

Los scripts son necesarios cuando necesita añadir un proceso personalizado a la línea de ensamblaje. Algunos ejemplos de dónde pueden resultar útiles los scripts son las tareas siguientes:

Manipulación o cálculo de atributos

Necesita calcular el valor de un atributo de salida basándose en uno o más atributos de entrada.

Filtrado de datos

Desea procesar solamente las entradas que coinciden con un conjunto de criterios determinado.

Comprobación de la coherencia o validez de los datos

Necesita informar de los valores de datos no válidos o corregirlos.

Control de flujo

Desea alterar temporalmente la operación de actualización del conector que está utilizando.

Inicialización

Desea ejecutar algunos procedimientos de inicialización antes de iniciar la línea de ensamblaje.

Cada uno de estos casos mencionados, y muchos otros que no se han mencionado, requieren scripts.

Ejemplos

Vaya al subdirectorio `examples/scripting` de la instalación de IBM Security Directory Integrator.

Modelo de datos interno: entradas, atributos y valores

Cuando los componentes de IBM Security Directory Integrator acceden a la información de sistemas conectados, convierten los datos de tipos específicos del sistema en una representación interna utilizando objetos Java. En la salida, los componentes se convierten a la inversa, partiendo de este modelo de datos interno a los tipos nativos del sistema de destino. Se utiliza esta misma representación interna si desea pasar datos a/de líneas de ensamblaje. Por lo tanto, es fundamental comprender cómo funciona el modelo de datos interno de IBM Security Directory Integrator.

Si observamos con detenimiento el momento en que un componente recibe un valor de datos, vemos que se crea un objeto de *atributo* de IBM Security Directory Integrator correspondiente utilizando el nombre del atributo que se está leyendo. El propio valor de datos (o valores, si es un atributo con varios valores) se convierte al objeto Java apropiado (`Java.lang.String` o `java.sql.Timestamp`) y se asigna al atributo. Si revisa la documentación de la API de IBM Security Directory Integrator, verá que el objeto de atributo proporciona varios métodos útiles como, por ejemplo, `getValue()`, `addValue()` y `size()`. Esto permite crear, enumerar y manipular los valores de atributos directamente desde scripts. También puede crear instancias de objetos de atributo nuevos según sea necesario, como se muestra en este ejemplo de correlación avanzada del atributo `objectClass` de un directorio:

```
var oc = system.newAttribute( "objectClass" );  
  
oc.addValue( "top" );  
oc.addValue( "person" );
```

```
oc.addValue( "organizationalPerson" );
oc.addValue( "inetOrgPerson" );

ret.value = oc;
```

Los atributos en sí se reúnen en un objeto de almacenamiento de datos denominado *objeto de entrada*. El objeto de *entrada* es el portador de datos principal del sistema y IBM Security Directory Integrator proporciona acceso a objetos de entrada importantes registrándolos como variables de script. Un ejemplo excelente es el objeto de entrada *work* de la línea de ensamblaje, que se utiliza para pasar datos entre componentes de la línea de ensamblaje (así como entre líneas de ensamblaje). Este objeto de entrada es local a cada línea de ensamblaje y está disponible como la variable de script *work*.

IBM Security Directory Integrator proporciona algunos métodos abreviados y funciones cómodas cuando trabaja en JavaScript, de modo que la correlación avanzada específica anterior puede codificarse simplemente de este modo:

```
ret.value = [ "top", "person", "organizationalPerson", "inetOrgPerson" ];
```

La función de correlación avanzada admite entradas y matrices de JavaScript para pasar varios valores de atributo.

Por ejemplo, en una correlación de atributos de entrada (que hace que los atributos correlacionados se muestren en la entrada *work* en el retorno), supongamos que tiene la asignación

```
ret.val = anentry;
```

para el atributo denominado "last". Supongamos además que *work* está vacía al principio y que *anentry* contiene los atributos "cn", "sn" y "mail".

Después de la correlación de atributos, *work* contendrá los atributos "cn", "sn" y "mail", **no** un único atributo denominado "last" con el valor "anentry". En resumen, lo que sucede en la correlación de atributos es que cuando una correlación de atributos devuelve un objeto de entrada, se fusiona con la entrada receptora - *work* o *conn*, según en qué correlación esté (de entrada o de salida).

Nota: En IBM Security Directory Integrator, aprovechando los objetos jerárquicos, puede evitar este comportamiento encapsulando primero una entrada de un atributo antes de que se realice la correlación de atributos. Por ejemplo,

```
// es la entrada para devolver
e = system.newEntry();
e.setAttribute("some", "value");
```

```
// Crear un objeto de atributo. No es necesario proporcionar un nombre ya que
la correlación utilizará el nombre de la correlación actual.
attr = system.newAttribute(null);
```

```
// añadir la entrada al objeto de atributo y devolverlo en lugar del objeto de entrada
attr.addValue(e);
```

```
return attr;
```

Si se ha entrado esto como correlación de atributos avanzada para el atributo "last", después de la correlación de atributos, la entrada *work* no contendrá un atributo denominado "last". Este atributo es una entrada, a su vez formada por dos atributos denominados "some" y "value".

Si observa los Javadocs, verá que el objeto de entrada ofrece distintas funciones para trabajar con entradas y sus atributos y valores, incluidos `getAttributeNames()`, `getAttribute()` y `setAttribute()`. Si desease crear y añadir un atributo a la entrada `work` de la línea de ensamblaje, podría utilizar el script siguiente, por ejemplo, en un componente de complemento de software o de script:

```
var oc = system.newAttribute( "objectClass" );  
  
oc.addValue( "top" );  
oc.addValue( "organizationalUnit" )  
  
work.setAttribute( oc );
```

Recuerde que en este caso **no** se tiene la opción de utilizar una matriz de JavaScript para establecer el valor:

```
oc.addValue( ["top", "organizationalUnit"] ); // No funciona como la correlación avanzada
```

Este código hará que el atributo `oc` obtenga un solo valor, que a su vez es una matriz de series.

Los objetos de entrada también pueden contener *propiedades*. Las propiedades son contenedores de datos como los atributos, excepto que tienen únicamente un valor. Mientras que para almacenar el contenido de los datos se utilizan los atributos, las propiedades retienen información de parámetros, que le permite mantener esta información aparte. No se muestran las propiedades con la selección de la correlación de atributos, ni en la lista de entradas `work`, pero se puede acceder a éstas de forma parecida a los atributos desde el script. Las funciones de entrada como `getProperty()` y `setProperty()` se utilizan para esto y funcionan directamente con valores de propiedades, que pueden ser cualquier tipo de objeto Java, como los valores de atributo. No hay un objeto de propiedad intermedio como cuando se trabaja con atributos.

En muchos casos, puede restringir el modelo de datos a una entrada que contenga cero o más atributos, cada uno con cero o más valores, es decir un esquema simple.

Ésta es una de las cualidades de IBM Security Directory Integrator: la simplificación y coordinación de las representaciones de datos y esquemas. Pero también supone un reto cuando se tiene que manejar información con una estructura más compleja. Sin embargo, dado que el valor de un atributo puede ser cualquier tipo de objeto Java, incluido otro objeto de entrada (con sus propios atributos y valores), IBM Security Directory Integrator permite trabajar con datos estructurados jerárquicamente.

Este modo más elaborado y estructurado de manejar objetos jerárquicos se describe en el apartado “Trabajo con objetos de entrada jerárquicos”.

Trabajo con objetos de entrada jerárquicos

Un método alternativo de trabajar con datos estructurados jerárquicamente es aprovechar el soporte de objetos jerárquicos del objeto de entrada de IBM Security Directory Integrator.

A diferencia de versiones anteriores, IBM Security Directory Integrator Versión 7.1.1 y posterior es compatible con el concepto de objeto de entrada jerárquico. El objeto de entrada representa la raíz de la jerarquía y cada atributo representa un nodo en dicha jerarquía. Siguiendo esta lógica, los valores de cada atributo son hojas en la jerarquía. También existe una API para atravesar la jerarquía. Esta API

es una implementación parcial de la especificación DOM 3. Sólo se han implementado unas cuantas clases de esta especificación:

- `Org.w3c.dom.Document` – implementada por la clase de entrada.
- `Org.w3c.Element` – implementada por la clase de atributo.
- `Org.w3c.Attr` – implementada por la clase de propiedad.
- `Org.w3c.Text` y `org.w3c.CDATASection` – implementada por la clase de valor de atributo.

Estas clases son el conjunto mínimo de clases proporcionadas por la especificación DOM necesario para representar datos jerárquicos. La API anterior a la v7.0 no reconoce las jerarquías (por ejemplo, no puede acceder/modificar/eliminar elementos hijo) por motivos de compatibilidad con versiones anteriores. Por este motivo la API de DOM sólo puede manipular una estructura jerárquica.

Para hacer que la estructura de entrada sea compatible con las versiones anteriores de forma predeterminada, la entrada utiliza siempre atributos simples. La entrada sólo será jerárquica bajo demanda - después de llamar a una de las API de DOM que se han proporcionado recientemente. Esto hace que sólo los componentes que reconocen la jerarquía de la entrada puedan utilizarla, no es necesario cambiar el resto de los componentes para que sigan ejecutándose. A partir de IBM Security Directory Integrator v7.0, se ha introducido una nueva notación de nombres para proporcionar a los usuarios un método más fácil de crear árboles jerárquicos. Un nombre que contiene un punto se considera un nombre compuesto formado por nombres simples; estos nombres se separan mediante puntos. Cuando un nombre compuesto de este tipo se pasa a una entrada jerárquica, ésta la descompone en nombres simples y crea la jerarquía que describe el nombre compuesto.

Por ejemplo, si ejecuta el siguiente código JavaScript:

```
// crear un nuevo objeto de entrada vacío
var entry = new com.ibm.di.entry.Entry(true);
// crear una nueva rama de 2 niveles
entry.setAttribute("firstLevelChild.secondLevelChild", "level2Value");
// busca la rama que ya existe y crea un código nuevo en el nivel 3
entry.setAttribute("firstLevelChild.secondLevelChild.thirdLevelChild", "level3Value");
```

se creará la siguiente estructura:

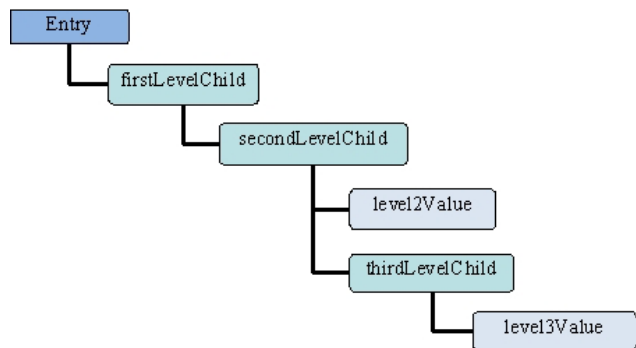


Figura 1. Entrada jerárquica simple

Nota: Es importante saber que los nombres no se descomponen en nombres simples hasta que la estructura de entrada no se convierte en una entrada jerárquica. Por ejemplo, si la entrada es una entrada simple y sólo se utilizan los

métodos antiguos, se crearía la misma estructura simple antigua:

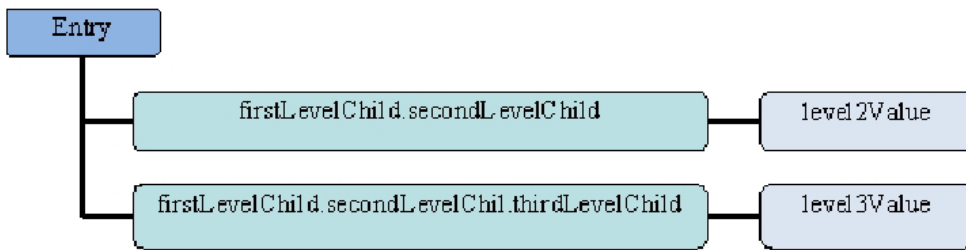


Figura 2. Entrada simple tradicional

En algunos casos puede ser necesario que existan puntos en los nombres de los atributos de entrada, por ello el objeto de entrada también comprende los caracteres de escape (actualmente sólo están soportados `\\` y `\\.`).

Nota: Cuando se trabaja desde script, la barra inclinada invertida es un carácter de escape correcto si se utiliza otra barra inclinada invertida. Por ejemplo, si se necesita la siguiente jerarquía:

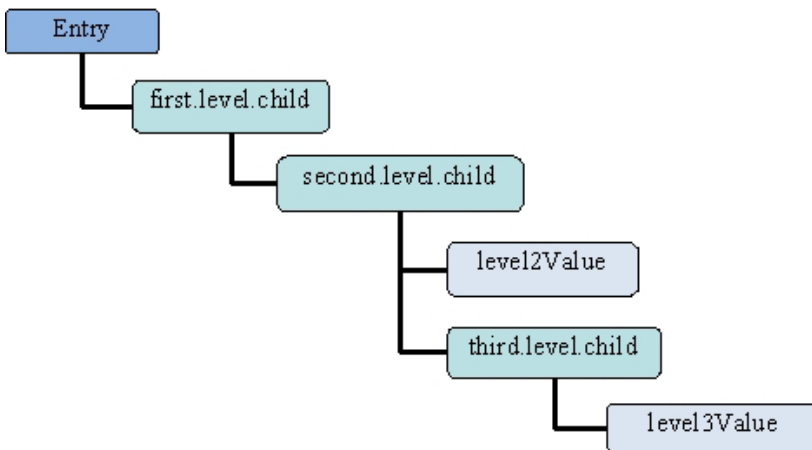


Figura 3. Otra entrada jerárquica simple

el script siguiente la crearía:

```
var entry = new com.ibm.di.entry.Entry(true);
entry.setAttribute("first\\.level\\.child.second\\.level\\.child", "level2Value");
entry.setAttribute("first\\.level\\.child.second\\.level\\.child.third\\.level\\.child", "level3Value");
```

Para mantener la compatibilidad con versiones anteriores de IBM Security Directory Integrator cuando se trabaja implícitamente con entradas jerárquicas, todos los métodos antiguos que exponen las clases de entrada y de atributo se han cambiado ligeramente. Por ejemplo, el método `Entry.getAttributeNames()` devolverá una matriz de vías de acceso completas a las hojas que están disponibles del árbol. En cuanto a la estructura anterior, el método `getAttributeNames` devuelve la matriz:

```
["first\\.level\\.child.second\\.level\\.child", "first\\.level\\.child.second\\.level\\.child.third\\.level\\.child"]
```

El método `Entry.size()` devuelve el número total de elementos de la matriz que el método `getAttributeNames()` devuelve. En este caso, el método `size()` devolvería 2 (el número de hojas del árbol completo) en lugar de 1 (como podría esperarse, teniendo en cuenta que el objeto de entrada sólo tiene un atributo como hijo).

Esto se debe a que los métodos `getAttributeNames()` y `size()` sólo funcionan con estructuras simples. Para obtener el tamaño real del hijo, tendría que utilizar la API de DOM del modo siguiente `Entry.getChildNodes().getLength()`;

Si la entrada es una entrada simple, la API antigua se comportará igual que en las versiones anteriores.

Objeto de atributo

El objeto de atributo se ha mejorado con la posibilidad de crear estructuras jerárquicas. Estas estructuras siguen la especificación DOM y por ello el objeto de atributo reconoce los conceptos de espacio de nombres XML.

La clase de atributo también se ha tenido que ampliar para proporcionar nuevos métodos para la funcionalidad jerárquica. Los métodos `getValue/setValue/addValue` también son compatibles con las versiones anteriores y sólo devuelven los valores que el elemento determinado tiene. La diferencia está en que cuando se accede a cada valor mediante la API de DOM, se acomoda en una clase de valor de atributo y se trata como un nodo de texto. Para obtener los elementos hijo de un atributo (por ejemplo, atributos y valores de atributo) es necesario utilizar la API de DOM.

El hijo de atributo de una entrada también puede conmutar la estructura de la entrada a una entrada jerárquica cuando se accede a cualquiera de sus métodos de DOM. A diferencia de la clase de entrada, la clase de atributo lo hace implícitamente y no proporciona un método para conmutar explícitamente.

IBM Security Directory Integrator v7.0 da soporte a extensiones para que las posibilidades de script del servidor puedan acceder fácilmente a estructuras complejas. Consulte el apartado “Navegación en scripts” en la página 56 para obtener más detalles.

Objeto de valor de atributo

Esta clase representa un valor del árbol jerárquico. Según la especificación DOM, los valores del árbol siempre son series. Sin embargo, la clase de valor de atributo ha aceptado objetos de todo tipo en las últimas versiones. Esto también es válido en la versión actual. La única diferencia para el objeto de valor de atributo está en que cuando se accede a él mediante DOM, devuelve una representación de tipo serie del objeto contenido. Para que la clase de valor de atributo represente un nodo y tenga un valor al mismo tiempo en los términos que la especificación DOM define, debe implementar la interfaz `org.w3c.dom.Text` u `org.w3c.dom.CDATASection`. El valor de atributo implementa ambas interfaces y puede representar cualquiera de estos nodos según lo que se necesite.

Objeto de propiedad

Los atributos pueden tener cero, uno o más objetos de propiedad. La clase de propiedad implementa la interfaz `org.w3c.dom.Attr` y de este modo representa los atributos según los conceptos de DOM. Mediante la utilización de propiedades puede declarar prefijos/espacios de nombres según los conceptos de XML.

Transferencia de objetos

La correlación de un atributo de una entrada a otra siempre copiará el atributo de origen.

Por ejemplo:

```
entry.appendChild(conn.getFirstChild());
// o
entry.setAttribute("name", conn.getFirstChild());
```

Incluso cuando el atributo no es un hijo de primer nivel de la entrada también se copia. Esto se puede lograr mediante el script:

```
entry.a.b.c.d.appendChild(conn.e.f.g);
```

Para mover un objeto de atributo entre entradas sin clonarlo, primero deberá separarlo de su padre anterior y luego adjuntarlo a su nuevo padre.

Por ejemplo:

```
var src = entry1.b.source;
entry1.b.removeChild(src);
entry2.a.target.appendChild(src);
```

Cuando se mueve un objeto de atributo de un padre a otro padre en la misma entrada, el atributo se mueve automáticamente. No se realiza ninguna clonación.

Por ejemplo:

```
entry.a.target.appendChild(entry.b.source);
```

En este ejemplo, el atributo "source" se separa de su padre ("entry.b") y después se adjunta al atributo "entry.a.target". No se realiza ninguna clonación.

Si no desea eliminar el objeto de atributo del origen, puede añadir una copia del atributo del modo siguiente:

```
entry.a.target.appendChild(entry.b.source.clone());
```

Navegación en scripts

El motor de scripts de IBM Security Directory Integrator le permite acceder fácilmente a los atributos de una entrada haciendo referencia a ellos únicamente mediante el nombre; por ejemplo, `entry.attrName` devuelve el atributo con el nombre `attrName`.

1. El motor de JavaScript resuelve nombres basándose en el objeto de contexto en que se ha solicitado el nombre. Por ejemplo, si se efectúa la llamada `entry.a`, el nombre `entry` es el objeto de contexto y `a` es el nombre del objeto hijo que debe resolverse. El motor de JavaScript utiliza una interpretación de izquierda a derecha para evaluar cada objeto de contexto hasta que se resuelve el último. Según el diagrama de abajo, la llamada siguiente, `entry.a.b.c`, se resuelve utilizando este procedimiento: Se busca el objeto `entry` para utilizarlo como objeto de contexto para el primer paso.
2. Se busca el objeto de contexto para un nombre `a`. El objeto `entry` sólo tiene un hijo con el nombre `a`. Se considera que ese hijo es el siguiente objeto de contexto para el siguiente paso.
3. Se busca el objeto de contexto para un nombre `b`. El objeto de contexto tiene dos hijos denominados `b`. Se colocan en una lista y se devuelve dicha lista.

- La operación final busca la lista devuelta en la operación anterior para el nombre *c*. Cada elemento de la lista tiene como mínimo un hijo denominado así. Se obtienen todos y se ponen en una lista, que es el resultado real de resolver la expresión completa.

El siguiente diagrama de ejemplo de objeto de entrada lo ilustra:

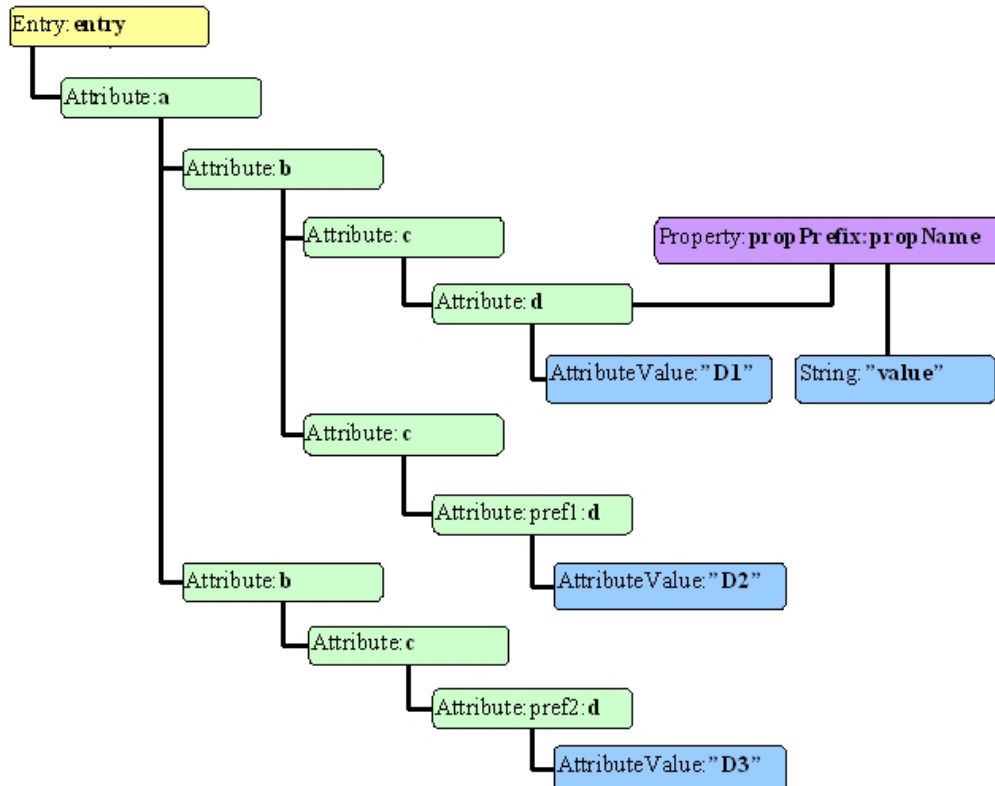


Figura 4. Ejemplo de objeto de entrada jerárquica

El motor de scripts proporcionado por IBM Security Directory Integrator permite utilizar más nombres arbitrarios en el proceso de resolución de hijos. Por ejemplo, si el nombre del hijo contiene puntos, podría hacerse referencia a este nombre utilizando la siguiente sintaxis en corchetes que se muestra a continuación:

```
trabajo["{espacionombres}nombre:Containing\.invalid\.characters"]
```

Tenga en cuenta que los puntos se especifican para denotar que forman parte del nombre local y no deben ser tratados como separadores de vía de acceso por el motor de script.

Según el objeto de contexto actual en el que se realiza una operación, el resultado final puede ser diferente. IBM Security Directory Integrator se ajusta a la manipulación de objetos estándar que proporciona el motor de JavaScript implementando varias mejoras en los objetos siguientes:

Entrada

Cuando el objeto de contexto es una instancia de este tipo, el mecanismo de resolución de nombres buscará la sintaxis siguiente:

- @<name> – busca el objeto de entrada para una propiedad con el nombre especificado. El objeto resuelto es nulo o el objeto correlacionado con dicha propiedad.

- <prefix>:<localName> – busca el objeto de entrada para un hijo con un prefijo igual a <prefix> y un nombre local igual a <localName>. El objeto resuelto puede ser nulo o un objeto de atributo existente.
- <localName> – busca el objeto de entrada para el primer hijo con un nombre local igual a <localName>. El objeto resuelto puede ser nulo o un objeto de atributo.
- {namespaceURI}<localName> – busca la entrada para el primer atributo hijo que pertenece al namespaceURI especificado y con el mismo nombre local que el nombre especificado.

Nota: Si se proporciona un prefijo, se ignorará y el mecanismo de resolución de nombres sólo buscará los namespaceURI y localName especificados. El objeto resuelto puede ser nulo o un objeto de atributo.

Atributo

Cuando el objeto de contexto es una instancia de este tipo, el mecanismo de resolución de nombres buscará la sintaxis siguiente:

- @<prefix>:<localName> y @<localName> – busca el atributo para objetos de propiedad con el mismo prefijo y/o el nombre local o sólo el nombre local especificado. Devuelve un valor nulo si ninguna propiedad coincide con el nombre especificado o devuelve un único objeto de propiedad.
- @{namespaceURI}<localName> – busca el atributo para una propiedad que pertenece al namespaceURI especificado y tiene el mismo nombre local que el nombre especificado.

Nota: Si se proporciona un prefijo, se ignorará y el mecanismo de resolución de nombres sólo buscará los namespaceURI y localName especificados. El objeto resuelto puede ser nulo o un objeto de propiedad.

- [<index>] – especifica la posición del valor en el atributo que debe recuperarse. Si utiliza esta notación, no puede acceder a un hijo de este atributo. Devuelve un valor nulo o el objeto en la posición especificada.
- <prefix>:<localName> y <localName> – busca el atributo para un hijo o los hijos con el prefijo especificado y/o el nombre local. Devuelve un valor nulo o el hijo con el nombre especificado (si sólo hay uno) o una lista de nodos con todos los atributos hijo que coinciden con el criterio.
- {namespaceURI}<localName> – busca el atributo para todos los atributos hijo que pertenecen al namespaceURI especificado y con el mismo nombre local que el nombre especificado.

Nota: Si se proporciona un prefijo, se ignorará y el mecanismo de resolución de nombres sólo buscará los namespaceURI y localName especificados. El objeto resuelto puede ser nulo, un único objeto de atributo o una lista de nodos que contenga todos los atributos que coinciden con el nombre de la búsqueda.

Lista de nodos

Cuando el objeto de contexto es una instancia de este tipo, el mecanismo de resolución de nombres buscará la sintaxis siguiente:

- [<index>] – especifica la posición del elemento en la lista de nodos que debe recuperarse. Devuelve un valor nulo o el objeto en la posición especificada. Genera una excepción si el índice está fuera de los límites.
- @<prefix>:<localName> y @<localName> – busca cada uno de los elementos de la lista de nodos para una propiedad con el mismo prefijo

y/o el nombre local. Devuelve un valor nulo, un objeto de propiedad (si sólo se encuentra uno) o una lista de todos los objetos de propiedad que se han encontrado en la lista de nodos.

- `@{namespaceURI}<localName>` – busca cada uno de los atributos para una propiedad que pertenece al namespaceURI especificado y que tiene el mismo nombre local que el nombre especificado.

Nota: Si se proporciona un prefijo, se ignorará y el mecanismo de resolución de nombres sólo buscará los namespaceURI y localName especificados. El objeto resuelto puede ser nulo o un objeto de propiedad (si sólo se encuentra uno) o una lista de nodos de todos los objetos de propiedad que se han encontrado en la lista de nodos.

- `<prefix>:<localName>` y `<localName>` – busca cada uno de los elementos de la lista de nodos para un hijo que tenga el mismo prefijo y/o el nombre local. Devuelve un valor nulo, un objeto de atributo (si sólo encuentra uno) o una lista de nodos de todos los objetos de atributo que se han encontrado en la lista de nodos.
- `{namespaceURI}<localName>` – busca cada uno de los atributos para todos los atributos hijo que pertenecen al namespaceURI especificado y con el mismo nombre local que el nombre especificado.

Nota: Si se proporciona un prefijo, se ignorará y el mecanismo de resolución de nombres sólo buscará los namespaceURI y localName especificados. El objeto resuelto puede ser nulo, un único objeto de atributo o una lista de nodos que contenga todos los atributos que coinciden con el nombre de la búsqueda.

Ahora dispone de las opciones siguientes:

1. La posibilidad de acceder a todos los elementos *d* haciendo referencia a ellos empezando desde la parte superior; por ejemplo `entry.a.b.c.d` – esto devuelve un objeto de tipo de lista de nodos con todos los atributos *d* que coinciden con esta vía de acceso. En nuestro ejemplo esto devuelve los tres elementos *d*.
2. La posibilidad de acceder a los atributos especificando el prefijo y el nombre local; por ejemplo, `entry["a.b.c.pref1:d"]` – esto devuelve un único atributo, el que tiene un prefijo "pref1".
3. La posibilidad de acceder a cada atributo de una lista de nodos utilizando la notación `[]`; por ejemplo, `entry.a.b.c.d[0]` – esto devuelve un único atributo, es decir, el primer elemento *d* de la estructura anterior.
4. La posibilidad de navegar por los elementos utilizando la notación `[]`; por ejemplo, `entry.a.b[0].c.d` – esto devuelve una lista de atributos, pero esta vez contiene todos los atributos *d* de la primera rama *b*.
5. La posibilidad de obtener la propiedad de un atributo (es decir, un atributo de un elemento utilizando el convenio de denominación de DOM) utilizando la notación `@ notation`; por ejemplo, `entry.a.b.c.d[0]["@propPrefix:propName"]` – esto devuelve un objeto de tipo serie que contiene el valor de dicha propiedad (es decir, el valor del atributo según DOM). Tenga en cuenta que `propPrefix` y `propName` están separados con dos puntos (":") y que si la propiedad tiene un prefijo, es obligatorio especificar ambos prefijos y el nombre para que pueda encontrarse la propiedad. Como alternativa, se pueden utilizar el espacio de nombres y el nombre de propiedad para buscar una propiedad que tiene un prefijo.
6. La posibilidad de llamar a métodos de un atributo antes de buscar un hijo con el nombre del método que el usuario desea ejecutar; por ejemplo,

`entry.a.b.[0].getChildNodes()` – esto devuelve un objeto de lista de nodos que contiene todos los valores de atributo que el primer atributo *b* contiene.

7. La posibilidad de acceder a atributos de entrada mediante el nombre; por ejemplo, `entry.attrName` – esto devuelve el atributo correlacionado con la clave *attrName*.
8. La posibilidad de acceder a propiedades de entrada utilizando la notación `@`; por ejemplo, `entry.@propName` – esto devuelve el objeto correlacionado como una propiedad con la clave *propName*.
9. La posibilidad de acceder a nodos hijo especificando el espacio de nombres al que pertenecen los hijos. Por ejemplo, `work.a.b[{someNamespace}c]` – esto devuelve todos los objetos *c* que pertenecen al espacio de nombres "someNamespace".

Nota:

1. Si el nombre de un atributo es igual que el nombre de un método de una entrada/atributo, se llamará al método. Si desea acceder al atributo, debe utilizar los métodos de objeto igual que antes (es decir, `Entry.getAttribute("getAttribute");`)
2. Si se utiliza una entrada simple, el motor de scripts no convertirá la entrada en una entrada jerárquica a menos que se especifique el espacio de nombres del atributo que se busca. Por ejemplo: `entry["{ns}element"]`. El motor de scripts convertirá automáticamente la entrada a una entrada jerárquica si el objeto de contexto es de tipo atributo. Por ejemplo, en este caso: `entry.attr.child`.
3. Si el atributo que se busca contiene puntos en su nombre y la entrada es simple, debe utilizar la notación con corchetes en lugar de la notación con puntos o forzar la entrada para convertirla en jerárquica antes de resolver el nodo hijo. Por ejemplo, si la entrada es simple, no puede acceder al atributo `http.body` utilizando la llamada `entry.http.body`. Para poderlo hacer, en su lugar debe utilizar: `entry["http.body"]` o llamar a `entry.enableDOM()` antes de llamar a `entry.http.body`.
4. Si tiene intención de utilizar la referencia recuperada de la expresión como, por ejemplo, `a.b.c.d` más de una vez, es recomendable asignar dicha referencia a una variable local puesto que cada evaluación repetida de la misma expresión producirá una carga adicional para recuperar la misma referencia.
5. Si, por ejemplo, se utiliza la expresión `entry.a.b.c[0].d`, hará referencia a un objeto de atributo; pero si se utiliza `entry.a.b[0].c.d`, hará referencia a un objeto de lista de nodos. Para reconocer el objeto referido puede seleccionar el nombre del objeto, por ejemplo:

```
var obj = a.b.c.d;
if (obj.getClass().getSimpleName().equals("Attribute") ) {
    // manejar como atributo
} else {
    // manejar como una lista de atributos
}
```

Si, por ejemplo, necesita manejar cada elemento que devuelve una expresión, incluso si tan solo se devuelve un elemento, puede utilizar una estructura de bucle `for/in` como la siguiente:

```
for (obj in entry.a.b.c.d) {
    // obj será un objeto de tipo atributo
}
```

Ahora está disponible el mismo uso con el objeto de entrada, por ejemplo:

```

for (obj in work) {
    // obj será un atributo de entrada
}

```

6. La opciones del motor de scripts también permiten asignar valores. Por ejemplo, son válidas las expresiones siguientes:

entry.a.b.c.d[0]["@propPrefix:propName"] = "valor nuevo";

Cambia el valor de la propiedad. Si la propiedad no existe, se creará.

entry.a.b.c.d[0][0] = "valor nuevo";

Sustituye el valor de atributo en la posición 0.

entry.a.b.c.d[0][1] = "otro valor";

Añade otro valor al atributo (si el índice es igual que el tamaño de la matriz de valores).

entry.a.b.c.d[0][a.b.c.d[0].size()] = "valor añadido";

Si necesita añadir un nuevo valor a la lista de objetos, pero no conoce la posición del último elemento, puede utilizar `Attribute#size()` para obtener el número de hijos que tiene este elemento.

new com.ibm.di.entry.Entry().@propName = "unValor";

Creará una nueva propiedad en el objeto de entrada con el nombre `propName` si no existe y establecerá su valor en la serie "unValor".

entry.a.b.c[1] = "valor";

Resuelve `entry.a.b.c` como una lista de nodos y añade automáticamente como un valor el nuevo objeto de serie al segundo elemento del objeto de lista de nodos resuelto. Si, por ejemplo, `entry.a.b.c` se resuelve en un atributo, el nuevo valor de serie sustituirá al segundo valor del atributo `c` resuelto.

entry.a.b.c[2]["pref3:d"] = "valor";

Añade un nuevo hijo `pref3:d` al tercer atributo `c` de la lista `entry.a.b.c`. Tenga en cuenta que `entry.a.b.c[2]` se resuelve en un atributo y en una lista.

entry.a.b.c["{namespaceURI}:d"] = "miValorTexto";

Establece un valor en el primer atributo hijo de `c` que tenga un nombre local igual a `d` y que pertenezca al espacio de nombres `namespaceURI`. Si no se encuentra este atributo, se crea uno nuevo y se le asigna el valor. Cuando se crea un atributo, también puede asociar el espacio de nombres a un prefijo. En nuestro caso, podría realizarse utilizando lo siguiente: `entry.a.b.c["{namespaceURI}prefix:d"] = "myTextValue";`.

entry["{http://ibm.com/xmlns}/first.second.third"] = null;

Primero se intentará encontrar el elemento con el nombre local "first" del espacio de nombres "http://ibm.com/xmlns/". Si falla, creará el elemento y, a continuación, intentará resolver su elemento hijo "second". Si falla, lo creará y establecerá su espacio de nombres a uno del primer elemento. Finalmente, se intentará resolver el tercer elemento. Si falla, se creará el último elemento sin valores. Esto equivale a `entry["{http://ibm.com/xmlns}/first.{http://ibm.com/xmlns}/second.{http://ibm.com/xmlns}/third"] = null;`

Creación de la estructura anterior con script

```

//
crear una nueva entrada que contendrá la estructura
var entry = new com.ibm.di.entry.Entry();
// crear la primera rama
var d = entry.newAttribute("a.b.c.d");
// crear una nueva propiedad y asignarle un valor

```

```

d["@propPrefix:PropName"] = "valor";
// crear un nuevo valor del atributo d
d[0] = "D1";
// añadir un nuevo valor para el atributo entry.a.b
entry.a.b.appendChild(entry.createElement("c"));
// crear un nuevo atributo d con un prefijo en el segundo atributo c
// y asignar la serie "D2" como su primer valor
entry.a.b.c[1]["pref1:d"] = "D2";
// crear un nuevo hijo del atributo a
entry.a.appendChild(entry.createElement("b"));
// elegir el segundo atributo de la lista de nodos entry.a.b y crear un nuevo hijo denominado c
entry.a.b[1].appendChild(entry.createElement("c"));
// crear el hijo d del atributo c
entry.a.b[1].c["pref2:d"] = "D3";

```

Navegación utilizando XPath

La clase de entrada proporciona métodos adecuados para navegar y recuperar datos basados en expresiones XPath. La utilización de XPath para consultar datos de una entrada es mucho más avanzada que utilizar cualquier navegación simple en scripts. Es mucho más fácil implementar una búsqueda y/o una lógica de coincidencia de valores en una única expresión que escribir scripts de varias líneas para conseguir el mismo resultado.

La clase de entrada proporciona los métodos siguientes:

- `NodeList getNodeList (String xPath);`
- `Attribute getFirstAttribute(String xPath);`
- `String getStringValue(String xPath);`
- `Number getNumberValue(String xPath);`
- `Boolean getBooleanValue(String xPath);`

Simplificación de estructuras jerárquicas

Cuando se trabaja con datos jerárquicos a veces es necesario simplificar los nodos de la jerarquía. Para ello puede escribir un script para atravesar el árbol o utilizar uno de los métodos siguientes:

- `getElementsByTagName(String namespace, String localName);`
- `getElementsByTagName(String tagName);`

Estos métodos atraviesan el árbol y buscan elementos con el nombre y el espacio de nombres especificados. La API de DOM permite que estos métodos acepten el carácter "*" tanto para nombres como para espacios de nombre. Este carácter representa un comodín, que se utiliza para comparar cualquier nombre/espacio de nombres de elemento. La utilización de este carácter para el nombre simplifica el árbol al devolver todos los nodos de atributo en una lista de nodos. Debe tener en cuenta que la estructura de la jerarquía no ha cambiado. La lista de nodos que se devuelve es únicamente un contenedor de los nodos del elemento que se han encontrado en el árbol.

Si ejecuta el script siguiente en la entrada desde la estructura de la sección "Navegación en scripts" en la página 56:

```
var list = entry.getElementsByTagName("*");
```

la variable *list* contendrá la estructura siguiente:

```

list
|
+ a
|
+ b

```

```
|
+ c
|
+ d
|
+ c
|
+ pref:d
|
+ b
|
+ c
|
+ pref2:d
```

Excepciones

Se genera una excepción en los casos siguientes:

- Si se llama al método `entry.appendChild(newAttr)` y la entrada ya contiene un atributo con el nombre del objeto `newAttr` que se ha pasado al método.
- Si para alguno de los métodos definidos por el Documento/Nodo/Elemento e implantados por las clases de entrada/atributo se pasa un parámetro inesperado.
- Se genera una excepción `ArrayIndexOutOfBoundsException` si el script proporcionado hace referencia a un índice de un atributo/lista de nodos que no existe.

Integración de los scripts en su solución

Como ya se ha explicado, los scripts se utilizan cuando se necesitan procesos personalizados en la solución de integración. Los métodos recomendados con IBM Security Directory Integrator dividen este proceso personalizado en dos categorías: transformación de los atributos y control de flujo.

Nota: se trata de un convenio y no de una limitación o una norma que impone el sistema. La necesidad de procesar datos personalizados surge inevitablemente en un punto identificable del flujo de datos (por ejemplo, antes de comenzar el proceso, antes de procesar una entrada determinada, después de un error, etc.), por lo tanto, si sitúa el código tan cerca de este punto como sea posible, las soluciones resultarán más fáciles de comprender y mantener.

El lugar más lógico para realizar la transformación de atributos corresponde a las **Correlaciones de atributos**, tanto de entrada como de salida. Si necesita calcular un atributo nuevo necesario para la lógica de scripts u otros conectores que siguen en la línea de ensamblaje, la mejor forma es hacerlo en una **Correlación de entrada**, si es posible. Alternativamente, si debe transformar atributos para un solo origen de salida, puede evitar que el objeto de entrada **work** se sobrecargue con transformaciones específicas de salida colocándolos en la **Correlación de salida** del conector en cuestión.

La segunda categoría de la lógica personalizada, el control de flujo, se implementa mejor en los complementos de software que se invocan en el punto del flujo de trabajo automatizado donde se necesita la lógica. El acceso a estos puntos de control es sencillo desde el Editor de configuración. La implementación del proceso personalizado consiste simplemente en identificar el punto de control correcto y añadir el script en la ventana de edición adecuada.

Los **componentes de script** de la línea de ensamblaje, bloques independientes de código de script, también proporcionan un lugar para crear su propio proceso personalizado y luego permiten volver a situar el código en la línea de ensamblaje. Aunque los componentes de script se utilizan con frecuencia durante la fase de prueba y depuración, también pueden jugar un papel importante en una configuración de producción. Simplemente recuerde que debe asignar a los componentes un nombre evidente e incluir algún tipo de documentación en el propio script para describir ⁷ por qué ha implementado esta lógica en un componente de script y no en una **Correlación de atributos** o en un **Complemento de software**.

Aunque es importante identificar correctamente el punto de control adecuado para la entrada del script, también es igualmente importante limitar el ámbito del script para cubrir el objetivo concreto asociado al punto de control. Si mantiene sus unidades de lógica independientes unas de otras, hay más posibilidades de que sean reutilizables y menos de que se interrumpan cuando los componentes se vuelven a ordenar o a utilizar en otros contextos. Una forma de crear código reutilizable es crear funciones propias en la **Biblioteca de scripts** (o en un complemento de software **Prólogo**) que implemente la lógica que se utiliza frecuentemente, en lugar de copiar y pegar el mismo código en varios lugares.

A continuación, se ofrece un resumen de los métodos más recomendados que debe tener en cuenta cuando cree sus soluciones:

- Realice las manipulaciones de atributos en las Correlaciones de atributos.
- Coloque el control de flujo (filtrado, validación, ramificación, etc.) en los complementos de software y, donde sea necesario, componentes de script de línea de ensamblaje.
- Utilice el comportamiento automatizado siempre que sea posible; por ejemplo, el flujo de trabajo de la línea de ensamblaje y las modalidades de conector.
- Simplifique la solución manteniendo las líneas de ensamblaje cortas y focalizadas.
- Coloque la lógica que se utiliza con frecuencia en los bloques discretos; por ejemplo, scripts de la sección Recursos.
- Considere su reutilización.

Vale la pena volver a mencionar que aunque los métodos descritos anteriormente son los recomendados, es posible que se encuentre en situaciones en las que debe desviarse del convenio establecido. Si es este el caso, la documentación de la solución resulta vital para mantener y mejorar su trabajo con el tiempo.

Control de la ejecución con scripts

El motor de expone varias clases y objetos a los que es posible acceder, y que se pueden leer y modificar desde scripts creados por el usuario en una línea de ensamblaje. Estos objetos representan el estado de la línea de ensamblaje y de todo el entorno de IBM Security Directory Integrator en cualquier momento. Al modificar cualquiera de estos objetos, se modifica el entorno de IBM Security Directory Integrator y, por consiguiente, ello afecta a la ejecución del proceso de integración.

Nota: se pueden aplicar cambios a instancias de un componente o línea de ensamblaje. También se pueden realizar cambios en parámetros operativos como,

7. A otros y a usted mismo cuando repase el código en otro momento.

por ejemplo, parámetros del sistema o Java. También se pueden realizar cambios en el archivo de configuración, o Config. En este caso, las instancias nuevas de objetos Config reflejan estos cambios.

Para obtener más información sobre los objetos globales, consulte los Javadocs que se incluyen como parte del producto IBM Security Directory Integrator seleccionando **Ayuda > Javadocs** en el Editor de configuración.

Se puede encontrar una descripción de todas las clases e instancias disponibles en el paquete de instalación.

Si comprende las clases y las interfaces que se exponen, comprenderá mejor los elementos del motor de IBM Security Directory Integrator, así como las relaciones entre ellos.

Utilización de variables

Es importante diferenciar entre el objeto de contenedor estándar de los datos que se están procesando (el objeto de entrada) y otras variables genéricas y tipos de datos proporcionados por JavaScript, además de los creados por el usuario. Las únicas limitaciones son su creatividad y las posibilidades del lenguaje de script respecto a lo que se puede situar en los scripts en las soluciones de IBM Security Directory Integrator. No obstante, cuando manipula datos en el contexto del flujo de datos, debe tener en cuenta y utilizar la estructura del objeto de entrada.

Los objetos de entrada transportan atributos que son, a su vez, el contenedor de los valores de los datos. Los valores de los atributos son objetos propiamente dichos (`java.lang.String`, `java.util.Date` y estructuras más complejas). Un valor de atributo puede ser también otro objeto de entrada con su propio conjunto de atributos y valores. La función de IBM Security Directory Integrator es comprender cómo se almacenan los datos en el sistema conectado y también cómo se convierten estos tipos nativos en la representación de datos propia del sistema, y desde dicha representación, situada en objetos Java.

Si conoce la clase del valor del atributo, puede acceder correctamente e interpretar este valor. Por ejemplo, si un atributo `java.lang.String` contiene un valor de coma flotante que desea utilizar como coma flotante, primero debe transformar manualmente este valor, mediante el lenguaje de creación de scripts, en algún tipo de datos numérico.

Cuando crea variables o procesos que no están directamente relacionados con el flujo de datos del proceso de integración y los objetos globales disponibles, se aplica el principio siguiente: puede declarar y utilizar cualquier variable (objetos) habilitada mediante el lenguaje de scripts. La finalidad de estas variables es ayudarle a obtener un objetivo específico asociado al punto de control en el que incluye el script. Las variables sólo deben servir como almacenamientos intermedios temporales y no deben intentar afectar al estado del entorno de IBM Security Directory Integrator.

Utilización de propiedades

Durante la vida de una línea de ensamblaje, el servidor de IBM Security Directory Integrator facilita cierto número de propiedades de componente, relacionadas con el entorno de ejecución de la línea de ensamblaje, que se pueden consultar en los scripts: desde componentes de scripts o enganches de componente. Se puede definir hasta una propiedad (*lastCallStatus*).

Puede acceder a las propiedades utilizando un objeto de línea de ensamblaje como, por ejemplo, un conector y llamar a su método `get(nombre_propiedad)` para extraer el valor de `nombre_propiedad`; como alternativa, utilice el método `put(nombre_propiedad, valor_propiedad)` para definir la propiedad en el valor que desee. Cuando se definen propiedades, el nombre de propiedad o el valor de propiedad no pueden ser nulos; si uno de ellos es nulo, se generará una excepción con el mensaje adecuado.

El conjunto de propiedades disponibles es el siguiente:

Tabla 6. Propiedades de componente disponibles durante la ejecución de una línea de ensamblaje

Propiedad	Utilización
<code>numErrors</code>	El número de errores producidos.
<code>numAdd</code>	Número total de entradas que la línea de ensamblaje ha añadido (realizadas por los conectores en modalidad Sólo adición).
<code>numModify</code>	Número total de entradas que la línea de ensamblaje ha modificado (realizadas por los conectores en modalidad Actualizar).
<code>numDelete</code>	Número total de entradas que la línea de ensamblaje ha suprimido (realizadas por los conectores en modalidad Supresión).
<code>numGet</code>	Número total de entradas que la línea de ensamblaje ha recuperado (realizadas por los conectores en modalidad Iterador).
<code>numGetTries</code>	Número total de veces que la línea de ensamblaje ha intentado recuperar una entrada (realizado por los conectores en modalidad Iterador).
<code>numGetClient</code>	Número total de clientes aceptados (disponible para los conectores en modalidad Servidor).
<code>numGetClientTries</code>	Número total de veces que la línea de ensamblaje ha intentado acceder al siguiente cliente conectado (realizado por los conectores en modalidad Servidor).
<code>numCallreply</code>	Número total de operaciones de Llamada/Respuesta que la línea de ensamblaje ha ejecutado (realizadas por los conectores en modalidad CallReply).
<code>numLookup</code>	Número total de operaciones de Búsqueda que la línea de ensamblaje ha ejecutado (realizadas por los conectores en modalidad Actualizar/Suprimir/Buscar).
<code>numNoChange</code>	Número total de entradas que la línea de ensamblaje ha procesado pero ha dejado sin modificar.
<code>numSkipped</code>	Número total de entradas que la línea de ensamblaje ha pasado por alto.
<code>numIgnored</code>	Número total de entradas que la línea de ensamblaje ha ignorado (realizadas por los conectores en modalidad Actualizar/Delta).
<code>lastCallStatus</code>	Contiene el estado para la ejecución de la línea de ensamblaje. No es únicamente una propiedad de sólo lectura y el usuario la puede modificar. El valor de esta propiedad es "fail" o "success" según la ejecución de línea de ensamblaje.
<code>lastConn</code>	Entrada Conn de la última operación de conector. Antes de la primera operación de conector, <code>lastConn</code> tiene un valor nulo.
<code>lastError</code>	El último error como un objeto Java.
<code>hooksInvoked</code>	Una <code>java.util.List</code> de los nombres que los complementos de software han invocado la última vez que se ha invocado el componente. Los nombres son nombres internos.

Tabla 6. Propiedades de componente disponibles durante la ejecución de una línea de ensamblaje (continuación)

Propiedad	Utilización
success	Esta propiedad se define como 'true' si la última operación fue satisfactoria; de lo contrario, se define como 'false'.
endOfData	'True' cuando el componente de iterador llega al fin de los datos; de lo contrario, es 'false'. La modificación de esta propiedad no tiene efecto.

Nota: Si se intenta modificar una propiedad de sólo lectura, se generará una excepción con el mensaje adecuado.

Ejemplo

Para ilustrar la utilización de estas propiedades de componente, supongamos que tiene un conector del sistema de archivos llamado *FS* y varios componentes de script. El siguiente código JavaScript se encuentra en el enganche "GetNext satisfactorio" de FS:

```
if(work.getString("ID") == null)
throw new java.lang.Exception("Missing ID");
//para ejecutar correctamente el ciclo de LE necesito un ID, por lo tanto se
genera una excepción
```

Y este código JavaScript se encuentra en el enganche "DefaultOnError" de FS:

```
if(FS.get("lastError").getMessage().equals("Missing ID")) {
//Se puede corregir añadiendo un ID que ayudará a ejecutar la LE
work.setAttribute("ID", "SomeID"); //añadir el ID
if(FS.get("fixErrors") == null) {
var vector = new java.util.Vector();
vector.add(FS.get("lastError"));
FS.put("fixErrors", vector); //guardar errores corregidos en la propiedad personalizada
} else { //Ya he corregido antes un error similar
var vector = FS.get("fixErrors");
vector.add(FS.get("lastError"));
FS.put("fixErrors", vector); //guardar errores corregidos en la propiedad personalizada
}
FS.put("lastCallStatus", "success");
} else { //No se ha podido corregir el error
if(FS.get("notFixErrors") == null) {
var vector = new java.util.Vector();
vector.add(FS.get("lastError"));
FS.put("notFixErrors", vector); //guardar los errores no corregidos en mi propiedad personalizada
} else {
var vector = FS.get("notFixErrors");
vector.add(FS.get("lastError"));
FS.put("notFixErrors", vector); //guardar los errores no corregidos en mi propiedad personalizada
}
FS.put("lastCallStatus", "fail");
}
```

Por último, en un componente de script, considere el código siguiente:

```
main.logmsg("AL Cycle status: " + FS.get("lastCallStatus"));
//imprimir el estado de LE para este ciclo de LE
//También se puede informar de los errores producidos
// durante la ejecución de LE mediante la propiedad personalizada, "vector"
```

Puntos de control para scripts

Scripts en una línea de ensamblaje

Las líneas de ensamblaje tienen previsto un comportamiento previamente programado estándar. Si desea desviarse de este comportamiento estándar, puede hacerlo implementando su propia lógica empresarial mediante scripts.

Componente de script

Puede añadir componentes de script a la línea de ensamblaje además de conectores pulsando el botón derecho del ratón en **Insertar componentes...** > **Scripts** > **Script** en el componente o sección adecuados en la ventana Componentes de línea de ensamblaje del editor de líneas de ensamblaje. El componente de script se inicia una vez por cada entrada procesada por la línea de ensamblaje y puede colocarse en cualquier lugar de la línea de ensamblaje.

Nota: los iteradores se continúan procesando en primer lugar, incluso si coloca el componente de script delante de ellos en la línea de ensamblaje.

Para obtener más información, consulte el apartado “Componentes de script” en la página 24.

Complementos de software de línea de ensamblaje

Los complementos de software de línea de ensamblaje (es decir, complementos de software que se aplican a la línea de ensamblaje de manera global, no a cada componente individual) se encuentran en la pestaña **Complementos de software** de la línea de ensamblaje. Estos complementos de software se ejecutan solamente una vez por cada ejecución de la línea de ensamblaje o, en el caso de la **Solicitud de conclusión**, cuando algún proceso externo indica a la línea de ensamblaje que concluya. Sin embargo, si inicia varias veces la línea de ensamblaje (por ejemplo, utilizando el conector de línea de ensamblaje), entonces también iniciará los complementos de software varias veces.

Los complementos de software incluidos en un conector sólo se evalúan y ejecutan (si está definido y no vacío) cuando se ejecuta el conector en el que están definidos. Consulte el apartado “Scripts en un conector” en la página 75 para obtener más información.

Ganchos de servidor

Los complementos de software de servidor le permiten escribir código JavaScript para responder a sucesos y errores que se producen a nivel de servidor. A diferencia de los complementos de software de línea de ensamblaje y de componente, los complementos de servidor se almacenan en archivos de script separados. Estos archivos se guardan en la carpeta *serverhooks* del directorio de la solución actual y deben contener funciones de script indicadas específicamente. El servidor de IBM Security Directory Integrator y las instancias de configuración proporcionan un método para que los componentes de IBM Security Directory Integrator invoquen complementos de software a nivel de servidor personalizado. Un complemento de software de servidor es un nombre de función que se define en un archivo de script. Las implementaciones de función se proporcionan simplemente dejando los archivos de script en el directorio “serverhooks” del directorio de soluciones.

Estos scripts, además de que pueda llamarlos el servidor cuando se producen determinados sucesos, también pueden invocarse desde scripts. Las llamadas a estos complementos de software están sincronizadas para evitar posibles problemas de ejecución de hebras.

Durante el proceso de inicio, IBM Security Directory Integrator carga y ejecuta todos los scripts de usuario que se encuentran en el subdirectorio *serverhooks*. Los scripts pueden o no contener declaraciones de función. Un script que no tiene ninguna declaración de función se ejecuta una vez durante el arranque antes de que se inicie alguna instancia de configuración. El código que define funciones

estándar del complemento de software de servidor de IBM Security Directory Integrator tiene el prefijo "SDI_", y se ejecuta en distintos puntos durante la operación.

Todas las funciones de complemento de software de servidor de IBM Security Directory Integrator tienen la signatura de JavaScript siguiente:

```
/**
 * @param NameOfFunction Instancia de configuración que invoca la función
 * @param source Componente que invoca la función
 * @param user Información sobre parámetros arbitraria del origen
 */
function SDI_functionName(Name_of_function, source, user) {
}
```

Los parámetros "NameOfFunction" y "source" siempre proporcionan acceso a la instancia de configuración y al componente que efectúa la llamada, respectivamente. El parámetro "user" se utiliza con fines distintos en las diversas funciones de complemento de software.

A continuación se indican los nombres de función estándar que son llamados por distintos componentes de IBM Security Directory Integrator:

Nombre de función	Llamada por (origen)	Parámetro de usuario y valor esperado
SDI_ALStarted	Instancia de configuración	Se llama cuando se inicia una línea de ensamblaje. user = Línea de ensamblaje que se ha iniciado <i>valor de retorno ignorado</i>
SDI_ALStopped	Instancia de configuración	Se llama cuando se detiene la línea de ensamblaje. user = Línea de ensamblaje que se ha detenido <i>valor de retorno ignorado</i>
SDI_ConfigStarted	Servidor	Se llama cuando se inicia una instancia de configuración. user = Instancia de configuración <i>valor de retorno ignorado</i>
SDI_ConfigStopped	Servidor	Se llama después de detener una instancia de configuración. user = Instancia de configuración <i>valor de retorno ignorado</i>

Nombre de función	Llamada por (origen)	Parámetro de usuario y valor esperado
SDI_Shutdown	Servidor/instancia de configuración	Se llama inmediatamente antes de que el servidor de IBM Security Directory Integrator termine la Máquina virtual Java (por ejemplo, System.exit()). user = Estado de salida (número entero) <i>valor de retorno ignorado</i>

El acceso a las funciones de complemento de software de servidor de IBM Security Directory Integrator se realiza mediante el método `main.invokeServerHook()`. Esta función se sincroniza para impedir que más de una hebra ejecute un mismo complemento de software a la vez. Las llamadas se invocan de forma síncrona, por lo que el llamante espera a que finalice la función. Como consecuencia, debe tener cuidado en no permanecer demasiado tiempo en un complemento de software de servidor.

Como se ha indicado anteriormente, los scripts se definen y se ponen a disposición del usuario a través de los archivos que se crean en el subdirectorio “serverhooks” del directorio de la solución. Los scripts que contienen información confidencial deberían cifrarse con la API de servidor antes de añadirlos al directorio. La herramienta `serverapi/cryptutils` está disponible para cifrar archivos de script. Observe que IBM Security Directory Integrator intenta automáticamente descifrar los archivos con extensión `.jse`, por esto preferiblemente los archivos cifrados deberán tener esta extensión.

Además, los archivos del directorio `serverhooks` se cargan y ejecutan primero ordenando los nombres de archivo según un método de clasificación en el que se tienen en cuenta las mayúsculas y minúsculas y en el que se sigue la secuencia de clasificación estándar de la plataforma. Todos los archivos del directorio de nivel superior se cargan antes de procesar los archivos de los subdirectorios.

A continuación se indican algunos ejemplos de complemento de software de servidor:

- Si desea que un objeto personalizado se cargue siempre en IBM Security Directory Integrator para utilizarlo en scripts propios, puede crear una instancia del objeto a partir de un fragmento de código JavaScript que se enlaza al complemento de software de servidor cuando se inicia IBM Security Directory Integrator. Esto le proporciona mayor control que simplemente hacer referencia a la clase contenida en la carpeta Bibliotecas de Java del navegador de configuración.
- Una o más líneas de ensamblaje personalizadas que ha iniciado crea un registro de auditoría para estos sucesos o propaga estos sucesos a otros sistemas mediante algún tipo de transporte (SNMP, HTTP, JMS, etc.).
- Una política de seguridad corporativa implementada por el usuario y que se invoca cada vez que se carga una configuración o se inicia una línea de ensamblaje.

Cómo llamar a complementos de software de servidor desde un script: La clase `com.ibm.di.server.RS` (variable de script "main") tiene un método para invocar complementos de software de servidor:

```
/**
 * Invoca un complemento de software de servidor.
 *
 * @param name Nombre del complemento de software (también nombre de archivo)
 * @param caller Objeto que invoca al complemento de software.
 * @param userInfo Información arbitraria para el complemento de software del llamante
 */

public Object invokeServerHook(
    String name,
    Object caller,
    Object userInfo) throws Exception;
```

Esta llamada puede devolver un objeto Java (de cualquier tipo), por lo que, aunque IBM Security Directory Integrator pasa esto por alto durante la ejecución del complemento de software de servidor, el usuario puede utilizar los valores devueltos en sus propias llamadas de script.

En un script de una línea de ensamblaje, puede realizar lo siguiente:

```
main.invokeServerHook("MyCustomHook", this, "custom information");
```

Acceso a los componentes de la línea de ensamblaje que se encuentran dentro de la misma

Cada uno de los componentes de la línea de ensamblaje está disponible como una variable de script previamente registrada con el nombre que el usuario selecciona para el componente.

Tenga en cuenta que puede cargar dinámicamente los componentes con llamadas de script a funciones de tipo `system.getConnector()`, si bien esto debe hacerlo un usuario con experiencia.⁸

Pase de parámetros de la línea de ensamblaje

Hay tres formas de pasar datos a una línea de ensamblaje:

- Generar su propia entrada inicial en la línea de ensamblaje, por ejemplo, en un script de prólogo.
- Pasar datos desde uno o más iteradores.
- Iniciar la línea de ensamblaje con los parámetros de otra línea de ensamblaje utilizando el conector o el componente de función de la línea de ensamblaje, o utilizando una llamada de API.

Si desea iniciar una línea de ensamblaje con parámetros de otra línea de ensamblaje, tiene dos opciones:

- Utilizar el método **TCB** (bloque de llamada de tareas), que es el preferido. El TCB se detalla más adelante.
- Proporcionar directamente una entrada work inicial.

8. El objeto de conector que se obtiene de esta llamada es un objeto de *interfaz de conector*, y es la parte específica del origen de datos de un conector de línea de ensamblaje. Cuando cambia el *tipo* de un conector, en realidad está cambiando la inteligencia de su origen de datos (la interfaz de conector) que proporciona la funcionalidad para acceder a los datos de un sistema, un servicio o un almacén de datos específico. La mayor parte de la funcionalidad de un conector de línea de ensamblaje, incluyendo las correlaciones de atributos, los criterios de enlace y los complementos de software, la proporciona el kernel de IBM Security Directory Integrator y se mantiene intacta cuando se conmutan los tipos de conector.

Nota: Estas dos opciones se proporcionan por motivos de compatibilidad con versiones anteriores.

TCB (bloque de llamada de tareas)

El TCB (bloque de llamada de tareas) es un tipo especial de objeto de entrada que un llamante utiliza para establecer varios parámetros para una línea de ensamblaje.

Uso básico: El TCB (bloque de llamada de tareas) es un tipo especial de objeto de entrada que un llamante utiliza para establecer varios parámetros para una línea de ensamblaje. El TCB puede proporcionar al usuario una lista de parámetros de entrada o salida especificados por una línea de ensamblaje, incluyendo los códigos de operación definidos en la pestaña **Operaciones** de la línea de ensamblaje, así como habilitar el llamante para definir parámetros para los conectores de la línea de ensamblaje. El TCB consta de las secciones lógicas siguientes:

- La entrada work inicial pasada a la línea de ensamblaje:
tcb.setInitialWorkEntry()
- Los parámetros del conector: tcb.setConnectorParameter()
- Las reglas de correlación de entrada o salida para la línea de ensamblaje, definidas en la pestaña **Operaciones** del Editor de configuración
- Un objeto *acumulador* que recibe todas las entradas work de la línea de ensamblaje: tcb.setAccumulator()

Por ejemplo, iniciar una línea de ensamblaje con una entrada work inicial y establecer el parámetro *filePath* de un conector denominado MyInput como d:\myinput.txt se consigue con el código siguiente:

```
var tcb = system.newTCB(); // Crear un TCB nuevo
var myIWE = system.newEntry(); // Crear un objeto de entrada nuevo
myIWE.setAttribute("name","John Doe"); // Añadir un atributo a myIWE
tcb.setInitialWorkEntry ( myIWE ); // Establecer la IWE y los parámetros
// Observe que debido a que esto es una cadena JavaScript, debemos hacer que la barra
// inclinada se interprete de forma literal añadiendo un carácter de escape
// o utilice una barra inclinada invertida (sintaxis de Windows)
tcb.setConnectorParameter ( "MyInput", "filePath", "d:\myinput.txt" );

var al = main.startAL ( "MyAssemblyLine", tcb ); // Iniciar la línea de ensamblaje con tcb
al.join(); // Esperar a que la línea de ensamblaje finalice
```

Inicio de una línea de ensamblaje con operaciones: Las líneas de ensamblaje se pueden definir con *Operaciones*; un concepto mediante el cual se define un número de correlaciones de entrada para la línea de ensamblaje. Según cómo se invoca la línea de ensamblaje, se activa una correlación de entrada diferente. Dentro de la línea de ensamblaje necesitará comprobar la entrada op para averiguar qué operación está activa y utilizar componentes de rama para adaptar el flujo dentro de la línea de ensamblaje a la operación pertinente.

Uno de las formas de iniciar una línea de ensamblaje con una operación es mediante un TCB y código de script.

Si una línea de ensamblaje denominada "al1" tiene las operaciones siguientes: "Default", "Op1" y "Op2", este script iniciará la línea de ensamblaje con la operación establecida en "Op1":

```
var tcb = system.newTCB("al1");
tcb.setALOperation("Op1");
main.startAL(tcb);
```

Si no se especifica ninguna operación se iniciará la línea de ensamblaje con la operación establecida en "Default":

```
var tcb = system.newTCB("a11");
main.startAL(tcb);
```

En caso de que la línea de ensamblaje no tenga una operación Default (por ejemplo, sólo las operaciones "Op1" y "Op2"), el segundo script generará una excepción.

Encontrará más información sobre el concepto de operaciones de línea de ensamblaje en el apartado titulado "Creating new components using Adapters" en la publicación *Referencia*.

Utilización de un acumulador: Como se ha indicado anteriormente, también puede pasar un objeto acumulador a una línea de ensamblaje con TCB. Un acumulador puede ser cualquiera de las siguientes clases o interfaces:

java.util.Collection

Todas las entradas work se clonan y se añaden a la colección; por ejemplo, ArrayList, Vector, etc.

com.ibm.di.connector.ConnectorInterface (Interfaz de conector)

El método putEntry() para esta interfaz de conector se llama con la entrada work al final de cada ciclo de la línea de ensamblaje.

com.ibm.di.parser.ParserInterface (Analizador)

Se llama al método writeEntry() para este analizador con la entrada work al final de cada ciclo de la línea de ensamblaje.

com.ibm.di.server.AssemblyLine Component (Conector de línea de ensamblaje)

Se llama al método add() para este conector de línea de ensamblaje con la entrada work al final de cada ciclo de la línea de ensamblaje.

Si el acumulador no es una de estas clases o interfaces, se devuelve una excepción.

Por ejemplo, para acumular todas las entradas work de una línea de ensamblaje en un archivo XML puede utilizar el siguiente script:

```
var parser = system.getParser ( "nombre_ejemplo.XML" ); // Obtener un analizador
// Configurar para escribir en archivo
parser.setOutputStream ( new java.io.FileOutputStream ( "d:/accum.xml" ));
parser.initParser(); // Inicializarlo.
tcb.setAccumulator ( parser ); // Establecer el analizador
// en tcb

var al = main.startAL ( "MyAssemblyLine", tcb ); // Iniciar línea de ensamblaje con tcb
al.join(); // Esperar a que la línea de ensamblaje finalice

parser.closeParser(); // Cerrar el analizador - esto vacía y
// cierra el archivo de salida.
```

Además, puede configurar un conector en lugar de programar manualmente el analizador como en el script siguiente:

```
var connector = system.getConnector("myFileSysConnWithXMLParser");
tcb.setAccumulator ( connector );

var al = main.startAL( "MyAssemblyLine", tcb);
al.join();

connector.terminate();
```

Normalmente, el usuario inicializa el TCB y a continuación el TCB es utilizado por la línea de ensamblaje. Si la línea de ensamblaje tiene una especificación de operaciones, el TCB vuelve a correlacionar los atributos de entrada con la entrada

work inicial, como espera la línea de ensamblaje, y procede del mismo modo para definir el objeto de resultados. Esto se lleva a cabo para que la interfaz de llamada externa a una línea de ensamblaje pueda permanecer igual incluso si cambian los nombres internos de la entrada work en la línea de ensamblaje. Una vez se pasa el TCB a una línea de ensamblaje, no debe esperar nada más del TCB. Utilice getResult() y getStats() de la línea de ensamblaje para recuperar el objeto de resultado y las estadísticas.

La correlación de resultados de TCB se efectúa antes del Epílogo y de ese modo puede acceder al resultado final antes de que lo obtenga el llamante de la línea de ensamblaje.

Inhabilitación de los componentes de la línea de ensamblaje: En IBM Security Directory Integrator, puede especificar que ciertos componentes de línea de ensamblaje no deben crearse ni inicializarse durante a inicialización de la línea de ensamblaje. Esto se lleva a cabo inhabilitando estos componentes utilizando el TCB.

Los componentes de la línea de ensamblaje están habilitados por omisión.

Para habilitar o inhabilitar un componente de una línea de ensamblaje, debe llamar al método `com.ibm.di.server.TaskCallBlock.setComponentEnabled(String name, boolean enabled)` en el objeto TCB de la línea de ensamblaje. El argumento `name` del método especifica el nombre del componente que debe habilitarse o inhabilitarse. El argumento `enabled` del método especifica si el componente debe habilitarse o inhabilitarse.

La habilitación o inhabilitación real de los componentes de la línea de ensamblaje se produce en el método `com.ibm.di.server.TaskCallBlock.applyAllSettings(AssemblyLineConfig alc)`. Este método se invoca al inicializar la línea de ensamblaje. A medida que progresa la inicialización de la línea de ensamblaje los componentes que se han marcado como inhabilitados no se crean ni se inicializan.

Si un componente de bucle (LOOP) se inhabilita, también se inhabilitarán todos los componentes que se encuentren en el mismo.

Aun cuando un componente se haya inhabilitado desde la GUI del Editor de configuración, puede habilitarse utilizando llamada de script:

```
com.ibm.di.server.TaskCallBlock.setComponentEnabled(String name, boolean enabled)
```

Suministro de una IWE (entrada work inicial)

El suministro de una entrada work inicial (IWE) es un método alternativo de pasar parámetros utilizando un TCB y se da soporte al mismo por motivos de compatibilidad con versiones anteriores.

Cuando se inicia una línea de ensamblaje con la llamada **system.startAL()** desde un script, se pueden pasar parámetros a la línea de ensamblaje definiendo valores de atributo o de propiedad en la entrada work inicial ,a la que se accede a través de la variable *work*. Por tanto, es tarea suya aplicar dichos valores para definir parámetros de conector; por ejemplo, en el complemento de software **Prólogo – Init** de la línea de ensamblaje utilizando la función `nombreConector.setParam()`.

Nota: Debe borrar la entrada work con la llamada `task.setWork(null)`, de lo contrario los iteradores de la línea de ensamblaje pasan a través del primer ciclo.

Puede examinar el resultado de la línea de ensamblaje, que es la entrada work cuando la línea de ensamblaje se detiene, utilizando la función getResult(). Consulte también el apartado sobre el conector proporcionado en tiempo de ejecución en la publicación *Referencia*.

El siguiente es un ejemplo de cómo pasar un valor de parámetro del conector con una IWE:

```
var entry = system.newEntry();
entry.setAttribute ("userNameForLookup", "John Doe");

// Aquí comienza la línea de ensamblaje
var al = main.startAL ( "EmailLookupAL", entry );

// esperar a que finalice la línea de ensamblaje
al.join();

var result = al.getResult();

// se supone que la línea de ensamblaje establece el atributo mail en su entrada de trabajo
task.logmsg ("Correo electrónico devuelto = " + result.getString("mail"));
```

Scripts en un conector

Correlación de entrada y Correlación de salida

La correlación de atributos personalizada se efectúa en estas pestañas. Cuando se selecciona el atributo, debe seleccionar el recuadro de selección **Correlación avanzada** y especificar el script en la ventana de edición. Recuerde que después de realizar todos los procesos necesarios, debe asignar el valor del resultado obtenido a ret.value, por ejemplo:

```
...
ret.value = myResultValue;
```

Como alternativa, en IBM Security Directory Integrator puede utilizar la palabra clave *return* seguida del valor simple que desee pasar como resultado:

```
return "mystring";
```

Complementos de software del conector

Los complementos de software proporcionan el medio de responder a determinados sucesos que se producen y alteran temporalmente las funciones básicas de un conector. Cuando crea complementos de software de script, tiene acceso a los objetos globales, aunque es posible que algunos objetos estándar no estén disponibles en todos los complementos de software. Para obtener información acerca de la disponibilidad temporal de los objetos, consulte "AssemblyLine and Connector mode flowcharts" en la publicación *Referencia*. También tiene control completo sobre el entorno, la línea de ensamblaje, el conector, las entradas y los atributos. Los complementos de software le proporcionan diferentes puntos de control para personalizar el flujo de proceso. Consulte el apartado "Flujo y complementos de software de una línea de ensamblaje" en la página 34.

Definición de los parámetros internos mediante scripts

Puede establecer los parámetros de conexión de un conector utilizando el script siguiente:

```
myConnector.setParam ( "filePath", "examples/scripting/sample.csv" );
```

Generalmente esta tarea se realiza en el prólogo, pero también puede resultar muy útil cuando se va a ejecutar la línea de ensamblaje, siempre que detenga y reinicialice el conector.

```
myConnector.terminate();  
myConnector.setParam ( "filePath", "examples/scripting/sample.csv" );  
myConnector.initialize(null);
```

Scripts en un analizador

Los scripts en un analizador realmente hacen referencia a que puede implementar su propio analizador mediante un script. En la sección "Script Parser" de la publicación *Referencia* se incluye una descripción del proceso.

Java + Script ≠ JavaScript

JavaScript no es Java. Podría parecer que es igual que Java, pero simplemente es lo suficientemente parecido como para ocasionar realmente la confusión. JavaScript inicialmente se denominaba *Live!Script* cuando Netscape lo creó por primera vez. Aunque hay un amplio soporte para JavaScript, aprenderá que existen *dialectos*; por ejemplo, la versión de Microsoft, denominada *JScript*. Existe una definición estándar, que se conoce como *ECMAScript* y puede encontrar su especificación en la siguiente dirección URL: <http://www.ecma-international.org/>

Aunque la sintaxis es similar, Java y JavaScript tratan los datos y los tipos de datos de modo distinto. Esto es uno de los principales motivos de confusión, y debido a ello, de errores al trabajar con JavaScript.

Representación de datos

Java admite lo que se llaman *primitivas*, que son valores simples como enteros con signo, valores decimales y bytes sencillos. Las primitivas no proporcionan ninguna funcionalidad, sólo un contenido de datos no complejos. Puede utilizarlas en cálculos y expresiones, asignarlas a variables y pasarlas como parámetros en llamadas a función. Java también proporciona una gran cantidad de objetos que no sólo transportan el contenido de los datos (incluso datos muy complejos), también proporcionan inteligencia en forma de funciones de objeto, también llamadas métodos.

Cuando se realiza la llamada de script `task.logmsg("Hello, World")`, se está llamando al método `logmsg()` del objeto de tarea.

Muchas primitivas Java tienen objetos correspondientes. Un ejemplo son los enteros, que se pueden utilizar en su forma primitiva (*int*) o mediante la manipulación de objetos `java.lang.Integer`.

JavaScript no utiliza el concepto de primitivas. En su lugar, todos los datos se representan como objetos JavaScript. Además, JavaScript tiene sólo un pequeño conjunto de objetos nativos si se compara con el abundante vocabulario de datos de Java. Por lo tanto, mientras Java distingue entre objetos numéricos no fraccionales y sus equivalentes decimales – incluso distingue entre tipos con signo y sin signo, además de ofrecer objetos similares para distintos niveles de precisión – JavaScript agrupa todos los valores numéricos en un solo tipo de objeto denominado *número*.

Como consecuencia, puede obtener resultados aparentemente erróneos cuando compara valores numéricos en JavaScript:

```

if (miVal == 3) {
    // Realizar alguna acción aquí si miVal es igual a 3
}

```

Si una operación aritmética había establecido `miVal` o si éste hace referencia a un objeto decimal Java, el valor del objeto podría ser 3,00001 o 2,99999. Aunque se aproxima mucho a 3, no pasará la prueba de equivalencia anterior. Para evitar este problema en particular, puede convertir el operando en un objeto entero Java para asegurarse de que el valor sea con signo y no fraccional. Entonces la expresión booleana se comportará como esperaba.

```

if (java.lang.Integer( miVal ) == 3) { ...

```

O puede asegurarse de que las variables hagan referencia a objetos Java adecuados con los que comenzar. En general, tendrá que estar al tanto de los tipos de objetos que está utilizando.

Llamadas a función ambiguas

Java también proporciona un tipo primitivo llamado *char*, que puede contener un único valor de carácter. Un conjunto de caracteres podría representarse en Java como una matriz de primitivas de caracteres o manejarse como un objeto `java.lang.String`. Como se ha mencionado antes, JavaScript no reconoce las primitivas. Los datos de tipo carácter se deben tratar utilizando el objeto *string* de JavaScript. Aunque especifique un solo carácter en JavaScript ("a"), éste se considera una *serie*.

Ahora considere que cuando llama a una función Java desde el script, la función se compara con el método real utilizando el nombre de la función así como el número y los tipos de parámetros utilizados en la llamada. Esta comparación se lleva a cabo mediante la ampliación de LiveConnect a JavaScript. LiveConnect hace lo posible para reconocer a qué signatura de función se refiere, lo cual no resulta una tarea sencilla dado que Java y JavaScript representan tipos de parámetros de distintos modos. Pero JavaScript y LiveConnect realizan algunas conversiones internas, que intentan comparar tipos de datos Java y JavaScript.

El problema surge cuando tiene varias versiones de un solo método, y cada una de ellos toma un conjunto distinto de parámetros; en concreto, si dos funciones tienen el mismo número de parámetros, pero de tipos distintos, y estos tipos no se diferencian en JavaScript. Observemos un script de ejemplo que realizará un cifrado MD5 de una serie.

```

// Crear el objeto MessageDigest a partir de la tabla hash MD5
var md = new java.security.MessageDigest.getInstance( "MD5" );

// Obtener un valor de atributo EID como una matriz de bytes.
var ab = java.lang.String( "mensaje para cifrar" ).getBytes();

md.update( ab );

var retHash = md.digest();

```

La llamada `update()` anterior producirá un error con una excepción de evaluación que indica que la llamada a la función es ambigua. Esto es porque el objeto `MessageDigest` tiene varias versiones de esta función con signaturas similares: una que acepta un solo byte y otra que espera una matriz de bytes (`byte[]`). Se puede evitar esto si se puede encontrar otra variante del mismo método que acepte un número distinto de parámetros, aunque le proporcione una signatura identificable de modo exclusivo. Afortunadamente, `MessageDigest` proporciona algo adecuado: una versión de `update()` que acepta una matriz de bytes más un par de valores

numéricos (los parámetros de desplazamiento (offset) y longitud (length)). Por lo tanto se puede cambiar el código para utilizar en su lugar esta llamada:

```
md.update( ab, 0, ab.length );
```

Finalmente, siempre puede especificar la signatura exacta del método Java que desea utilizar indicándolo dentro de corchetes después del objeto:

```
md["update(byte[])"](ab);
```

Aquí se llama a la versión de la función `update()` declarada con un solo parámetro de matriz de bytes.

Datos Char/String en Java frente a series JavaScript

Tanto Java como JavaScript proporcionan el objeto `String`. Aunque estos dos tipos de objetos `String` se comportan de modo similar y ofrecen varias funciones análogas, difieren de modo significativo. Por ejemplo, cada tipo de objeto proporciona un método distinto de devolver la longitud de la serie. Con series Java puede utilizar el método `length()`. Por otro lado, las series JavaScript disponen de una *variable* `length`.

```
var jStr_1 = new java.lang.String( "Hola a todos" ); // Serie Java
task.logmsg( "la longitud de jStr_1 es " + jStr_1.length() );
```

```
var jsStr_A = "Hola a todos"; // Serie JavaScript
task.logmsg( "la longitud de jsStr_A es " + jsStr_A.length );
```

Esta sutil diferencia puede llevar a errores de sintaxis desconcertantes. Si se intenta llamar a `jsStr_A.length()` se producirá un error de ejecución, dado que este objeto no tiene un método `length()`.

Las comparaciones de series pueden llevar incluso a errores más desconcertantes.

```
var jsStr_A = "Hola a todos"; // Serie JavaScript
var jsStr_B = "Hola a todos"; // Serie JavaScript
```

```
if ( jsStr_A == jsStr_B )
    task.logmsg( "TRUE" );
else
    task.logmsg( "FALSE" );
```

Como se esperaba, obtendrá un resultado "TRUE" del fragmento de código anterior. No obstante, todo funciona de modo distinto con series Java.

```
var jStr_1 = java.lang.String( "Hola a todos" ); // Serie Java
var jStr_2 = java.lang.String( "Hola a todos" ); // Serie Java
```

```
if ( jStr_1 == jStr_2 )
    task.logmsg( "TRUE" );
else
    task.logmsg( "FALSE" );
```

Esto producirá el resultado "FALSE", dado que el operador de equivalencia anterior comparará para comprobar si las dos variables hacen referencia al *mismo objeto* en memoria, en lugar de comparar los valores. Para comparar los valores de serie Java, debe utilizar el método de serie adecuado:

```
if ( jStr_1.equals( jStr_2 ) ) ...
```

Pero aún hay más. Este fragmento de código siguiente le proporcionará un resultado "TRUE":

```

var jsStr_A = "Hola a todos"; // Serie JavaScript
var jStr_1 = java.lang.String( "Hola a todos" ); // Serie Java

if ( jsStr_A == jStr_1 )
    task.logmsg( "TRUE" );
else
    task.logmsg( "FALSE" );

```

Dado que JavaScript no puede operar sobre un tipo desconocido como un objeto de serie Java, primero convierte jStr_1 a una serie de JavaScript equivalente para realizar la evaluación.

En resumen, tenga en cuenta los tipos de objetos con los que trabaja. Y recuerde que las funciones de IBM Security Directory Integrator siempre devuelven objetos Java. Si se tienen en cuenta estos factores contribuirán a minimizar errores en el código de script.

Ámbito y nombre de las variables

JavaScript es un lenguaje relativamente informal y no requiere que defina las variables antes de que se asignen valores a éstas. Tampoco impone la comprobación estricta de tipos ni señala cuando se redefine una variable. Esto hace que sea sencillo y rápido trabajar con JavaScript, pero puede llevar fácilmente a código ilegible y a errores desconcertantes, especialmente dado que puede crear variables que sobrescriban las incorporadas.

Un error que puede ser un desafío para la depuración es declarar una variable con el mismo nombre que una incorporada, como work, conn y current, por lo que será necesario que se familiarice con los nombres reservados utilizados por IBM Security Directory Integrator.

Otro problema común se produce cuando crea variables nuevas que redefinen las existentes, que quizá se utilicen en bibliotecas de configuraciones o de scripts incluidas. Se pueden evitar errores si tiene conocimientos sobre la denominación de las variables y del *ámbito*. El ámbito define la esfera de influencia de las variables y en IBM Security Directory Integrator se distingue entre variables globales, que están disponibles en todos los complementos de software, los componentes de script y las correlaciones de atributos, y las que son locales para una función.

Para conocer mejor el ámbito, debe comprender primero que cada línea de ensamblaje tiene su propio motor de scripts y por lo tanto se ejecuta en su propio contexto de script. Cualquier variable que no se haya definido específicamente como local dentro de una declaración de función es global para ese motor de scripts. Por lo tanto, el código siguiente creará una variable global:

```
miVar = "Conócete a ti mismo";
```

Esta variable estará disponible desde este punto hasta que dure la línea de ensamblaje. Para hacer que esta variable sea local se requieren dos pasos: utilizar la palabra clave var cuando se declara la variable e incluir la declaración dentro de una función:

```
function miFunc() {
    var miVar = "Conócete a ti mismo";
}
```

Ahora miVar como se ha definido antes dejará de existir después del símbolo de cerrar paréntesis. Recuerde que no es suficiente con incluir la variable dentro de una función; también tiene que utilizar var para indicar que está declarando una variable local nueva.

```

var glbVar = "Esta variable tiene ámbito global";
glbVar2 = "Otra variable global";

function miFunc() {
    var lc1Var = "De ámbito local dentro de este bloque";
    glbVar3 = "Esta es global, dado que no se ha utilizado "var";
};

```

Una variable local, siempre que la declare dentro de una función, puede llamarla como desee. En cuanto sale del ámbito se restauran los tipos y valores anteriores

Aunque no es necesaria la palabra clave `var` para definir variables globales, se recomienda utilizarla. También se recomienda definir las variables al principio del script e incluir los comentarios necesarios para proporcionar al lector una idea del uso que se las va a dar. Este procedimiento no sólo mejora la legibilidad del código, también fuerza a que se tomen decisiones teniendo en cuenta la denominación y el ámbito de las variables.⁹

Creación de instancias de una clase de Java

Puede invocar clases Java externas; es decir, las que ha añadido al entorno de ejecución de IBM Security Directory Integrator, o la CLASSPATH, utilizando código de script.

Por ejemplo, si suponemos que desea utilizar la clase estándar `java.io.FileReader`, utilice el script siguiente:

```
var javafile = new java.io.FileReader ( "myfile" );
```

Ahora tiene un objeto llamado *javafile*; si lo utiliza puede invocar todos los métodos del objeto.

La misma técnica se utiliza para crear una instancia de sus propios objetos:

```
var myfile = new my.FileReader("myfile");
```

Utilización de valores binarios en scripts

Los valores binarios se pueden recuperar de los atributos mediante la función `getObject()` de la entrada. El valor del atributo binario propiamente dicho se devuelve como matriz de bytes. El siguiente es un ejemplo de JavaScript:

```

var x = conn.getObject("objectGUID");
for ( i = 0; i < x.length; i++ )
{
    task.logmsg ("GUID[" + i + "]: " + x[i]);
}

```

En este ejemplo se escriben varios números, entre -128 y 127, en el archivo de registro cronológico. Es posible que desee hacer algo más con sus datos. Si ha leído una contraseña de un conector, que la ha almacenado como una matriz de bytes, puede convertirla en una serie con este código:

```
password = system.arrayToString(conn.getObject("userpassword"));
```

9. La denominación de funciones funciona un poco distinto. Los lenguajes de programación como Java identifican una función por la combinación de su nombre más el número y tipo de parámetros. JavaScript sólo utiliza el nombre. De modo que si tiene varias definiciones de la misma función, JavaScript sólo "recordará" la última — independientemente de que esta definición tenga o no un número distinto de parámetros.

Utilización de valores de fecha en scripts

Cuando se trabaja con fechas en IBM Security Directory Integrator, esto implica utilizar instancias de `java.util.Date`. Con los lenguajes de script disponibles, puede implementar su propio mecanismo para manejar fechas; sin embargo no es una práctica habitual.

El motor de scripts de IBM Security Directory Integrator proporciona un mecanismo para analizar fechas. El objeto `system` tiene un método `parseDate(date, format)` al que se puede acceder en cualquier momento.

Nota: Cuando obtiene una instancia de `java.util.Date`, puede utilizar las bibliotecas y clases Java estándar para ampliar el proceso.

Este es un sencillo ejemplo de JavaScript que maneja fechas. Este código se puede colocar e iniciar desde cualquier punto de control del script:

```
var string_date1 = "07.09.1978";
var date1 = system.parseDate(string_date1, "dd.MM.yyyy");

var string_date2 = "1977.02.01";
var date2 = system.parseDate(string_date2, "yyyy.dd.MM");

task.logmsg(date1 + " es anterior a " + date2 + ": " +
    date1.before(date2));
```

En primer lugar, el código de script analiza dos valores de fecha (en formatos diferentes) en `java.util.Date`. A continuación, utiliza el método `java.util.Date.before()` estándar para determinar si la primera instancia aparece antes que la segunda. Después, la salida de este script se imprime en el archivo de registro.

Utilización de valores de coma flotante en scripts

Los ejemplos siguientes muestran cómo se pueden utilizar los valores de coma flotante en el código de script que cree. Todos estos ejemplos se han implementado en JavaScript. Aunque algunos ejemplos pueden repetirse utilizando otros lenguajes de scripts, es posible que la sintaxis sea diferente. El siguiente es un script sencillo que asigna valores de coma flotante a dos variables para saber su promedio. Este código se puede iniciar desde cualquier punto de control del script. La salida del archivo de registro es " r = 3.85 ".

```
var a = 5.5;
var b = 2.2;
var r = (a + b) / 2;
task.logmsg("r = " + r);
```

El ejemplo siguiente amplía este sencillo script. Tenga en cuenta que en el conector de entrada hay un atributo "Marks" con múltiples valores que contiene valores de serie (`java.lang.String`) que representan valores de coma flotante; una situación común. Este atributo se correlaciona con un atributo del conector de salida denominado "AverageMark", que contiene el valor promedio de todos los valores del atributo "Marks". El siguiente código se utiliza en la correlación avanzada del atributo "AverageMark":

```
// Devolver en primer lugar los valores del atributo "Marks"
var values = work.getAttribute("Marks").getValues();

// Establecer en cero las variables count y sum
var sum = 0;
var count = 0;
```

```
// Realizar un bucle por los valores, contándolos y sumándolos
for (i=0; i<values.length; i++)
{
    // utilizar la función Double() para convertir el valor en número
    sum = sum + new Number(java.lang.Double(values[i]));
    count++;
}

// Si count > 0, calcular el promedio
var average = (count > 0) ? (sum / count) : 0;

// Devolver el promedio calculado
ret.value = average;
```

La llamada central de este ejemplo es `java.lang.Double(values[i])` que se utiliza para convertir el valor de "Marks" indexado actualmente en un valor numérico que, a continuación, se puede utilizar en el cálculo del promedio.

Capítulo 3. Editor de configuración

El Editor de configuración de Eclipse de IBM Security Directory Integrator es la herramienta principal para desarrollar soluciones de IBM Security Directory Integrator. Le permite crear, mantener, probar y depurar archivos de configuración; se crea en la plataforma Eclipse para proporcionar un entorno de desarrollo que sea extenso y ampliable.

Para comprender los conceptos que se presentan aquí con mayor facilidad, debe familiarizarse con los conceptos comunes de Eclipse como, por ejemplo, editores, vistas y otros puntos de ampliación comunes de Eclipse.

Modelo de proyecto

Con la llegada del Editor de configuración (CE) basado en Eclipse, el desarrollo de soluciones en IBM Security Directory Integrator (IBM Security Directory Integrator) no se basa, como en las versiones anteriores a la versión 7.0, en el desarrollo de archivos de configuración simples, sino en un modelo de proyecto y en *espacios de trabajo*. Al utilizar el modelo de proyecto y el espacio de trabajo, puede llevar a cabo su trabajo de desarrollo; cuando ya esté preparado para desarrollar la solución, debe extraer un archivo de configuración del espacio de trabajo y enviarlo a un servidor de IBM Security Directory Integrator adecuado: el entorno de ejecución. Esto genera algunas ventajas, entre las cuales están las siguientes:

- Archivos de configuración más limpios; los elementos de configuración innecesarios y no utilizados no se exportarán desde el espacio de trabajo;
- Edición más eficaz de las configuraciones de IBM Security Directory Integrator;
- Mayor reutilización de componentes comunes;
- Posibilidad de utilizar sistemas de gestión de código fuente, tal como CVS.

Espacio de trabajo

Cuando inicia el Editor de configuración, se le solicita que seleccione un directorio de espacio de trabajo. El directorio de espacio de trabajo es el lugar donde el Editor de configuración almacena todos los proyectos y archivos de IBM Security Directory Integrator. Este directorio es distinto del directorio de soluciones, aunque el directorio de espacio de trabajo puede estar contenido en el directorio de soluciones. El Editor de configuración recoge archivos del espacio de trabajo para crear un archivo de configuración en tiempo de ejecución (*rs.xml*) que se puede ejecutar en un servidor de IBM Security Directory Integrator. Los archivos y proyectos del espacio de trabajo se pueden imaginar como el origen de varios archivos de configuración en tiempo de ejecución.

Directorio de instalación, directorio de soluciones y directorio de trabajo

El hecho de que existan tres directorios diferentes puede producir confusión. Como se ha descrito en la sección anterior, el directorio de espacio de trabajo pertenece al CE y es donde se almacenan los archivos de proyectos. Los directorios de instalación y de soluciones pertenecen al servidor de IBM Security Directory Integrator cuando éste se ejecuta. Pero cuando configura un conector en el Editor de configuración, por ejemplo, a menudo utilizará la función descubrir atributos; esto hará que el Editor de configuración abra el conector y ejecute operaciones en

el conector. Debido a que algunos conectores utilizan nombres de archivo para varios fines, puede ser confuso cuando se utilizan vías de acceso relativas. Por este motivo, el directorio de trabajo predeterminado para el CE siempre debe ser el mismo que el directorio de soluciones con el que trabaja principalmente¹⁰. Básicamente, porque se pueden crear varios servidores y directorios de soluciones en el CE. Puede trabajar con éstos y ejecutar líneas de ensamblaje en dichos servidores, pero las vías de acceso relativas de las configuraciones ya no son las mismas que cuando se utilizan conectores en el CE, a diferencia de cómo se utilizan cuando se ejecuta la línea de ensamblaje.

Considere una configuración con un conector de archivo que utilice abc.txt como archivo de entrada. Cuando utiliza Descubrir atributos en el CE, éste buscará un archivo en el directorio de trabajo del CE (el directorio de soluciones preferido especificado en el tiempo de instalación). Cuando ejecuta la línea de ensamblaje con este conector todo parece ir bien. A continuación, crea un nuevo servidor con un directorio de soluciones que no apunta al directorio de trabajo actual. Configura el conector y el conector resuelve el nombre de archivo en el lugar correcto (es decir, el archivo abc reside en el directorio de trabajo actual). Sin embargo, cuando ejecuta la línea de ensamblaje, ésta falla, porque no encuentra el archivo. Esto se produce porque el nuevo servidor se ejecuta en un directorio de trabajo diferente (el directorio de soluciones) que el CE.

De forma predeterminada, el directorio de trabajo del CE se establece en el directorio de soluciones del servidor predeterminado que se utiliza para ejecutar líneas de ensamblaje. Por lo tanto, si no cambia nada no debería encontrar ninguno de estos problemas salvo cuando crea directorios de soluciones nuevos o cambia el directorio de soluciones del servidor predeterminado.

Entorno de trabajo

El entorno de trabajo es la aplicación principal de la interfaz de usuario donde realiza toda la configuración y el desarrollo de soluciones de IBM Security Directory Integrator. El entorno de trabajo consta de varias vistas y editores que le permiten llevar a cabo esta tarea.

Vista Servidores de IBM Security Directory Integrator

No se inicia una instancia de servidor IBM Security Directory Integrator cada vez que ejecuta una línea de ensamblaje, a diferencia de lo que sucedía en versiones anteriores a la versión 7.0. En lugar de ello, se inicia automáticamente un servidor IBM Security Directory Integrator cuando inicia el Editor de configuración, que se utiliza para probar y ejecutar las líneas de ensamblaje que desarrolla. A diferencia de las versiones anteriores, cuando finaliza el objeto de las ejecuciones de prueba, este servidor no termina, sino que permanece activo, a la espera de la siguiente ejecución de prueba.

En la vista Servidores puede gestionar las definiciones de servidores. Los servidores que se definen aquí pueden ser servidores locales o remotos y son utilizados por los distintos proyectos IBM Security Directory Integrator creados por el usuario. Los servidores que tienen una vía de instalación definida se consideran servidores “locales” que pueden ser iniciados por el Editor de configuración.

10. El directorio de trabajo no debe confundirse con el espacio de trabajo. El espacio de trabajo es el área donde se almacenan los archivos de proyectos; el directorio de trabajo es la posición en el sistema de archivos que define todos los nombres de archivo que no están completamente calificados. Por motivos de portabilidad, debe ser el directorio de soluciones.

Se define automáticamente un servidor, que se denomina *Default*. Este servidor es el servidor predeterminado que utilizan los nuevos proyectos de IBM Security Directory Integrator. El servidor predeterminado se crea con valores del archivo de propiedades globales utilizando el directorio de soluciones definido por el usuario (por ejemplo, de la variable SDI_SOLDIR, a su vez configurada por el instalador).

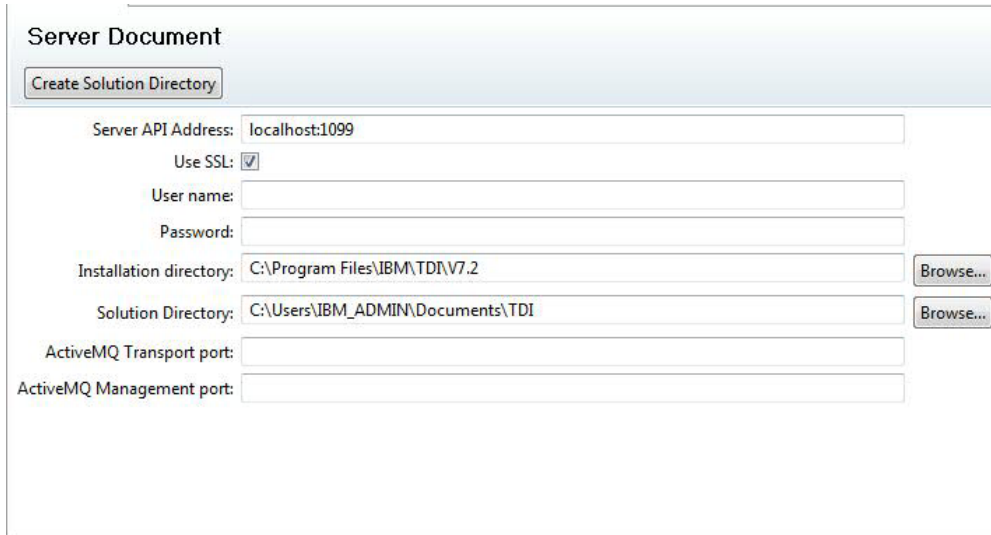


Figura 5. Definición del servidor de predeterminado de IBM Security Directory Integrator

El proyecto de IBM Security Directory Integrator

Para desarrollar una solución de IBM Security Directory Integrator, debe primero crear un proyecto de IBM Security Directory Integrator. En Eclipse, un proyecto es una colección de archivos y recursos relacionados.

Cuando se crea el proyecto, se llena con varias carpetas donde residen objetos de configuración comunes. A continuación se muestra el diseño de un proyecto nuevo de IBM Security Directory Integrator.

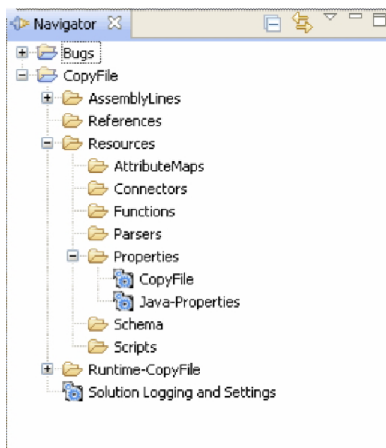


Figura 6. El árbol de proyecto de IBM Security Directory Integrator

La carpeta Líneas de ensamblaje contiene las líneas de ensamblaje del proyecto; mientras que la carpeta Recursos contiene todos los componentes que se

comparten o que son utilizados por la línea de ensamblaje, o que hacen referencia a la solución en general: propiedades, parámetros de registro cronológico, etc. Para crear nuevos recursos utilice el asistente **Archivo/Nuevo...** del menú principal o utilice los menús contextuales de cada carpeta (es decir, pulse con el botón derecho en la carpeta Líneas de ensamblaje para crear una nueva línea de ensamblaje).

El directorio Runtime (*Runtime-NombreProyecto*) es un directorio especial que contiene el archivo de configuración en tiempo de ejecución además de los archivos de propiedades personalizadas. Cada vez que se modifica un componente del proyecto (conector, línea de ensamblaje, archivo de propiedades, etc.), los archivos relacionados de este directorio también se actualizan. Todos los archivos generados se marcan como archivos derivados, lo cual significa que recibirá un aviso si intenta modificar el contenido. Tenga en cuenta que si cambia alguno de estos archivos, se sobrescribirán. La finalidad del directorio de ejecución es crear un directorio de archivos de ejecución para el proyecto que se pueden copiar, o reservar si utiliza control de origen, y puedan ser utilizados por un servidor de IBM Security Directory Integrator.

Archivos de configuración

El archivo de configuración (Config) ejecutado por un servidor es un documento XML compuesto en el cual las líneas de ensamblaje, los conectores, etc., forman parte del mismo documento. En el CE de Eclipse de IBM Security Directory Integrator, se asigna a cada uno de estos componentes su propio archivo físico. Estos archivos sólo contienen un objeto de configuración.

Uno de los motivos para dividir el archivo de configuración en archivos separados durante el desarrollo de soluciones era facilitar el compartimiento de los componentes. Además, el hecho de tener cada componente en su propio archivo sirve para los sistemas de control de origen como CVS y para el desarrollo multiusuario en que varias personas trabajan en distintas partes de la solución al mismo tiempo.

Las configuraciones anteriores a esta versión se pueden importar utilizando un asistente de importación. La configuración importada se divide en archivos de configuración individuales como resultado de este proceso. Otro método consiste en utilizar la opción **Archivo > Abrir archivo de configuración de Security Directory Integrator**, que importará la configuración anterior a la versión 7.0 a un nuevo proyecto. Sin embargo, tenga en cuenta que esta opción también vuelve a establecer la actualización automática en el archivo de origen. Consulte el apartado siguiente para obtener más información sobre la función de archivo enlazado.

Archivo de configuración en tiempo de ejecución

El archivo de configuración en tiempo de ejecución está oculto en la vista normal. Este archivo es el utilizado por los servidores de IBM Security Directory Integrator para ejecutar la solución y también es el archivo con el que trabajan directamente las versiones anteriores a la versión 7.0. Cada vez que guarda un archivo de configuración en el Editor de configuración, esto también hace que se actualice el archivo de configuración en tiempo de ejecución. Este archivo es también el que se transfiere a un servidor de IBM Security Directory Integrator para ejecutarlo. Este archivo es mantenido por el creador de proyectos y no debe ser modificado por el usuario final.

Puede configurar el proyecto para exportar el archivo de configuración en tiempo de ejecución automáticamente cuando cambie. Utilice el panel de propiedades del

proyecto para configurar el archivo con el que está enlazado el proyecto.

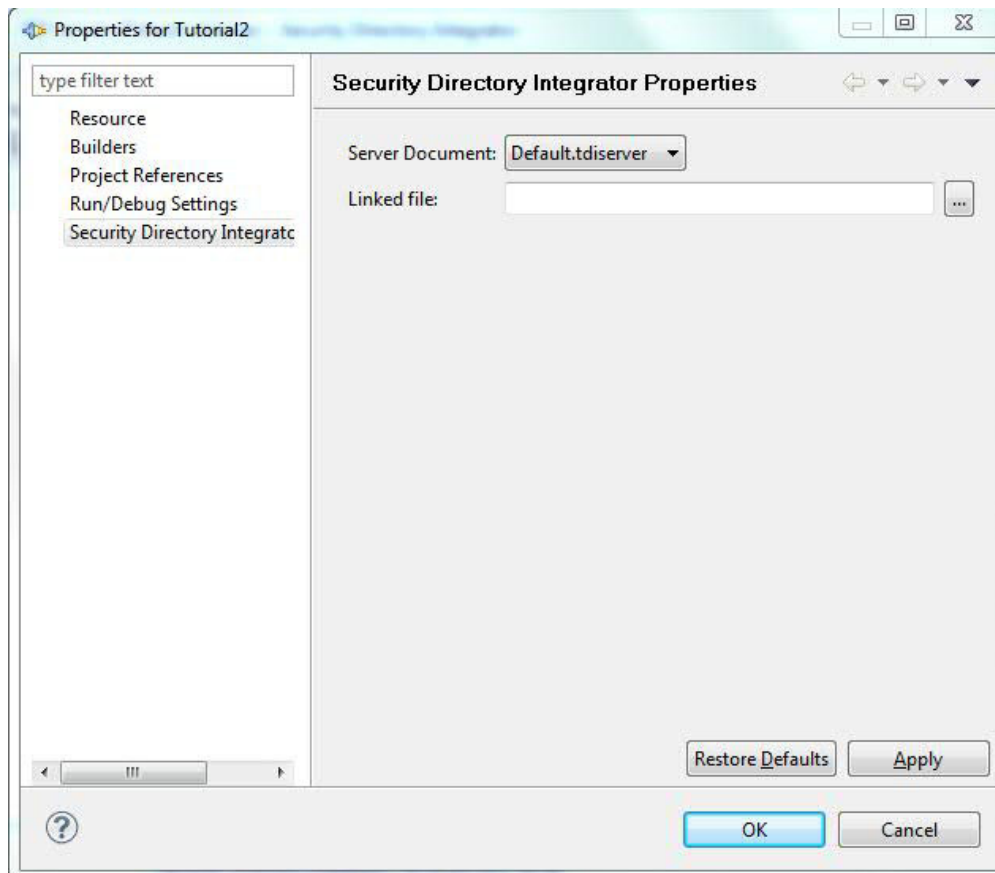


Figura 7. La ventana Propiedades de proyecto de IBM Security Directory Integrator

Cada vez que se actualiza la configuración en tiempo de ejecución, también se copia en el archivo especificado en la sección de propiedades del proyecto de IBM Security Directory Integrator. El archivo enlazado se establece automáticamente cuando se utiliza el mandato **Archivo > Abrir configuración de Directory Integrator**.

Creador de proyectos

Un creador de proyectos personalizado está asociado al proyecto y su objetivo es reunir todos los artefactos en un *archivo de configuración ejecutable*.

Este creador se puede ejecutar automáticamente cada vez que un recurso cambia, o manualmente mediante el elemento de menú estándar **Proyecto > Crear**. En ambos casos, el creador de proyectos de IBM Security Directory Integrator mantiene un archivo de configuración que se actualiza cuando se invoca. Cuando se han realizado cambios en un recurso (modificaciones, adiciones o eliminaciones), el creador actualiza el archivo de configuración ejecutable con esa actualización de recurso. Sólo se actuará en los archivos de configuración reconocidos, es decir, líneas de ensamblaje, pero no archivos .gif, por ejemplo. El archivo de configuración ejecutable normalmente está oculto ya que el nombre de archivo empieza con un punto (.). El archivo se denomina *.rs.xml* y está situado en la carpeta de proyectos. Este es el archivo de configuración compilado que se envía a un servidor de IBM Security Directory Integrator para ejecutarlo.

El creador reubicará y renombrará los componentes en la configuración de destino. Si la carpeta Recursos contiene tanto propiedades como conectores, se reubicarán en sus carpetas estándar en la configuración de destino (es decir, la carpeta Propiedades y Conectores).

El creador de proyectos también comprueba todos los componentes modificados para ver si existen errores obvios y problemas potenciales. Estos problemas se registran en la vista **Problemas** estándar de Eclipse. Cada elemento del problema contiene una descripción del problema además de la ubicación para que pueda efectuar una doble pulsación y activar el editor donde se ha identificado el problema.

Propiedades y sustitución

Cada proyecto de IBM Security Directory Integrator está asociado a un servidor de IBM Security Directory Integrator. El servidor asociado es el servidor que se utilizará para ejecutar las líneas de ensamblaje del proyecto. Debido a que el servidor puede estar situado en una máquina diferente, el modelo de proyecto debe proporcionar acceso a las propiedades mediante copias locales de almacenes de propiedades seleccionados.

Los almacenes de propiedades se pueden descargar desde el servidor cuando sea necesario cuando la copia local contiene dos valores para cada propiedad. Un valor es el que se ha descargado desde el servidor (valor remoto) y el otro es el valor establecido por el usuario (valor local). De este modo se pueden ver qué propiedades están en posible conflicto, además de poderse extraer el conjunto de propiedades que la solución utiliza. No existe ningún requisito para descargar un almacén de propiedades antes de añadir propiedades a él; sin embargo, cuando se ejecuta la solución, las propiedades con un valor local se comprobarán con el valor del servidor para evitar sobrescrituras accidentales de propiedades existentes.

La edición de archivos de propiedades se puede realizar abriendo (o creando) los archivos de propiedades en la carpeta Recursos. Una vez ha creado un archivo de propiedades, puede modificar su contenido y realizar carga y descarga. La carga y descarga se lleva a cabo para sincronizar las propiedades localmente con los del servidor.

Nota: IBM Security Directory Integrator utiliza actualmente el signo de igualdad "=" o dos puntos ":" como separador en los archivos de propiedades de pares clave/valor, cualquiera que sea el que aparezca en primer lugar. Por lo tanto, no está permitido el uso del signo de igualdad o de dos puntos en los nombres y valores de propiedades. El separador de clave/valor utilizado en los archivos de propiedades de IBM Security Directory Integrator V6.0 y versiones anteriores era únicamente el carácter ":". Por lo tanto, puede ser necesario editar los archivos de propiedades migrados desde V6.0 y versiones anteriores.

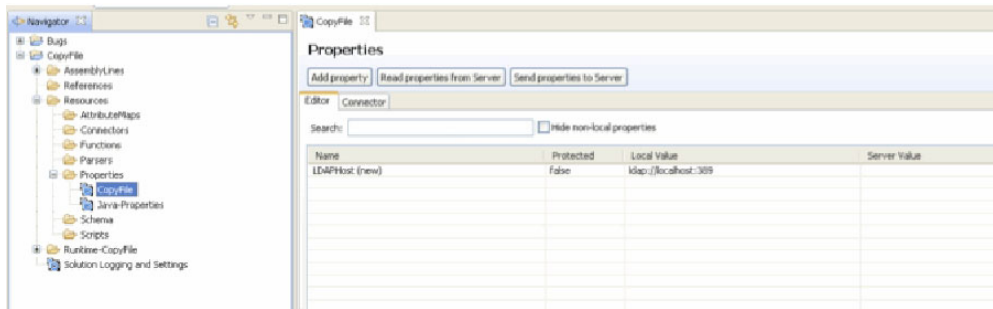


Figura 8. Vista Propiedades

Tenga en cuenta que los almacenes de propiedades personalizados con una vía de acceso relativa (por ejemplo, la configuración de conector utiliza una vía de acceso relativa) tienen un archivo correspondiente que se genera durante el tiempo de ejecución, en el directorio *Proyecto*. Cuando se crean nuevas propiedades personalizadas la vía de acceso predeterminada se establecerá en "{config.\$directory}/nombre_archivo.properties" donde *nombre_archivo* es el nombre que se asigna al nuevo almacén de propiedades. "{config.\$directory}" se resuelve en la ubicación del archivo de tiempo de ejecución.

De forma predeterminada, los almacenes de propiedades compartidas no se añaden a un proyecto (por ejemplo el almacén global, de la solución y del sistema). Todavía puede añadirlas al proyecto si desea mantener los cambios realizados en dichos archivos. Para ver los almacenes de propiedades compartidas, debe utilizar la vista Servidores o utilizar **Examinar almacenes del sistema** en la barra de herramientas principal.

Modelo de interfaz de usuario

La interfaz de usuario (UI) la proporcionan un conjunto de vistas, editores y otros recursos relacionados con la UI. Se proporcionan a través de mecanismos de punto de ampliación definidos según la plataforma Eclipse.

Aunque hay numerosas contribuciones de punto de ampliación en el CE, aquí sólo se listan los más importantes.

Tabla 7. Contribuciones del punto de ampliación del CE de Eclipse CE

Contribución	Descripción
Editores	Se proporcionan editores para todos los archivos de configuración más importantes: <ul style="list-style-type: none"> • Línea de ensamblaje (archivos .assemblyline) • Conector (archivos .connector) • Función (archivos .function) • Scripts (archivos .script) • Propiedades (archivos .tdiproperties) • Correlación de atributos (archivos .attributemap)
Vistas	Se proporcionan varias vistas para ayudar a visualizar distintos aspectos de los archivos de configuración.

Tabla 7. Contribuciones del punto de ampliación del CE de Eclipse CE (continuación)

Contribución	Descripción
Menús, barras de herramientas	Todas las acciones del CE que operan en objetos de configuración se definen como acciones estándar.
Asistentes	Se proporcionan varios asistentes para ayudar a crear nuevos proyectos y archivos de configuración.

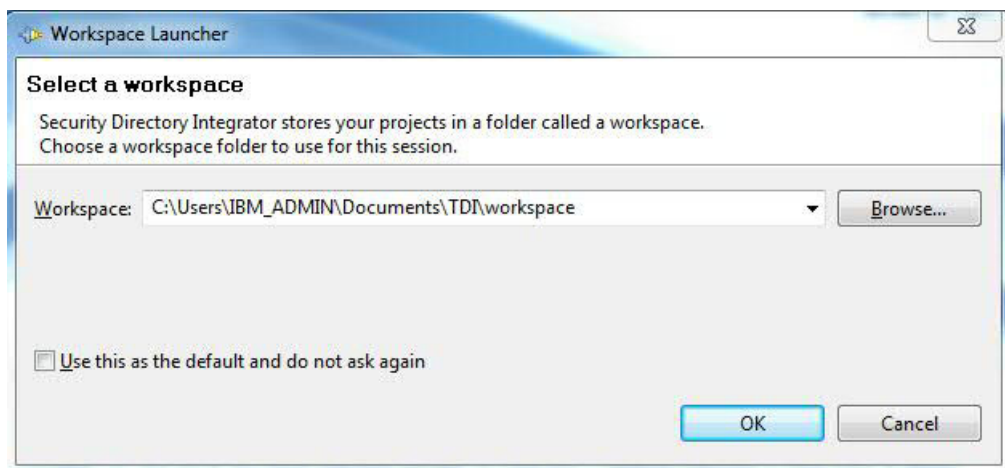
El CE sigue el paradigma MVC (modelo, vista, controlador). El editor para un archivo de configuración crea los indicadores que muestran el contenido del archivo de configuración. Los indicadores registran las barras de herramientas y los menús que utilizan con la infraestructura de Eclipse de modo que se puedan realizar contribuciones en ellos. Todas las acciones que afectan al archivo de configuración se implementan y se aportan utilizando los mecanismos estándar de Eclipse.

Interfaz de usuario

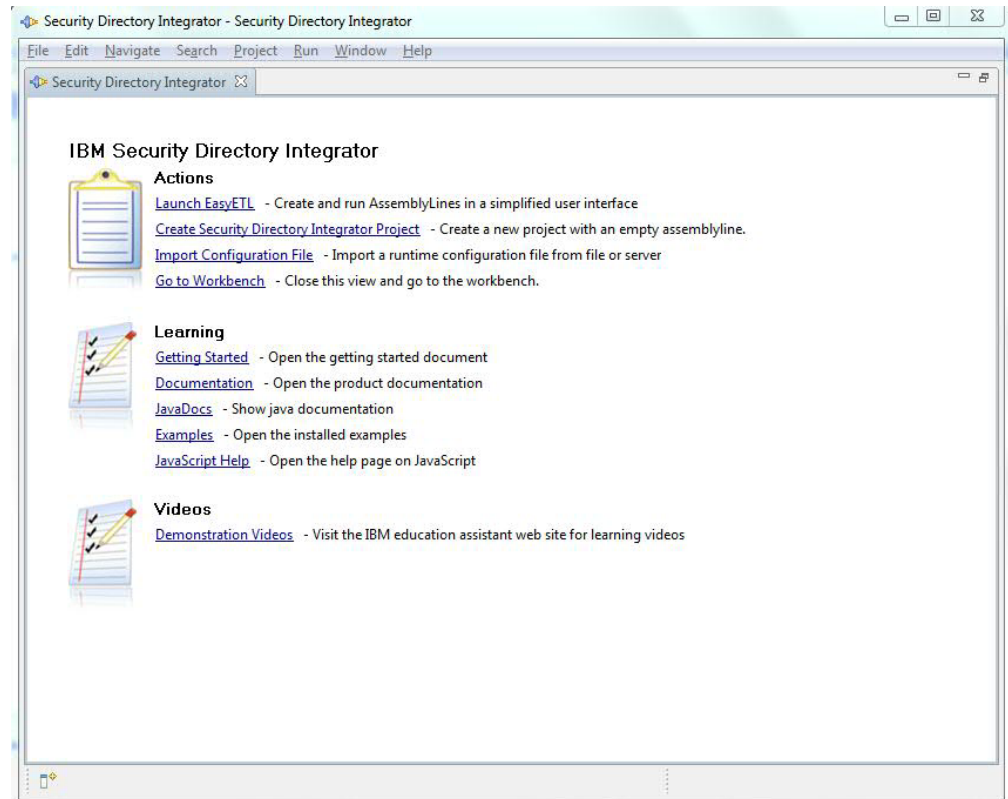
Ventana Aplicación

La primera vez que inicie el Editor de configuración (CE), le solicitará un directorio de espacio de trabajo. El directorio de espacio de trabajo es la ubicación del sistema de archivos donde se almacenan los proyectos. Puede cambiar este directorio más adelante en el menú **Archivo**.

Este diálogo aparecerá cada vez que inicie el CE a menos que seleccione el recuadro de selección para hacer que el directorio de espacio de trabajo especificado sea la ubicación predeterminada.

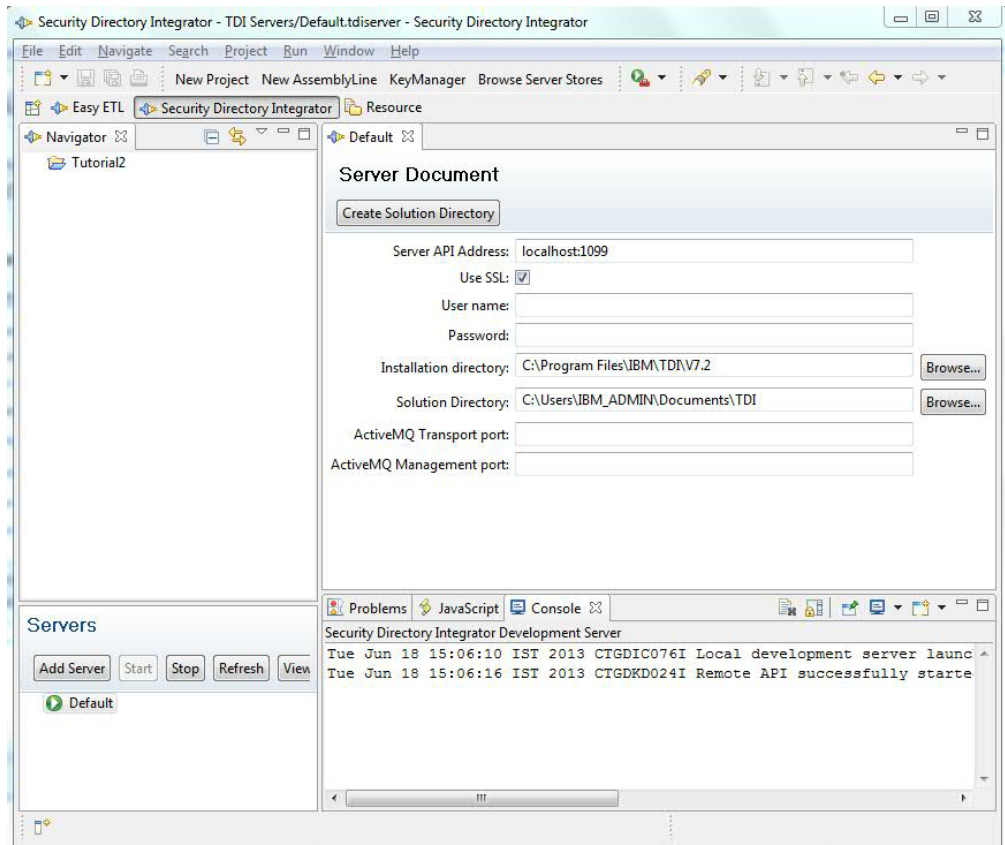


Después de este diálogo aparecerá la ventana de espacio de trabajo principal. Si es la primera vez que inicia el Editor de configuración, éste mostrará la pantalla de bienvenida:



Esta pantalla de bienvenida proporciona varios enlaces rápidos a tareas comunes y sitios de información. El enlace de documentación le remite a la documentación del producto configurado (propiedad del sistema *com.ibm.tdi.helpLoc*).

La pantalla de bienvenida se puede abrir de nuevo más tarde seleccionando **Ayuda** > **Bienvenida**. Cuando se cierra la pantalla de bienvenida, verá la ventana de espacio de trabajo:



Es la ventana principal en la que se gestionan los proyectos y configuraciones.

En esta imagen hay un editor abierto (para el documento Default.server) y varias vistas. Las vistas más importantes son las que aparecen en esta imagen.

El *Navegador* (parte superior izquierda) contiene todos los proyectos y archivos de origen para las configuraciones de servidor y soluciones de IBM Security Directory Integrator. El navegador también puede contener otros archivos y proyectos como, por ejemplo, archivos de texto, etc. El Editor de configuración trata los proyectos de IBM Security Directory Integrator de forma específica, por lo que el Editor no afecta a los demás archivos y proyectos. Esto se describe en la sección sobre el creador de proyectos.

La vista *Servidores* (parte inferior izquierda) muestra el estado de cada uno de los servidores que están definidos en el proyecto "Servidores de IBM Security Directory Integrator". Puede tener definidos tantos servidores como desee. La vista proporciona varias funciones para operar en los servidores y sus configuraciones. El botón renovar renueva el estado de todos los servidores de la vista.

El *área del editor* (parte superior derecha) es donde se muestran todos los editores. Cuando abre un documento, tal como una configuración de línea de ensamblaje, termina en esta área. Esta área está dividida verticalmente, con un área (parte inferior derecha) que contiene varias vistas con información pertinente. Las vistas más importantes son la vista Problemas, que muestra los posibles problemas de un componente de IBM Security Directory Integrator, el Registro de errores, que muestra los errores que se han producido al desarrollar soluciones, y finalmente la

vista Consola, que muestra el archivo de registro de consola para servidores de IBM Security Directory Integrator en ejecución (por ejemplo, los servidores iniciados por el Editor de configuración).

Vista Servidores

La vista Servidores contiene varias funciones útiles para gestionar instancias de servidor. Además de añadir y eliminar instancias de servidor, hay muchos mandatos asociados con los servidores y sus instancias de configuración activas y líneas de ensamblaje.

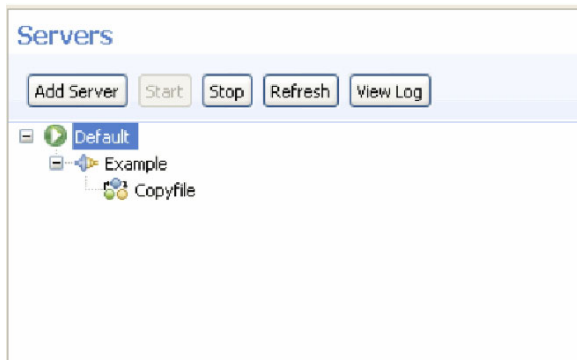


Figura 9. Vista Servidores del Editor de configuración

La barra de herramientas principal muestra los botones siguientes:

Añadir servidor

Utilice este botón para añadir otro servidor.

Iniciar Utilice este botón para iniciar un servidor "local" (por ejemplo, uno al que se pueda acceder desde el sistema de archivos).

Si tiene seleccionada una instancia de configuración, puede iniciar una o más de sus líneas de ensamblaje.

Detener

Utilice este botón para detener una solicitud al servidor, una instancia de configuración o una línea de ensamblaje.

Renovar

Utilice este botón para renovar el contenido de la vista Servidores.

Ver registro

Utilice este botón para ver el archivo de registro estándar, archivo "ibmdi.log" en un servidor "local".

El menú emergente para cada elemento de la vista muestra mandatos adicionales que se pueden ejecutar en función de la selección.

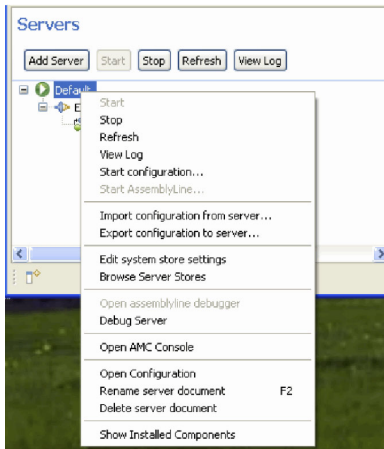


Figura 10. Vista de servidores; menú emergente

Los mandatos posibles son:

Iniciar Iniciar el servidor.

Detener

Detiene el servidor, la instancia de configuración o la línea de ensamblaje.

Renovar

Renueva la vista de servidores.

Ver registro

Abre el archivo ibmdi.log en un editor de texto.

Iniciar configuración...

Cuando se selecciona un servidor, puede iniciar una instancia de configuración en dicho servidor.

Iniciar línea de ensamblaje...

Cuando se selecciona una instancia de configuración, puede iniciar una línea de ensamblaje desde dicha instancia de configuración.

Importar configuración del servidor...

Permite importar una configuración del servidor a un proyecto de CE.

Exportar configuración al servidor...

Permite exportar el proyecto de CE a una configuración de tiempo de ejecución en el servidor seleccionado.

Editar valores de almacén del sistema

Abre los valores de almacén del sistema para el servidor seleccionado.

Examinar almacenes del sistema

Abre el explorador de datos de almacén del sistema para ver/editar las tablas de almacén del sistema.

Abrir el depurador de línea de ensamblaje

Conecta un depurador a la línea de ensamblaje seleccionada. Esto le permitirá depurar una línea de ensamblaje que ya esté en ejecución.

Depurar servidor

Abre una sesión de depuración de servidor para el servidor seleccionado.

Mostrar los componentes instalados

Muestra una lista de componentes instalados y sus versiones para el servidor seleccionado.

Abrir la consola AMC

Abre la consola AMC para el servidor seleccionado. Si el servidor se ha instalado sin AMC esta opción aparece atenuada.

Nota: La característica AMC está en desuso y se eliminará en una versión futura de IBM Security Directory Integrator.

Suprimir documento de servidor

Suprime el servidor de la vista de servidores.

Renombrar documento de servidor

Renombra el servidor seleccionado. Esto no tiene efecto en el mismo servidor; sólo es una representación local del servidor.

Las opciones de menú se habilitan e inhabilitan en función de la selección.

Editor de expresiones

El editor de expresiones está disponible en muchos contextos diferentes. Con frecuencia, cuando se especifican valores de parámetros, como parámetros de conexión, criterios de enlace, etc., puede utilizar el editor de expresiones en lugar de escribir un valor simple para el parámetro.

Usar propiedad

Esta opción le permite seleccionar una propiedad existente en los almacenes de propiedades o crear un nuevo par propiedad/valor.

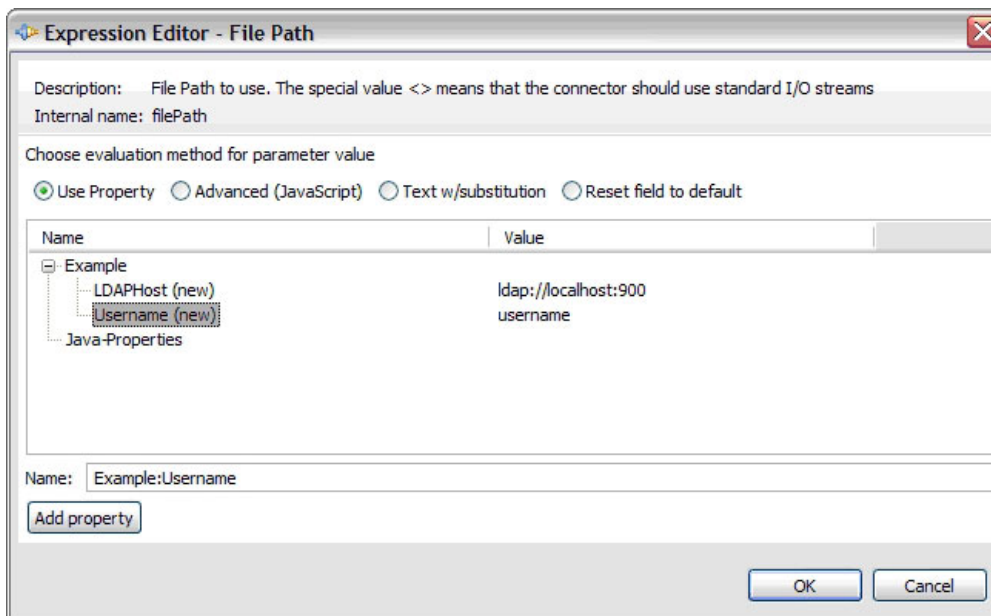


Figura 11. Editor de expresiones: propiedad simple

Cuando selecciona una propiedad, el campo de texto *Name* (Nombre) situado debajo del árbol se actualiza con la expresión para dicha propiedad. De forma predeterminada, la expresión incluye el nombre de almacén ("Example" en este caso) seguido de dos puntos y el nombre de propiedad "Username" en este caso). Cuando se pulsa **Aceptar**, la expresión del campo de texto se utiliza para el parámetro. Esto también significa que puede escribir la expresión directamente en este campo sin pasar por el árbol de propiedades. Por ejemplo, si sabe que existe

una propiedad llamada "FilePath" en el servidor donde se ejecutará esta solución, puede eliminar el nombre de almacén si no lo conoce (es decir, escribir "FilePath" sin el prefijo "store:").

Avanzado (JavaScript)

Con esta opción el valor se calcula como el resultado del código JavaScript que se especifica en el editor.

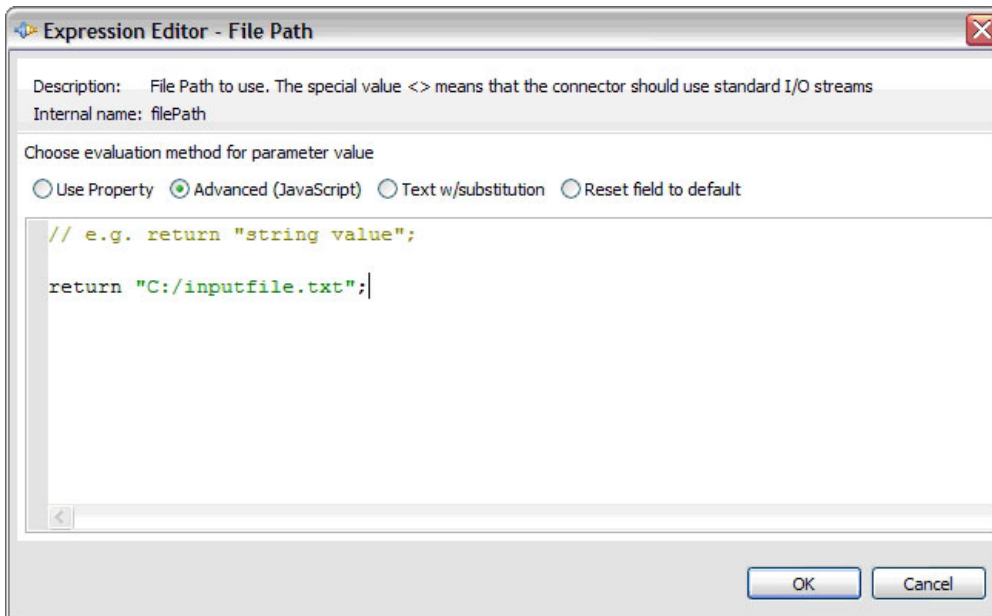


Figura 12. Editor de expresiones: Avanzado (JavaScript)

Texto con sustitución

Esto es la "Expresión de IBM Security Directory Integrator" existente en la versión 6.x. A partir de la versión 7 en adelante, se recomienda utilizar JavaScript para realizar una evaluación compleja de variables y propiedades. Sin embargo, esta opción es muy útil si desea especificar grandes cantidades de texto. Las opciones de sustitución son compatibles con las expresiones de la versión 6.

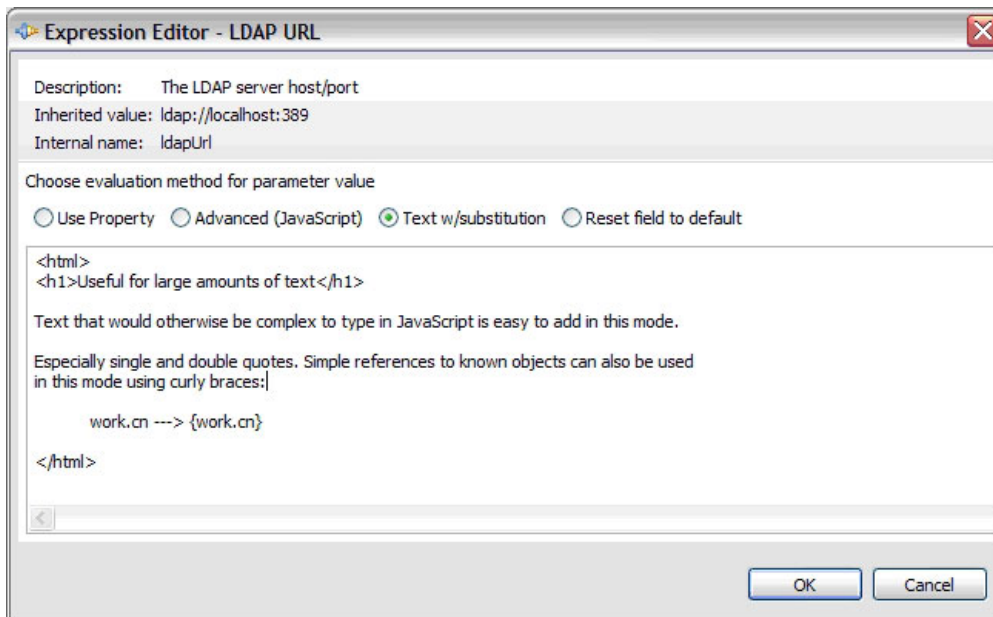


Figura 13. Editor de expresiones: texto con sustitución del tipo de v.6

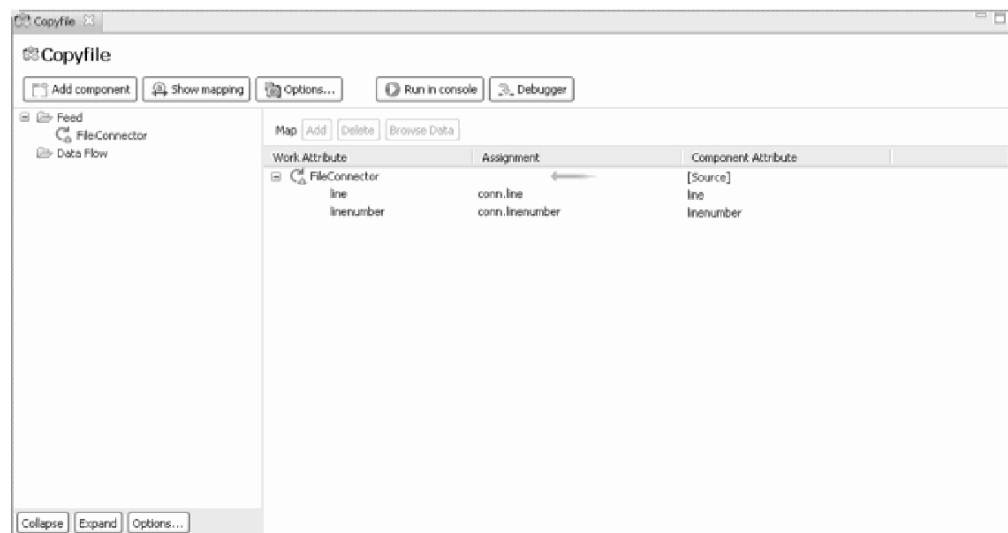
Restablecer el campo al valor determinado

La última opción se utiliza para restablecer el valor del parámetro a su valor predeterminado (también se conoce como valor heredado). Seleccione esta opción y pulse **Aceptar** para restablecer el valor del parámetro.

Editor de líneas de ensamblaje

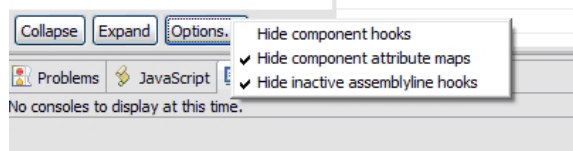
El editor de líneas de ensamblaje es el editor principal que se utiliza cuando se desarrollan soluciones de IBM Security Directory Integrator.

En este editor el usuario crea la línea de ensamblaje mediante la adición y configuración de componentes. Mientras procede, puede ejecutar la línea de ensamblaje para ver los efectos de los componentes añadidos.

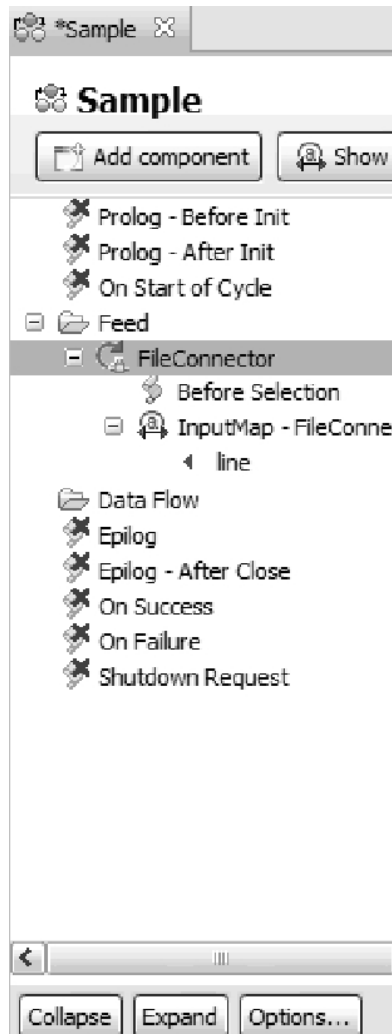


El editor de líneas de ensamblaje muestra dos secciones principales. La sección de la izquierda muestra los componentes y complementos de software de la línea de

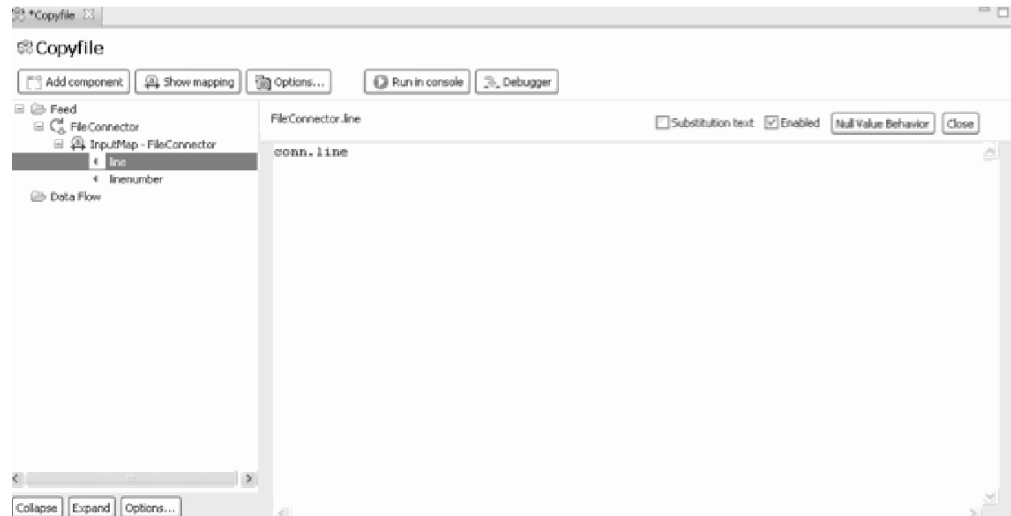
ensamblaje. En la barra de herramientas puede seleccionar el nivel de detalle que desea ver en el árbol.



El botón **Opciones...** le permite seleccionar qué parte de la línea de ensamblaje se muestra en la vista de árbol de componentes.

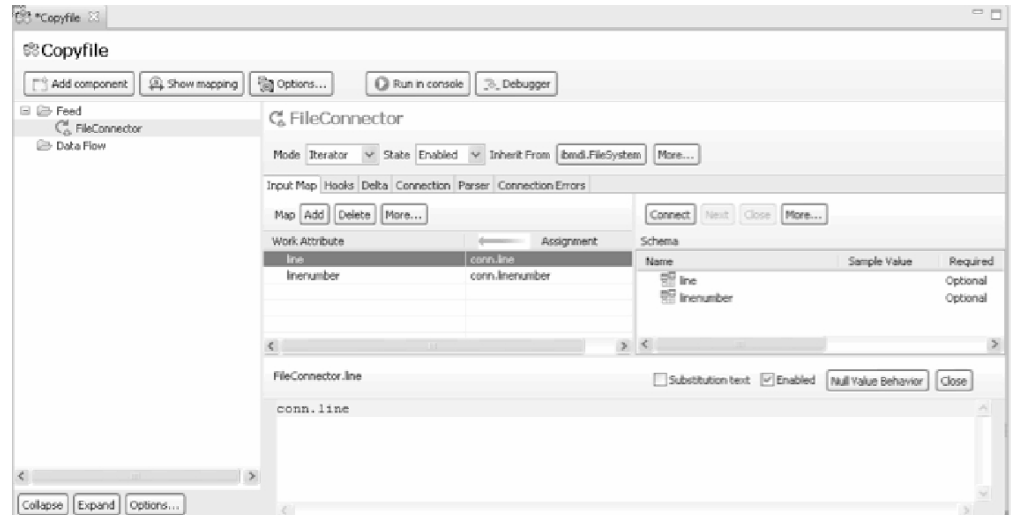


Esta imagen muestra todos los complementos de software de línea de ensamblaje además de las correlaciones de atributos de componentes en el árbol. Al seleccionar complementos de software o elementos de correlación de atributos en este árbol, el lado derecho proporcionará una vista más grande del elemento que dentro del mismo editor de componentes:



La sección de correlación de la derecha muestra las correlaciones para todos los componentes con una correlación de atributos. El primer nivel de este árbol es el nombre de componente con los elementos de correlación de atributos individuales debajo. Al seleccionar este elemento, se activa el editor de detalles para dicho elemento. Si muestra complementos de software en la vista de componentes de línea de ensamblaje, también puede efectuar una doble pulsación en ellos para activar el editor de scripts.

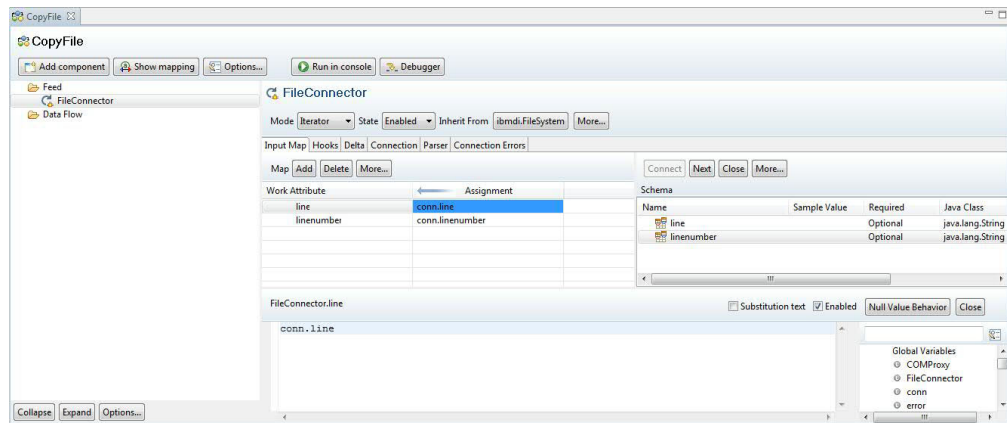
La imagen siguiente muestra cómo aparece el editor rápido con el script para la correlación de atributos.



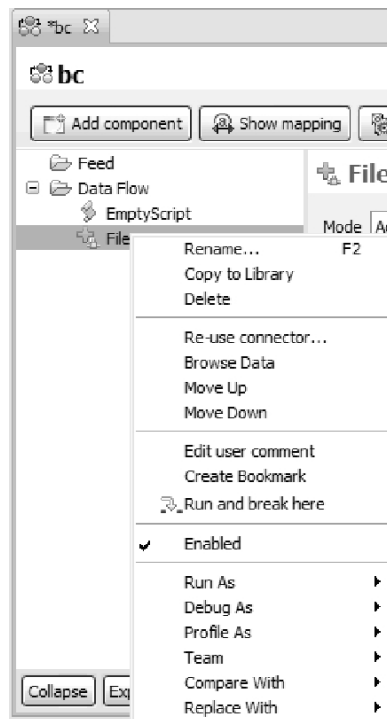
El editor rápido también muestra algunas opciones adicionales. El "Texto de sustitución" es el formato de "Expresión de IBM Security Directory Integrator de la versión 6, donde puede especificar texto y algunas macros de expansión simples. A partir de la versión 7, este formato está pensado principalmente para valores de texto grandes o complejos dado que JavaScript proporciona una sintaxis de expresión más potente.

La sección de flujo de componentes muestra todos los componentes de la línea de ensamblaje. Cuando se selecciona un componente del lado derecho del editor, se sustituye por la pantalla de configuración de dicho componente. Un atajo útil en el CE es Ctrl-M, que maximiza el editor o vista actual para llenar la pantalla de la

aplicación. Ctrl-M conmutará entre el tamaño maximizado y normal.



Al pulsar sobre un componente, una pantalla de configuración de dicho componente sustituye la vista de correlación de atributos global. También puede pulsar el botón derecho del ratón sobre el componente para acceder al menú emergente del componente.



Opciones de línea de ensamblaje

Desde el botón desplegable **Valores** puede seleccionar varias opciones.

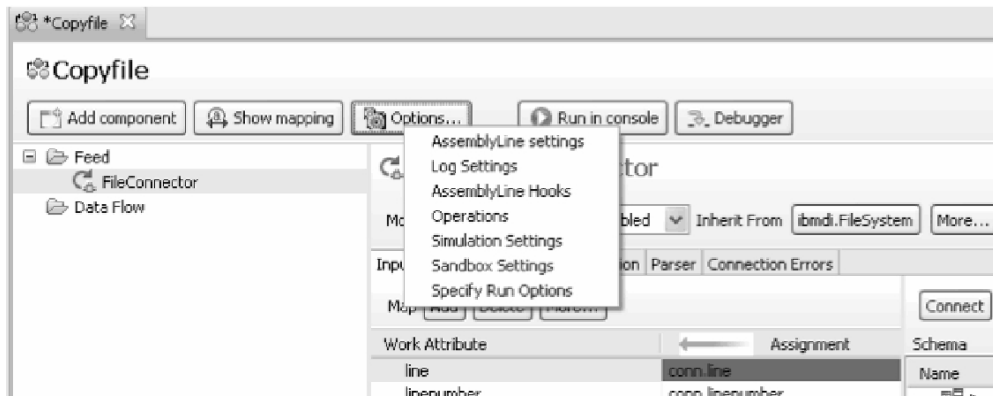


Figura 14. Menú de opciones de línea de ensamblaje

En el menú desplegable hay disponibles varias pantallas de opciones, que controlan diferentes aspectos de la línea de ensamblaje, tanto en lo relacionado con el diseño como con las opciones de tiempo de ejecución. Estas pantallas son las siguientes:

- “Valores de línea de ensamblaje”
- “Valores de los registros” en la página 102
- “Complementos de software de línea de ensamblaje” en la página 103
- “Operaciones de línea de ensamblaje” en la página 104
- “Valores de simulación” en la página 105
- “Valores de recinto de seguridad” en la página 106

Valores de línea de ensamblaje

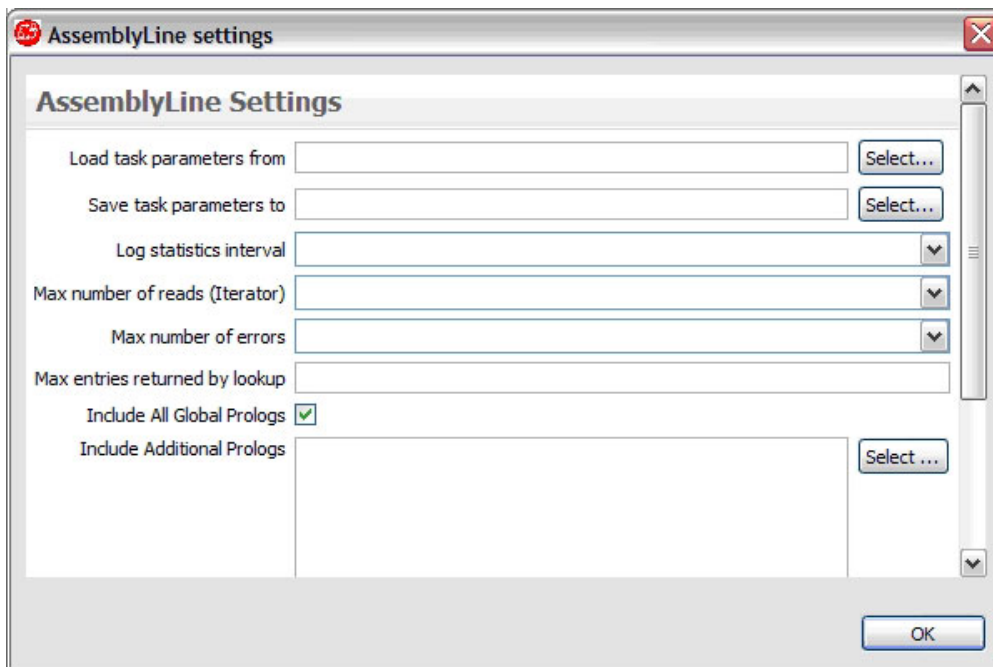


Figura 15. Valores de línea de ensamblaje

En esta ventana puede especificar opciones que afectan al modo de ejecución de la línea de ensamblaje.

Valores de los registros

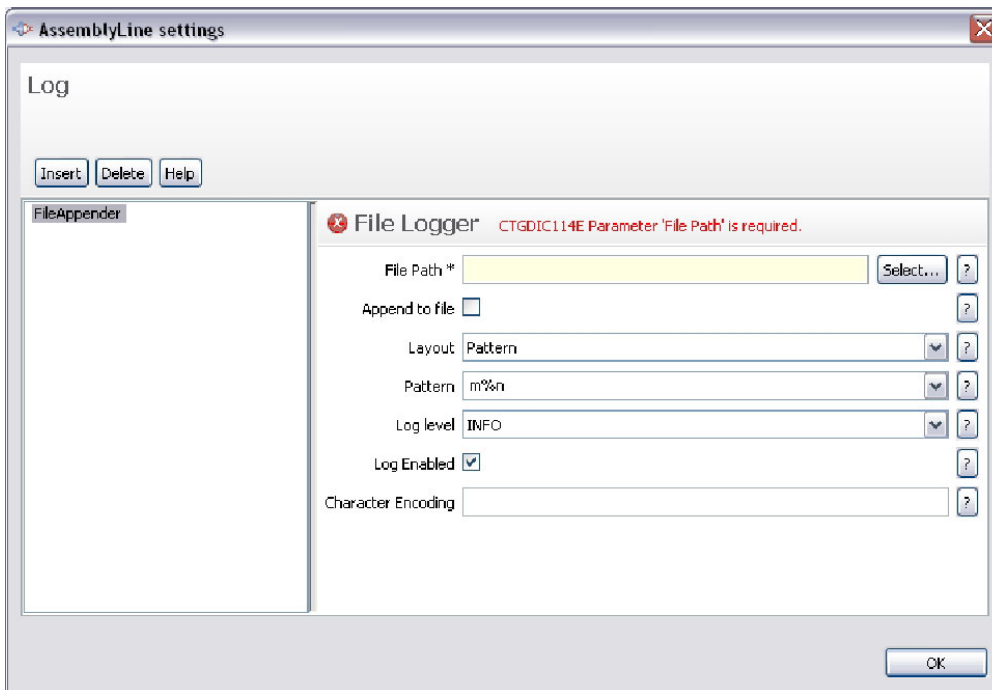


Figura 16. Valores de los registros de línea de ensamblaje

En esta ventana puede añadir registradores para esta línea de ensamblaje. Los registradores no son globales y sólo se activan para la línea de ensamblaje.

Complementos de software de línea de ensamblaje

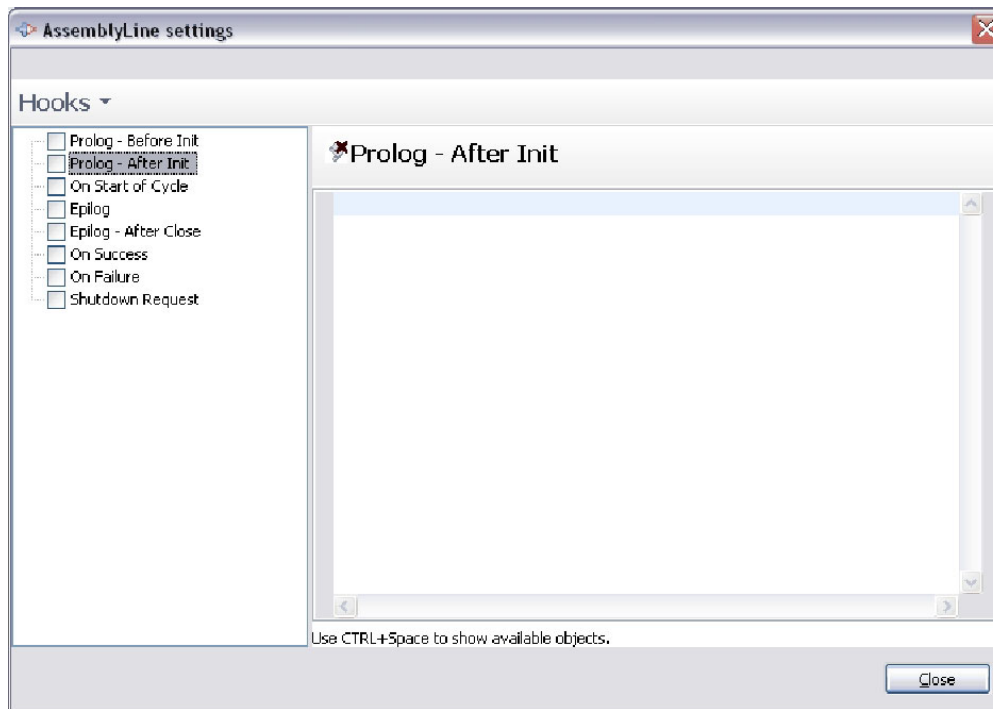


Figura 17. Complementos de software de línea de ensamblaje

En esta ventana puede habilitar/inhabilitar complementos de software de nivel de línea de ensamblaje. Los complementos de software habilitados también se mostrarán en el panel de componentes del editor de líneas de ensamblaje.

Operaciones de línea de ensamblaje

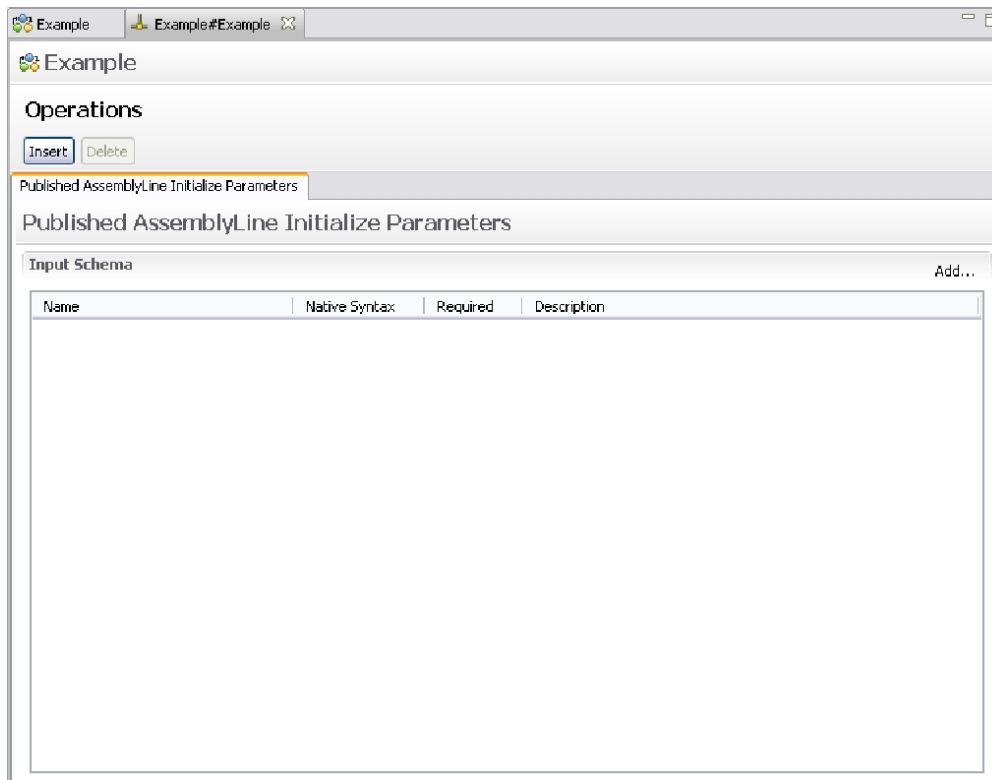


Figura 18. Operaciones de línea de ensamblaje

Esta ventana le permite definir operaciones de línea de ensamblaje. Consulte el apartado sobre operaciones de línea de ensamblaje, en "Creating new components using Adapters" (Creación de nuevos componentes utilizando adaptadores) en la publicación *Referencia* para obtener una descripción completa de las operaciones. El botón Insertar le permite añadir una nueva operación. **Parámetros de inicialización de la línea de ensamblaje publicada**, disponible de forma predeterminada, es una operación especial que se utiliza para proporcionar valores a una línea de ensamblaje antes de inicializar los componentes.

Valores de simulación

Simulation Settings

Proxy AssemblyLine

Server Name:

Config ID:

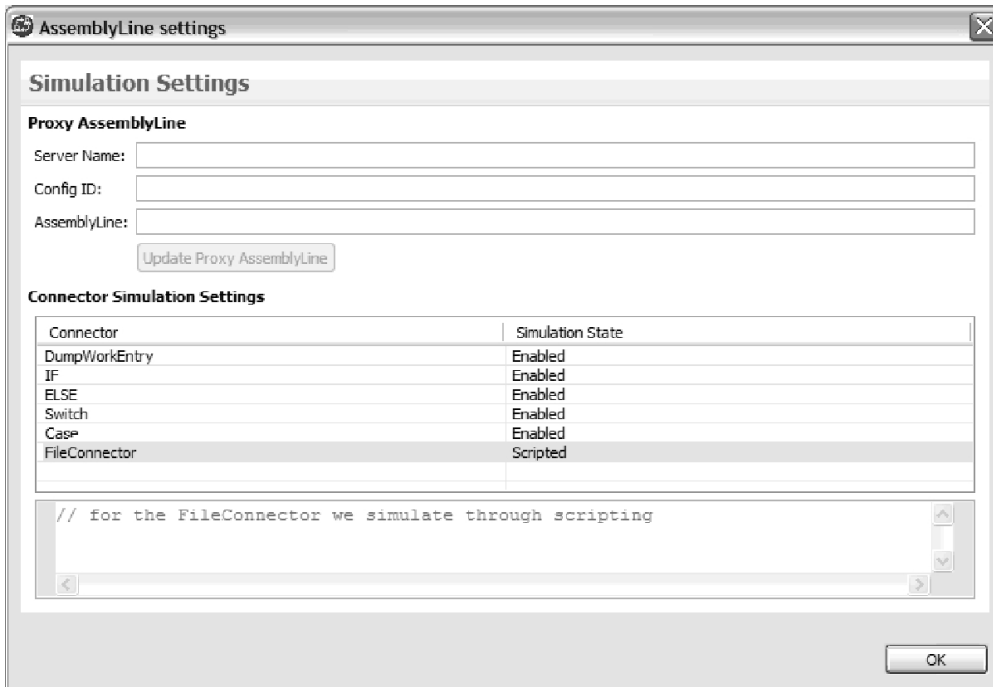
AssemblyLine:

Connector Simulation Settings

Connector	Simulation State
DumpWorkEntry	Enabled
IF	Enabled
ELSE	Enabled
Switch	Enabled
Case	Enabled
FileConnector	Enabled

Figura 19. Valores de simulación de línea de ensamblaje

Esta ventana le permite configurar los valores de simulación para cada componente. Consulte el apartado “Modalidad de simulación de línea de ensamblaje” en la página 195 para obtener más información. El panel mostrará un editor de scripts en la parte inferior cuando seleccione scripts para simulación:



También puede crear/actualizar la línea de ensamblaje del proxy que utiliza el código de simulación con el botón **Actualizar la línea de ensamblaje del proxy**.

Figura 20. Ventana Valores de simulación de línea de ensamblaje, con editor de scripts

Valores de recinto de seguridad

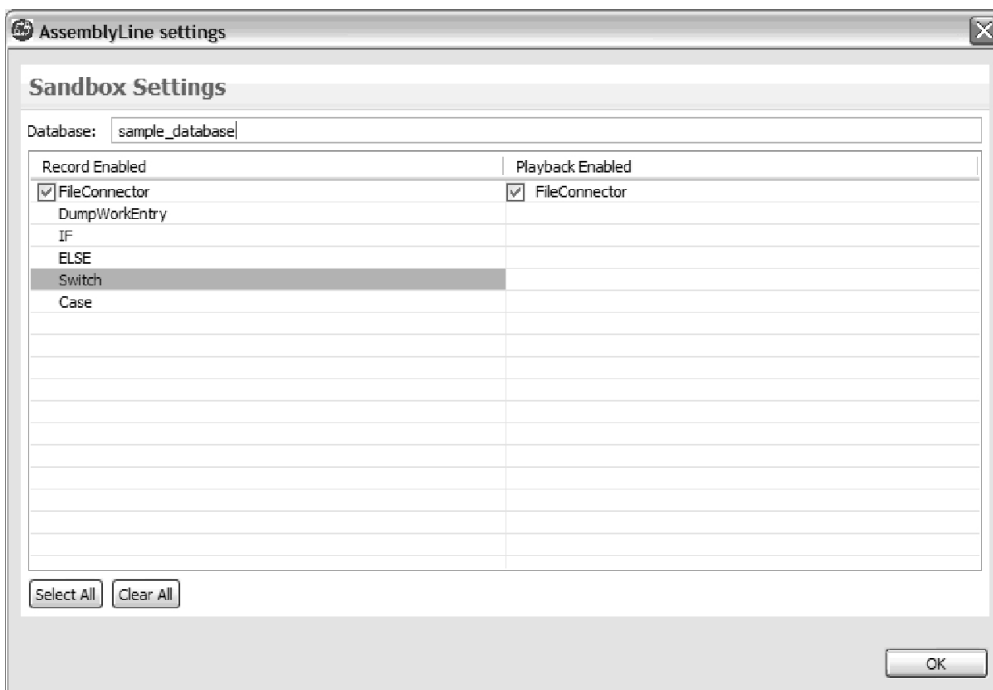


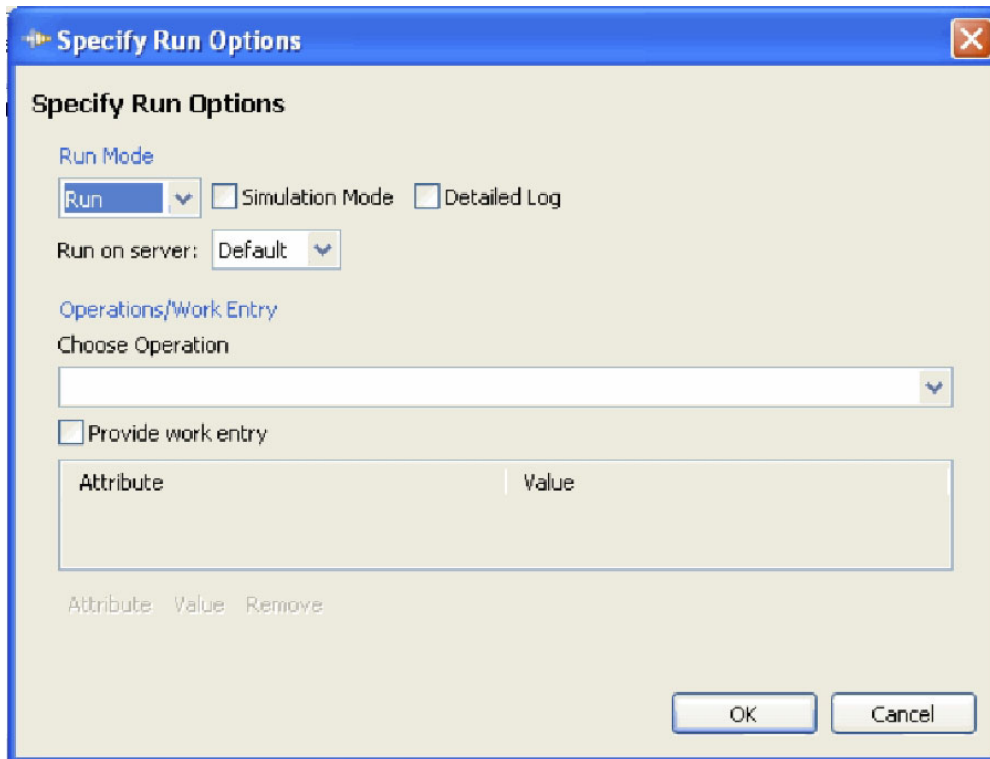
Figura 21. Valores de recinto de seguridad de línea de ensamblaje

Los valores del recinto de seguridad le permiten configurar qué componentes se habilitan para registro/reproducción cuando se ejecuta la línea de ensamblaje en la modalidad de registro o reproducción.

Para obtener más información sobre la funcionalidad del recinto de seguridad en IBM Security Directory Integrator, consulte “Recinto de seguridad” en la página 193.

Especificación de opciones de ejecución

El diálogo de opciones de ejecución permite configurar el modo en que se ejecuta la línea de ensamblaje.



Al seleccionar **Proporcionar entrada de trabajo**, se puede crear una entrada estática que se integra en la línea de ensamblaje cuando se inicia.

Figura 22. Diálogo Especificar opciones de ejecución

Paneles de componente

Cuando se abre un componente en la línea de ensamblaje se obtiene un panel del editor rápido en la parte inferior de la ventana Línea de ensamblaje.

Se obtiene para los componentes siguientes:

- “Rama IF/ELSE/ELSE-IF” en la página 108
- “Rama de conmutador/caso” en la página 108
- “Valor de atributo For-Each” en la página 110
- “Bucle condicional” en la página 110
- “Bucle de conector” en la página 111
- “Correlación de atributos” en la página 112

Rama IF/ELSE/ELSE-IF

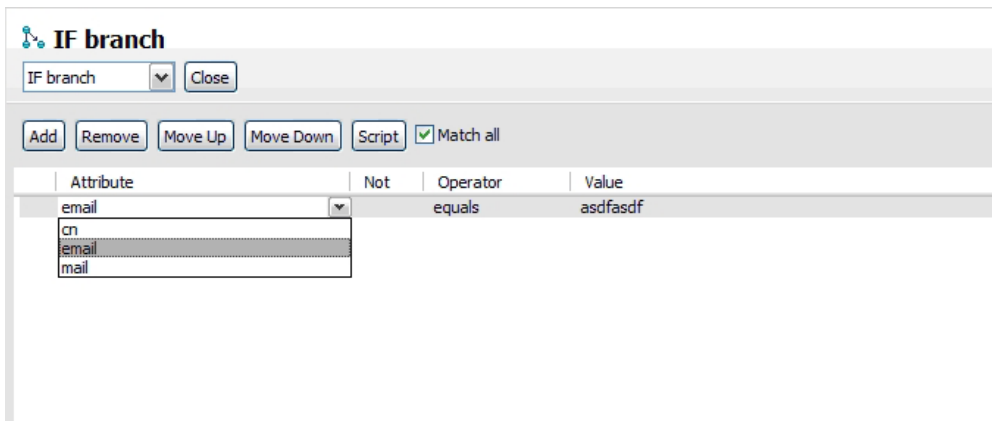


Figura 23. Editor rápido para la rama IF/ELSE/ELSE-IF

La rama IF y ELSE-IF le permite especificar expresiones simples y un script personalizado que devuelve *true* o *false*. Utilice el recuadro de selección **Coincidir todo** para devolver true sólo si todas las condiciones se evalúan como *true* (modalidad AND). Si se deselecciona sólo es necesario que coincida una de las condiciones (modalidad OR).

La rama ELSE no tiene parámetros.

Utilice el botón **Añadir** para añadir nuevas filas de controles de Atributo/Operador/Valor. Puede eliminar estas filas utilizando el botón **Suprimir**. El valor puede ser una constante o una expresión. Utilice el editor de expresiones para configurar la expresión pulsando el botón situado a continuación del campo de texto del valor. Los botones de mover se utilizan para reordenar las expresiones. Las expresiones se evalúan de arriba a abajo. También puede cambiar el tipo de rama (es decir, IF, ELSE, etc.) utilizando el desplegable que hay debajo del título.

Rama de conmutador/caso

La configuración del conmutador tiene varias opciones para seleccionar un valor. Este valor se utiliza para comparar los valores en las ramas de caso contenidas. Cuando son iguales la rama de caso se ejecuta. La rama de conmutador se ejecuta siempre.

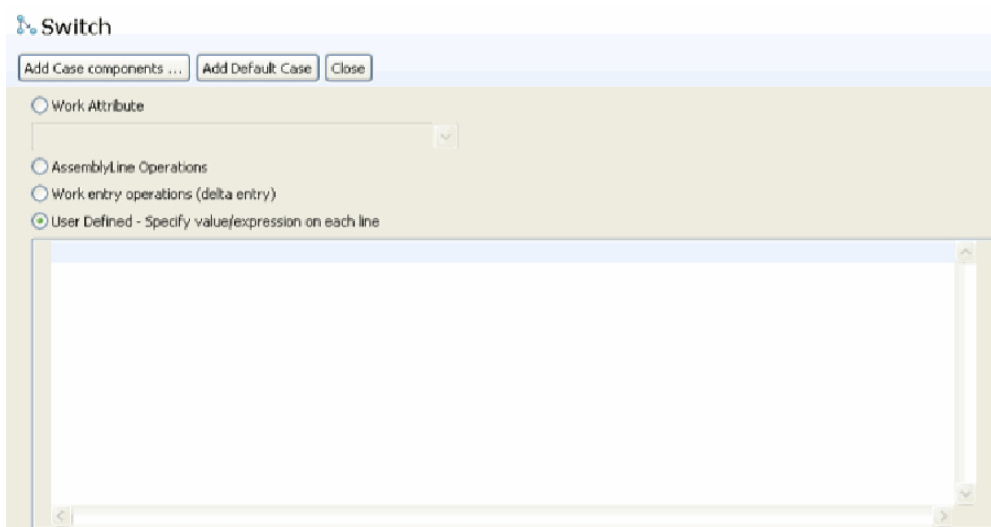


Figura 24. Rama de conmutador/caso

Opciones de selección que puede elegir para la rama de conmutador/caso son:

Atributo work

Seleccione de la lista de atributos work conocidos

Operaciones de línea de ensamblaje

Seleccione de la lista de nombres de operación de línea de ensamblaje conocidos.

Operaciones de entrada work

Utiliza el valor de operación de la entrada work (por ejemplo, el método `work.getOperation()`).

Definido por el usuario

Aquí puede especificar el valor que elija.

Utilice el diálogo **Añadir componentes de caso...** para generar las ramas de caso dentro de esta rama de conmutador. Según la selección que se realice, sugerirá automáticamente los valores que tienen sentido para la selección. Si elige operaciones de la entrada work, sugerirá todos los valores de operación conocidos (por ejemplo, añadir, modificar).

Utilice el botón **Añadir caso predeterminado** para añadir un caso predeterminado. El caso predeterminado se ejecutará cuando ninguna de las otras ramas de caso coincida con el valor del componente de conmutador.

Valor de atributo For-Each

Figura 25. Bucle de valor de atributo

Este componente forma un bucle basado en valores de un atributo. Especifique el nombre de atributo de entrada work para formar un bucle (nombre de atributo work) y el nombre de atributo de bucle. El nombre de atributo de bucle se establece en el valor de bucle actual del atributo de entrada work (por ejemplo, `foreach f in work.attr.values; set loopattr = f`).

Bucle condicional

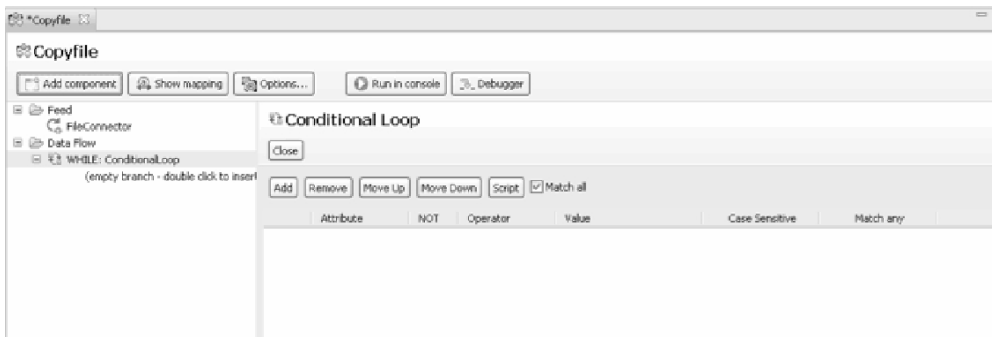


Figura 26. Bucle condicional

El bucle condicional le permite especificar expresiones simples y un script personalizado que devuelven true/false.

Utilice el botón **Añadir** para añadir nuevas filas de controles de Atributo/Operador/Valor. Puede eliminar estas filas utilizando el botón Suprimir. El valor puede ser una constante o una expresión. Utilice el editor de expresiones para configurar la expresión pulsando el botón situado a continuación del campo de texto del valor.

El recuadro de selección **Coincidir todo** determina si deben coincidir todas las líneas (**Coincidir todo** seleccionado) o si sólo es necesario que una sea true para que la rama ejecute.

Bucle de conector

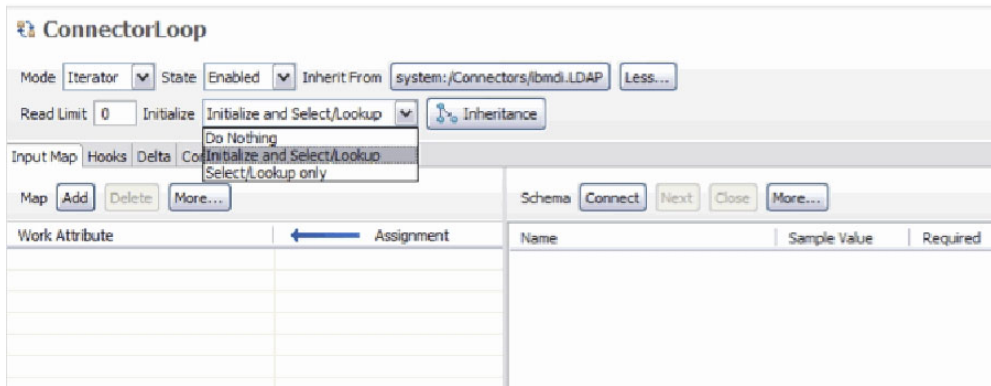


Figura 27. Bucle de conector

El bucle de conector utiliza el editor de conectores para configurar un conector. Aunque existen algunas diferencias que se muestran en la imagen anterior. La opción de inicializar ahora muestra las opciones correspondiente a un bucle de conector en lugar de las opciones de inicializar normales. Además, sólo se puede seleccionar Iterador y Buscar en el desplegable Modalidad.

Aparte de las pestañas usuales que se muestran para un conector, también hay un correlación de atributos de salida que le permite configurar asignaciones dinámicas de los parámetros del conector en la pestaña **Parámetros del conector**. Es ligeramente diferente de la correlación de salida corriente porque muestra un esquema fijo, que es la lista de parámetros para el conector elegido. La parte del esquema tampoco tiene botones Conectar/Siguiente para actuar sobre el esquema.

Figura 28. Parámetros del conector en el bucle de conector

Correlación de atributos

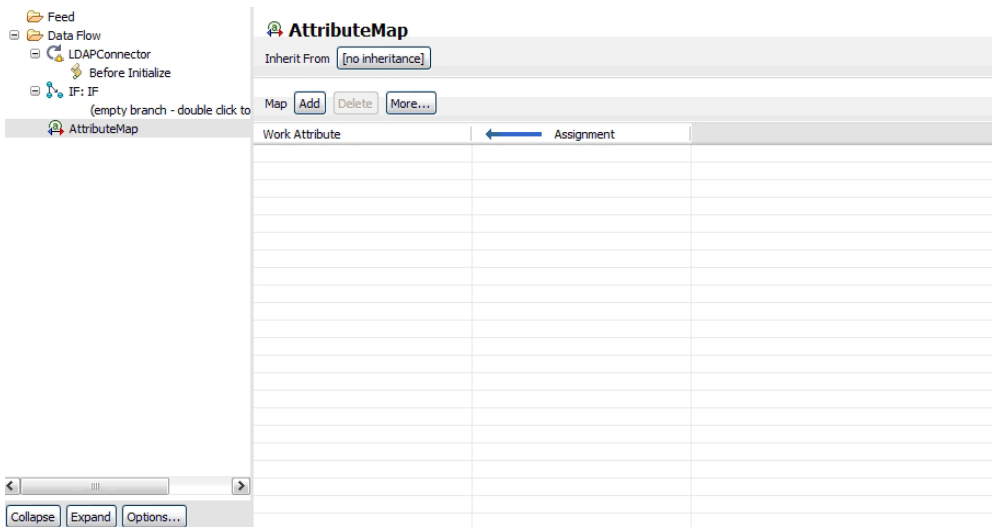


Figura 29. Componente de correlación de atributos independiente

El componente de correlación de atributos muestra un único panel en el que puede correlacionar atributos con la entrada work, de forma independiente de la correlación de atributos de otros componentes como, por ejemplo, un conector. El componente también ofrece reutilizar correlaciones de atributos de otros componentes como, por ejemplo, conectores, funciones y otros componentes de correlación de atributos de su biblioteca. Para obtener más información, consulte el apartado “Correlación de atributos y esquema” en la página 115.

Vista de documentación del usuario

En ocasiones, una línea de ensamblaje puede resultar bastante compleja. Como ayuda a la hora de leer la documentación, puede documentar las partes de la línea de ensamblaje mediante la vista de documentación.

En el diseño de la línea de ensamblaje, puede pulsar el botón derecho del ratón en un componente y seleccionar **Editar comentario de usuario** para llevar la vista de documentación a primer plano:

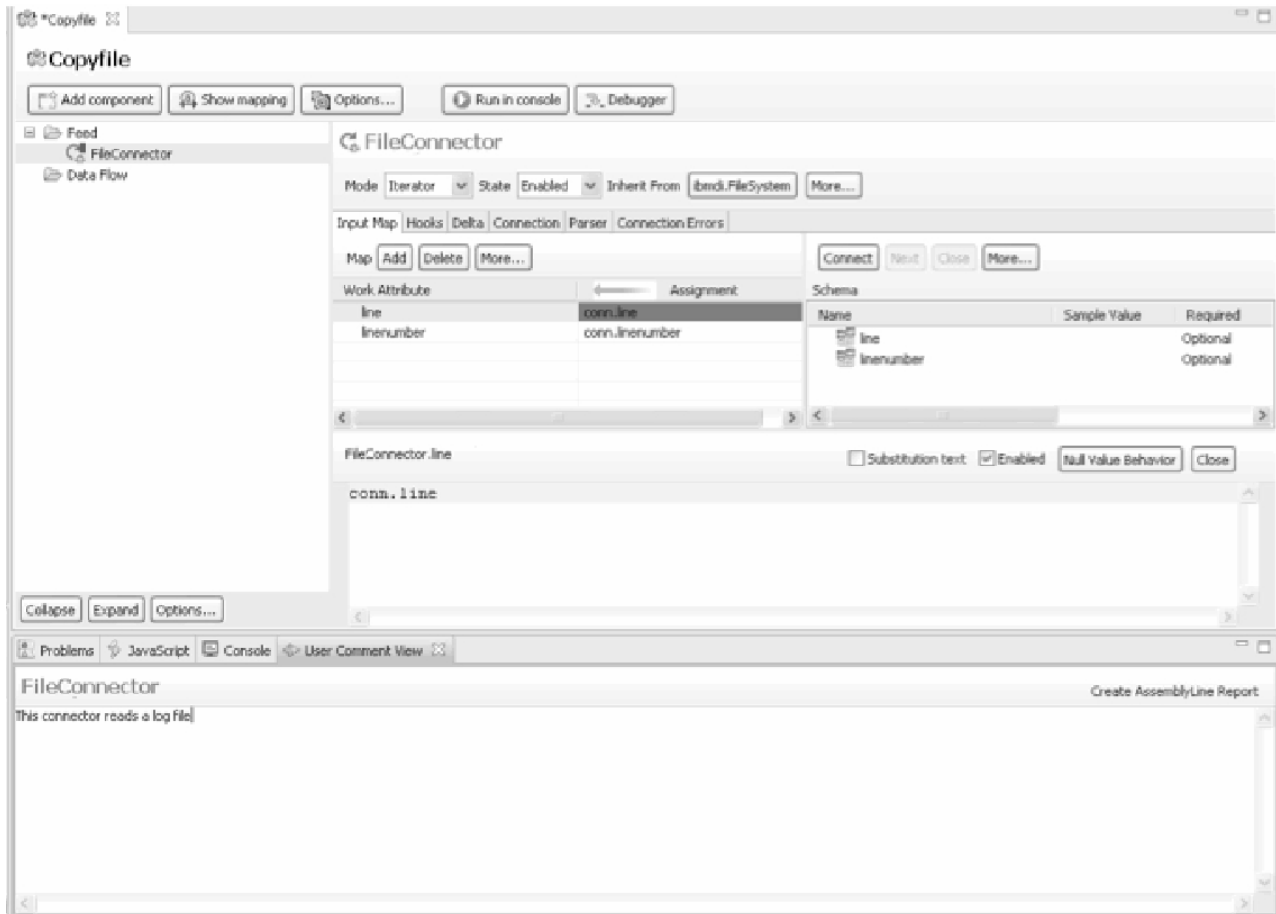


Figura 30. Vista de documentación del usuario

A medida que la selección varía en el editor de la línea de ensamblaje, la vista de documentación refleja la selección actual. El texto que especifique en la vista se guarda al guardar la línea de ensamblaje. Los componentes con un comentario aparecen en la esquina superior izquierda del icono de componente.

El botón **Crear un informe de línea de ensamblaje** creará un informe con todos los comentarios de usuario especificados en la línea de ensamblaje. La plantilla de informe utilizada se denomina `UserCommentsReport.xsl` en el directorio `dir_instalación_TDI/XSLT/ConfigReports`.

Figura 31. Informe de línea de ensamblaje de ejemplo

Ventana Ejecutar línea de ensamblaje

Cuando se ejecuta la línea de ensamblaje se obtiene una ventana que muestra la salida del registro.

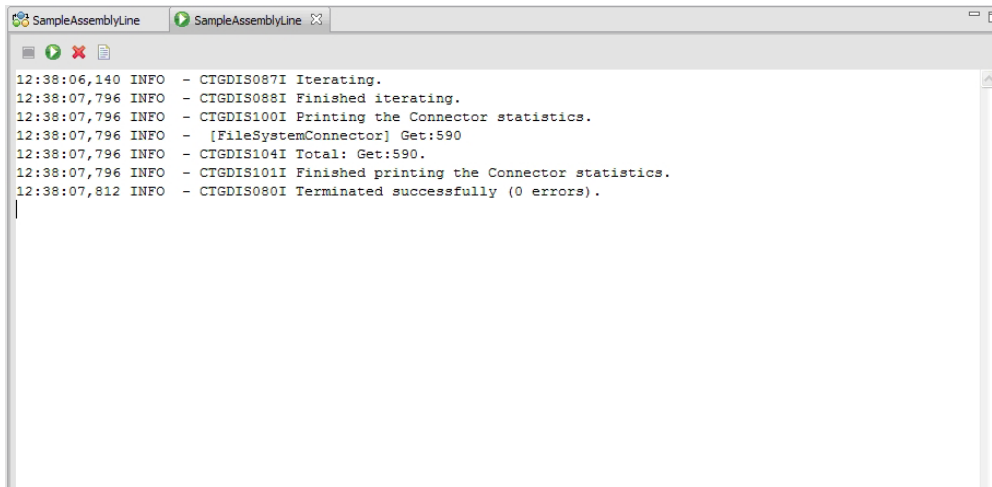


Figura 32. Registro de la consola

En esta pantalla, también puede detener una línea de ensamblaje en ejecución y reiniciarla más adelante cuando haya terminado.

Nota: Detener una línea de ensamblaje significa que el Editor de configuración envía una notificación de detención a la instancia de configuración (normalmente el servidor local predeterminado) que ejecuta la línea de ensamblaje; no concluye la hebra inmediatamente, sino que detiene la ejecución en cuanto el servidor vuelve a obtener el control. Esto es diferente en las versiones anteriores en las que pulsar el botón **Detener** haría que se concluyera todo el proceso del servidor que ejecutaba la línea de ensamblaje.

Los otros dos botones son para borrar la ventana de registro y abrir el archivo de registro en otra ventana del editor. La ventana de registro sólo muestra las últimas cien líneas para evitar problemas de falta de memoria.

El archivo de registro se escribe en un archivo temporal utilizando el prefijo "tdi_ce_al_log" y la extensión ".log". El archivo se coloca en el directorio temporal específico de la plataforma, que con frecuencia se define en la variable de entorno TEMP/TMP. El archivo de registro se suprime automáticamente cuando se cierra la ventana Ejecutar línea de ensamblaje, pero en caso de que la aplicación o la máquina dé un error puede que sea necesario eliminar manualmente estos archivos de registro. El editor que debe utilizarse para este archivo es de forma predeterminada el editor de texto simple, pero se puede cambiar correlacionando la extensión ".log" con otro editor (incluyendo editores externos). Utilice la opción de menú **Ventanas > Preferencias** para abrir el siguiente diálogo:

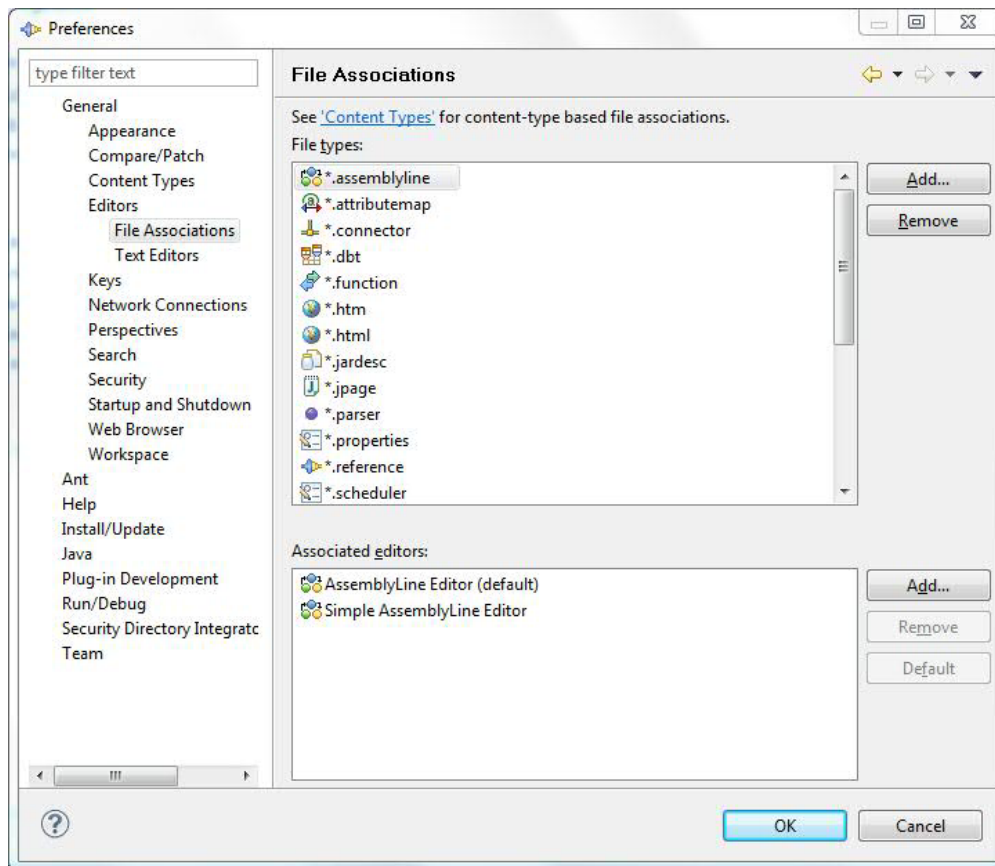


Figura 33. Preferencias de asociaciones de archivos del Editor de configuración

Aquí puede añadir la extensión “.log” y asociarla a un editor.

Correlación de atributos y esquema

La correlación de atributos se realiza utilizando el panel de correlación de atributos en la línea de ensamblaje o en el editor de componentes.

En el editor de líneas de ensamblaje puede añadir atributos pulsando el botón derecho del ratón en la sección de correlaciones de atributos y seleccionando añadir atributo o utilizando el botón **Añadir** en la barra de herramientas tal como se muestra a continuación.

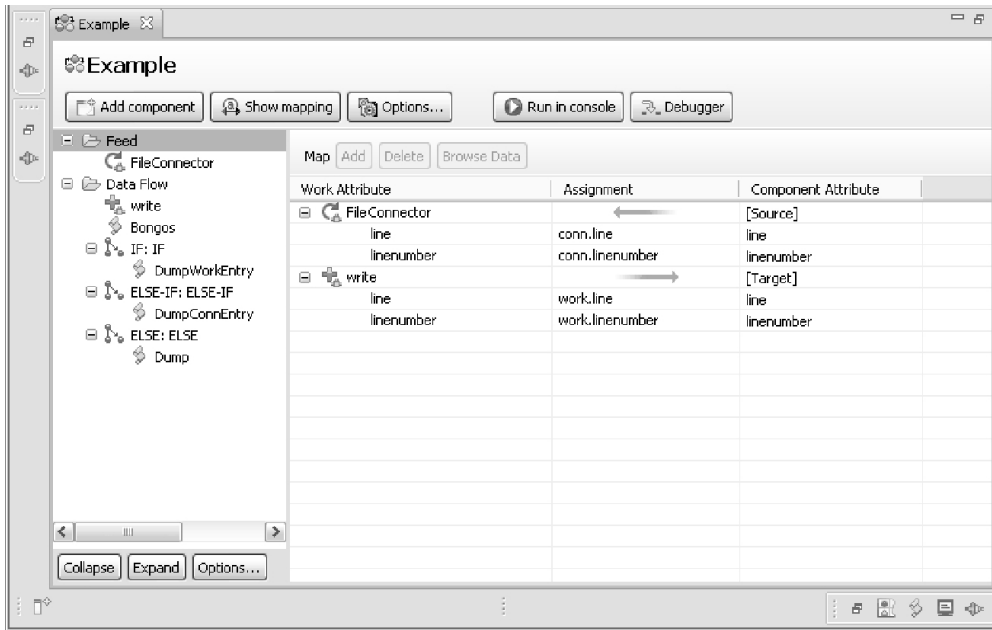


Figura 34. Correlación de atributos

Esta ventana no muestra el esquema para los componentes de la línea de ensamblaje. Para trabajar con el esquema, abra el editor para el componente seleccionando el componente en el árbol del lado izquierdo.

El escenario típico para la correlación de atributos es descubrir primero el esquema para el componente. Cuando se descubre un esquema, el CE ejecuta un trabajo en segundo plano que ejecuta el método de esquema de consulta del componente. Si no se devuelve ningún esquema, el CE le preguntará si desea leer una entrada para intentar derivar el esquema a partir de ésta. A continuación, el resultado se rellenará en el esquema para el componente que se está editando.

La imagen siguiente muestra el contenido del esquema de entrada para un componente después de descubrir los atributos. Si por algún motivo un componente no le proporciona un esquema, puede añadir elementos de esquema manualmente utilizando el botón **Añadir...** en la barra de herramientas o reutilizar un esquema de otra configuración de componente con la opción **Cambiar la herencia**.

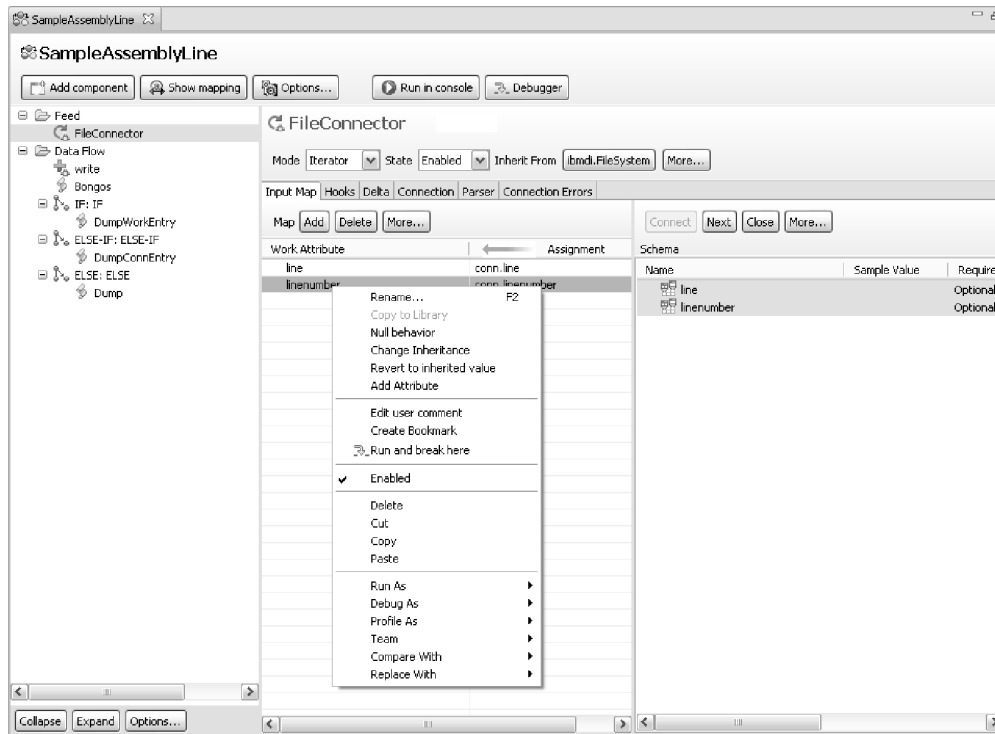


Figura 35. Correlación de atributos, con atributos descubiertos

También puede utilizar el menú desplegable de la barra de título para cambiar la herencia para la configuración de esquema.

Con un esquema, puede arrastrar y soltar elementos individuales en la correlación de atributos o utilizar la función **Correlacionar atributo** desde el menú contextual y modificar la correlación si es necesario.

Figura 36. Modificación de la herencia de Correlación de atributos

Nota: La funcionalidad de arrastrar y soltar depende hasta cierto punto del entorno de ventanas. En concreto, en sistemas UNIX, CDE (Common Desktop Environment) no proporciona esta funcionalidad, por lo tanto para configurar la

configuración deberá utilizar la función **Correlacionar atributo** del menú contextual.

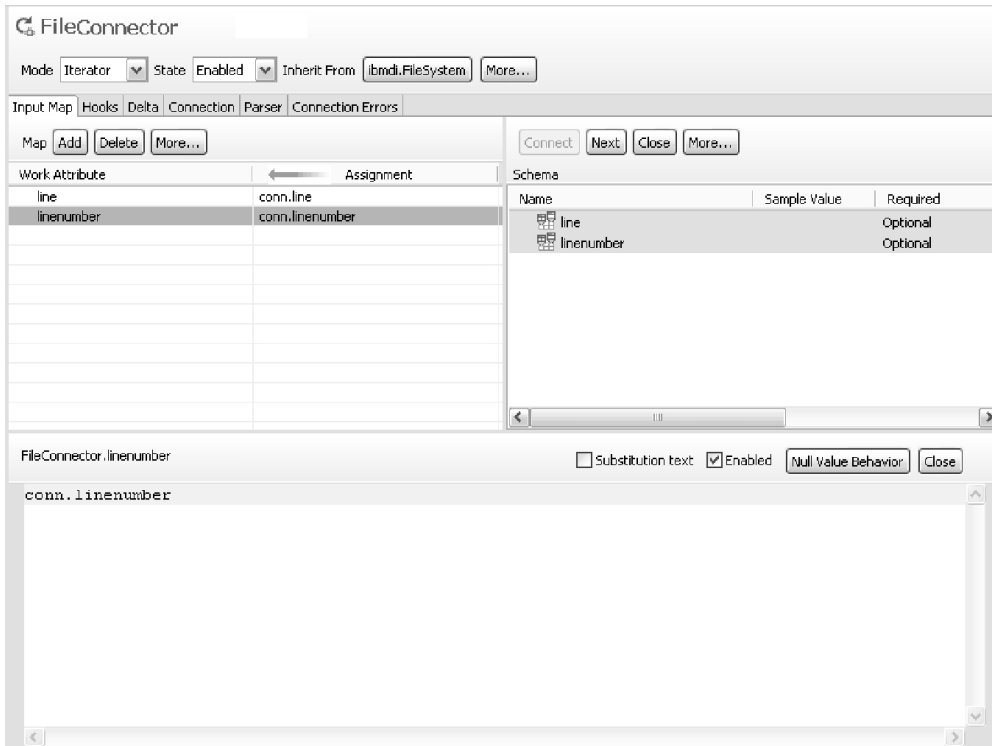


Figura 37. Correlación de atributos, con la ventana de edición de JavaScript para atributo individual

Si no tiene ningún esquema o desea añadir atributos independientemente del esquema, ciertamente puede hacerlo. Utilice el botón **Añadir** para añadir un nuevo atributo a la correlación. Proporcione un nombre para el atributo y se asignará una expresión "conn.nombre-atributo" o "work.nombre-atributo" al nuevo atributo. Esto se puede llevar a cabo en las ventanas del editor de líneas de ensamblaje del editor de conectores.

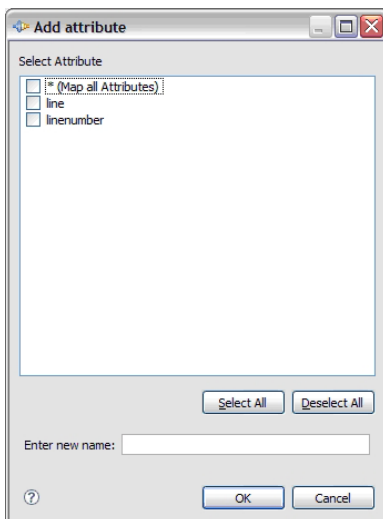


Figura 38. Diálogo Añadir atributo

Aparece un diálogo con un campo de texto editable en el que puede escribir el nombre del nuevo atributo. La lista anterior contiene todos los nombres de atributo conocidos del esquema; puede seleccionar los que desea añadir a la correlación de atributos.

Cuando añada más componentes a la línea de ensamblaje, puede arrastrar atributos entre ellos cuando tenga sentido. Si arrastra un componente hasta otro componente, se correlacionarán todos los atributos correlacionados con el componente de destino. También puede arrastrar atributos desde la correlación de atributos hasta los componentes del panel izquierdo, mostrando todos los componentes de la línea de ensamblaje. Esto realizará una correlación simple de todos los elementos que arrastre. Es similar a arrastrarlos hasta el componente en el panel de correlación de atributos.

El concepto de Correlación de atributos se describe de una forma bastante completa, con muchos ejemplos, en la publicación *Cómo empezar*.

En función del conector y de la modalidad en que se ha configurado, siempre habrá diferentes pestañas en la ventana de configuración del conector.

- Los conectores que están en una modalidad compatible con la entrada desde un sistema conectado siempre tienen una sección denominada **Atributos de entrada**.
- Los conectores que están en una modalidad compatible con la salida desde un sistema conectado siempre tienen una sección denominada **Atributos de salida**.
- Algunos conectores admiten modalidades tanto para entrada como para salida. Si lo ha configurado de este modo, verá una sección **Atributos de entrada** así como una sección **Atributos de salida**.

Correlaciones de atributos externos

Las correlaciones de atributos pueden heredar de archivos de correlaciones de atributos externos. Un archivo de correlación de atributos externos es un archivo de texto que contiene elementos de correlaciones de atributos iguales que los que aparecen en la pantalla de correlación real. La diferencia es que el archivo externo utiliza un formato diferente de la estructura XML interna. Esto facilita la configuración de la correlación de atributos para cualquier conector sin tener que ir al CE. El CE ofrece esta opción en el diálogo de herencia de las correlaciones de atributos:

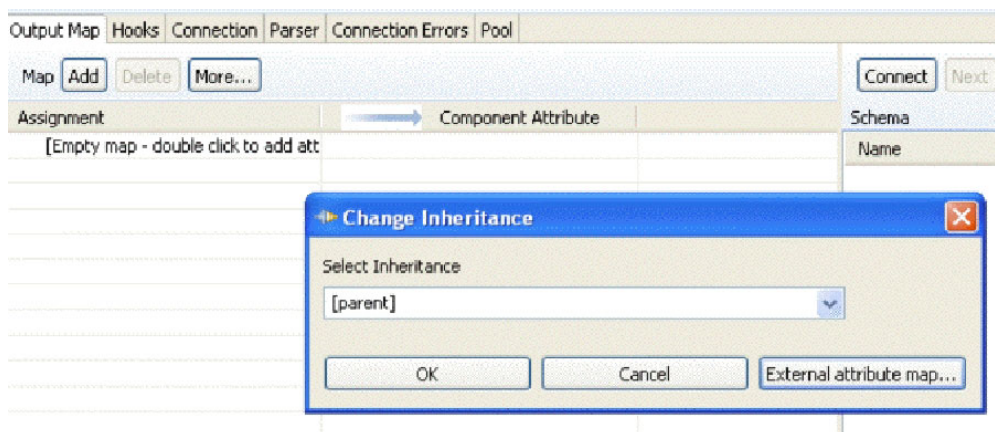


Figura 39. Correlación de atributos: diálogo de herencia

Pulse el botón **Correlación de atributos externos...** para elegir un archivo existente o escriba "archivo:" seguido de la vía de acceso completa al archivo de correlaciones de atributos. Si desea utilizar nombres de vía de acceso relativos, coloque un punto y una barra inclinada (./) del ante del nombre de archivo.

Correlación de atributos de entrada:

La correlación de atributos de entrada es el proceso de mover datos desde el origen de datos a la entrada work de la línea de ensamblaje. Las correlaciones de atributos de entrada se muestran en la ventana Correlaciones de atributos del conector, cuando se activa en el editor de conectores, con una flecha que apunta al conector desde una entidad referida como "[Origen]". También se muestran en la ventana Esquema, en Correlación de atributos de entrada.

Antes de empezar

Para poder configurar la correlación de atributos de entrada, el conector debe configurarse en una modalidad que admita entrada, en la barra de herramientas del conector. Las modalidades que admiten entrada normalmente son Iterador, Buscar y Servidor.

A continuación, en la sección **Correlación de entrada**, debe seleccionar los atributos del origen de entrada que desea procesar en la línea de ensamblaje.

Acerca de esta tarea

Los conectores que deben configurarse para correlación de atributos de entrada pueden residir en <espaciotrabajo>/Resources/Connectors o en su posición designada en la línea de ensamblaje.

Procedimiento

1. Pulse **Correlación de entrada**.
2. Pulse **Conectar** y, a continuación, **Siguiente** para obtener el esquema de muchos orígenes de datos. Algunos conectores o combinaciones de conector-analizador tienen esquemas predefinidos, mientras que otros le solicitan que lea una entrada de ejemplo del origen de datos y la examine para descubrir los atributos.
3. Por último, seleccione Atributos en la lista **Esquema** y, a continuación, arrástrelos a la correlación de atributos o añádalos manualmente con los botones Añadir y Eliminar. La correlación de atributos controla qué atributos se llevan a la línea de ensamblaje para el proceso, así como las transformaciones que se especifiquen.

Qué hacer a continuación

Estos atributos correlacionados se recuperan del origen de datos, en la entrada *Work*, y se pasan a los conectores siguientes de la sección Flujo de la línea de ensamblaje.

Si no ha creado el conector directamente en una línea de ensamblaje, para utilizar este conector en una línea de ensamblaje, arrastre el conector de su ubicación en <espaciotrabajo>/Resources/Connectors a la sección **Entrada de datos** de una línea de ensamblaje.

Correlación de atributos de salida:

La correlación de atributos de salida es el proceso de mover datos desde la entrada *work* de la línea de ensamblaje al destino de salida del sistema conectado. Las correlaciones de atributos de salida se muestran en la ventana Correlaciones de atributos del conector, cuando se activa en el editor de conectores, con una flecha que apunta desde el conector a una entidad referida como "[Destino]". También se muestran en la ventana Esquema, en Correlación de atributos de salida.

Antes de empezar

Para poder configurar la correlación de atributos de salida, el conector debe configurarse en una modalidad que admita salida, en la barra de herramientas del conector. La modalidad típica para salida es Sólo adición. Algunas modalidades, como CallReply, admiten tanto entrada como salida.

A continuación, en **Correlación de salida** en la sección Atributos de salida debe seleccionar los atributos de la entrada *work* de la línea de ensamblaje que desea para la salida en el sistema conectado.

Acerca de esta tarea

Los conectores que deben configurarse para correlación de atributos de salida pueden residir en <espaciotrabajo>/Resources/Connectors o en su posición designada en la línea de ensamblaje. Sin embargo, cuando el conector está únicamente en <espaciotrabajo>/Resources/Connectors, es decir, no es miembro de una línea de ensamblaje, no es fácil arrastrar atributos de entrada *work* a la correlación de atributos de salida. En este caso, arrastre el conector a la línea de ensamblaje, o cree las correlaciones manualmente, pulsando **Añadir** en la ventana Correlaciones de atributos; como alternativa puede pulsar el botón derecho del ratón en el conector de la ventana Correlaciones de atributos y seleccionar **Añadir elemento de correlación de atributos**.

Procedimiento

1. Pulse **Correlación de salida**.
2. Pulse **Conectar** para obtener el esquema del origen de datos. Algunos conectores o combinaciones de conector-analizador tienen esquemas predefinidos, que se mostrarán. Sin embargo, la mayoría de conectores, no.
3. Si el conector está en una línea de ensamblaje, arrastre los atributos de entrada *work* correlacionados previamente al conector de la ventana Correlaciones de atributos del editor de líneas de ensamblaje. Como alternativa, cree atributos manualmente; la comparación de nombres se produce durante la ejecución. Por ejemplo, un elemento de correlación de atributos de salida creado como `un_atributo` hace que un atributo de entrada *work* llamado `un_atributo` se correlacione con un atributo del sistema conectado con el mismo nombre.

Qué hacer a continuación

Estos atributos correlacionados se recuperan de la entrada *work* cuando se llama a este conector en el flujo de la línea de ensamblaje y se pasan como salida al sistema conectado.

Si no ha creado el conector directamente en una línea de ensamblaje, para utilizar este conector en una línea de ensamblaje, arrastre el conector de su ubicación en

<espaciotrabajo>/Resources/Connectors a la sección **Flujo** de una línea de ensamblaje.

Editor de conectores

El editor de conectores se utiliza cuando se editan archivos de conector o cuando se utiliza la función **Editar** en un conector de la línea de ensamblaje.

La creación de un conector se describe en el apartado “Creación de un conector”.

El editor utiliza los mismos indicadores que en los asistentes y diálogos emergentes para los conectores. El editor consta de seis pestañas en las que se muestran paneles de configuración para distintos aspectos del conector. En la parte superior están los atributos principales del conector como, por ejemplo, su modalidad, el estado y otras opciones generales del conector.

Las pestañas son:

1. “Correlaciones de atributos de entrada y salida” en la página 123
2. “Complementos de software” en la página 124
3. “Conexión” en la página 125
4. “Analizador” en la página 125
5. “Criterios de enlace” en la página 126
6. “Errores de conexión” en la página 128
7. “Delta” en la página 130
8. “Agrupación” en la página 132
9. “Herencia del conector” en la página 132

Creación de un conector

La creación de un conector implica decidir dónde residirá el conector y los parámetros iniciales se asignarán al conector.

Antes de empezar

Debe decidir dónde residirá el conector; existen dos lugares posibles:

1. Los conectores indicados para reutilización y compartimiento de recursos residen en el directorio <workspace>/Resources/Connectors. Normalmente es el mejor lugar para crear y mantener los conectores. Los conectores definidos de este modo se añaden a las líneas de ensamblaje arrastrándolos hasta el lugar adecuado.

A continuación, puede modificar estos pocos valores del conector de modo que cumpla el rol que tiene designado en la línea de ensamblaje, pero dejando la mayoría de los valores heredados y, por lo tanto, sin modificar respecto a sus definiciones en la sección Recursos.

2. También puede crear un conector directamente en una línea de ensamblaje; los conectores definidos de este modo son definiciones ad hoc que únicamente son válidas en el contexto de esa línea de ensamblaje concreta.

Acerca de esta tarea

Los conectores son la columna vertebral de cualquier solución creada en IBM Security Directory Integrator, ya que establecen la conexión con los sistemas con los que se desea intercambiar datos.

Procedimiento

1. Pulse el botón derecho del ratón y seleccione **Recursos > Connectors > Nuevo conector...** en el espacio de trabajo o seleccione **Archivo > Nuevo > Conector**
2. Navegue hasta la ubicación en la que desea situar el nuevo conector y asígnele un nombre.
 - a. La ubicación recomendada es <workspace>/Resources/Connectors.
 - b. Como alternativa, puede crear el nuevo conector directamente en la línea de ensamblaje. Navegue hasta la ubicación en la línea de ensamblaje de destino; la sección Canal de información para conectores en modalidad Iterador y Servidor o Flujo para las otras modalidades.
3. Pulse **Finalizar** para crear el conector.
4. En la pestaña **Conexión**, establezca la modalidad del nuevo conector en la modalidad que desee.
5. Establezca los parámetros de conexión para este conector en la pestaña **Conexión**; los parámetros necesarios se indican con un (*). Algunos conectores requieren que también se configure un analizador en la pestaña **Analizador**.
6. Vaya a la ventana **Correlación de atributos** de la ventana de configuración del conector para descubrir o definir el esquema para este origen de datos: pulse **Descubrir atributos** para obtener el esquema para el origen de datos. Algunos conectores o combinaciones de conector-analizador tienen esquemas predefinidos. Cuando no es así, se le solicita que lea una entrada de ejemplo del origen de datos y la examine para descubrir los atributos.

Qué hacer a continuación

Una vez definido el conector, ya está preparado para configurar qué partes de la información conocidas como *atributos* deben fluir desde y hacia la línea de ensamblaje. Este proceso se denomina *Correlación de atributos*, desde el punto de vista de la línea de ensamblaje. Por lo tanto, la definición de atributos de correlación desde el sistema conectado, a través del conector de entrada, hasta la entrada de trabajo en la línea de ensamblaje se realiza en la *Correlación de atributos de entrada*; y la definición inversa, la correlación de atributos desde la entrada de trabajo, a través del conector de salida, hasta el sistema conectado se realiza en la *Correlación de atributos de salida*.

Correlaciones de atributos de entrada y salida

La pestaña Correlación de atributos muestra las correlaciones de atributos de entrada y salida y los esquemas para un componente.

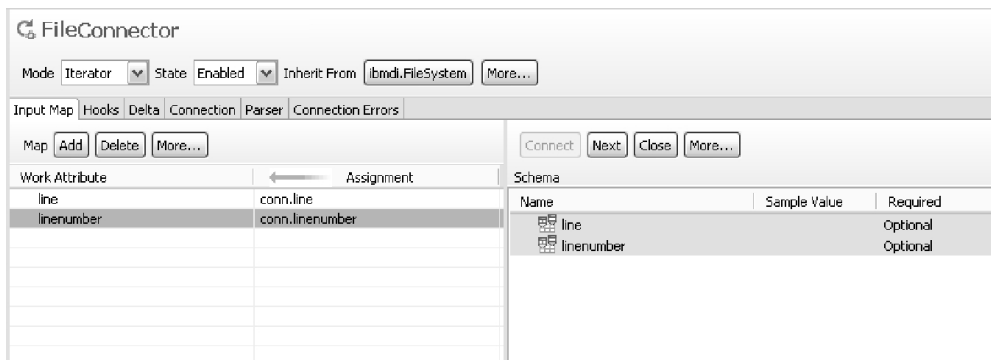


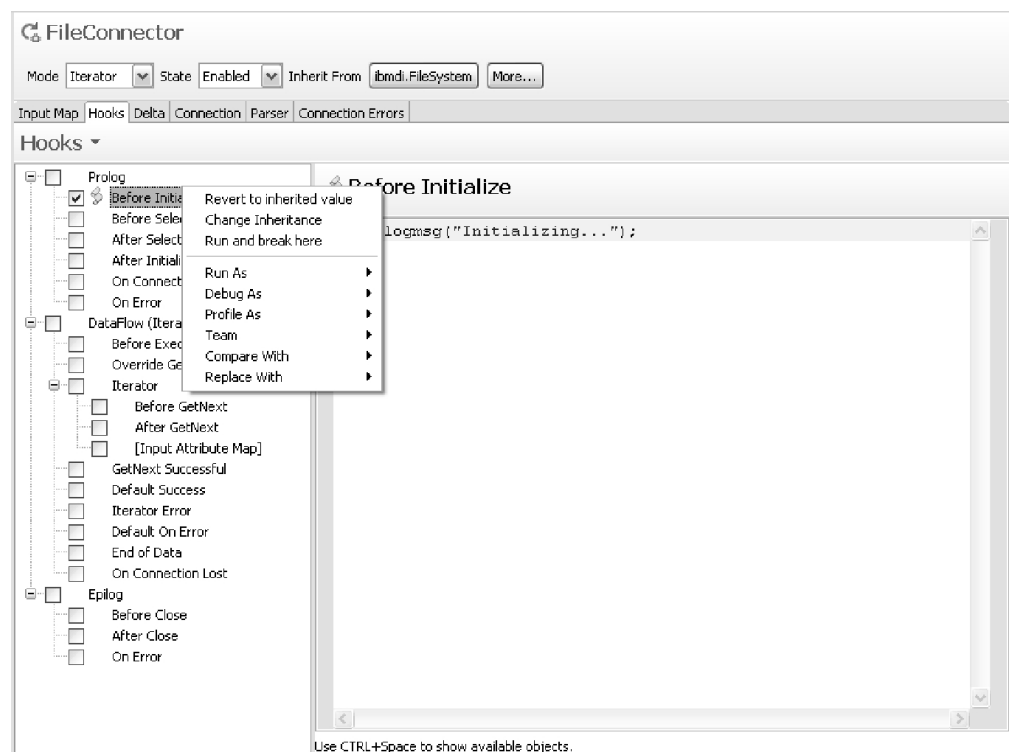
Figura 40. Ventana Correlación de atributos

Consulte la sección “Correlación de atributos y esquema” en la página 115 en el editor de líneas de ensamblaje para obtener una descripción de esta ventana.

Complementos de software

La pestaña Complementos de software muestra todos los complementos de software para el conector.

Utilice el recuadro de selección para habilitar/inhabilitar un complemento de software y seleccionar el complemento de software para editar su contenido. Cuando se modifica el contenido de un complemento de software, el complemento de software se habilita automáticamente. Los complementos de software que contienen código de script tendrán un icono de script en la vista de árbol para que pueda identificar rápidamente si un complemento de software tiene o no contenido. Tenga en cuenta que si un complemento de software está habilitado, se ejecutará cuando se llegue a él en el flujo de ejecución, tanto si contiene script como si no.



Consulte la sección “Flujo y complementos de software de una línea de ensamblaje” en la página 34 para obtener una descripción de los distintos complementos de software, tanto al nivel de línea de ensamblaje como a nivel de componente individual.

Conexión

Figura 41. Pestaña Conexión

Los parámetros de la pestaña Conexión son muy específicos para el componente que desea configurar. Consulte la especificación individual del componente en la publicación *Referencia*.

Analizador

Si un conector puede utilizar (o necesita) un analizador, también habrá una pestaña para él.

Utilice el botón de la barra de herramientas **Seleccionar analizador** para cambiar el analizador para el conector.

Por ejemplo, el analizador de lector de líneas tiene una pantalla de configuración parecida a la que se muestra a continuación:

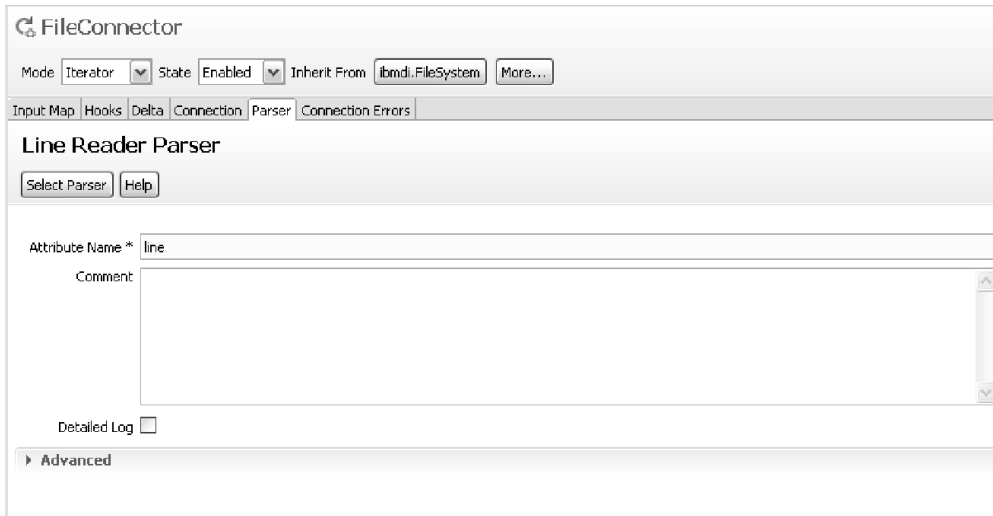


Figura 42. Analizador de lector de líneas

Criterios de enlace

Cuando un componente necesita un criterio de enlace, se muestra la pestaña Criterios de enlace.

El hecho de que un componente presente o no una pestaña Criterios de enlace no sólo depende del tipo de componente, sino que también depende de su modalidad. Consulte el apartado “Criterios de enlace” en la página 21 para obtener más información.

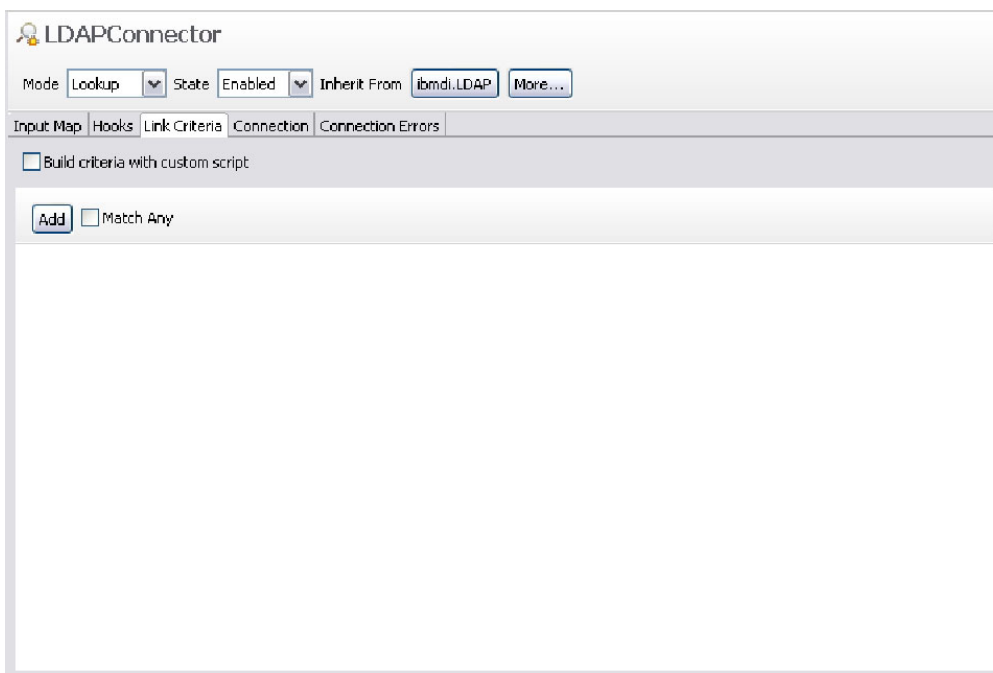


Figura 43. Pestaña Criterios de enlace

Utilice el botón **Añadir** para añadir una nueva fila de controles a la vista. Utilice el botón **Suprimir** para eliminar filas individuales. En la vista se especifica el nombre de atributo, el operando (por ejemplo, igual a, contiene) y el valor de coincidencia. El valor de coincidencia puede ser una constante o una expresión. Utilice el botón



para presentar el editor de expresiones:

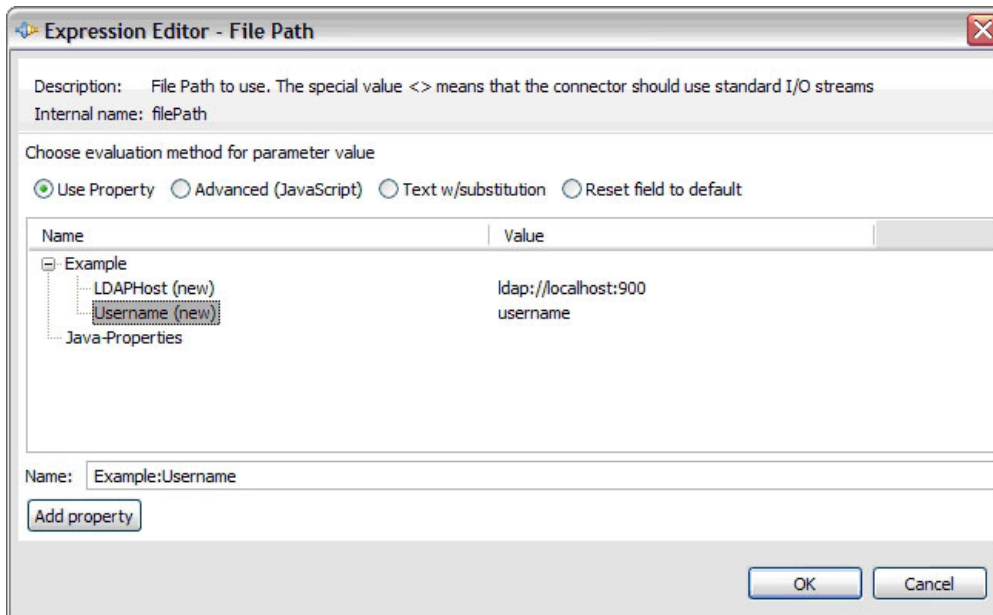


Figura 44. Ventana Editor de expresiones, modalidad simple

En el editor de expresiones puede seleccionar una propiedad de uno de los archivos de propiedades o utilizar la modalidad Avanzado cuando se devuelve un valor basado en código JavaScript. Seleccione el recuadro de selección **Avanzado (JavaScript)** para conmutar entre la selección de propiedades y el código JavaScript. Sólo puede utilizar una de las dos opciones.

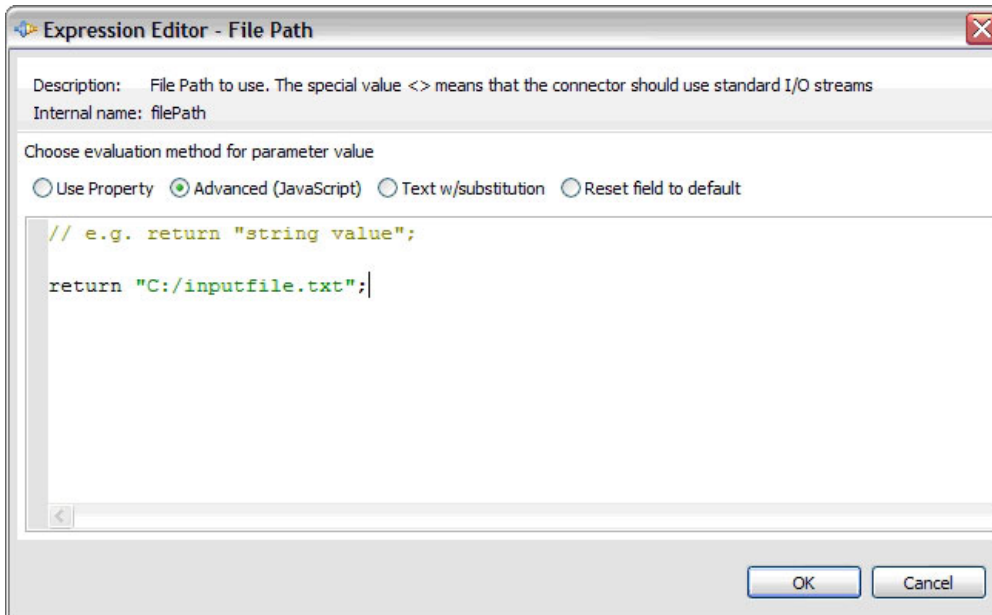
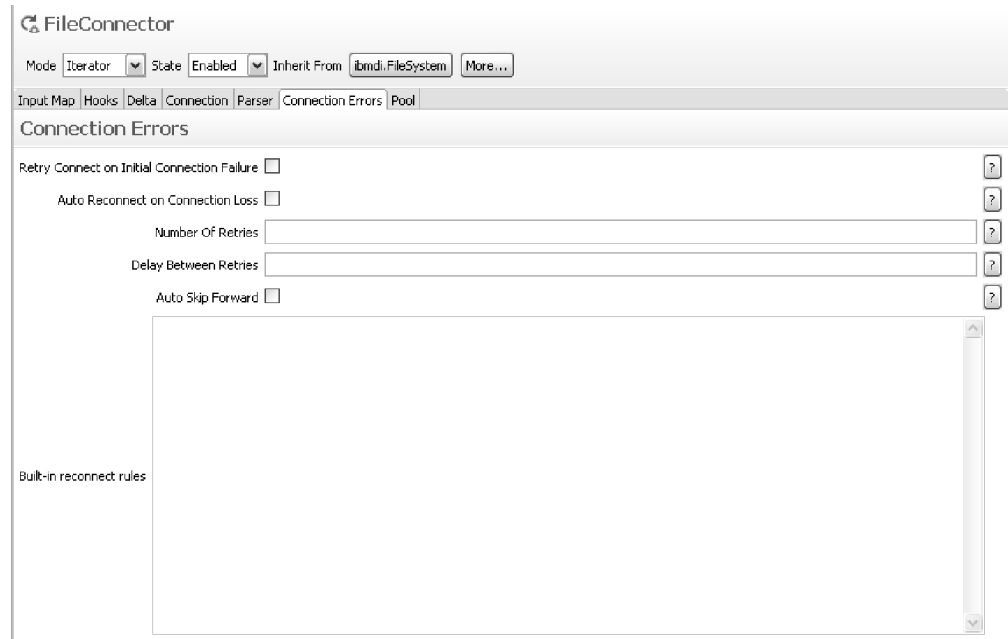


Figura 45. Ventana Editor de expresiones, modalidad Avanzado (JavaScript)

Errores de conexión

La pestaña Errores de conexión es donde configura la capacidad de recuperación de las conexiones, es decir, el comportamiento automatizado que se debe seguir cuando falla la conexión que ha establecido el componente.

Opcionalmente, cuando selecciona **Reintentar conexión en caso de fallo de conexión inicial**, puede configurar si el intento de conexión generará una excepción si falla durante el inicio o si lo reintentará. Si se establece este distintivo, y no se puede establecer una conexión cuando se inicializa el conector, se realizará un intento de "reconexión"; en realidad no es una reconexión, ya que no se ha establecido una conexión en primer lugar, pero generalmente se trata del mismo mecanismo que para las situaciones que pueden producirse cuando se pierde una conexión establecida.



Para conexiones establecidas, los parámetros siguientes tienen el significado siguiente:

Reconexión automática si se pierde la conexión

Si se establece este distintivo, y se ha perdido la conexión después de inicializar el conector, se realizará un intento de reconexión.

Número de reintentos

El número de veces que se realizará un intento de reconexión cuando se produce un problema, antes de desistir. Si se produce un problema más adelante, se realizará el mismo número de intentos.

Intervalo entre reintentos

El número de segundos que se esperará entre cada intento de reconexión y antes del primer intento de reconexión.

Salto adelante automático

Después de una reconexión, saltar adelante automáticamente tantas veces como el número de lecturas satisfactorias.

Reglas de reconexión incorporadas

Están relacionadas con el motor de reglas de reconexión; consulte la sección correspondiente en la publicación *Instalación y administración*

para obtener más información.

Delta

Configure Delta

Enable Delta ?

Unique Attribute Name ?

Delta Store Delete ?

Read Deleted ?

Remove Deleted ?

Return Unchanged ?

Commit After every database operation ?

Row Locking Read committed ?

Faster algorithm ?

Allow duplicate Delta keys ?

Change Detection Mode Use all Attributes for change detection ?

Attribute List ?

Esta pestaña sólo está disponible en modalidad Iterador.

Los parámetros de esta pestaña tienen el significado siguiente:

Habilitar delta

El conmutador maestro para el motor delta para este conector. Si no se selecciona, los parámetros siguientes no se habilitan.

Nombre de atributo exclusivo

El nombre de un atributo, o una opción de varios atributos de entrada separados por "+", que contiene un valor exclusivo en un origen de datos determinado. Los orígenes de datos con claves duplicadas no pueden estar sujetos a la función delta, excepto cuando **Permitir claves Delta duplicadas** está habilitado. Consulte el apartado "Detección delta" en la página 17 para obtener más información.

Almacén delta

La tabla del almacén del sistema que contiene la información delta de ejecuciones anteriores para este conector, para poder detectar las diferencias en ejecuciones posteriores.

Leer entradas suprimidas

Si se selecciona, la línea de ensamblaje proporcionará las entradas suprimidas en la ejecución de línea de ensamblaje cuando el iterador finalice la iteración, es decir, finalice la entrada. El código de operación indicará que esta entrada se ha suprimido en el origen de entrada. Tenga en cuenta que las entradas marcadas para suprimir no se suprimen del almacén Delta a menos que también se habilite el distintivo Eliminar entradas suprimidas.

Eliminar entradas suprimidas

Si se selecciona, las entradas suprimidas del origen de entrada se suprimen del almacén Delta, de modo que no se volverán a detectar en las ejecuciones posteriores.

Devolver las entradas sin cambios

Si se selecciona, las entradas sin cambios de esta ejecución se proporcionan a la línea de ensamblaje.

Comprometer

Selecciona cuándo se confirmarán los cambios realizados en el almacén Delta como resultado de la repetición mediante la entrada. Las opciones son:

- Después de cada operación de base de datos
- Al final del ciclo de la línea de ensamblaje
- Al cerrarse el conector
- Sin compromiso automático

El valor por defecto es **Después de cada operación de base de datos**.

Bloqueo de fila

Selecciona el nivel de aislamiento de transacción de la conexión con el almacén Delta. Este parámetro aborda la necesidad del bloqueo de fila en la tabla Almacén Delta cuando varios clientes de base de datos acceden a los mismos datos estableciendo un *nivel de aislamiento de transacción*. Al establecer un nivel de aislamiento mayor, se reducen las anomalías de transacción conocidas como 'lecturas incorrectas', 'lecturas no repetibles' y 'lecturas fantasma' mediante los bloqueos de fila y tabla.

Las opciones son:

- READ_UNCOMMITTED
- READ_COMMITTED
- REPEATABLE_READ
- SERIALIZABLE

El valor por defecto es READ_COMMITTED. Para obtener más información, consulte la sección "Bloqueo de fila" en la página 223.

Lista de atributos

Lista de atributos separados por comas cuyos cambios serán detectados o ignorados durante el proceso de cálculo de cambios. Los cambios de los atributos listados se verán afectados por el parámetro **Cambiar modalidad de detección**, que especifica si dichos cambios deben detectarse o ignorarse. Para obtener más información sobre este parámetro y el siguiente, consulte la sección "Detectar o ignorar cambios sólo en atributos específicos" en la página 224.

Modalidad de detección de cambios

Especifique si se deben detectar o ignorar los cambios en los atributos listados en el parámetro **Lista de atributos**. Los valores posibles son:

- IGNORE_ATTRIBUTES
- DETECT_ATTRIBUTES
- DETECT_ALL

Algoritmo más rápido

Si se selecciona, indica a la línea de ensamblaje que utilice un algoritmo más rápido para calcular los cambios, a costa de utilizar más memoria.

Permitir claves Delta duplicadas

Cuando se habilita la característica Delta para Registro de cambios/Conector de detección de cambios a la vez que se ejecutan líneas de ensamblaje, es posible que una entrada se modifique más de una vez. Estas modificaciones provocarán que se reciba la entrada por segunda vez y, de este modo, se lanzará una excepción de duplicación de clave delta. Al seleccionar este parámetro, las entradas con atributos de clave duplicada

(especificadas en el parámetro **Nombre de atributo exclusivo**) pueden ser procesadas por conectores de iterador con datos habilitados.

Agrupación

La definición de agrupación para un conector sólo está visible cuando el conector reside en la biblioteca de conectores (es decir, un archivo de Project/Resources/Connectors). Los conectores que se abren desde la línea de ensamblaje no tendrán esta pestaña.

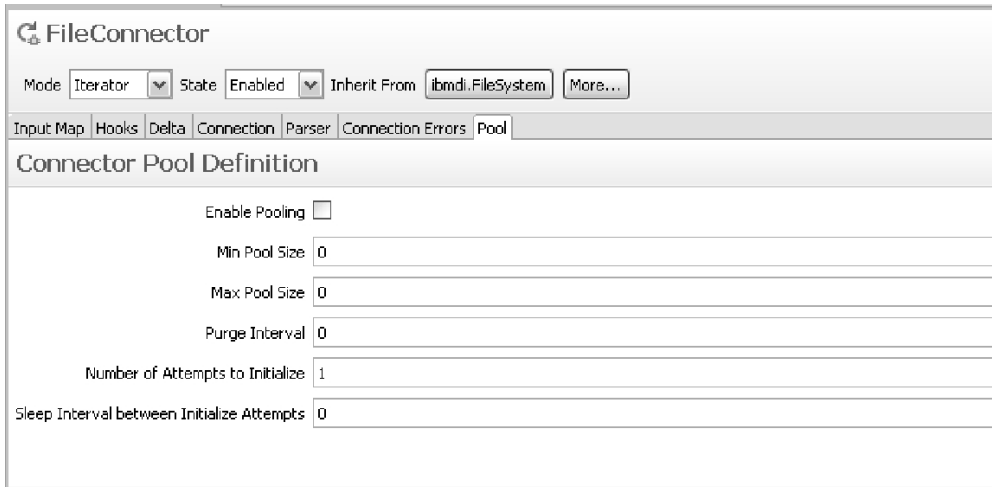


Figura 46. Pestaña Agrupación: Definición de agrupación de conectores

Cuando en una línea de ensamblaje se utiliza un conector perteneciente a una agrupación, la línea de ensamblaje mostrará la pestaña siguiente:

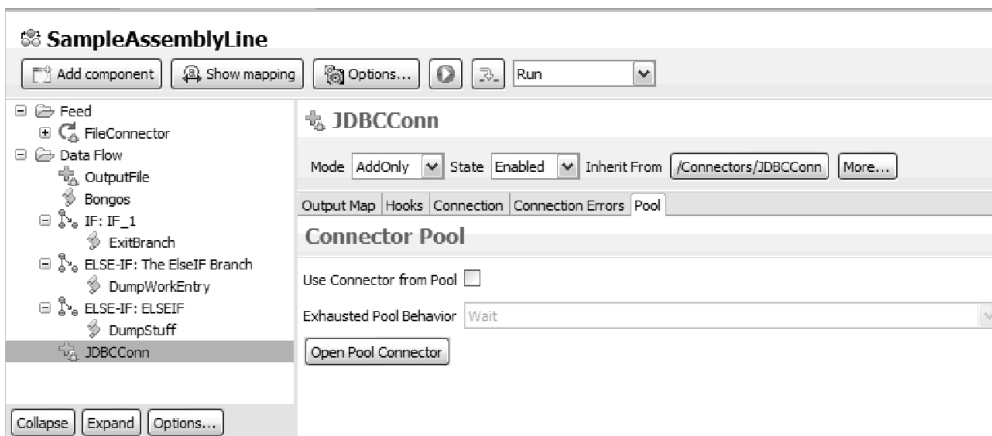


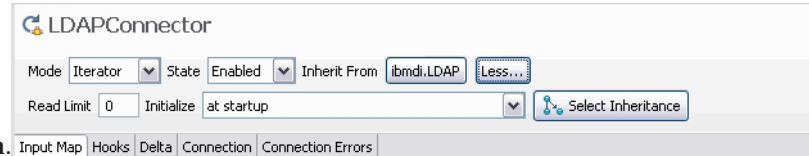
Figura 47. Pestaña Agrupación: Conector de línea de ensamblaje

Utilice el botón **Abrir conector de agrupación** para abrir el conector perteneciente a una agrupación.

Herencia del conector

Además de poder cambiar la herencia en varios lugares del editor, también puede utilizar el botón Herencia del conector para obtener una lista completa de los valores de herencia.

Utilice el botón **Más...** para expandir la cabecera del editor de conectores y pulse el



botón **Herencia**.

El botón **Herencia** presentará un diálogo que muestra todos los valores de herencia del conector:

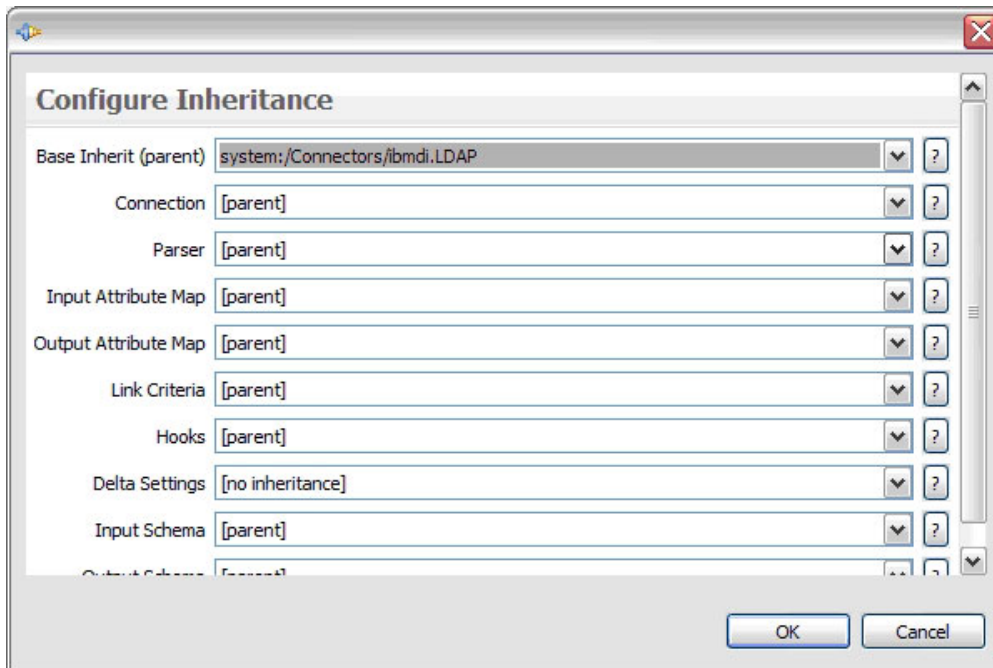


Figura 48. Editor de conectores: Configurar herencia

La notación [padre] indica que el elemento se ha heredado del componente listado en el campo **Herencia base (padre)**.

Editor de servidores

El editor de servidores es el lugar donde define cómo acceder a un servidor de IBM Security Directory Integrator.

El Editor de configuración define siempre un servidor denominado "Predeterminado". En la vista de servidores IBM Security Directory Integrator, se pueden añadir nuevos al proyecto.

Server Document

Create Solution Directory

Server API Address: localhost:1099

Use SSL:

User name:

Password:

Installation directory: C:\Program Files\IBM\TDI\W7.1.1

Solution Directory: C:\TDI_Workspace

ActiveMQ Transport port:

ActiveMQ Management port:

Figura 49. Editor de documentos del servidor

La dirección de API del servidor es la dirección host:port del servidor IBM Security Directory Integrator. Si especifica el directorio de instalación, el Editor de configuración puede iniciar el servidor. Antes de iniciar un nuevo servidor IBM Security Directory Integrator, configure todos los parámetros y utilice el botón **Crear un directorio de solución** para crear los archivos necesarios para el servidor. También puede especificar puertos exclusivos para el transporte y la gestión de Apache ActiveMQ.

Editor de esquemas

El editor de esquemas gestiona archivos de esquemas de diseño.

Estos archivos pueden ser utilizados por correlaciones de entrada y salida en otros editores. El esquema se refiere sólo al tiempo del diseño (es decir, sólo al CE) y se utiliza típicamente cuando se dispone de esquemas de gran tamaño que no deben aparecer en el archivo de configuración del tiempo de ejecución.

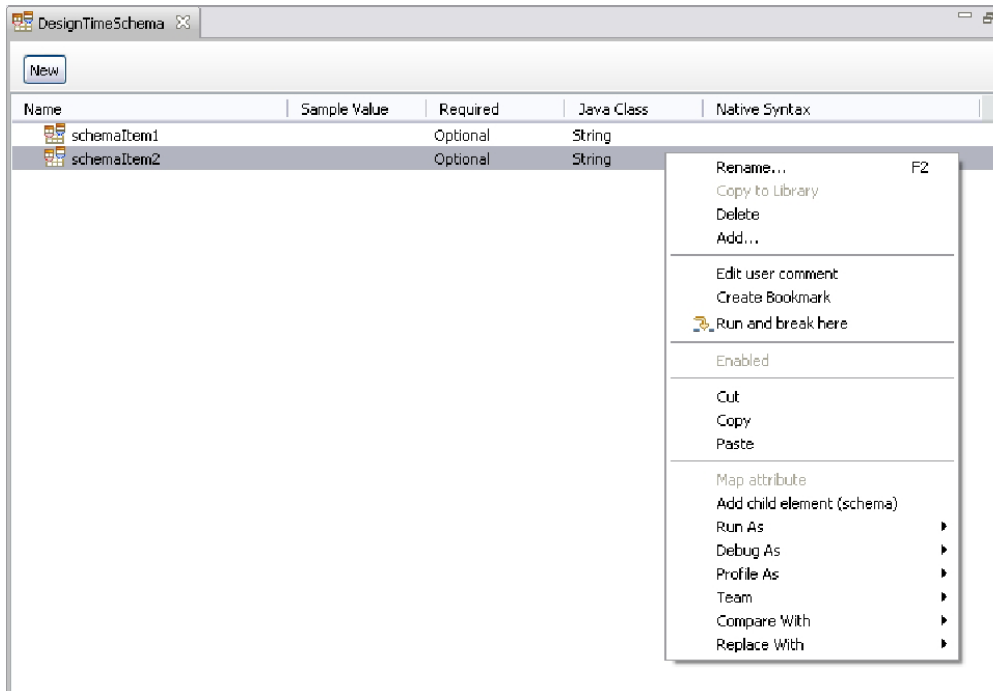


Figura 50. Editor de esquemas

El editor proporciona el botón **Nuevo** para añadir un nuevo elemento de esquema de nivel superior y un menú contextual para operar en elementos de esquema existentes.

Explorador de datos

El explorador de datos proporciona una vista detallada en un sistema de destino. Actualmente tan solo los conectores LDAP y JDBC proporcionan detalles adicionales para un conector. El explorador de datos se abre pulsando el botón derecho del ratón en un conector de la biblioteca o de una línea de ensamblaje.

En el navegador puede pulsar el botón derecho del ratón y seleccionar **Examinar datos** para abrir una nueva ventana del editor en la que puede examinar los datos de la configuración de conexión actual mediante el conector.

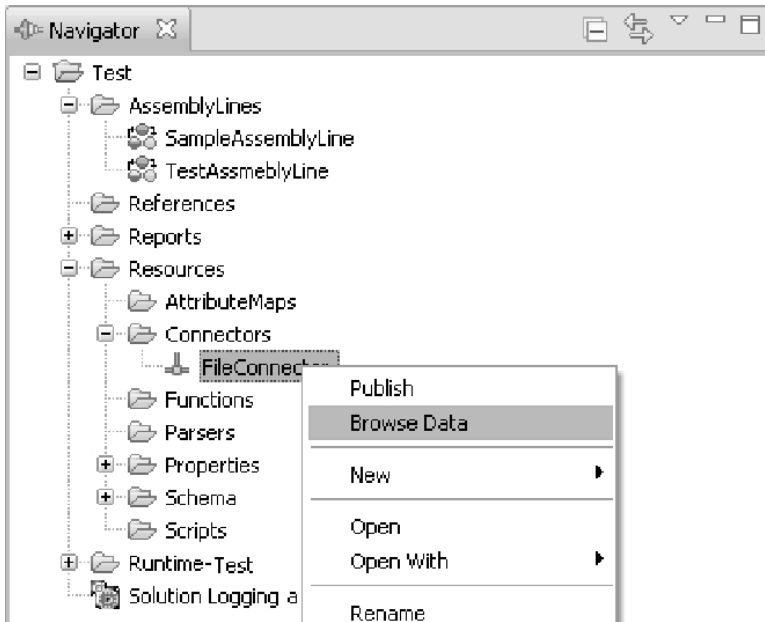


Figura 51. Explorador de datos

En la línea de ensamblaje puede hacer lo mismo y abrir una ventana nueva para el explorador de datos:

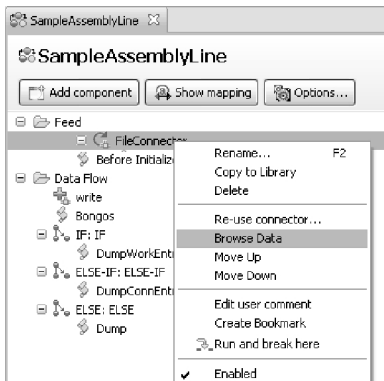


Figura 52. Explorador de datos

Explorador de datos genérico

Es el explorador de datos que se utiliza para los conectores sobre los cuales el Editor de configuración no tiene ningún conocimiento explícito. Proporciona un modo simple de navegar en un conjunto de resultados de un origen de datos.





Figura 53. Explorador de datos genérico

La parte superior muestra una lista de los atributos leídos del conector y una barra de herramientas. Los atributos de la lista tienen recuadros de selección; cuando se selecciona un atributo, se crea una correlación de atributos para dicho atributo utilizando una expresión `conn.attributename -> work.attributename` simple. Cuando se deselecciona un atributo, se elimina la correlación del atributo para dicho atributo.

La barra de herramientas tiene las funciones siguientes:

Tabla 8. Barra de herramientas del explorador de datos

Conmutar todo	Este mandato conmutará los recuadros de selección para cada atributo de la lista. Esto producirá una modificación en la correlación de atributos para todos los atributos listados.
Acumular	Este mandato conmuta independientemente de si la lista de atributos descubiertos se han acumulado o no. Cuando se acumulan atributos, cada registro leído del conector se fusiona con la lista existente de atributos. Cuando no se acumulan, todos los atributos se eliminan antes de que el siguiente registro se muestre en la lista.
	Pulse este botón para cerrar la conexión. Cuando se cierra la ventana del editor para el explorador, la conexión se cierra automáticamente. Normalmente se necesita cerrar la conexión antes de leer el siguiente registro si se han modificado los valores de la conexión en este editor.
	Pulse este botón para leer el siguiente registro del conector. Cuando no se devuelven más registros del conector, se muestra un mensaje a la izquierda de la barra de herramientas para indicar que no hay más entradas del conector. Volver a pulsar este botón después de esta condición hace que el conector empiece a leer desde el principio de su conjunto de resultados.

La parte inferior de la pantalla muestra dos pestañas. La primera pestaña es la pestaña **Detalles** que contiene detalles sobre la selección actual. Para el explorador de datos genérico esta pestaña siempre estará vacía.

La segunda pestaña es la pestaña **Conexión**. Esta pestaña muestra la configuración de conexión para el conector. Puede modificar los parámetros de conexión y guardarlo como hace al abrir el editor de conectores.

Explorador de datos de secuencia

El explorador de datos de secuencia se utiliza cuando un conector utiliza un analizador. El explorador de datos de secuencia primero inicializará el conector e intentará obtener la corriente de entrada y mostrará los primeros 20 K de datos de entrada en la pestaña de detalles.

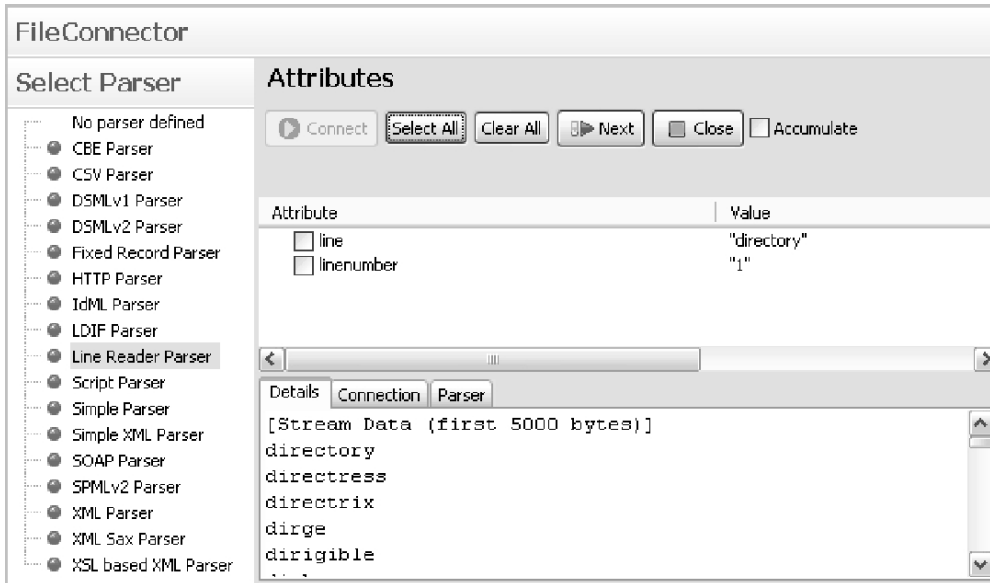


Figura 54. Explorador de datos de secuencia

El lado izquierdo ahora contiene una lista **Seleccionar analizador** en la que puede seleccionar un analizador, para intentar leer el contenido de la corriente de entrada para ver si coincide con sus expectativas. Cuando selecciona un analizador, la pestaña **Analizador** se actualiza con el formulario de configuración para ese analizador. Cada vez que cambie el analizador, el conector se cerrará para que pueda ver con facilidad si un analizador puede interpretar la entrada seleccionando de forma continua un analizador seguido de leer siguiente.

Nota: Cuando selecciona un analizador de la tabla, también modifica la configuración del conector para utilizar ese analizador. Cuando cierra y guarda (o utiliza la función **Archivo > Guardar**), efectivamente actualiza la configuración con los parámetros que ha configurado actualmente.

Explorador de datos JDBC

El explorador de datos JDBC muestra todas las tablas y vistas en el lado derecho. La pestaña de detalles muestra información para el elemento seleccionado de la lista.

Figura 55. Explorador de datos JDBC

La pestaña **Detalles** muestra la información del sistema obtenida desde el objeto de conexión JDBC. La vista de árbol de la izquierda también muestra todas las tablas y vistas con sus columnas como entradas hijo. Cuando se selecciona una tabla o una vista, la pestaña de detalles también se llenará con la sintaxis para la toda la tabla. Por ejemplo, si selecciona la tabla "IDI_PS_DEFAULT", debería ver algo similar a:

Figura 56. Detalles de Tabla de JDBC

Si selecciona una columna de una tabla o una vista, verá los detalles únicamente para esa columna.

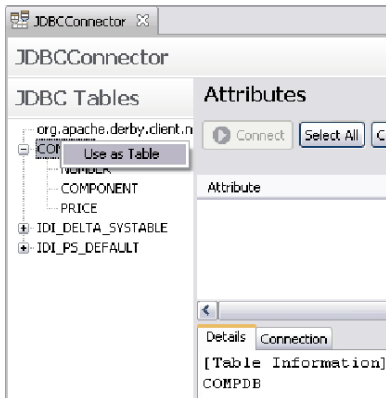



Figura 57. Opción "Utilizar como tabla"

También puede pulsar el botón derecho del ratón en un nombre de tabla y utilizar la función **Utilizar como tabla** para actualizar el parámetro **Nombre de tabla** de configuración del conector JDBC.

El botón  de la barra de herramientas de la cabecera **Tablas de JDBC** realiza un redescubrimiento de la conexión. Sólo debe utilizarlo si el descubrimiento inicial falla o si ha cambiado la dirección URL de JDBC en la pestaña de conexión.

Explorador de datos LDAP

El explorador de datos LDAP muestra el esquema y los prefijos de contexto (bases de búsqueda) que el servidor LDAP proporciona.

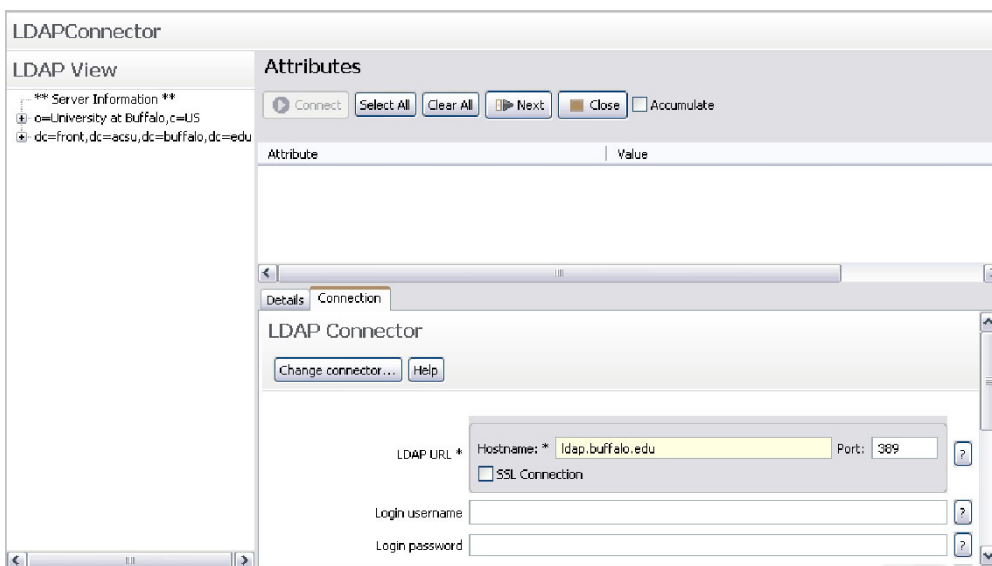


Figura 58. Explorador de datos LDAP

La Vista de LDAP contiene información del servidor y las bases de búsqueda que el servidor LDAP proporciona. Según la selección que se realice de este árbol, se ven resultados diferentes. Si selecciona uno de los nodos que no son de tipo

esquema, verá un volcado detallado para la entrada específica en la pestaña de detalles como se muestra a continuación:

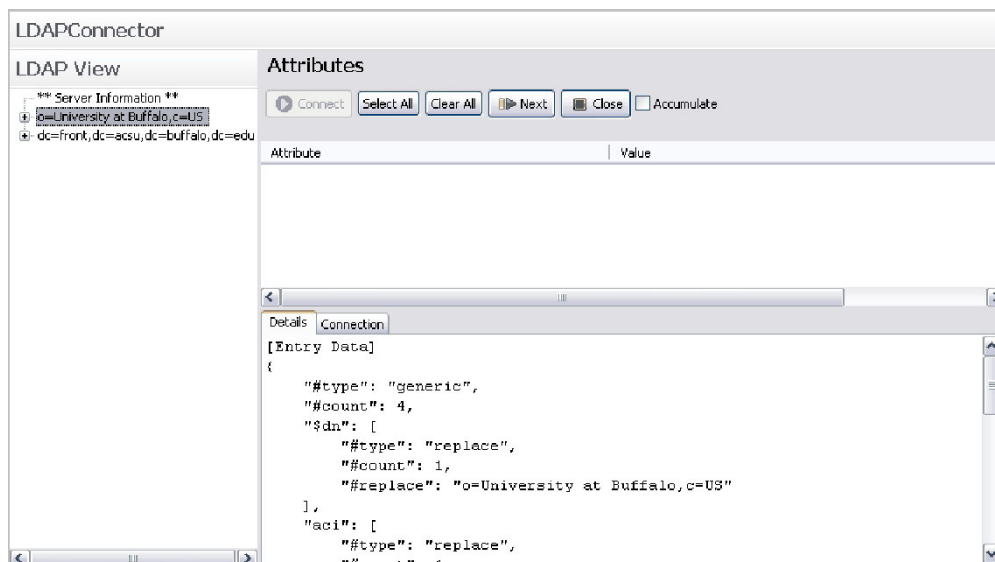


Figura 59. Entrada del explorador de datos LDAP

Cuando se selecciona un elemento de esquema, se muestran los detalles en la pestaña de detalles y además la lista de atributos se actualiza con la información de ese elemento de esquema. Esto es muy útil si va a leer o escribir en un esquema específico:

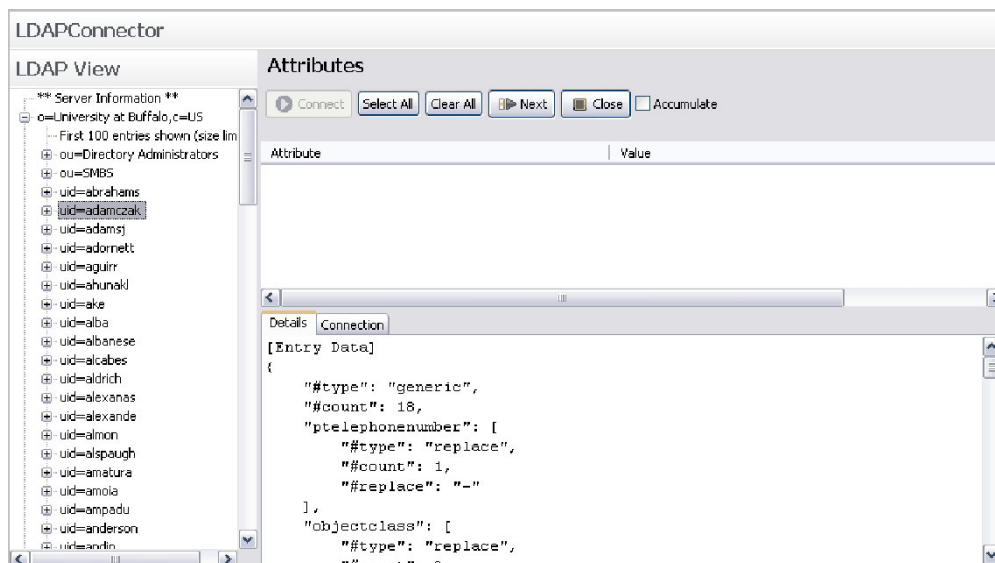


Figura 60. Elemento de esquema del explorador de datos LDAP

La selección de una clase de objeto del nodo de esquema le permite crear rápidamente una correlación de atributos para dicha clase. Ahora la columna de valores contiene información sobre el atributo. El valor "MAY" significa que es opcional, mientras que "MUST" significa que es necesario (al añadir entradas). El valor entre paréntesis muestra la clase de objeto que define el atributo; las clases de objeto LDAP son jerárquicas.

También puede actualizar rápidamente el parámetro **Base de búsqueda** del conector LDAP seleccionando **Utilizar como base de búsqueda** en el menú contextual.

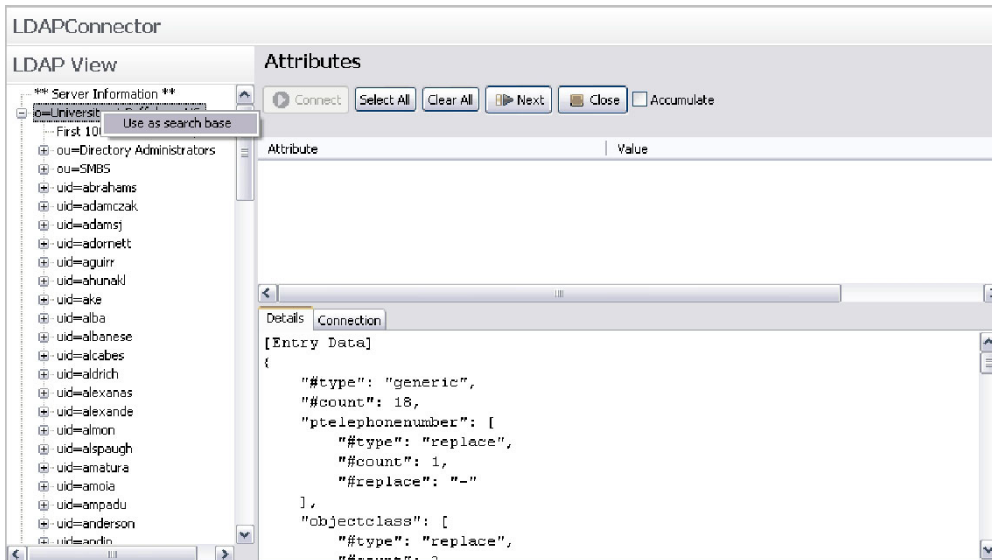


Figura 61. Opción de menú contextual "Utilizar como base de búsqueda"

Editor de formularios

El editor de formularios se utiliza para personalizar el formulario de parámetros de conexión para un componente. Esto sólo puede aplicarse a componentes de la carpeta Recursos (excepto los archivos de propiedades).

Para personalizar un formulario, debe abrir el componente con el *Editor de formularios*.

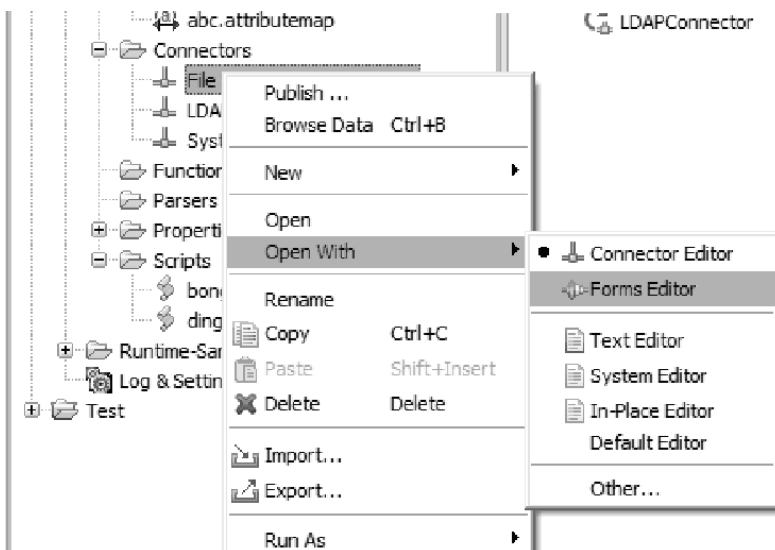
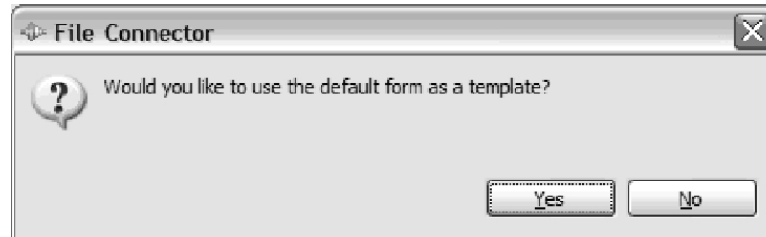


Figura 62. Menú contextual - opción Editor de formularios

Observe que cuando selecciona un editor distinto del editor predeterminado para un archivo, el CE recordará esta opción, de modo que la próxima vez que efectúe una doble pulsación en el archivo, el archivo se abrirá con el editor que ha

utilizado la última vez. Para abrir el componente con el editor predeterminado, simplemente seleccione el editor adecuado en este menú (normalmente el editor que está en primer lugar).

Si el componente que abre no tiene ningún formulario personalizado, se le solicitará que rellene el formulario con el formulario predeterminado:



Si selecciona **Sí** se creará una definición de formulario inicial basada en el formulario predeterminado para el componente. En este caso se utiliza el conector del sistema de archivos en el ejemplo y produce la siguiente pantalla:

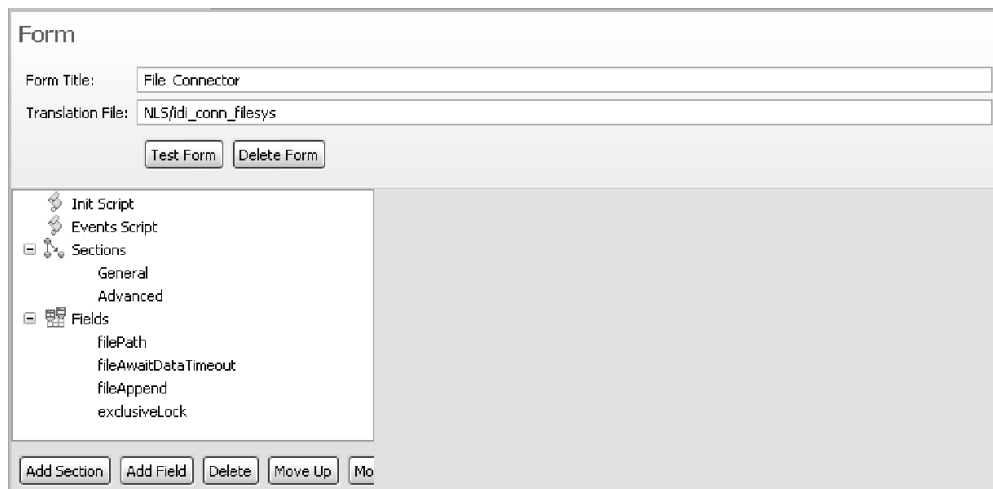


Figura 63. Pantalla Editor de formularios predeterminado para el conector del sistema de archivos

Los distintos elementos que aparecen en este formulario tienen la función siguiente:

Título del formulario

Es el título principal del formulario del componente (déjelo en blanco para no tener ningún título). Es lo que el usuario ve en la parte superior del formulario personalizado.

Archivo de traducción

Es el archivo que se utiliza para traducir las etiquetas y la información sobre herramientas del formulario. Si se define un archivo de traducción, se intenta traducir todas las etiquetas y toda la información sobre herramientas. Si crea una etiqueta con "file_name" como texto, la traducción intentará recuperar una serie del archivo de traducción utilizando "file_name" como clave. Los mismos archivos de traducción son archivos de propiedades simples con "key=value" en cada línea.

Para localizar un formulario debe crear archivos con el identificador de entorno local. Por lo tanto, para una traducción al francés deberá crear un archivo llamado *nombre-base_fr.properties*.

Formulario de prueba

Este botón mostrará una ventana de diálogo con la definición de formulario actual.

Suprimir formulario

Este botón suprimirá el formulario del componente. Debe cerrar y seleccionar **guardar** para que su efecto sea permanente.

Script de inicialización

El script de inicialización se ejecuta cuando se carga el formulario. Es donde se sitúa el código para inicializar el estado del formulario y cualquier variable de script global para el formulario.

Script de sucesos

Cuando el valor de un campo cambia, el formulario ejecutará un manejador de sucesos definido en este script. Cada campo tiene un nombre interno que el componente utiliza. Por ejemplo, FileConnector utiliza "filePath" como nombre interno para el parámetro **File path**. Puede ver todos los nombres de parámetros de componente cuando elige rellenar el formulario con el formulario predeterminado en la sección Campos. Como reacción a los cambios realizados en el formulario, deben escribirse manejadores de sucesos en el editor de scripts de sucesos utilizando el nombre interno con el sufijo "_changed" como nombre de función de script. A continuación se muestra un ejemplo del conector LDAP que inhabilita dos campos basándose en el método de autenticación:

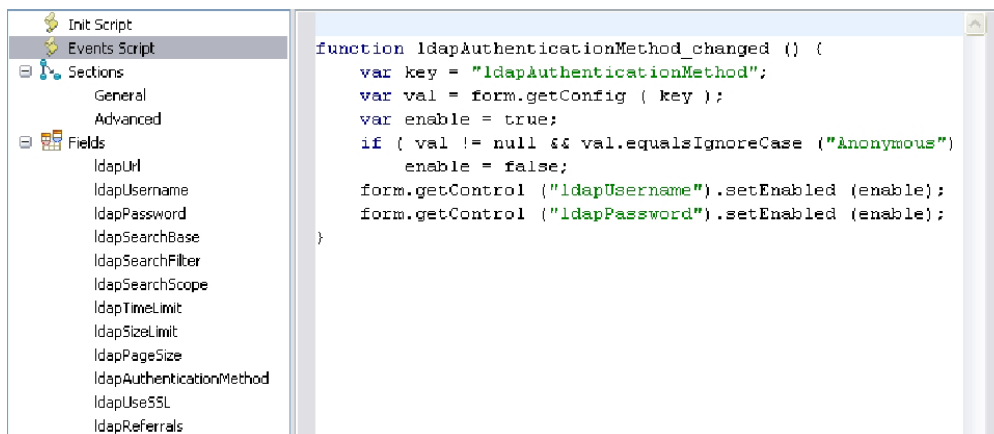


Figura 64. Editor de formularios, script de sucesos en el conector LDAP

Secciones y campos

El formulario está formado por secciones y campos. Para gestionar secciones y campos, puede añadir, eliminar y reordenar utilizando la barra de herramientas situada debajo del árbol.



- **Añadir sección** – añade una sección nueva y vacía al formulario.
- **Añadir campo** – añade un campo nuevo al formulario. Los nombres de campo deben ser exclusivos.

- **Suprimir** – elimina una sección o un campo del formulario.
- **Mover hacia arriba/hacia abajo** – reorganiza el orden de las secciones y los formularios. Cuando hay secciones definidas, el orden de los campos lo define la sección no la lista de campos. Cuando no hay secciones definidas, el orden del árbol determina el orden de los campos del formulario.

Secciones

Esta parte es opcional. Si no especifica secciones, el formulario mostrará todos los campos en su formulario, todos a la vez.

Las secciones se utilizan para organizar los campos del formulario. Las secciones son como carpetas que el usuario puede expandir o contraer para mostrar/ocultar su contenido. Cuando se define una sección, debe especificar si la sección se expande inicialmente y qué campos mostrará la sección. En el ejemplo del conector del sistema de archivos, hay dos secciones.

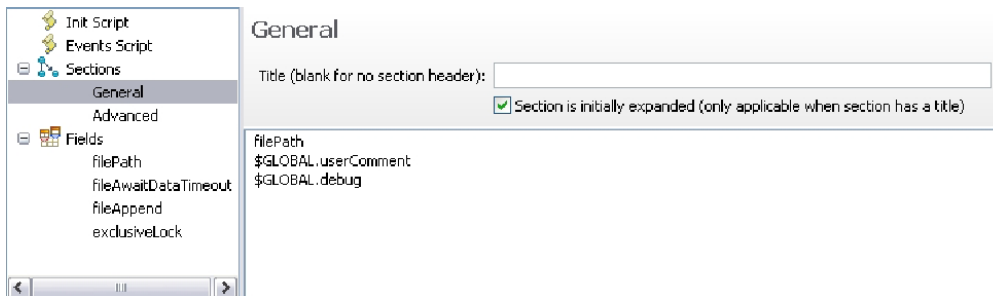
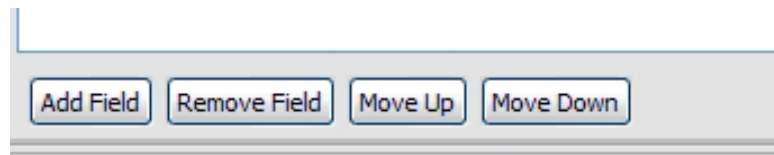


Figura 65. Editor de formularios - sección General

La primera sección es la sección *General*. Esta sección no tiene título, lo que significa que no habrá ninguna cabecera de sección en la que el usuario puede pulsar para expandir o contraer el contenido. Esto hace que sea una sección estática puesto que no se puede ni contraer ni expandir. Puede añadir, eliminar y reordenar los campos de la lista de campos utilizando la barra de herramientas situada al final del panel:



La segunda sección es la sección *Avanzado*:

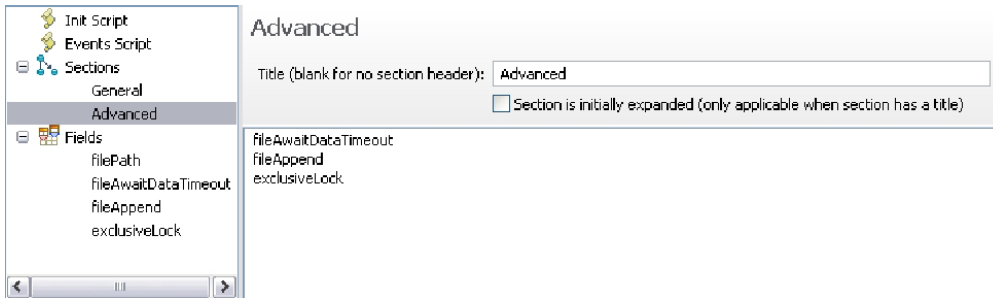


Figura 66. Editor de formularios - sección Avanzado

Esta sección tiene un título, pero no se expande inicialmente. Esto hará que el formulario muestre el formulario con el título de sección contraído y los campos que contiene están ocultos hasta que el usuario expande la sección.

Campos

Los campos son los parámetros para el componente. Si vuelve a utilizar un componente como conector del sistema de archivos debe proporcionar los parámetros necesarios en el formulario o establecer el parámetro en la sección “Script de inicialización” para que el componente pueda funcionar correctamente. Cada campo muestra su definición cuando se selecciona.

Figura 67. Editor de formularios - definiciones de campos

Este panel muestra la definición del parámetro de conector “filePath”. Por orden de aparición:

Tabla 9. Editor de formularios - definición de parámetro

Campo	Descripción
Etiqueta	La etiqueta que el formulario mostrará.
Información sobre herramientas	La información sobre herramientas que se muestra cuando el usuario mueve el ratón sobre el campo de entrada.

Tabla 9. Editor de formularios - definición de parámetro (continuación)

Campo	Descripción
Tipo de campo	<p>El tipo de campo de entrada:</p> <ul style="list-style-type: none"> • Serie para entrada de texto de una sola línea • Desplegable (editable) para un campo de entrada desplegable editable • Desplegable (no editable) para un desplegable con un conjunto fijo de selecciones • Booleano para un recuadro de selección • Área de texto para un campo de entrada de texto de varias líneas • Texto estático para mostrar texto simple (es decir, no entrada) • Contraseña para un campo de entrada protegido de una sola línea • Editor de scripts para editar scripts • Componente personalizado para un control SWT/JFace definido por el usuario
Selección de modalidad	<p>Este campo opcional puede especificar las modalidades de componente en las que se excluye o incluye el campo. Especifique las modalidades separadas con una coma y con un signo menos para excluir.</p> <p>“Iterador” – sólo mostrar en modalidad Iterador “-Iterador” – no mostrar en modalidad Iterador “Iterador,Buscar” – sólo mostrar en modalidad Iterador y Buscar</p>

Las tres pestañas situadas en la parte inferior le permiten especificar, botones, valores desplegables para listas desplegables y el nombre de clase Java para el componente personalizado.

Asistentes

Existen varios asistentes (procedimientos por pasos asistidos gráficamente) en el Editor de configuración de IBM Security Directory Integrator. Son los siguientes:

1. “Asistente Importar configuración”
2. “Nuevo asistente de componente” en la página 150
3. “Características del formulario de configuración del conector” en la página 155

Asistente Importar configuración

Puede importar archivos de configuración de versiones anteriores utilizando el asistente Importar configuración.

En este asistente el usuario elige el proyecto de destino y qué componentes desea importar.

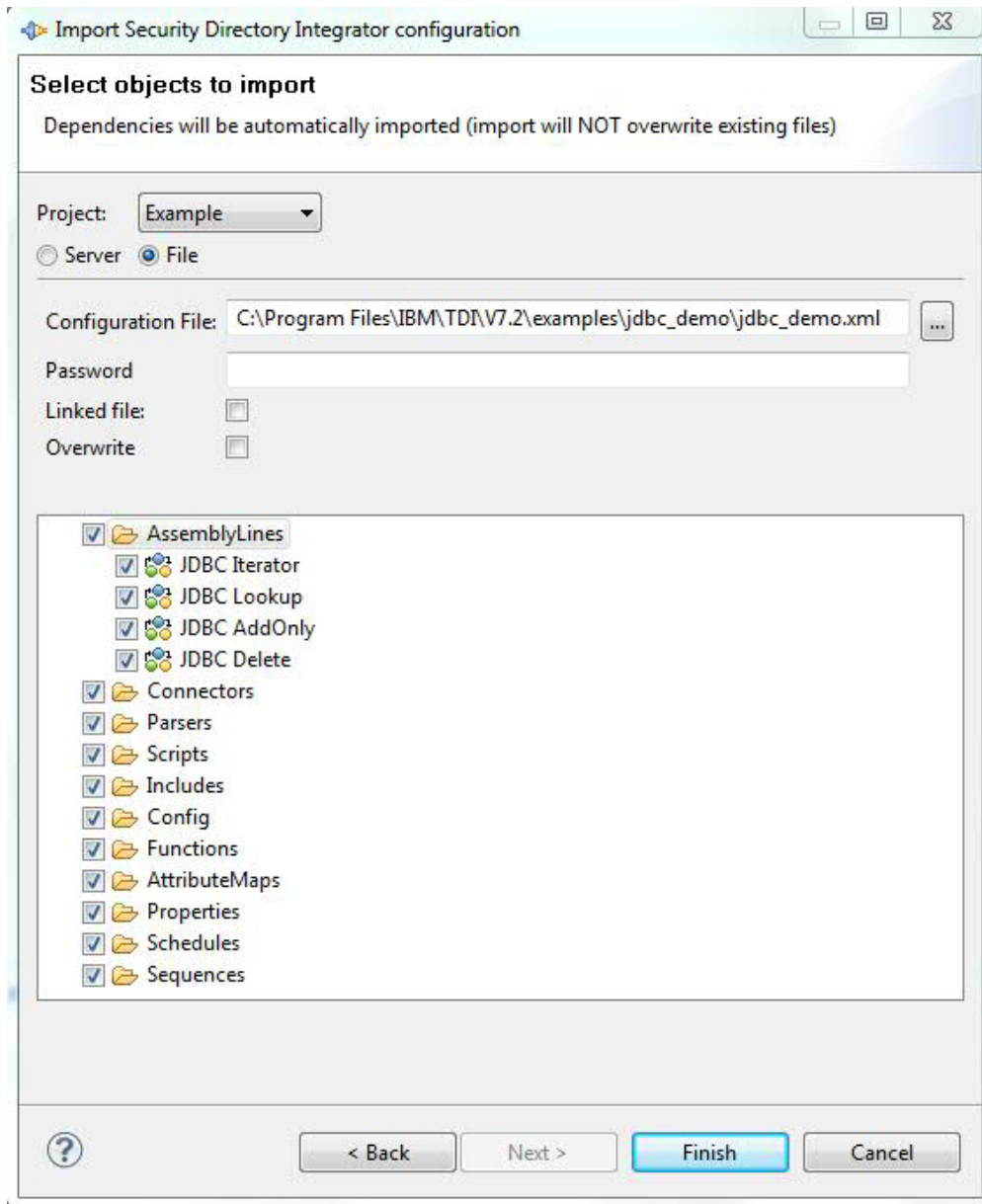


Figura 68. Asistente Importar configuración

De forma predeterminada, se seleccionan todos los componentes para importar. Sin embargo, puede seleccionar únicamente los que le interesen. Si selecciona una única línea de ensamblaje que utiliza conectores en el archivo de configuración, estos conectores también se importarán automáticamente.

El campo de entrada **Proyecto** es el proyecto de destino al que se importa la configuración. Seleccione la opción en blanco para crear un nuevo proyecto.

El campo de entrada **Archivo de configuración** es el archivo de configuración que se va a importar. Si la configuración está protegida mediante contraseña, escriba la contraseña en el campo de entrada **Contraseña**.

Si se selecciona el campo **Archivo enlazado**, cualquier cambio realizado en el proyecto importado se vuelve a grabar en el archivo desde donde se ha importado el proyecto. Puede cambiar este valor en las propiedades del proyecto borrando o

cambiando el nombre del archivo del campo **Archivo enlazado**, tal como se ilustra a continuación:

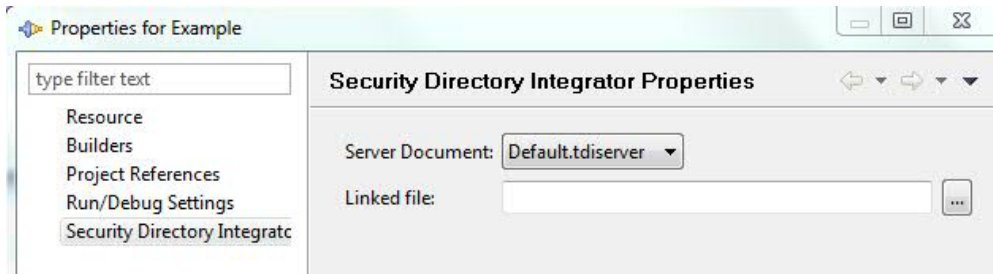


Figura 69. Campo Archivo enlazado

También puede importar configuraciones a partir de servidores. Conmute la vista del servidor seleccionando el botón de selección **Servidor**.

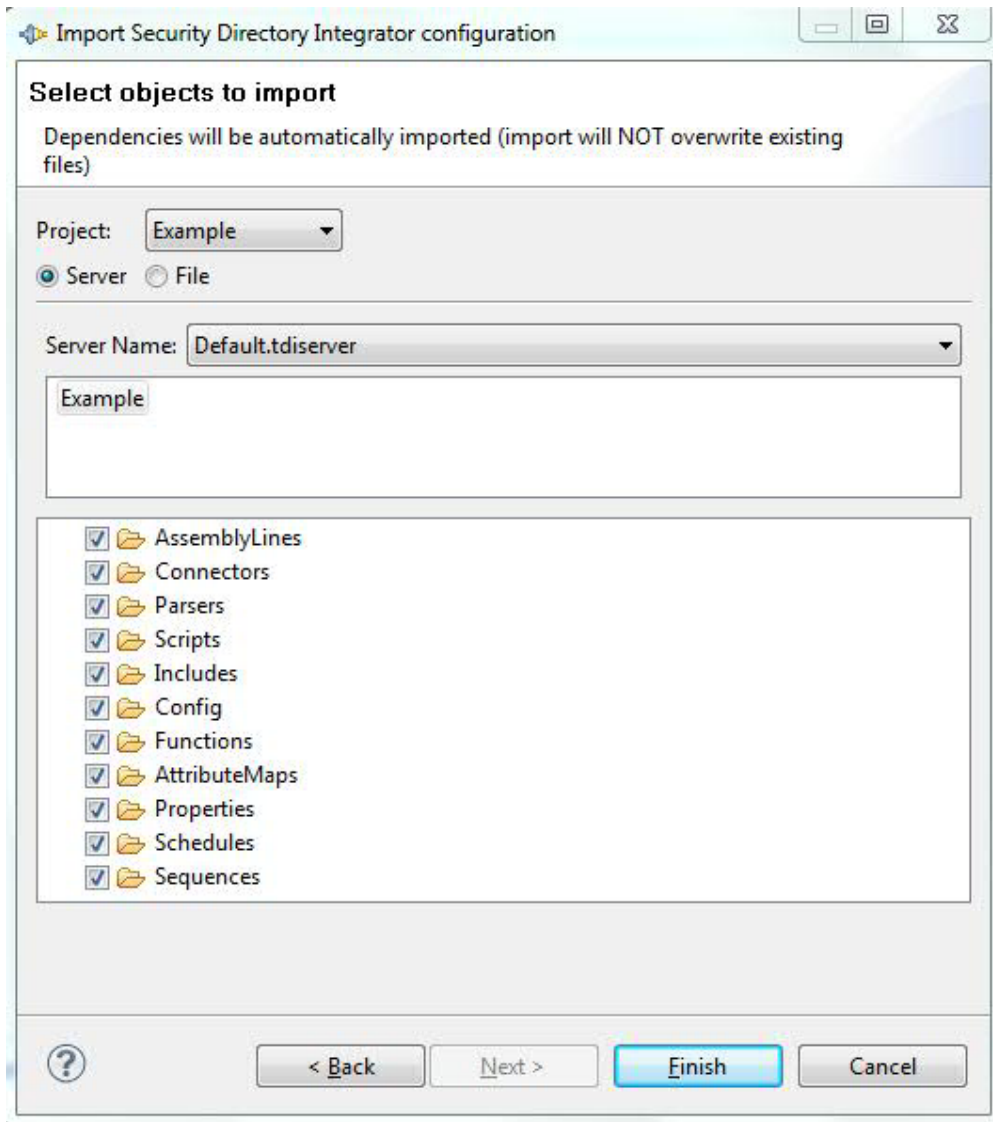


Figura 70. Asistente Importar desde servidor

Para iniciar este tipo de importación, seleccione el servidor de la lista de servidores en el campo **Nombre de servidor**. Una vez que haya seleccionado un servidor, se mostrará la lista de configuraciones en la lista debajo del nombre del servidor. Puesto que se trata de una operación de la red, es posible que algunos controles se inhabiliten durante la actualización de la lista. De la lista de configuraciones, seleccione la configuración que desea importar. Al seleccionar una configuración, se descarga del servidor y rellena el árbol siguiente con su contenido para que pueda seleccionar los componentes que incluirá en la importación.

Nuevo asistente de componente

Este asistente se invoca cuando se utiliza **Añadir componente** en una línea de ensamblaje o cuando se utiliza **Archivo > Nuevo...** en la barra de menús principal.

Cuando crea un nuevo componente en la carpeta Recursos, obtiene un diseño del asistente ligeramente distinto. Por ejemplo, para un nuevo conector, el asistente es similar a éste:

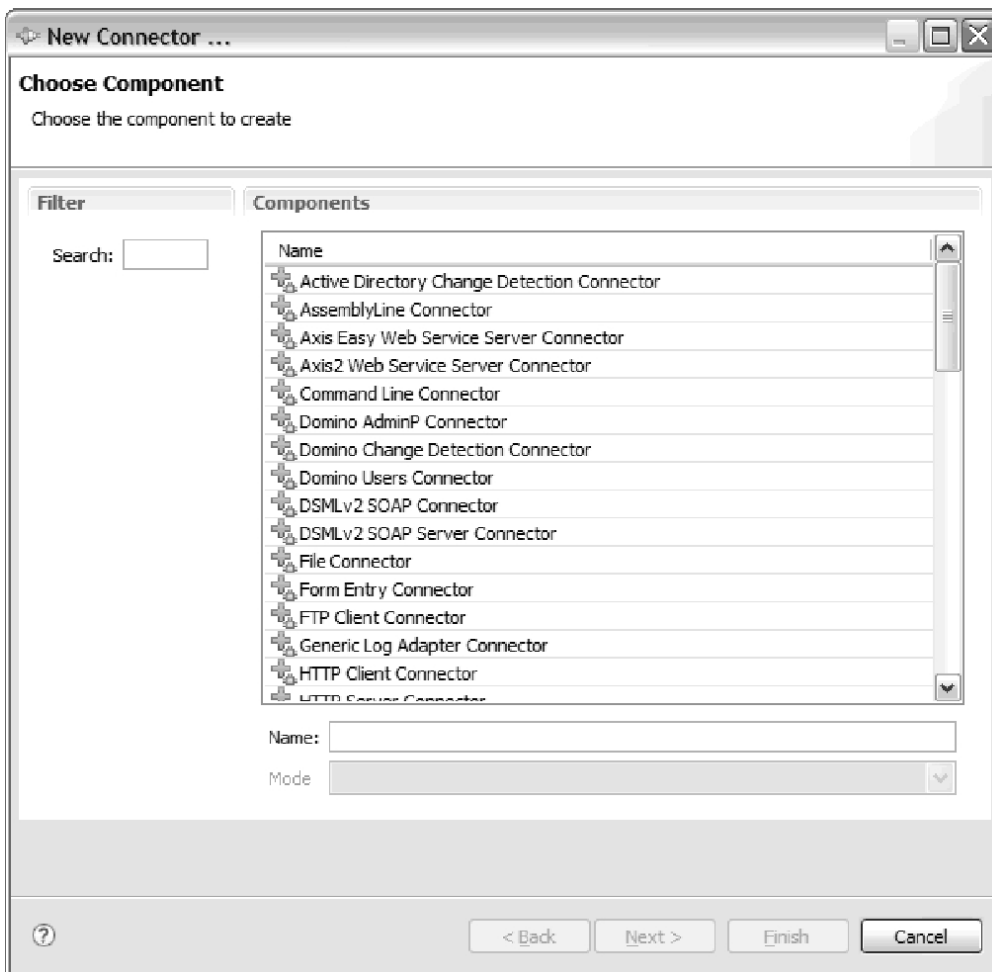


Figura 71. Asistente Nuevo conector en la carpeta Recursos

El nombre de componente se utilizará como nombre de archivo sugerido en la carpeta; es recomendable cambiar este nombre por otro significativo.

Cuando crea un nuevo componente en una línea de ensamblaje, obtiene muchas más opciones en este asistente.

La primera página del asistente es la página de selección del tipo de componente. En esta página seleccionará el tipo de componente que desea añadir o crear. El lado izquierdo contiene una lista de filtros que seleccionarán los componentes pertinentes basándose en la etiqueta de la lista.

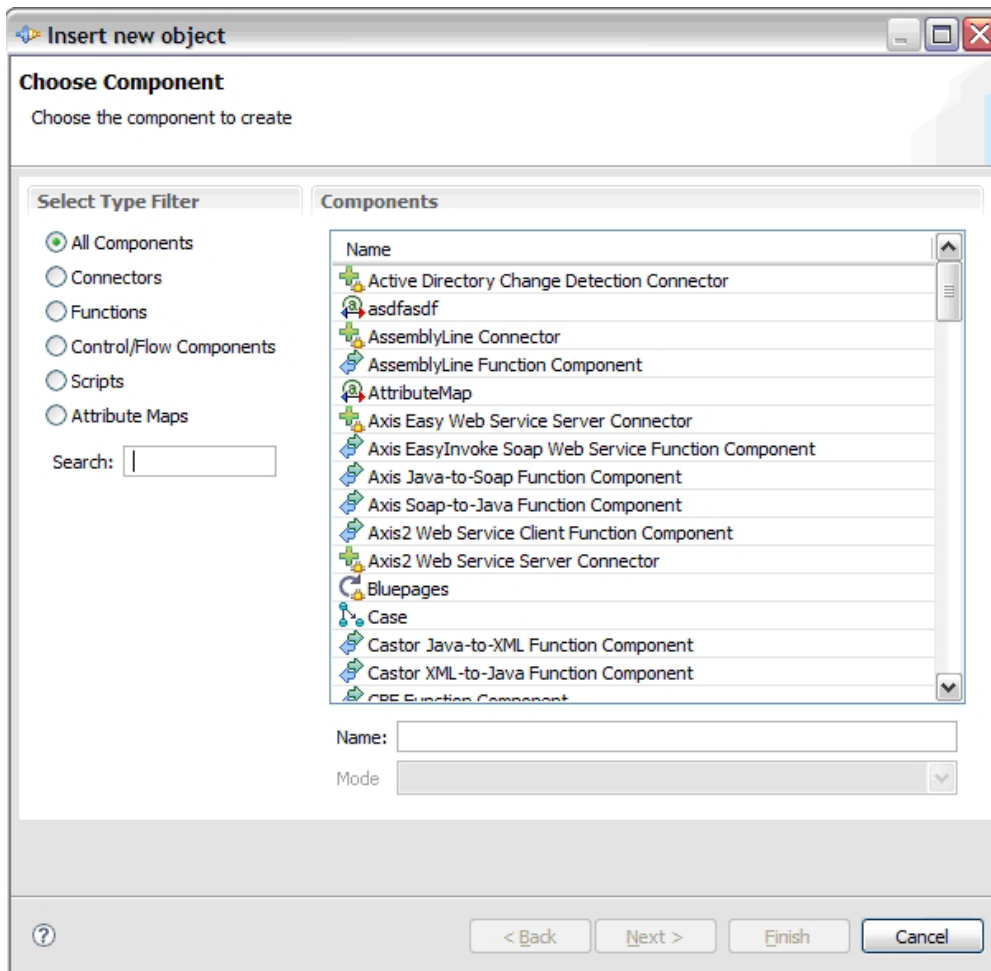


Figura 72. Asistente para Componente nuevo

Puede filtrar el contenido escribiendo en el campo de búsqueda. A medida que escriba, la lista se comparará con lo que ha escrito en el campo de búsqueda (se ignoran las mayúsculas y minúsculas en la coincidencia).

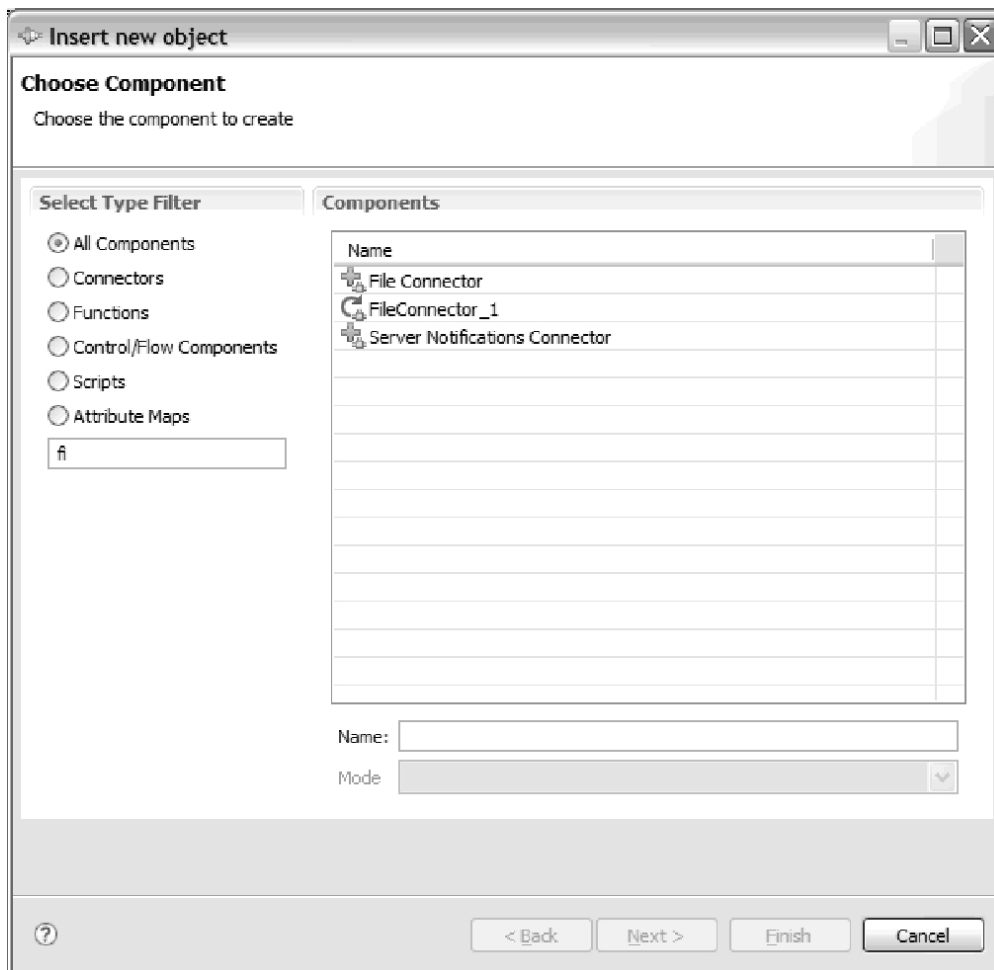


Figura 73. Asistente para Componente nuevo, con filtro

En este momento puede seleccionar finalizar el asistente y el componente se insertará en la línea de ensamblaje.

Si selecciona un conector, se mostrarán una serie de formularios para definir correctamente el conector. Para todos los demás tipos, sólo puede finalizar el asistente y configurar el componente en la línea de ensamblaje.

Después de seleccionar el tipo se muestra el panel Configuración de conector.

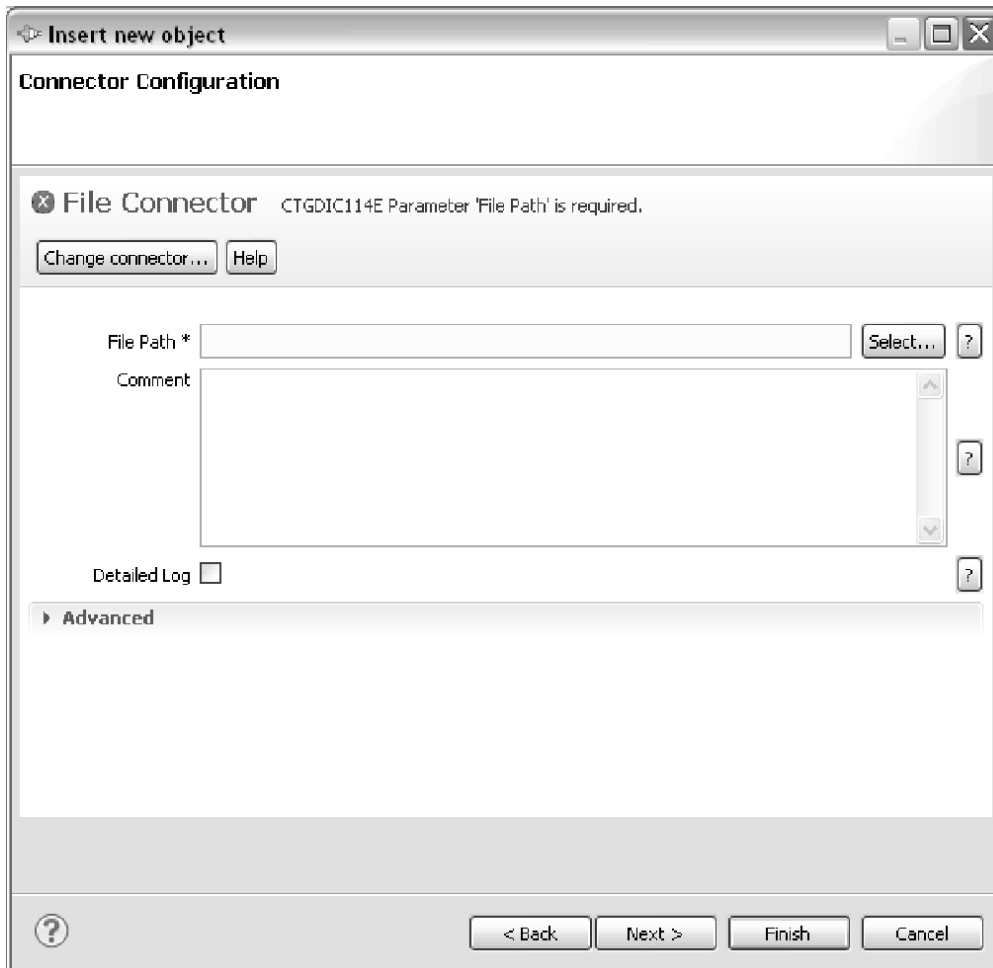


Figura 74. Panel Configuración de conector

Consulte también “Características del formulario de configuración del conector” en la página 155 para obtener información específica para completar este tipo de formulario.

El paso siguiente muestra la configuración del analizador si el conector puede utilizar uno.

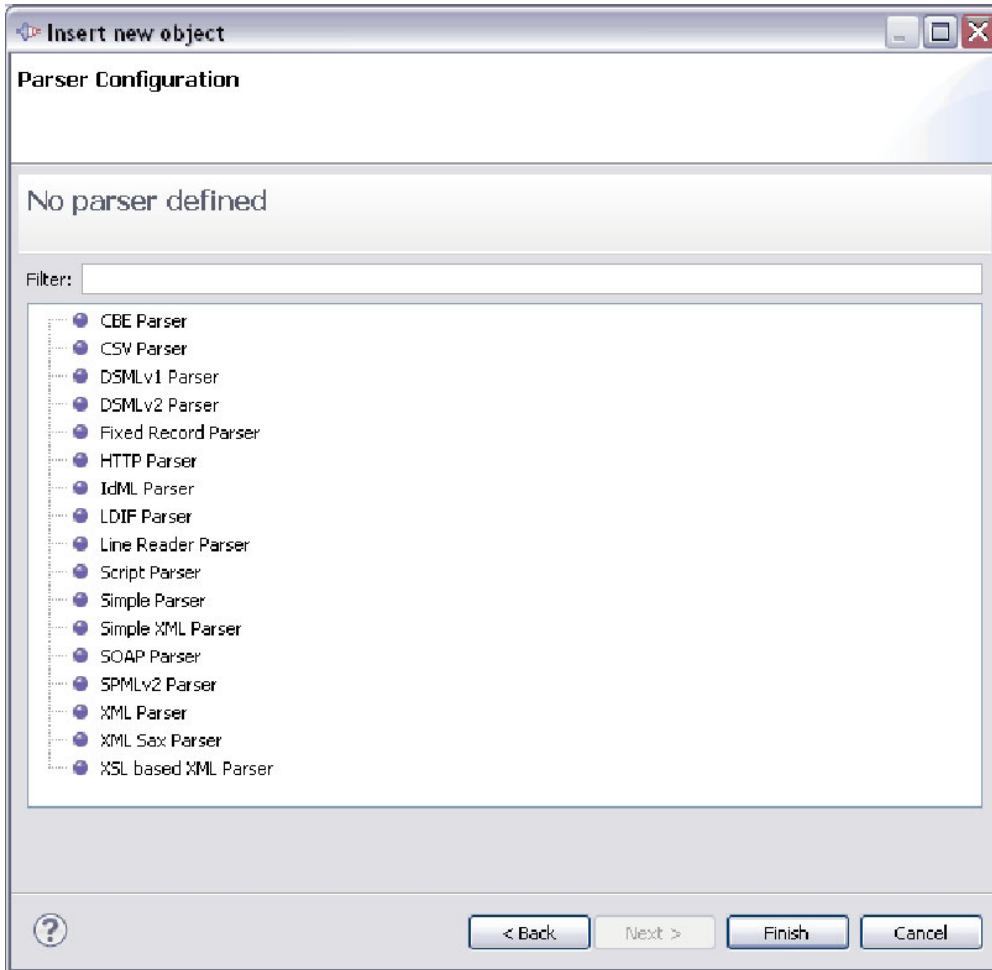


Figura 75. Panel de configuración del analizador

Utilice el botón **Seleccionar analizador** para seleccionar el analizador para el componente:

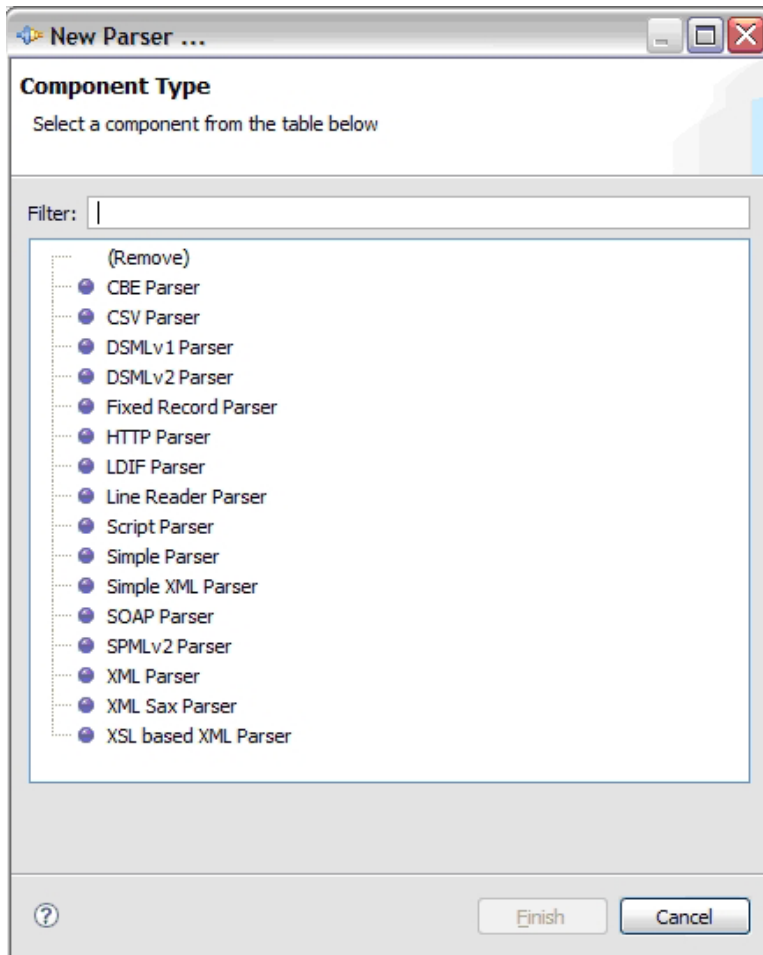


Figura 76. Diálogo Selección de analizador

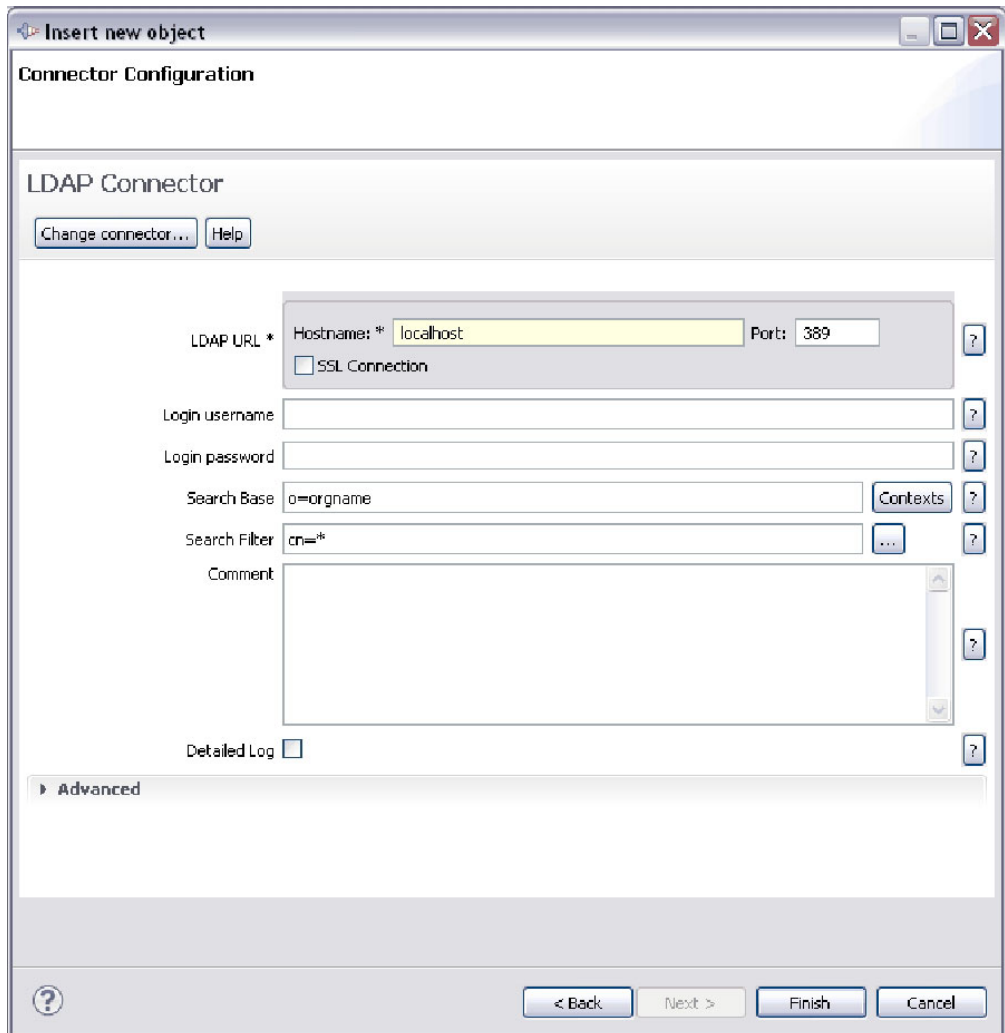
Puede eliminar el analizador actual del componente seleccionando la opción "(Eliminar)".

Características del formulario de configuración del conector

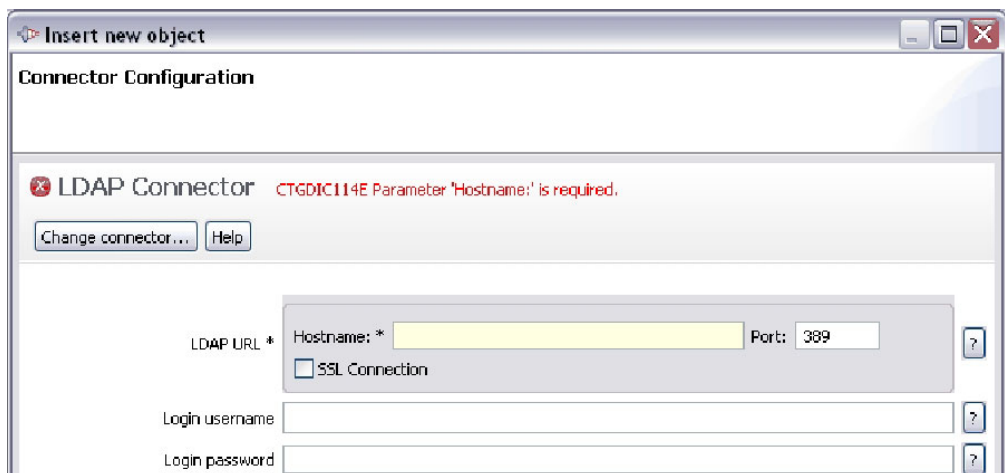
Se pueden agrupar los campos de un formulario en secciones opcionales con título, y los campos tienen una propiedad que especifica si el campo es necesario.

En la ventana Configuración del conector, los campos necesarios se indican mediante un * seguido del nombre del campo.

Los valores que se heredan tendrán una etiqueta azul.



En el ejemplo anterior, la configuración del conector LDAP muestra dos secciones con un campo necesario. Cuando un campo necesario no tiene ningún valor, el formulario lo marcará en el título.



Al pasar el ratón por encima del icono o texto rojo, verá los mensajes de error y a qué campos hacen referencia los errores. Esto es útil cuando existe más de un error en el formulario.

Ejecución y depuración de líneas de ensamblaje

El Editor de configuración dispone de varios mecanismos para ayudarle a desarrollar líneas de ensamblaje, que incluyen recursos para probar y depurar su lógica.

Informes de la línea de ensamblaje

Los informes de la línea de ensamblaje se pueden ejecutar desde el menú contextual del navegador.

Pulse con el botón derecho del ratón sobre una línea de ensamblaje y seleccione el submenú **Crear informe de línea de ensamblaje**. Este menú contiene todas las plantillas de informe que se encuentran en el directorio `dir_instalación_TDI/XSLT/ConfigReport`.

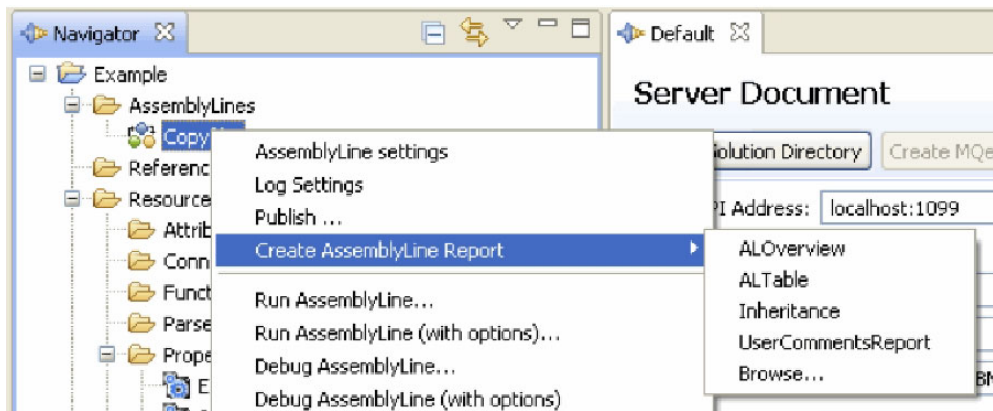


Figura 77. Mandato "Crear informe de línea de ensamblaje"

Seleccione la opción **Examinar...** para examinar el sistema de archivos local para una plantilla de informe.

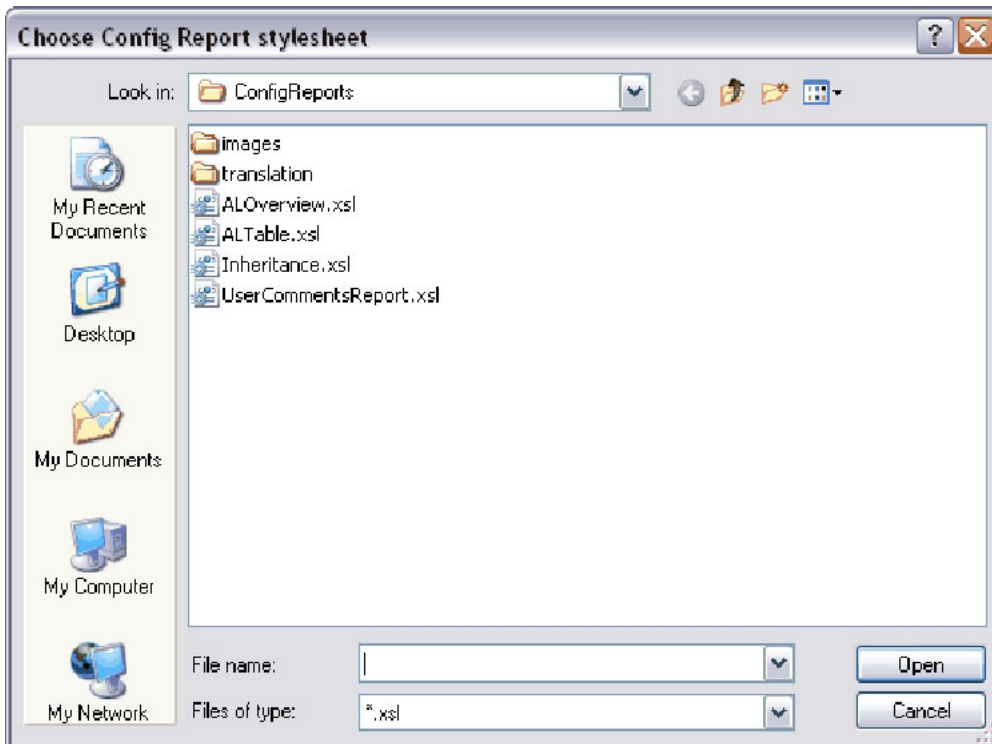


Figura 78. Diálogo Elegir hoja de estilos de informe de configuración

Al seleccionar una y pulsar **Abrir**, el informe se generará y se colocará en el directorio Reports del proyecto como puede verse en la siguiente imagen. El editor asociado con la extensión de archivo *.html* se abre para ver el informe, normalmente el navegador de Internet predeterminado.

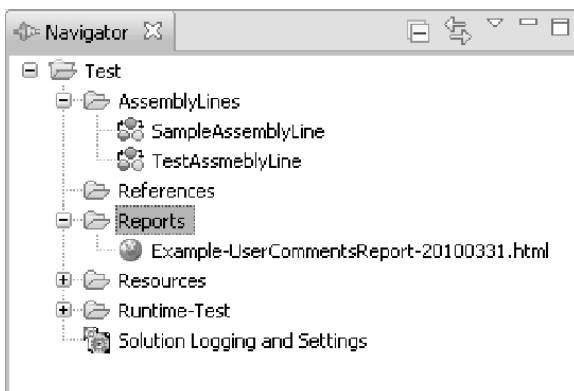


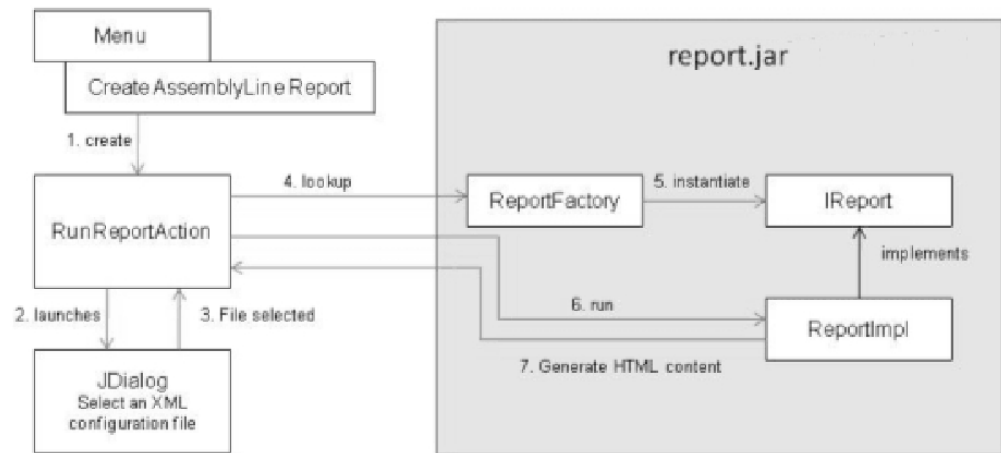
Figura 79. Carpeta Informes en la jerarquía de proyectos

El informe de la línea de ensamblaje genera nombres de archivo de informe basados en la línea de ensamblaje con la fecha actual insertada.

Descripción general de los informes de línea de ensamblaje basados en XML

Puede generar un informe del elemento de configuración seleccionado en función de una hoja de estilo de informes o del archivo XML de configuración de informes especificado. El diagrama siguiente muestra la arquitectura de los informes de

línea de ensamblaje basados en XML.



Formato de archivo XML de configuración de informes

El archivo XML de configuración de informes de línea de ensamblaje tiene el siguiente formato:

```
<tdiReport>  
<reportClass>com.ibm.di.report.aloverview.AssemblyLineOverview</reportClass>  
  <reportConfig>  
    <report specific configuration>  
  </reportConfig>  
</tdiReport>
```


El archivo XML de configuración tiene los siguientes elementos:

- **reportClass** - especifica el nombre de la clase Java del informe.
- **ReportFactory** - crea una instancia de una instancia del informe.
- **reportConfig** - contiene parámetros específicos de informe.
-

Ejecución de la línea de ensamblaje

Cuando se desarrolla la línea de ensamblaje se puede probar ejecutándola hasta completar o paso a paso a través de los componentes, uno a uno.

Hay dos botones para ejecutar la línea de ensamblaje. El primer botón (icono de

reproducción, ) inicia la línea de ensamblaje y muestra la salida en la vista de la consola. El segundo botón (**Depurador**) ejecuta la línea de ensamblaje con el depurador.

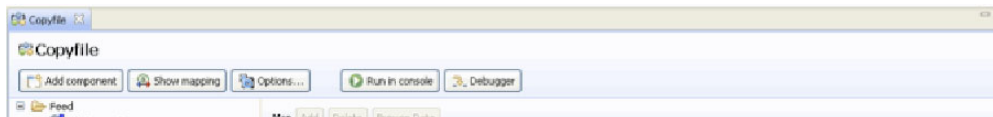
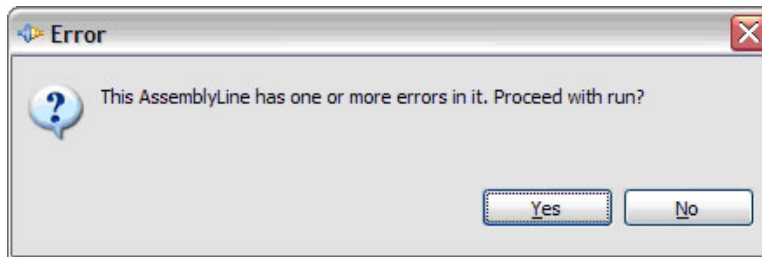


Figura 80. Tres opciones para iniciar una línea de ensamblaje

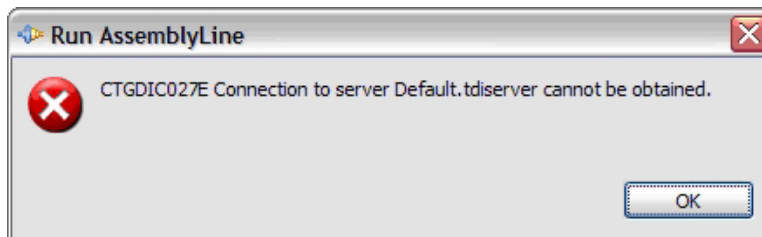
El proceso de iniciar una línea de ensamblaje se realiza en tres pasos.

Si la línea de ensamblaje contiene errores (tales como correlaciones de salida faltantes), se le pedirá que confirme la ejecución de la línea de ensamblaje:

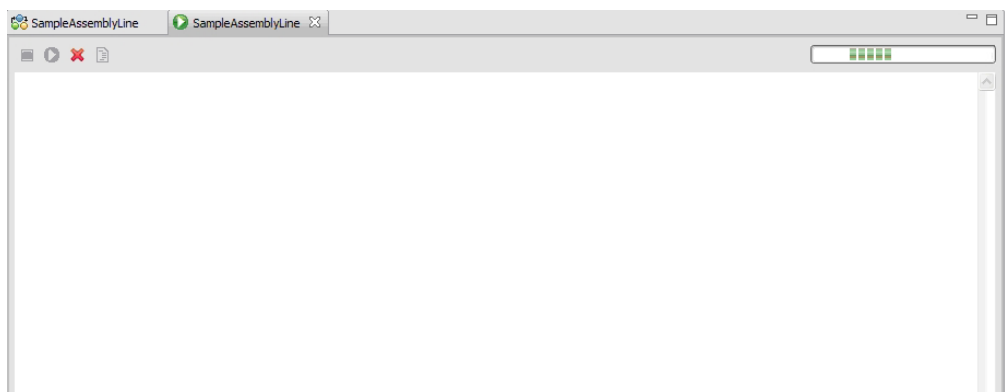


Si se muestra este diálogo, debe examinar la vista Problemas para ver qué errores pueden interrumpir la línea de ensamblaje. A menudo, estos errores se detectan durante el desarrollo y a pesar de ello el usuario desea ejecutar la línea de ensamblaje. En este caso, pulse Intro o el botón **Sí** para ejecutar la línea de ensamblaje.

La comprobación siguiente es ver si el servidor de IBM Security Directory Integrator está disponible. Si el servidor no es accesible, verá este mensaje:



Cuando se ejecuta una línea de ensamblaje desde el Editor de configuración, el primer paso es cuando el Editor transfiere la configuración en tiempo de ejecución al servidor y espera a que se inicie la línea de ensamblaje. En este paso verá una barra de progreso en la parte superior derecha de la ventana. Además, el botón de la barra de herramientas para detener la línea de ensamblaje aparece atenuado porque todavía no se ha iniciado la línea de ensamblaje.



El segundo paso es cuando se ejecuta la línea de ensamblaje. La barra de progreso girará y normalmente comenzará a ver mensajes en la ventana de registro. Ahora puede detener la línea de ensamblaje pulsando el botón de detención (en el extremo izquierdo) de la barra de herramientas.

Nota: El botón de detención sólo es efectivo si el servidor obtiene el control en la ejecución de la hebra. Si la hebra está ejecutando algo fuera de IBM Security Directory Integrator, pulsar el botón de detención puede ser inefectivo.

```

13:41:54,265 INFO - line (replace): ' 1: aload_1'
13:41:54,265 INFO - lineNumber (replace): '18'
13:41:54,265 INFO - CTGDIS004I *** Finished dumping Entry
13:41:55,265 INFO - CTGDIS003I *** Start dumping Entry
13:41:55,265 INFO - Operation: generic
13:41:55,265 INFO - Entry attributes:
13:41:55,265 INFO - line (replace): ' 2: invokespecial #3; //Method com/ibm/di/config/base/BaseCo
13:41:55,265 INFO - lineNumber (replace): '19'
13:41:55,265 INFO - CTGDIS004I *** Finished dumping Entry
13:41:56,265 INFO - CTGDIS003I *** Start dumping Entry
13:41:56,281 INFO - Operation: generic
13:41:56,281 INFO - Entry attributes:
13:41:56,281 INFO - line (replace): ' 5: aload_0'
13:41:56,281 INFO - lineNumber (replace): '20'

```

Si tiene varias ventanas de línea de ensamblaje abiertas, puede saber cuáles de las líneas de ensamblaje correspondientes se están ejecutando puesto que sus nombres llevarán un prefijo '*' (asterisco).

Cuando la línea de ensamblaje se detiene (normalmente o al pulsar el botón de detención), la barra de progreso desaparece y se habilita el elemento de la barra de herramientas para volver a ejecutar la línea de ensamblaje. El botón de detención ahora está inhabilitado porque la línea de ensamblaje ya no se está ejecutando.

```

CTGDIS266E Error in InitConnectors. Exception occurred: java.lang.Exception: CTGDIT004E The 'filePath' parameter
CTGDIS100I Printing the Connector statistics.
[FileSystemConnector] Not used
[IF] Branch True:0, Branch False:0
[OutputFile] Errors:1
CTGDIS104I Total: Errors:2.
CTGDIS101I Finished printing the Connector statistics.
CTGDIS077I Failed with error: CTGDIT004E The 'filePath' parameter must be set to use the File Connector to a ope

```

Figura 81. Ventana Registro de la consola

Puede borrar la ventana de registro en cualquier momento. La ventana de registro sólo muestra las últimas cien líneas del registro de línea de ensamblaje, pero cada mensaje del registro se graba en un archivo de registro temporal de modo que se puede abrir el archivo de registro en una ventana de editor diferente utilizando el botón Ver registro (en el extremo derecho).

Nota: Puede cambiar el tamaño del almacenamiento intermedio de la ventana de registro desde el valor predeterminado de 300 líneas a otro valor. Para ello, seleccione **Ventana > Preferencias > Preferencias de Security Directory Integrator > Número máximo de líneas para la ventana Ejecutar línea de ensamblaje**.

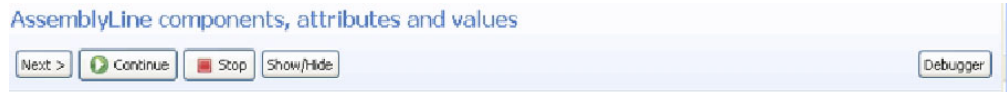
Una vez la línea de ensamblaje ha terminado, puede volver a ejecutar la línea de ensamblaje con el botón de reejecutar.

Observe el texto en azul de la ventana de registro. Si aparece ese texto, puede mantener pulsada la tecla Control mientras pulsa en la palabra para ir a esa parte de la línea de ensamblaje.

El repetidor y el depurador

El repetidor y el depurador son herramientas integradas en el Editor de configuración que pueden ayudarle a desarrollar líneas de ensamblaje de forma interactiva.

Puede ejecutar el repetidor en dos modalidades. Una es el depurador avanzado normal (activado mediante el botón **Depurador**, consulte “El depurador” en la página 164) donde se tiene acceso a todas las partes de la línea de ensamblaje y la otra es el repetidor (activado por el botón **Repetidor de datos**, consulte “El repetidor de datos”) que ofrece una vista más simple de los componentes y el flujo. Puede conmutar entre los dos pulsando el botón situado en la vista de columna:



En la vista de repetidor puede conmutar a la vista de depurador pulsando el botón **Depurador**. Inversamente, para conmutar desde el depurador al repetidor, pulse el botón **Repetidor de datos**.



El repetidor de datos

El repetidor de datos ofrece una vista de columna de todos los conectores de la línea de ensamblaje. Al pasar por la línea de ensamblaje, se mostrarán los datos leídos o grabados de cada componente. Todos los datos que aparecen en estas tablas siempre posteriores al momento en que un conector completa su operación.

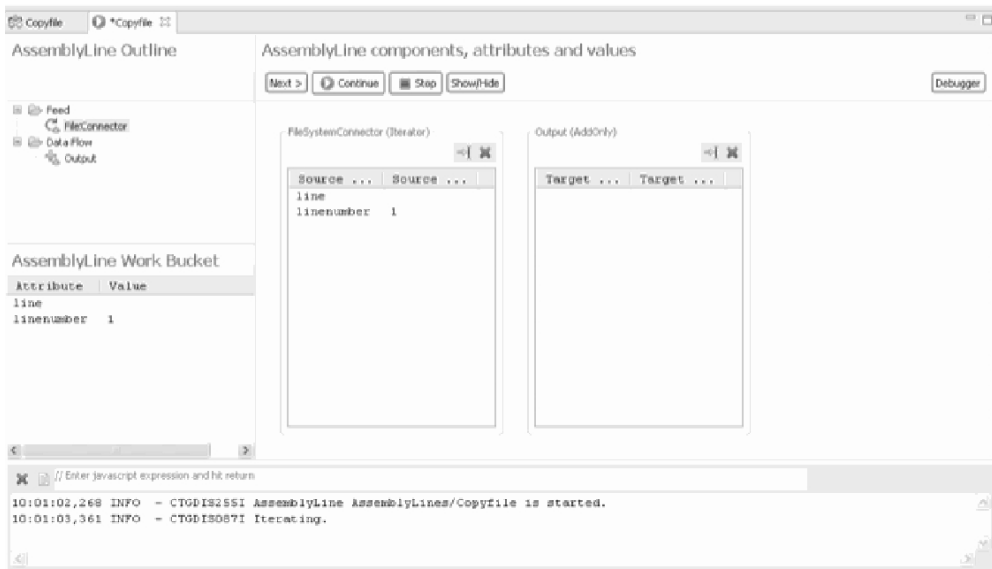



Figura 82. Ventana principal Repetidor de datos

En la parte de la derecha, el repetidor de datos muestra componentes con una correlación de atributos de forma vertical. Cada componente se puede eliminar de

la vista pulsando el botón de cierre o deseleccionando el componente en el diálogo

Mostrar/ocultar. El botón de la izquierda del botón de cierre () de cada componente es atajo de “Ejecutar aquí”.

Los botones **Siguiente** y **Ejecutar** se utilizan para repetir un componente cada vez o para ejecutar la línea de ensamblaje hasta su finalización. El botón **Detener** se utiliza para pausar la ejecución de una línea de ensamblaje o para terminar una línea de ensamblaje en pausa. La parte izquierda del repetidor de datos muestra la descripción de la línea de ensamblaje y la entrada de trabajo a continuación. En la vista de esquema, puede optar por volver a iniciar la línea de ensamblaje desde el menú contextual. De este modo, se iniciará el depurador en una nueva sesión y se ejecutará hasta el componente seleccionado. Éste es un modo rápido de llegar al depurador desde el repetidor de datos.

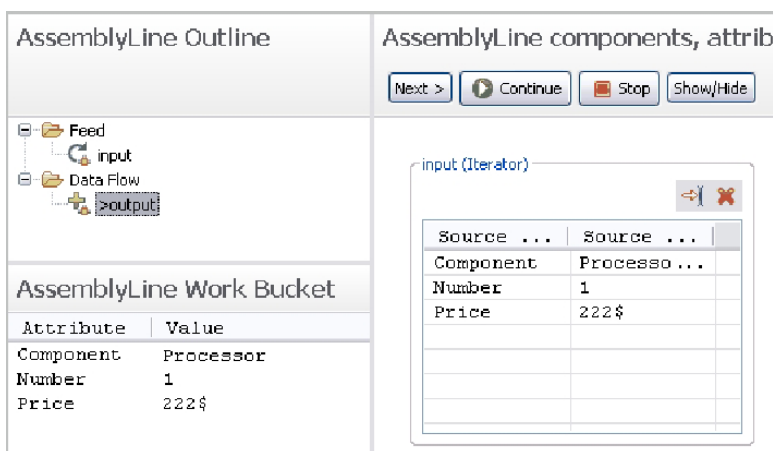


Figura 83. Botón **Mostrar/ocultar** del repetidor de datos

El diálogo **Mostrar/ocultar** componentes permite escoger los componentes que deben mostrarse en la vista.

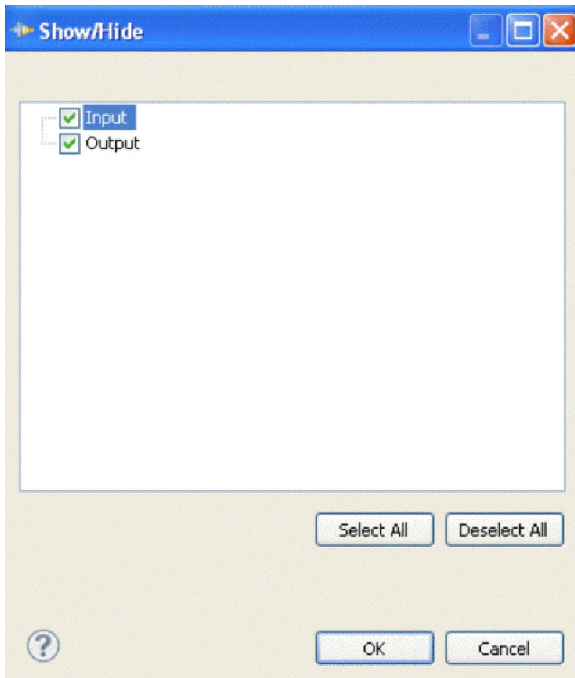


Figura 84. Diálogo Mostrar/ocultar componentes

El depurador

Al seleccionar la vista **Depurador**, se mostrará un diseño muy similar a la vista del repetidor, pero en la vista del depurador verá más información de los componentes de la línea de ensamblaje y también tendrá la ventana de visualización con las expresiones personalizadas. El árbol de componentes de la línea de ensamblaje cuenta con una casilla de verificación para cada elemento que puede marcar o desmarcar para establecer o eliminar un punto de interrupción. Además, existen más botones de mandato que permiten acceder a componentes y ganchos.

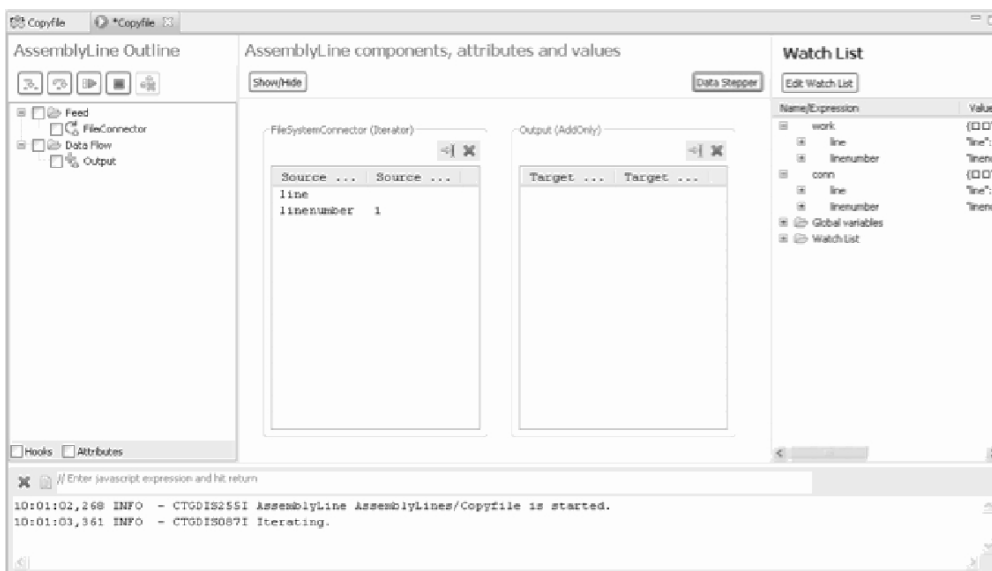


Figura 85. Ventana Depurador

(En la ventana del depurador puede resultar útil maximizar el editor con Ctrl-M par obtener una mejor visión general.)

El árbol de la izquierda muestra los componentes y complementos de software de la línea de ensamblaje. Puede conmutar el recuadro de selección para cada elemento para establecer un punto de interrupción allí. Efectúe una doble pulsación en el elemento para ver el script o para entrar el script para una interrupción condicional.

La imagen siguiente muestra la pestaña de interrupción condicional después de efectuar una doble pulsación en el complemento de software **Antes de GetNext**. Tenga en cuenta también que puede ocultar todos los complementos de software que estén inactivos. Mostrar todos los complementos de software le permite establecer un punto de interrupción independientemente de si el complemento de software está o no activo. El recuadro de selección **Atributos** le permite ocultar las correlaciones de atributos de la vista de árbol.

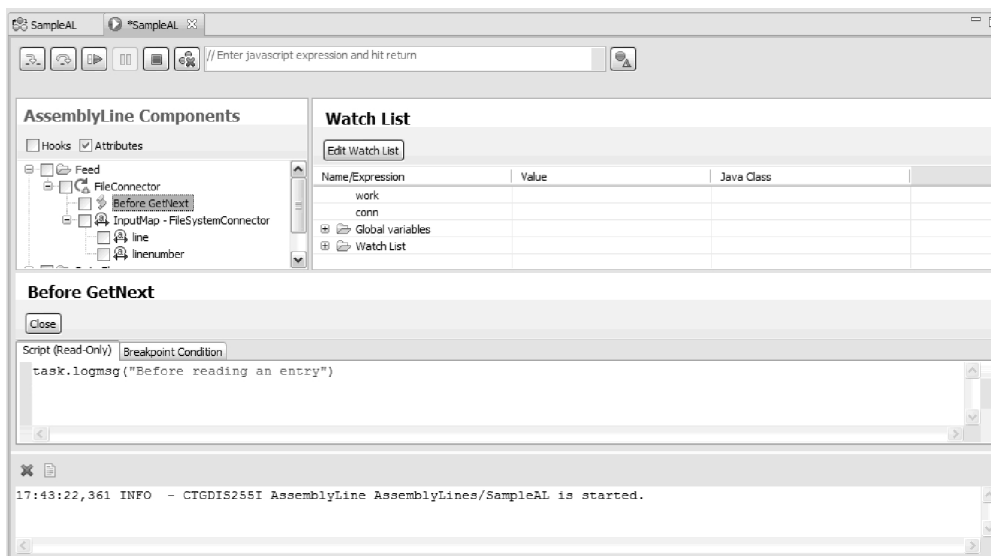


Figura 86. Depurador antes en Antes de GetNext

El panel de correlación de atributos muestra los componentes y sus asignaciones y además el valor asignado al atributo de trabajo. El valor es una instantánea de la última interrupción de la línea de ensamblaje y **Valor anterior** es el valor de instantánea antes de ésta. Si ejecuta paso a paso la línea de ensamblaje, los valores reflejarán las correlaciones y los scripts que afectan a la entrada de trabajo.

Si se produce un error, el repetidor mostrará el mensaje de excepción y el rastreo de pila en un diálogo aparte. El archivo de registro (en la pantalla) no incluye este rastreo de pila. Puede elegir no mostrar este diálogo seleccionando el recuadro de selección o en la página de preferencias del Editor de configuración.

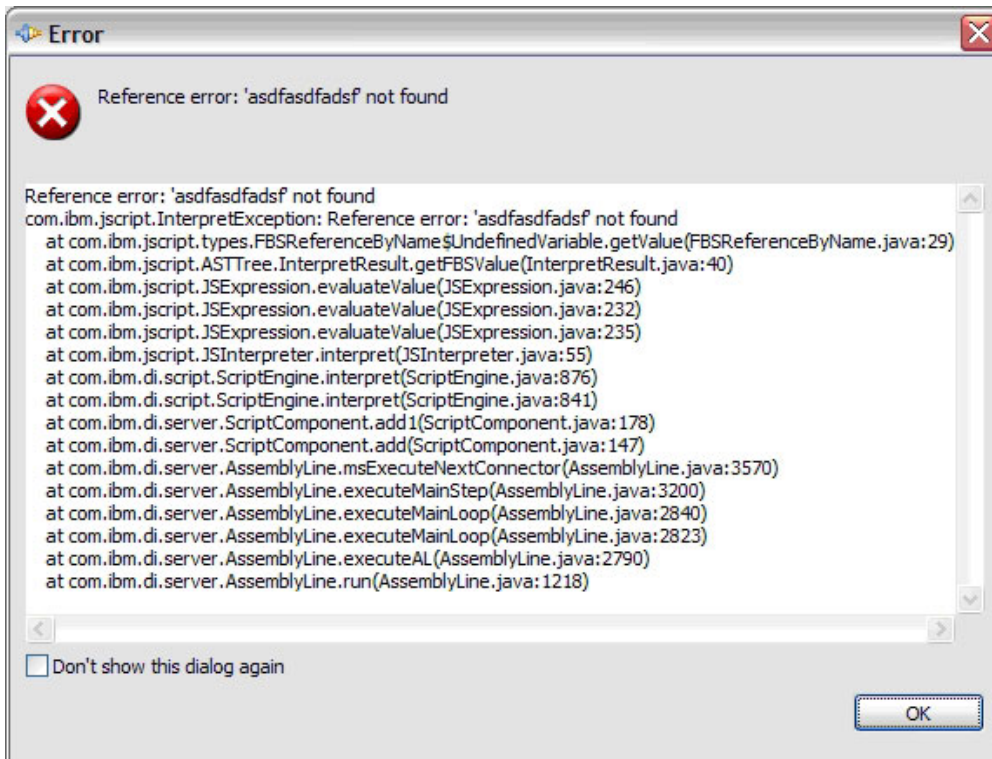


Figura 87. Diálogo de error: Seguimiento de la pila

Ejecución paso a paso de scripts

Cuando se alcanza un punto de interrupción que contiene JavaScript, puede ejecutar paso a paso el script y ejecutarlo línea a línea. La pestaña del script mostrará el script que va a ejecutarse; puede seguir el flujo utilizando el mandato **Ejecutar paso a paso** (el botón más a la izquierda de los dos botones de Repetidor).

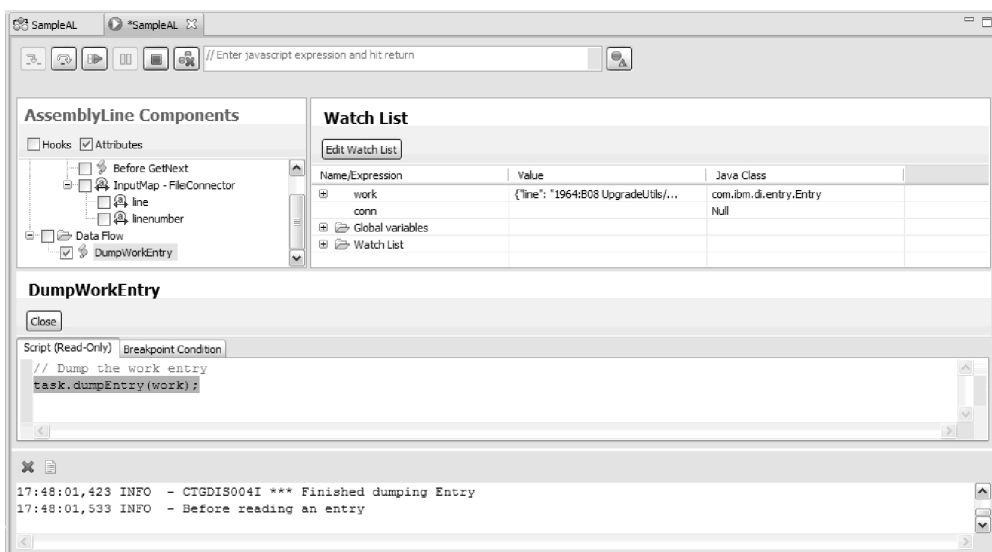


Figura 88. Ventana del Depurador: ejecutar paso a paso un script, línea a línea.

Cuando se ejecuta paso a paso el script, se resalta cada línea antes de ejecutarla. También puede utilizar la función **Evaluar** para mostrar variables de motor de scripts mientras ejecuta paso a paso un script.

Cuando esté a punto de ejecutarse una función de script, puede utilizar el botón **StepInto** para acceder a la función de JavaScript. Si la función se define fuera del contexto actual (por ejemplo, gancho, script de correlación de atributos) se sustituirá la ventana.

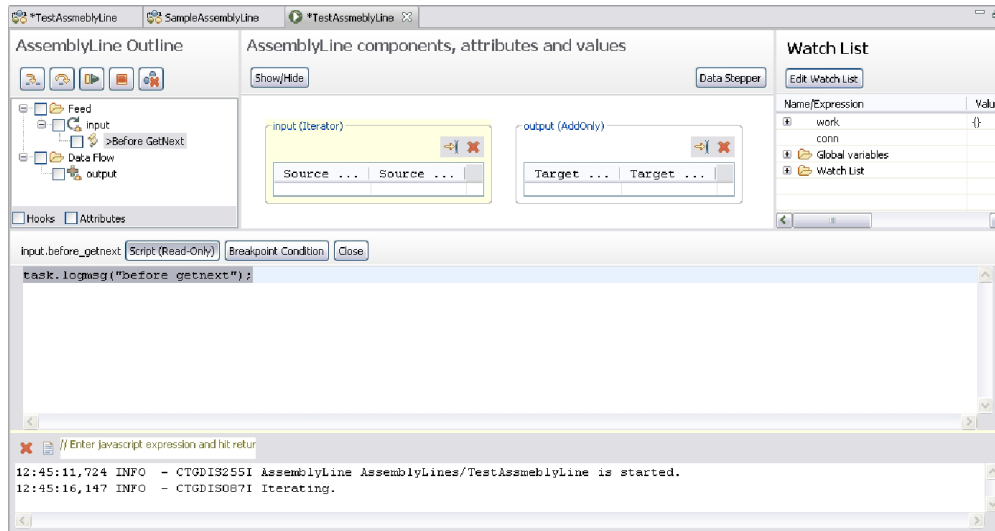


Figura 89. Función StepInto

En esta línea de ensamblaje, hemos definido una función denominada myfunc1() en el gancho antes de inicio. Le llamaremos desde el componente Script1 para mostrar el aspecto que tendrá:

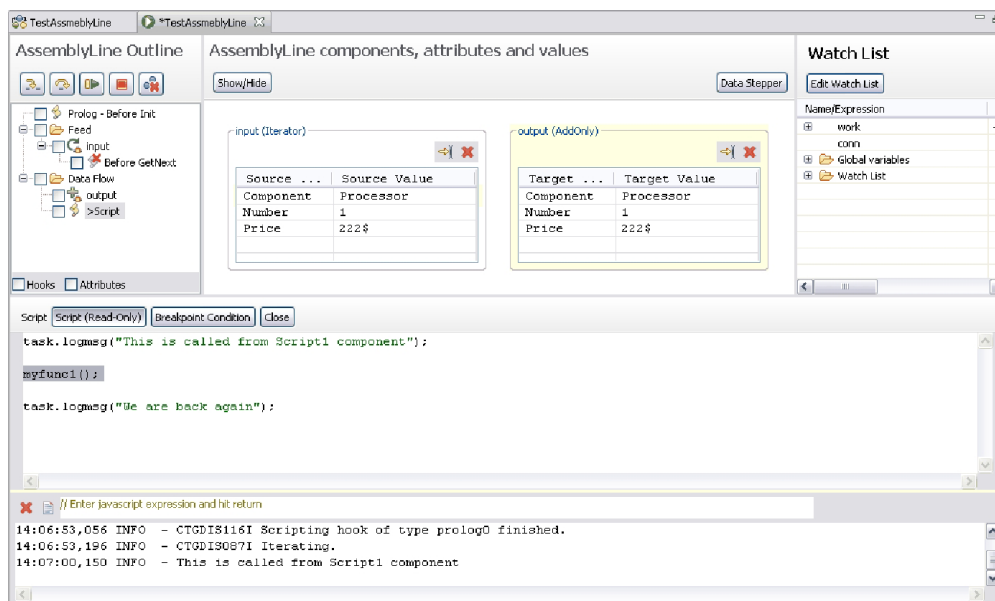


Figura 90. Función Stepped-into

Ahora podemos pulsar **StepOver** para continuar con la sentencia siguiente o pulsar **StepInto** para seguir la llamada de función JavaScript:

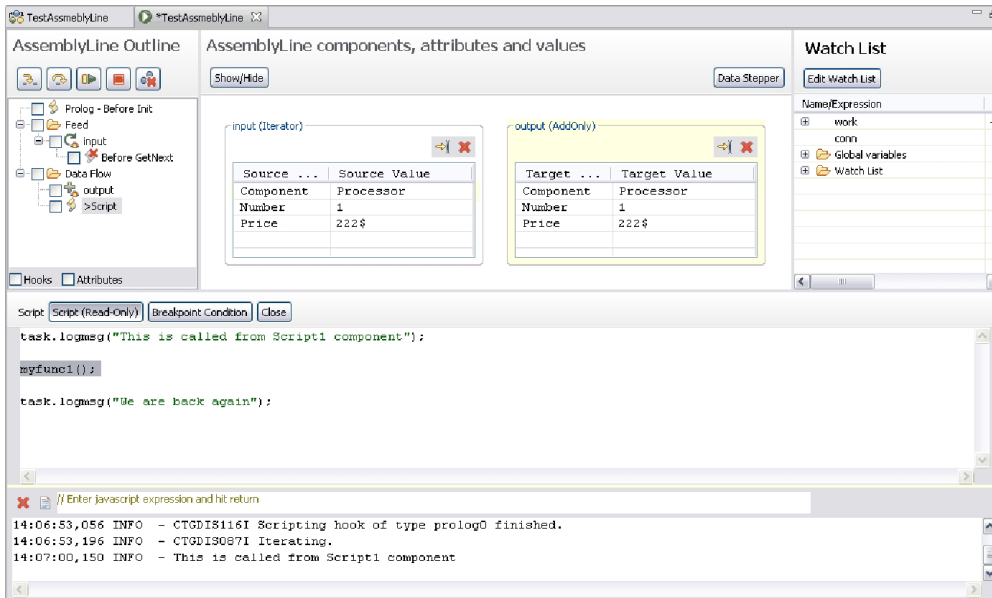
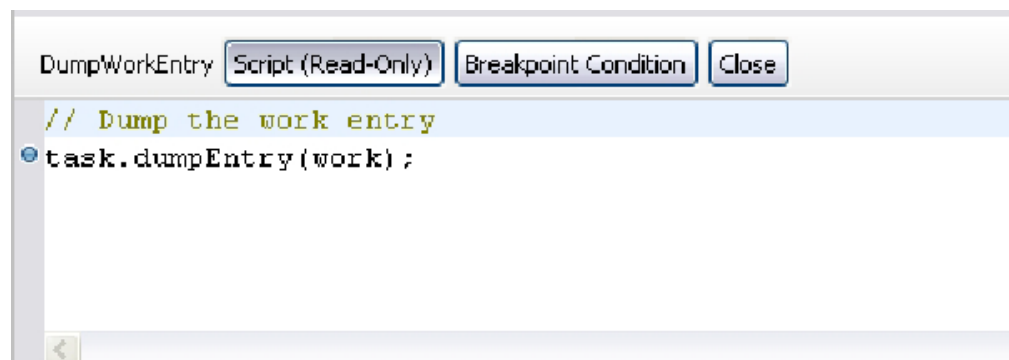


Figura 91. Seguir la llamada de función

Paso relativo a las causas del editor de script para modificar el script desde el gancho *antes de inicializar*. Tenga en cuenta que la etiqueta modificada se muestra en el lugar en el que se define el script.

También puede establecer puntos de interrupción dentro de los scripts. Abra el editor para un script o una correlación de atributos y efectúe una doble pulsación en el margen izquierdo. Aparecerá una viñeta en azul para indicar que hay establecido un punto de interrupción para esa ubicación. Vuelva a efectuar una doble pulsación para eliminarlo. También puede pulsar con el botón derecho del ratón en el margen izquierdo o en el campo de texto para conmutar los puntos de interrupción.



Depuración de servidor

Depurar un servidor de IBM Security Directory Integrator significa que cada línea de ensamblaje iniciada en un servidor de IBM Security Directory Integrator establece automáticamente una sesión de depuración con el Editor de configuración como si se hubiera iniciado en la modalidad de pasos.

La depuración de servidor se activa seleccionando **Depurar servidor** en el menú desplegable de un servidor en la vista Servidores. Si selecciona esta opción, el CE se conecta al servidor y establece una propiedad de Java que apunta al CE para la depuración.

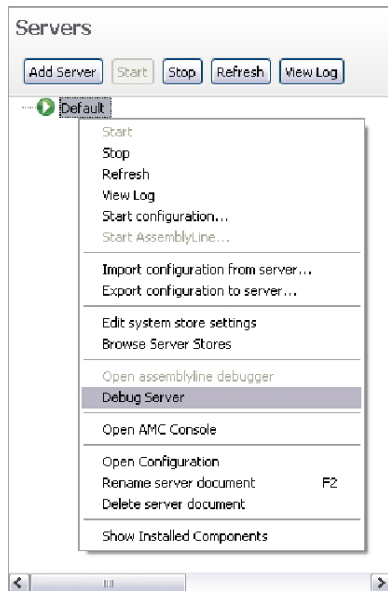
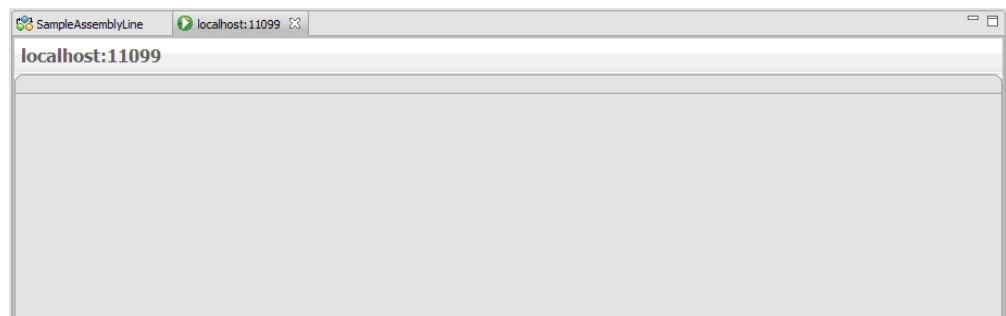


Figura 92. Opción Depurar servidor

La selección de **Depurar servidor** abre una nueva ventana en la que aparecen las líneas de ensamblaje a medida que se inician en el servidor. Esto es diferente a iniciar de forma activa una línea de ensamblaje en el Editor de configuración. Cuando inicia una sesión de depuración desde el Editor de configuración, la sesión tiene su propia ventana y no aparece en esta ventana de depuración de servidor.



Cada línea de ensamblaje iniciada por otro Editor de configuración o componente en el servidor de destino tendrá su propio panel de repetidor dentro de esta ventana. Consulte “El repetidor y el depurador” en la página 162 para obtener una descripción del panel de repetidor.

Opciones de ejecución

Puede especificar opciones adicionales cuando ejecuta una línea de ensamblaje. Estas opciones se guardan para que cada vez que ejecute la línea de ensamblaje utilice estas opciones.

Puede seleccionar la modalidad de ejecución, la operación de la línea de ensamblaje y proporcionar una entrada work inicial (IWE) a la línea de ensamblaje.

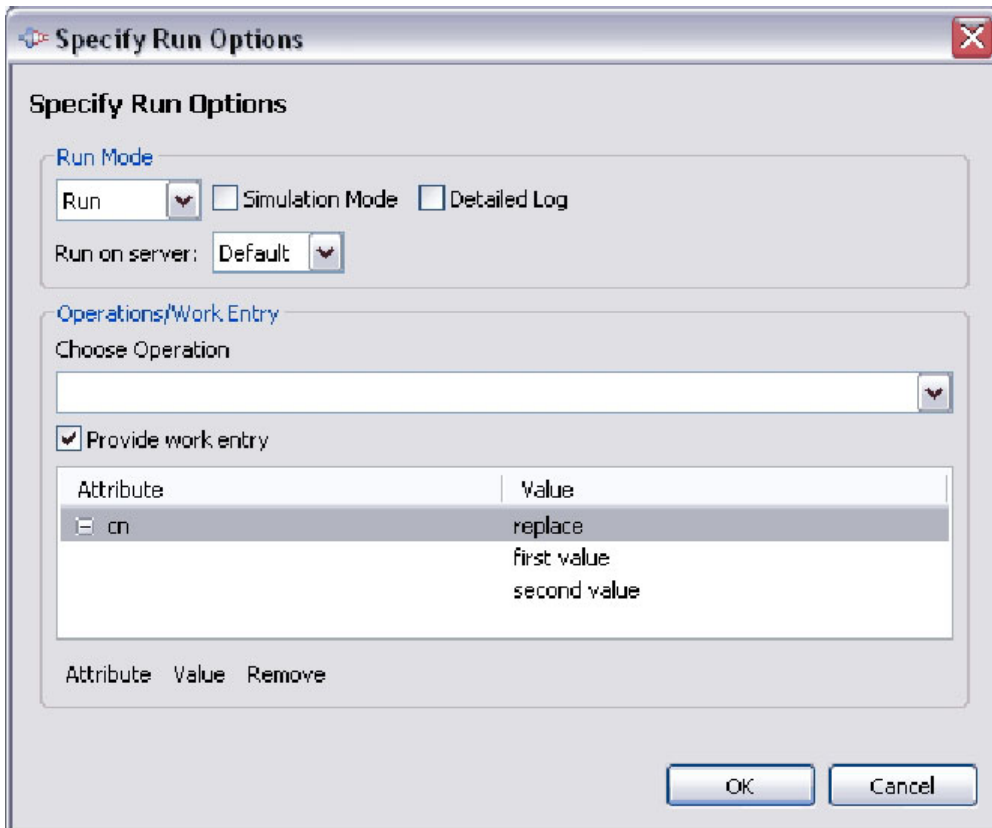


Figura 93. Ventana Ejecutar con opciones

Utilice los botones **Atributo** y **Valor** para añadir atributos y valores a la entrada work inicial.

Selección del servidor

Cuando ejecuta una línea de ensamblaje, ésta se ejecutará en el servidor de desarrollo local. Este servidor lo inicia el CE y su directorio de soluciones se encuentra en el proyecto *Servidores de TDI*. Cuando cree un nuevo servidor de IBM Security Directory Integrator, puede cambiar el servidor preferido para un proyecto determinado mediante el diálogo **Propiedades** del proyecto:

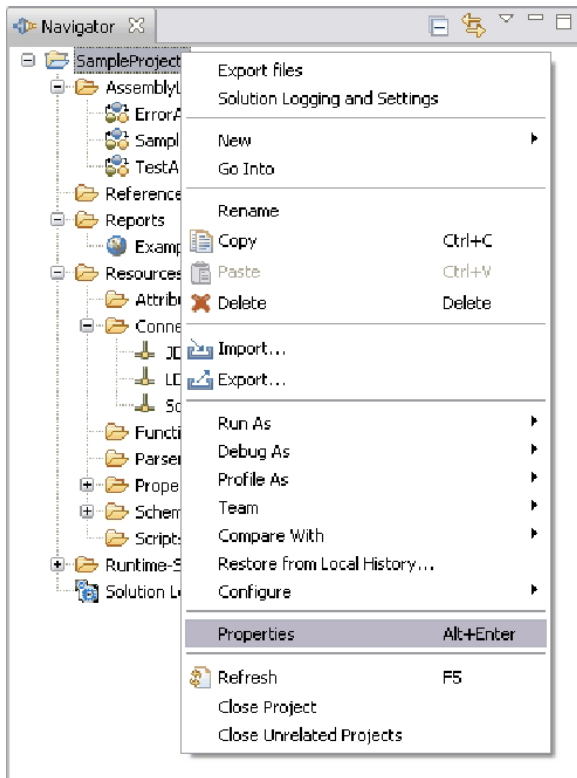


Figura 94. Opción del menú desplegable *Propiedades del proyecto*

La selección de este elemento presentará el diálogo de propiedades para el proyecto. Seleccione **Propiedades de Directory Integrator** para cambiar el servidor predeterminado para el proyecto.

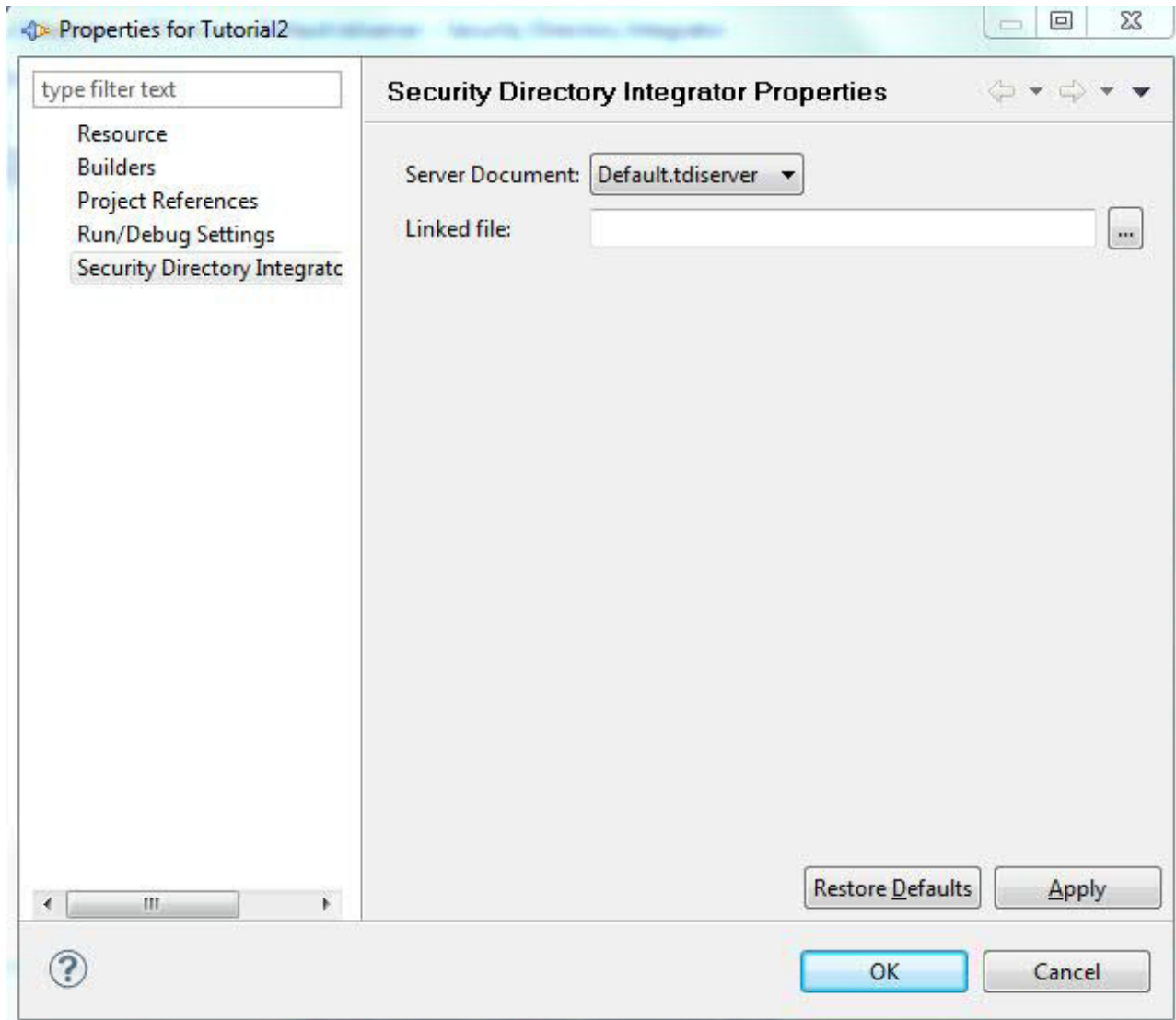


Figura 95. Propiedades del proyecto

Soporte de equipo

El Editor de configuración de IBM Security Directory Integrator incluye los plug-ins de Eclipse que permiten compartir proyectos entre usuarios utilizando un repositorio de control de código fuente.

IBM Security Directory Integrator incluye bibliotecas CVS que dan soporte a las conexiones de tipo pserver, pserverssh2, ext y extssh. Para obtener una información más a fondo sobre los plug-ins de CVS, consulte el sitio de CVS de Eclipse en <http://www.eclipse.org/eclipse/platform-cvs>.

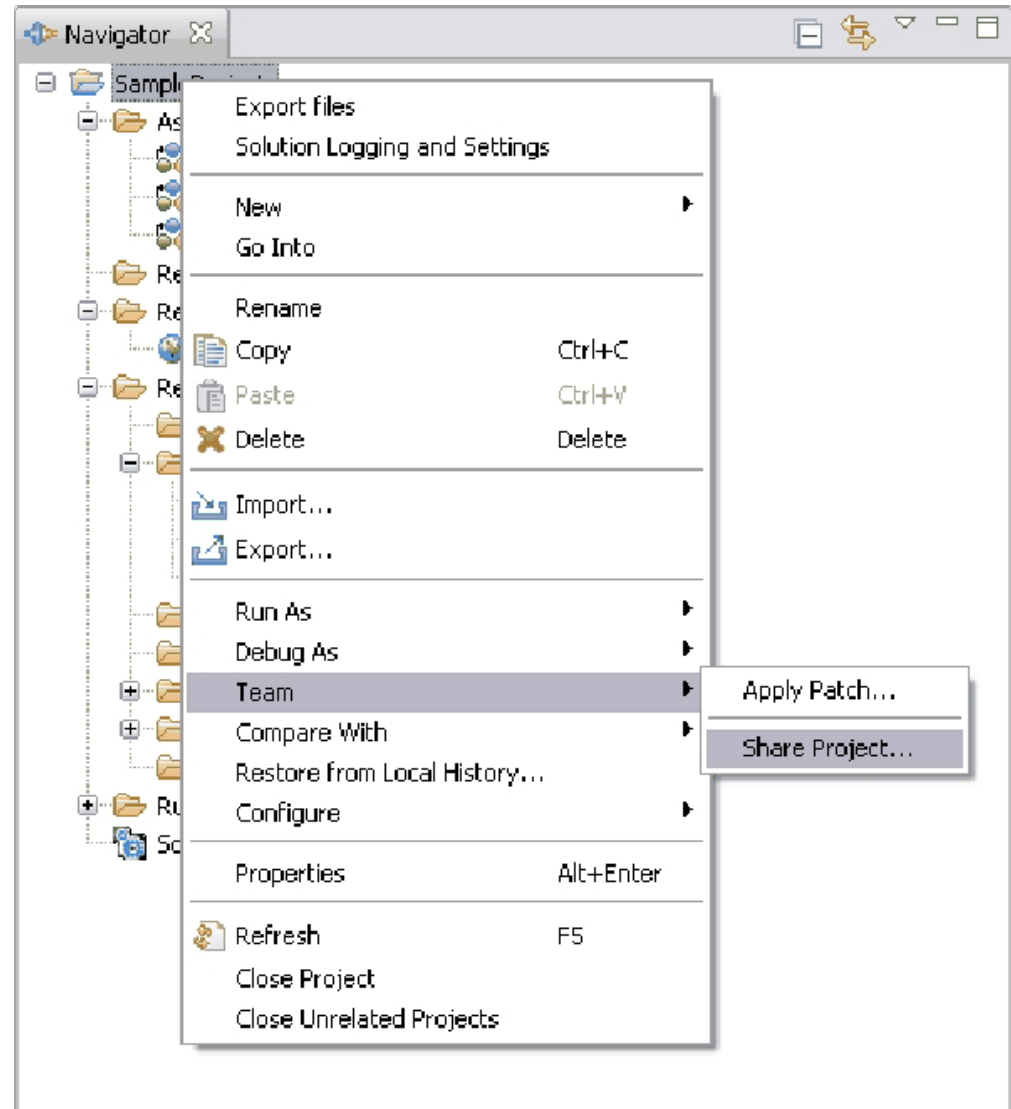
Para utilizar los recursos de compartimiento de equipo necesita acceder a un repositorio CVS. La mayoría de sistemas operativos pueden alojar repositorios CVS y existen paquetes binarios generalmente disponibles en la red. El sitio de CVS de Eclipse contiene preguntas frecuentes (FAQ) que le ayudarán con los problemas más corrientes que puede encontrar.

Para obtener ayuda para instalar y configurar un servidor CVS, consulte la wiki de CVS en <http://ximbiot.com/cvs>. Buscar en la "cómo instalar un servidor cvs" también presentará numerosos sitios web que describirán detalladamente cómo instalar y configurar un repositorio CVS en distintos sistemas operativos y plataformas.

Compartimiento de un proyecto

Puede compartir un proyecto con otros usuarios utilizando CVS.

Para compartir un proyecto, seleccione la opción **Equipo > Compartir proyecto...** en el menú desplegable de un proyecto.



Se abre otra pantalla en la que podrá especificar los parámetros del control de origen.

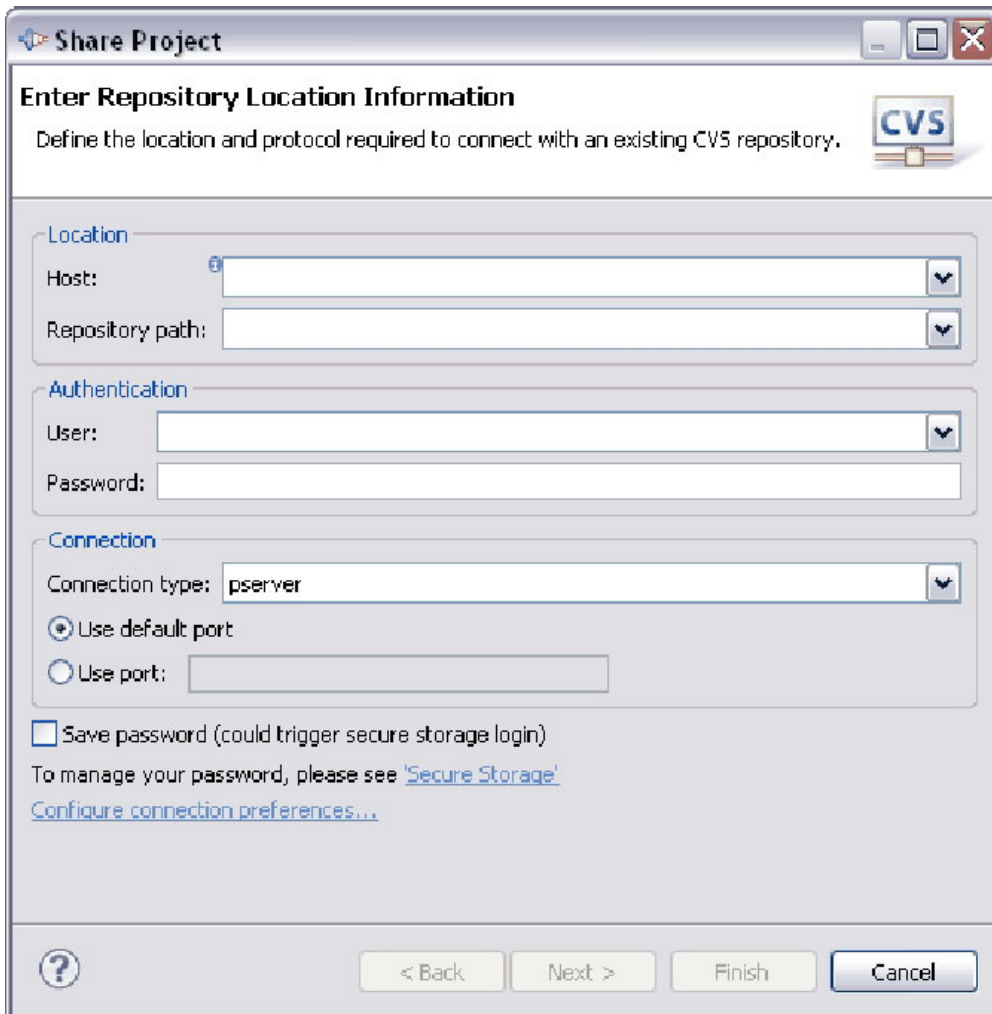
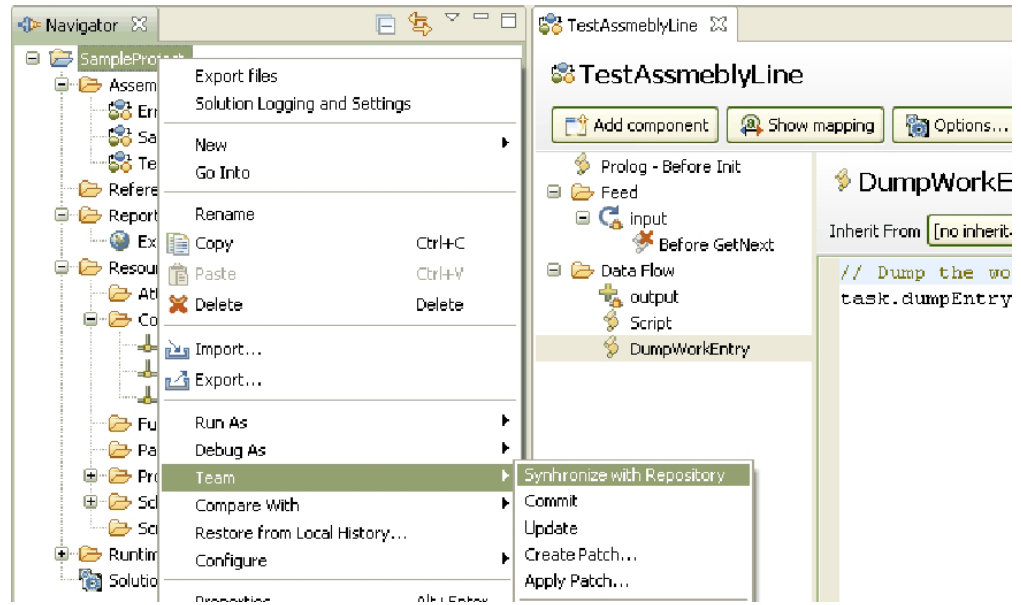


Figura 96. Ventana Compartir proyecto en CVS

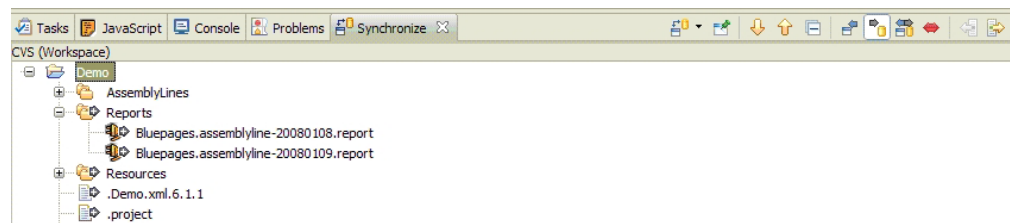
Complete el asistente para compartir el proyecto en el servidor CVS.

Nota: El repositorio sólo puede controlar correctamente los archivos reales del proyecto y la estructura de directorios que los contiene. Los directorios vacíos no se tienen en cuenta.

Cuando un proyecto se comparte, se puede sincronizar con el repositorio para confirmar sus propios cambios además de recibir los cambios que realizan otras personas.



Seleccione **Equipo > Sincronizar con repositorio** para abrir la vista Sincronizar:

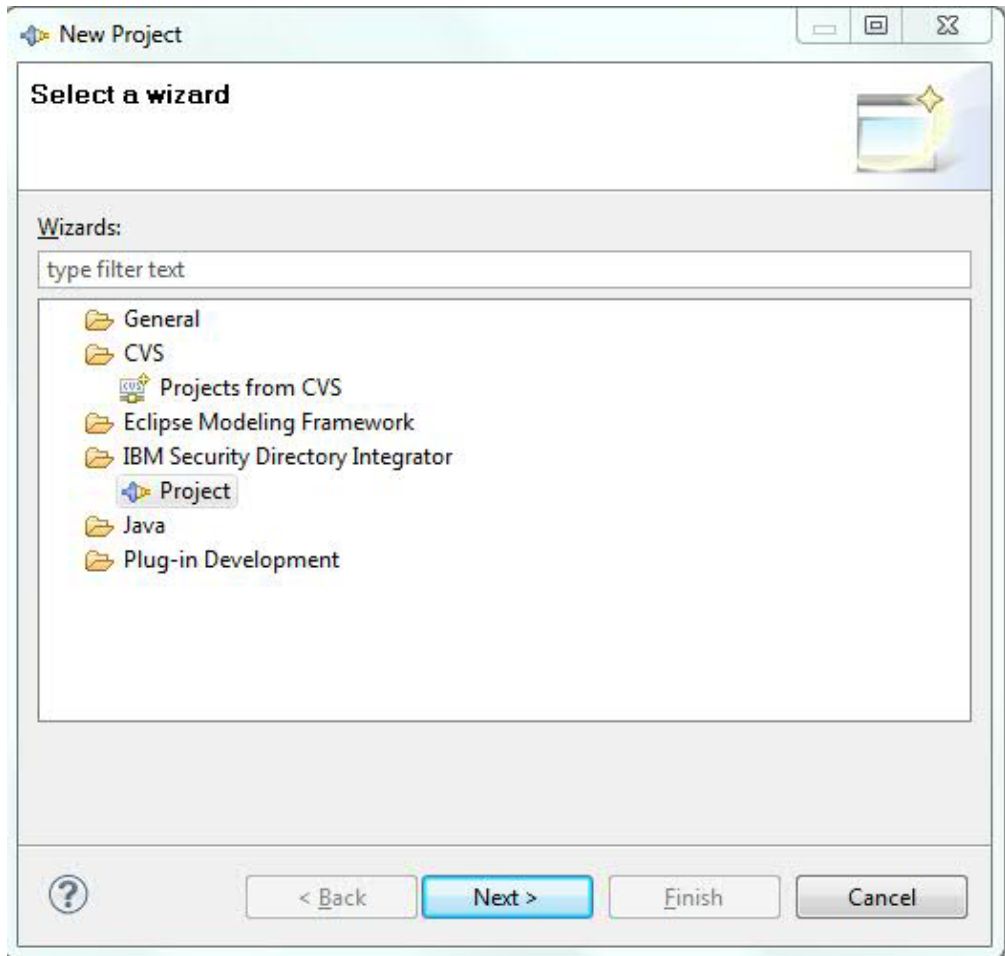


Esta vista muestra los archivos que es necesario actualizar además de los archivos en los que hay actualizaciones. Una vez confirmados los cambios, el navegador mostrará información adicional para los archivos.

Utilización de un proyecto compartido

Si desea utilizar un proyecto que alguien ha compartido, utilice el asistente Proyecto CVS.

Seleccione **Archivo > Nuevo > Nuevo proyecto...** en el menú principal y seleccione el asistente Proyecto CVS.



Finalice el asistente para recuperar el proyecto en el espacio de trabajo.

Vista Problemas

Cuando guarda un componente, el creador de proyectos de IBM Security Directory Integrator actualizará su archivo de configuración en tiempo de ejecución para reflejar los cambios que se han realizado.

El creador de proyectos también realizará una comprobación de validación en el componente e informará de los avisos y errores en la vista de problemas. En la vista de problemas verá avisos como los siguientes:

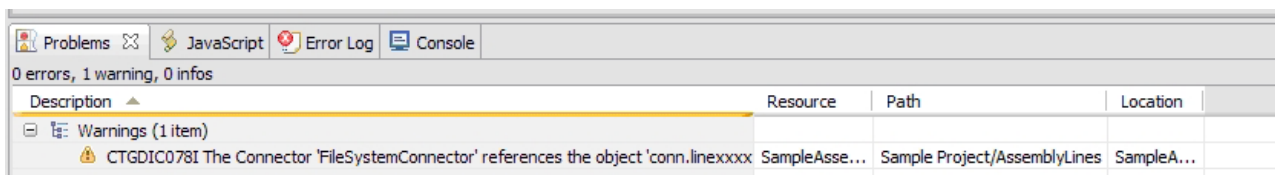


Figura 97. Ventana Vista Problemas

Cuando se efectúa una doble pulsación en una línea de esta vista se abrirá la ubicación donde se ha localizado el problema.

Los problemas que puede esperar ver en la vista de problemas son los siguientes elementos:

- Referencia a un elemento de esquema sin definir
Si utiliza construcciones como “conn.abc” y “abc” no está definido en el esquema, se obtiene este aviso.
- Referencia a un atributo work sin definir
Si utiliza construcciones como “work.abc” y “abc” no existe, se obtiene este aviso.
- Errores sintácticos en scripts
- Las líneas de ensamblaje con más de un conector en modalidad servidor

Mejoras en JavaScript

En cualquier lugar que edite scripts en el Editor de configuración, obtendrá un editor de texto, mejorado específicamente para editar código JavaScript.

Este editor proporciona terminación de código, coloración de sintaxis, además de marcar los errores sintácticos.

Algunas de las mejoras son:

- “Terminación de código”
- “Coloración de sintaxis” en la página 179
- “Comprobación de la sintaxis” en la página 180
- “Evaluación local” en la página 180
- “Editores externos” en la página 181

Terminación de código

El editor de JavaScript admite terminación de código.

A medida que escribe, el editor reaccionará a determinadas pulsaciones. El punto (.) activa la terminación de código que presenta un menú emergente que muestra todos los métodos relevantes, campos y características específicos de IBM Security Directory Integrator. La terminación de código también se puede activar manualmente mediante la combinación de teclas `Ctrl-<Espacio>`. La terminación de código incluye los tipos de script JavaScript estándar (serie, numérico, etc.) además de la terminación personalizada para objetos específicos de IBM Security Directory Integrator. Los objetos como *conn* y *work* proporcionan una lista de atributos disponibles además de los campos y métodos del objeto.

La terminación de código funciona siempre que el editor pueda derivar el nombre de clase Java de la expresión:

El editor también reaccionará a las comillas simples o dobles y a las llaves ("{}"). Cuando se escribe una comilla simple o doble, el editor inserta automáticamente otra comilla y sitúa el signo de intercalación entre las dos. Se hace así para que sea más fácil entrar constantes de tipo serie.

Cuando se escriben llaves, se sangra automáticamente para acomodar sangrado de bloques. Cuando se escribe una llave de apertura y se pulsa Intro, se insertarán separadores y el signo de intercalación se sitúa en la línea siguiente con el sangrado correcto. A la inversa, cuando se escribe la llave de cierre, se eliminará el sangrado de la llave.

Correlación y terminación de código

Cuando se añaden atributos en el CE, se generan expresiones simples basadas en el nombre de atributo. Cualquier nombre que contenga puntos u otros caracteres que no sean identificadores de objeto de script válidos, se especificarán en ["nombre de atributo"] (por ejemplo, conn["http.body"]). Que la entrada sea jerárquica o no es un hecho irrelevante cuando se utiliza esta notación puesto que funciona en ambos casos. Los atributos que son identificadores de javascript válidos se utilizan tal cual (por ej. conn.cn). Si tiene un esquema jerárquico, aun así se generará una expresión simple para la correlación. Si, por ejemplo, tiene una jerarquía de a->b->c y correlaciona la parte "c", se genera ["a.b.c"] para hacer referencia al atributo.

En el editor de scripts verá que la terminación de código funciona de un modo similar. La terminación de código mostrará las terminaciones en texto claro (por ejemplo, http.body) pero si pulsa Intro para terminar la expresión, se utilizará el mismo alojamiento para los nombres de atributo que no son nombres de objeto de script válidos.

Coloración de sintaxis

La coloración de sintaxis del editor es básica. El editor decora los comentarios y las series.

Comprobación de la sintaxis

Cuando se escribe, el editor comprueba la sintaxis del script en segundo plano. Cuando el script contiene errores, se muestra el error en los márgenes y se pone un garabato rojo debajo del texto para marcar el lugar en que el intérprete de JavaScript ha encontrado un error.

```
// This is a comment
a = "string value";
❌ if(a === 0) {
    asdfasdf;
}
```

Cuando se pasa el ratón por encima de la marca del error en la *regla* (el margen izquierdo), aparece el mensaje de error en una ventana emergente.

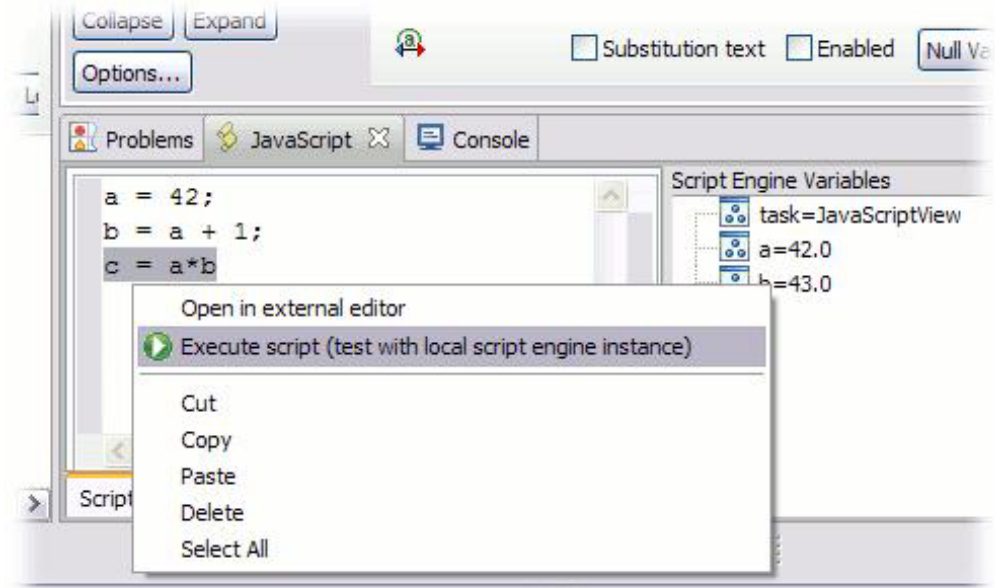


La regla de la izquierda está sincronizada con la ventana del texto y una marca está directamente relacionada con la línea de la ventana del texto. Si el texto no contiene errores esto se ve en la ventana, que no tendrá marcas en esta regla.

Sin embargo, la regla de visión general de la derecha muestra todos los errores del script completo independientemente de la posición del editor de texto. Cuando se pasa el ratón por encima de la marca en la regla, se obtiene el mismo mensaje de error emergente que en la regla de la izquierda. Al pulsar en el marcador, el editor se vuelve a situar en la línea en la que se ha identificado el problema.

Evaluación local

Cuando edita un script, puede pulsar el botón derecho del ratón y seleccionar **Ejecutar script** en el menú desplegable para realizar una evaluación rápida del texto seleccionado. Un entorno de prueba más detallado para scripts es la vista de JavaScript en la que puede ejecutar scripts y ver las variables del motor de scripts en una ventana aparte:



Editores externos

Puede editar el script con su editor de JavaScript preferido utilizando el menú contextual.

Pulse con el botón derecho del ratón en el campo de texto y seleccione **Abrir en un editor externo**. El editor se configura en la pestaña de preferencias **Ventana > Preferencias > Security Directory Integrator**:

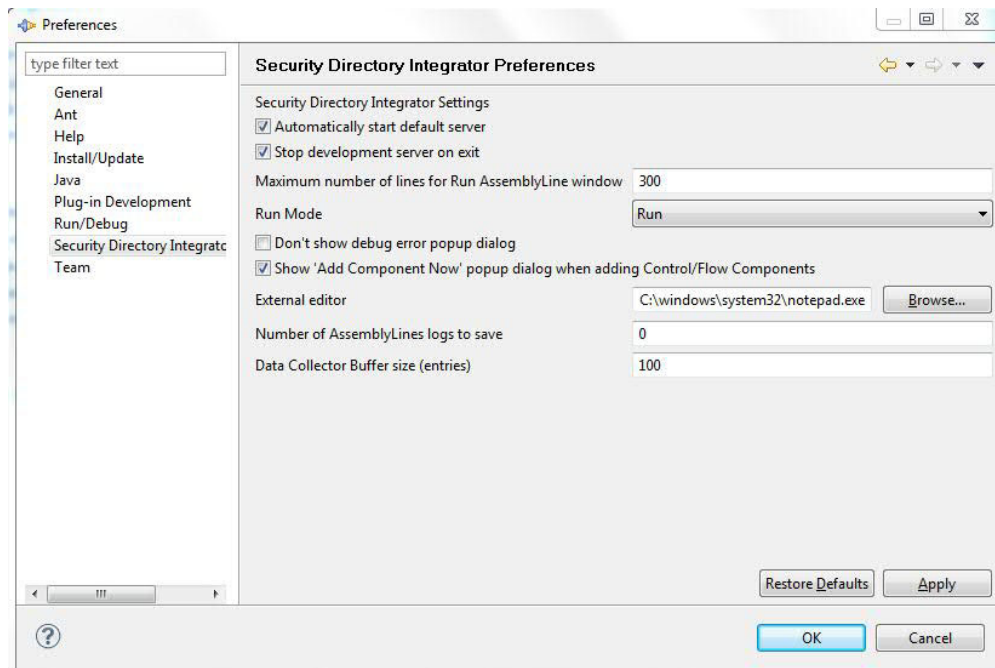
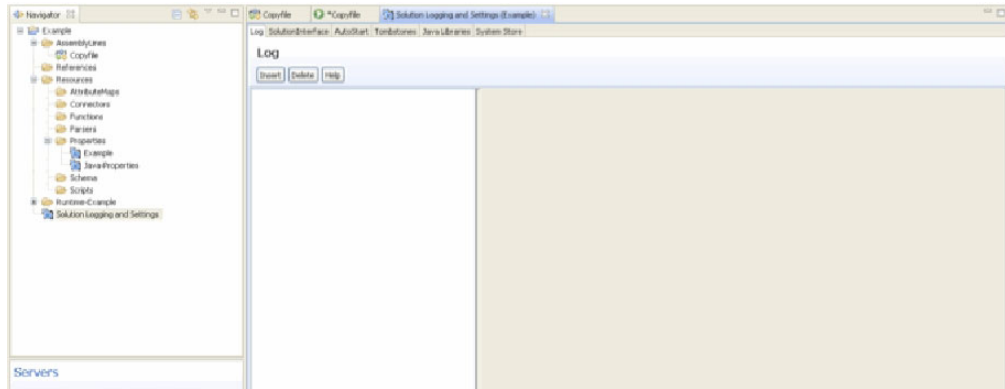


Figura 98. Ventana Preferencias del Editor de configuración

Registro de soluciones y valores

La ventana Registro de soluciones y valores le permite editar las áreas específicas de la solución del proyecto.

Puede abrir la ventana seleccionando un proyecto, o un archivo de proyecto, y efectuando una doble pulsación en **Registro de solución y valores** en el árbol de proyectos.



En Registro de solución y valores puede modificar los elementos siguientes:

- “Valores de almacén del sistema”
- “Registro cronológico” en la página 184
- “Lápidas” en la página 184
- “Bibliotecas Java” en la página 185
- “Inicio automático” en la página 185
- “Valores de interfaz de la solución” en la página 185

Valores de almacén del sistema

Los valores de almacén del sistema para un proyecto pueden alterar temporalmente el almacén de sistema predeterminado, definido por el servidor de IBM Security Directory Integrator. Cuando está habilitado, la configuración utilizará el almacén de sistema configurado en lugar del almacén del sistema del servidor.

Los valores del almacén del sistema pueden producirse en dos lugares:

1. A nivel de proyecto de IBM Security Directory Integrator en el espacio de trabajo; y
2. a nivel del servidor de IBM Security Directory Integrator.

Los valores a nivel de proyecto tienen prioridad sobre los valores a nivel de servidor; es decir, si ha definido un almacén de sistema específicamente para su proyecto, dicho almacén del sistema se utilizará durante la ejecución de los componentes en el Editor de configuración. Si no ha definido ningún valor del almacén de configuración en el proyecto, se utilizará el almacén del sistema del servidor que utiliza para ejecutar sus componentes.

Valores de nivel de proyecto

En el menú desplegable del título puede seleccionar plantillas predefinidas y guardar valores de almacén del sistema en archivos del espacio de trabajo. Tenga en cuenta que todas las tablas (delta, propiedades, etc.) se almacenarán en esta base de datos.

Los elementos de menú enumerados como **Derby Embedded**, etc., son plantillas predefinidas que puede cargar en el panel de configuración, tras lo cual podrá modificarlas según sus necesidades y, posteriormente, actualizar en la configuración. También puede utilizar la opción **Cargar plantilla...** en el sistema de archivos local.

Los cambios realizados en este panel se almacenan en el archivo de configuración (cuando se exporte el proyecto) y son utilizados por todas sus líneas de ensamblaje, a pesar de los valores del servidor en el que se ejecuten.

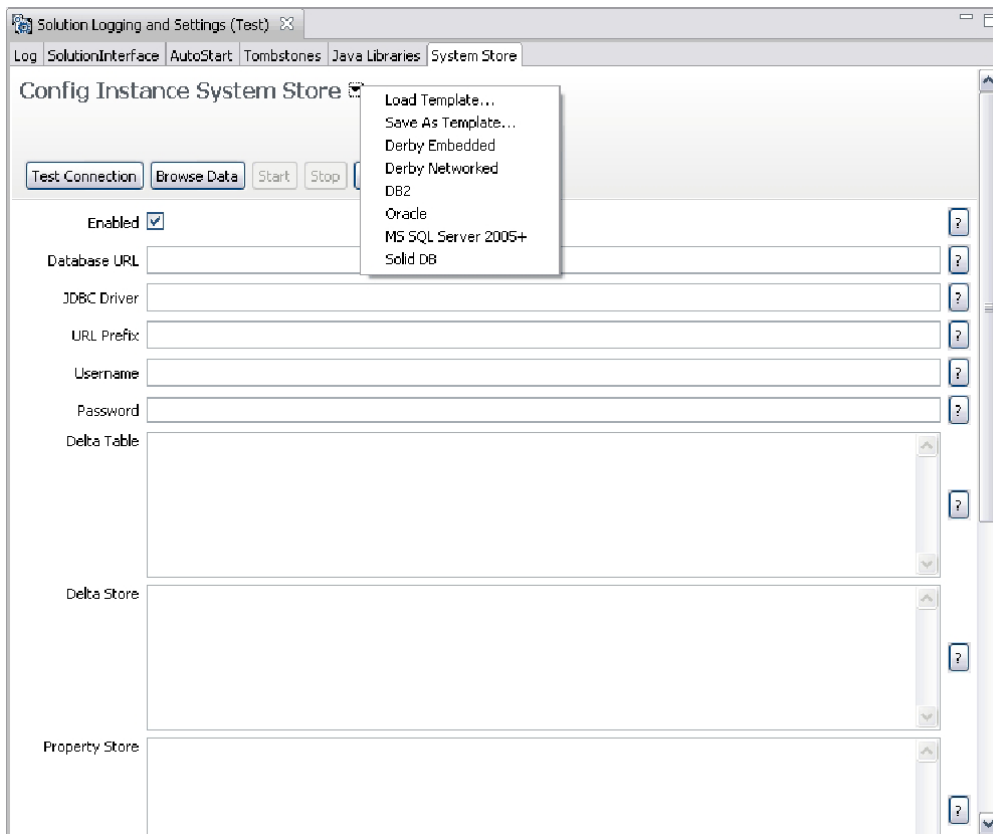


Figura 99. valores del almacén del sistema de configuración

Valores de nivel de servidor

Aparece la misma pantalla cuando selecciona **Editar valores de almacén del sistema** en el menú desplegable de un servidor en la vista Servidores. Esto actualizará y guardará los valores de almacén del sistema en el archivo de propiedades de solución. Es necesario reiniciar el servidor para activar los valores nuevos.

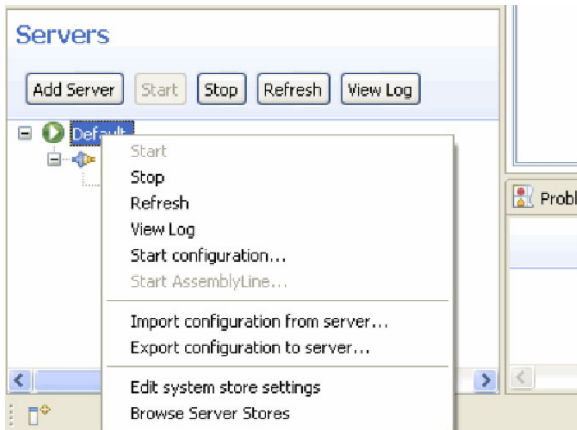
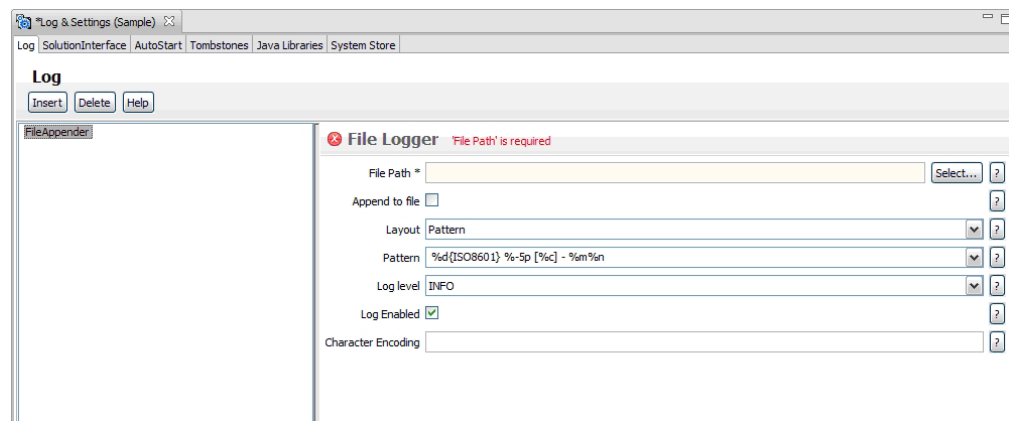


Figura 100. Menú contextual del documento de servidor

Registro cronológico

La vista **Registro cronológico** muestra los registradores para la solución.

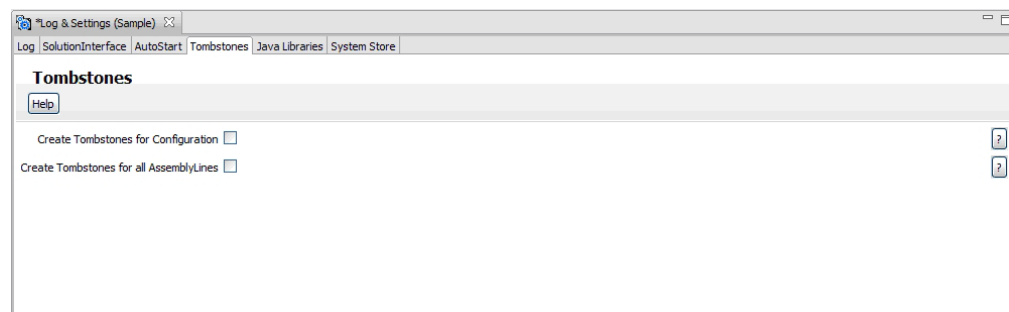
Puede añadir y eliminar registradores pulsando **Insertar** y **Suprimir** en la barra de título.



Para obtener más información, consulte la sección sobre registro y depuración en la publicación *Instalación y administración*.

Lápidas

La configuración de lápidas para un proyecto se encuentra en la pestaña **Lápidas**.



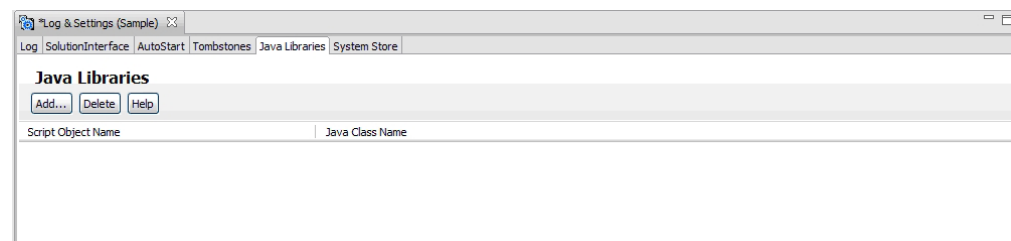
Una lápida es un registro de una ejecución de una línea de ensamblaje, que contiene estadísticas tales como el número de entradas leídas por iteradores, el número de registros omitidos, el número de registros actualizados, etc.

La selección de **Crear lápidas para configuración** hace que se genere una lápida cada vez que termina el archivo de configuración derivado de este proyecto, cargado en un servidor de IBM Security Directory Integrator.

La selección de **Crear lápidas para todas las líneas de ensamblaje** hace que se genere una lápida cada vez que termina una línea de ensamblaje de este proyecto, ejecutada en un servidor.

Bibliotecas Java

La pestaña **Bibliotecas de Java** muestra las clases Java que se cargan y definen de forma automática en cada instancia del motor de scripts.



Inicio automático

La pestaña **Inicio automático** muestra una lista en la que puede especificar el nombre de las líneas de ensamblaje que se inician automáticamente cuando se inicia la instancia de configuración.



Figura 101. Valores de Inicio automático

Puede añadir elementos a la lista pulsando **Insertar**; esto le permite elegir entre las líneas de ensamblaje existentes en el espacio de trabajo.

Puede eliminar líneas de ensamblaje en la lista Elementos de inicio seleccionando las líneas y pulsando **Suprimir**.

Valores de interfaz de la solución

Los valores de interfaz de la solución se pueden habilitar para proporcionar información adicional sobre la configuración. Esta información generalmente la utiliza la herramienta Webadmin (AMC) pero pueden utilizarla otros clientes que tengan acceso a la configuración de interfaz de la solución.

Los dos primeros campos de este panel son el nombre de solución y el estado habilitado. El valor predeterminado para el nombre de solución es el mismo nombre de proyecto; si el nombre de solución se deja en blanco, el archivo de configuración exportado no contendrá ningún ID de solución. El recuadro de

selección habilitado determina si la configuración se está utilizando o no.

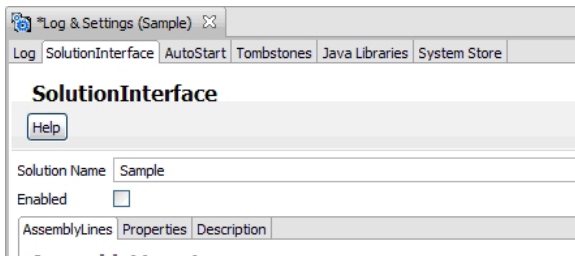


Figura 102. Valores de interfaz de la solución

Los valores de Interfaz de la solución se dividen en tres secciones, cada una de ellas con su correspondiente pestaña:

- “Líneas de ensamblaje”
- “Propiedades” en la página 187
- “Descripción” en la página 187

Líneas de ensamblaje

Esta pestaña muestra todas las líneas de ensamblaje de proyecto y se pueden seleccionar las que deben estar visibles para el usuario para iniciar/detener.

La línea de ensamblaje de estado es una línea de ensamblaje especial que se utiliza para informar del estado de la configuración en ejecución. Esta línea de ensamblaje simplemente proporciona comentarios personalizados al usuario sobre el estado de la configuración. La línea de ensamblaje se llama y debería devolver dos campos en su entrada work para informar del estado (`healthAL.result` y `healthAL.status`). Consulte la documentación de Action Manager (AM) (en línea en AMC o en la publicación *Instalación y administración*) para obtener más información sobre se utilizan estos campos en este contexto. El intervalo de sondeo especifica con qué frecuencia el cliente (por ejemplo, AMC o AM) llamará a la línea de ensamblaje de estado.

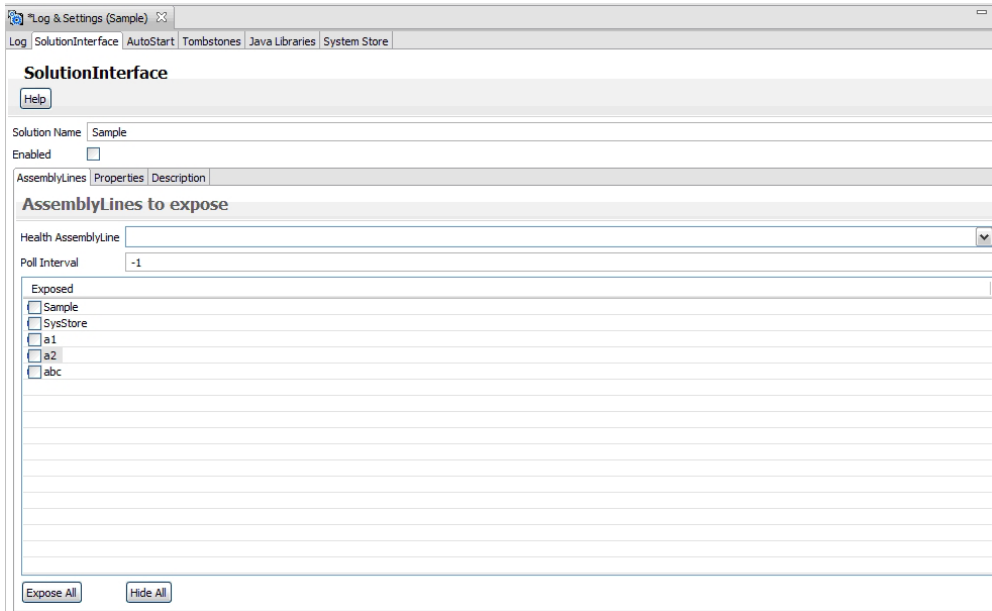


Figura 103. Valores de interfaz de la solución: Líneas de ensamblaje

Propiedades

Esta pestaña le permite definir qué propiedades ven los usuarios y cómo se presenta al usuario.

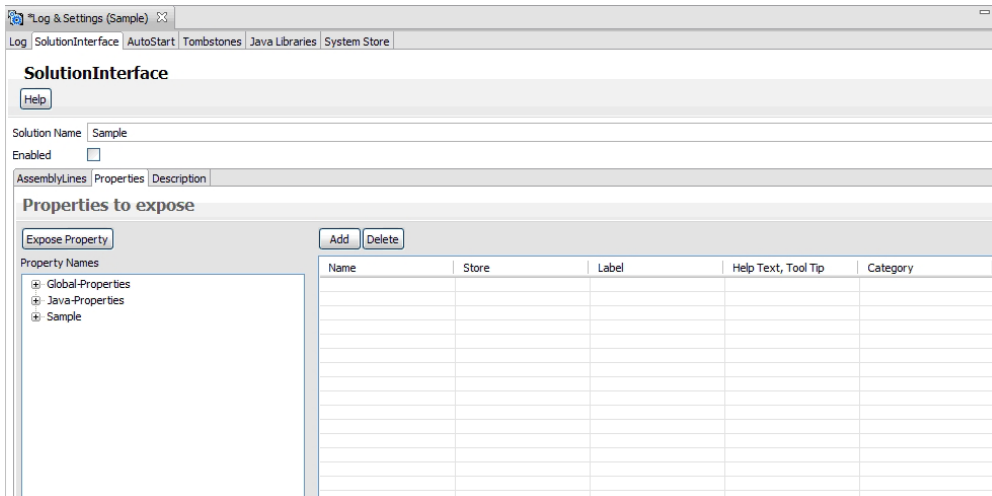


Figura 104. Valores de interfaz de la solución: Propiedades

Descripción

Esta pestaña le permite escribir un texto que describe la solución. Sólo tiene una finalidad de documentación; IBM Security Directory Integrator no lo utiliza de ningún otro modo.

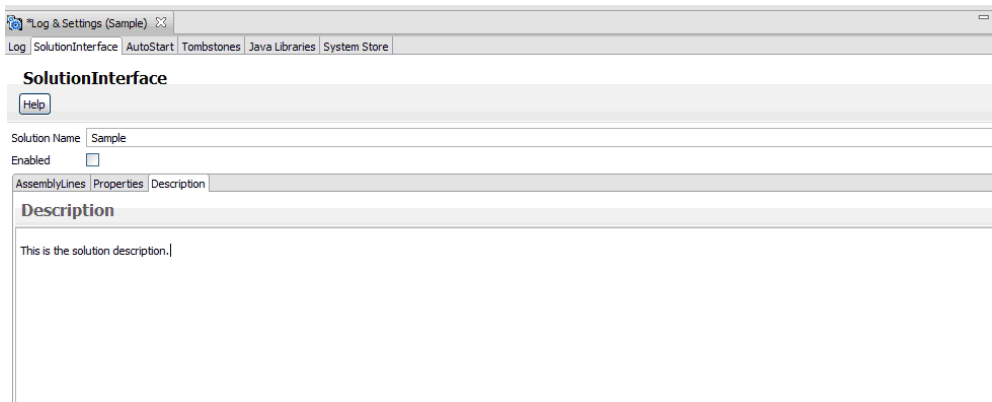


Figura 105. Valores de interfaz de la solución: Descripción

Propiedades de servidor

Puede editar las propiedades para un proyecto utilizando el editor de propiedades.

Cree un nuevo archivo de propiedades utilizando el asistente **Archivo > Nuevas > Propiedades** y escriba el nombre de un almacén de propiedades.

En el editor de propiedades puede intercambiar el contenido del almacén de propiedades utilizando los mandatos **Descargar** y **Cargar**:

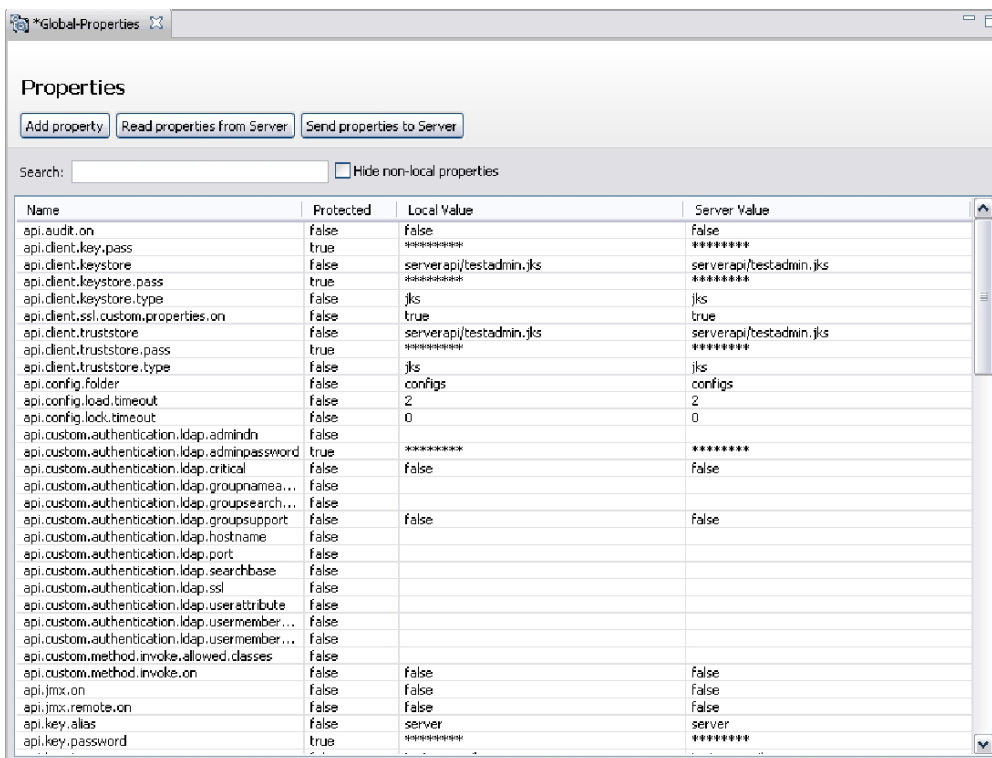


Figura 106. Ventana Editor de propiedades

Utilice **Descargar** para recuperar todas las propiedades del almacén de propiedades (por ejemplo, el archivo de propiedades asignado a este almacén).

Tenga en cuenta que estos valores se leen y pasan al CE desde el servidor IBM Security Directory Integrator actualmente seleccionado para el proyecto. En cambio, el botón **Cargar** actualiza el almacén de propiedades (a través del servidor actual) con los valores en el editor. Sólo se actualizan las propiedades que tienen un valor local.

Utilice el campo de texto de **Búsqueda** para mostrar las propiedades que corresponden al texto en este campo. Si selecciona **Ocultar propiedades no locales**, el editor sólo muestra las propiedades que tienen un valor local.

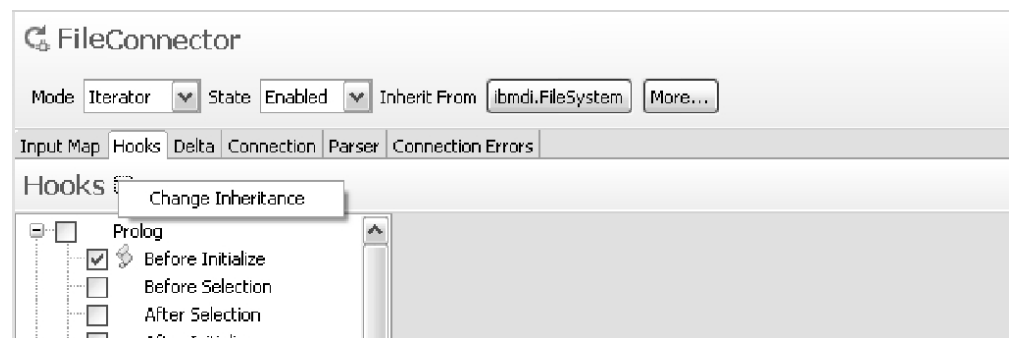
El creador de proyectos incluirá la configuración del almacén de propiedades en el archivo de configuración ejecutable. Sin embargo, los valores de propiedades de este documento sólo se transfieren si es necesario.

Nota: El orden en que se inicializan los almacenes de propiedades y en que se accede a ellos en el servidor de IBM Security Directory Integrator no está definido. Por lo tanto, las propiedades que definen parámetros de acceso (por ejemplo, nombres de archivo) de otros almacenes de propiedades no se pueden almacenar de forma fiable en un almacén de propiedades.

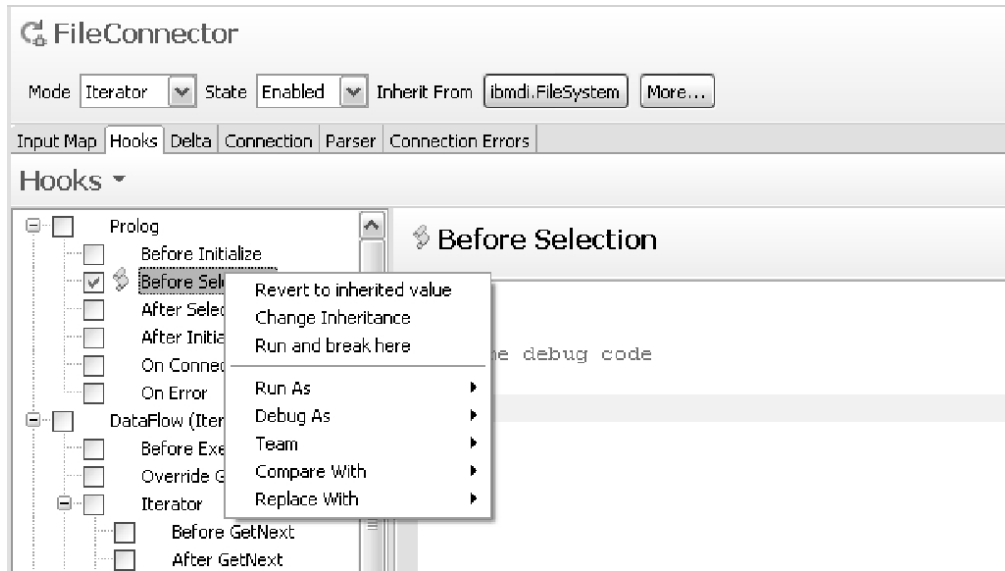
Herencia

Los componentes pueden heredar elementos de otros componentes arrastrando y soltando un objeto hasta la barra de título de un componente o una subsección del componente.

Para subsecciones también existe una opción de menú (**Cambiar la herencia**) disponible en la barra de título.



Para complementos de software y correlaciones de atributos existe una selección de herencia separada de elementos individuales donde puede seleccionar para heredar de un componente de script o de función en el espacio de trabajo.

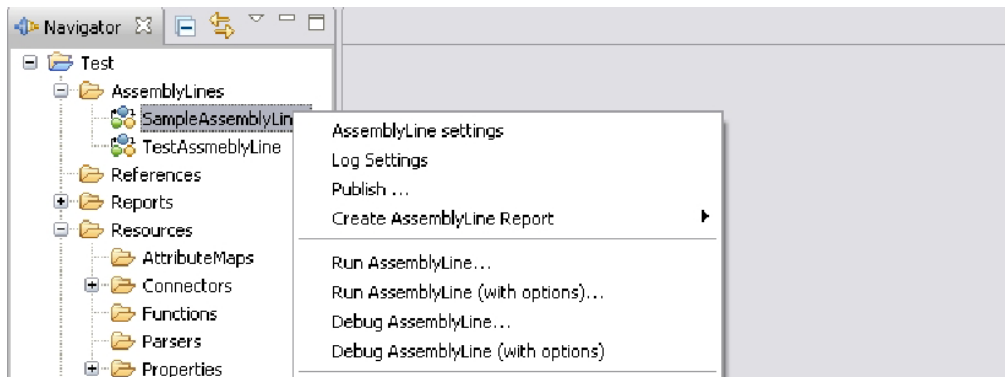


Cuando un elemento de configuración sustituye a un elemento heredado, la interfaz de usuario proporciona un modo para revertir al valor heredado mediante un elemento de acción de menú desplegable: **Revertir al valor heredado**.

Acciones y enlaces de teclas

El CE proporciona varias acciones para sí mismo además de para los objetos del entorno de trabajo. Estas acciones realizan operaciones específicas en objetos específicos.

Por ejemplo, **Ejecutar un informe de línea de ensamblaje** es una acción proporcionada para todos los archivos con una extensión `.assemblyline`. Cuando se pulsa con el botón derecho del ratón en una línea de ensamblaje en el navegador, el menú desplegable también incluirá este mandato además de otras contribuciones a este tipo de objeto.



Estas acciones también tienen asociada una definición de mandato. Una definición de mandato permite definir al usuario el atajo de teclado para un mandato. Esto se lleva a cabo en el panel **Ventana > Preferencias > Teclas**:

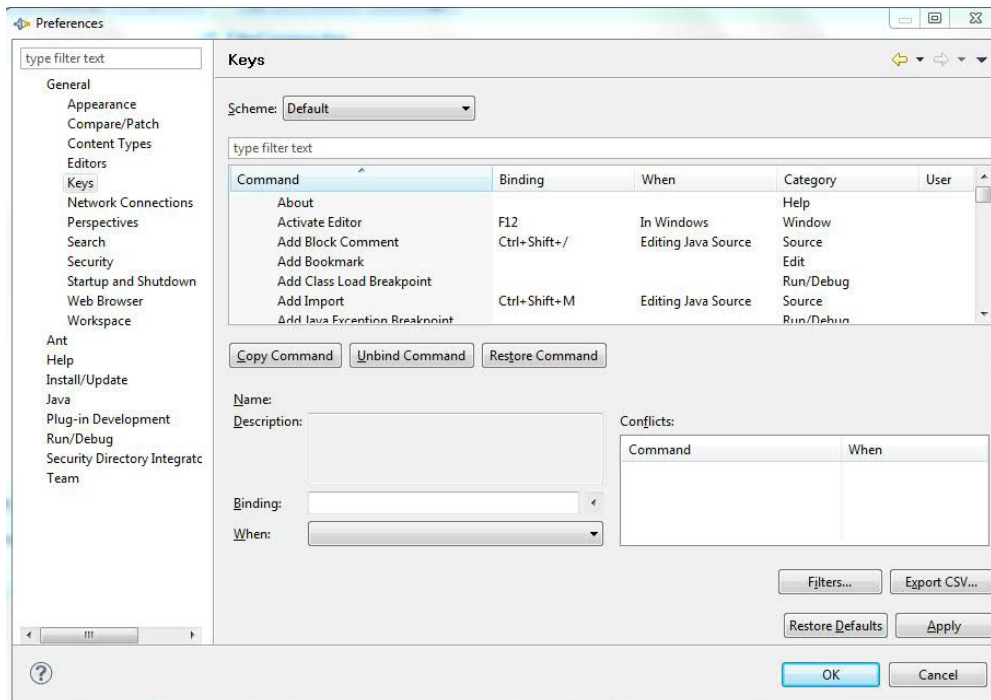
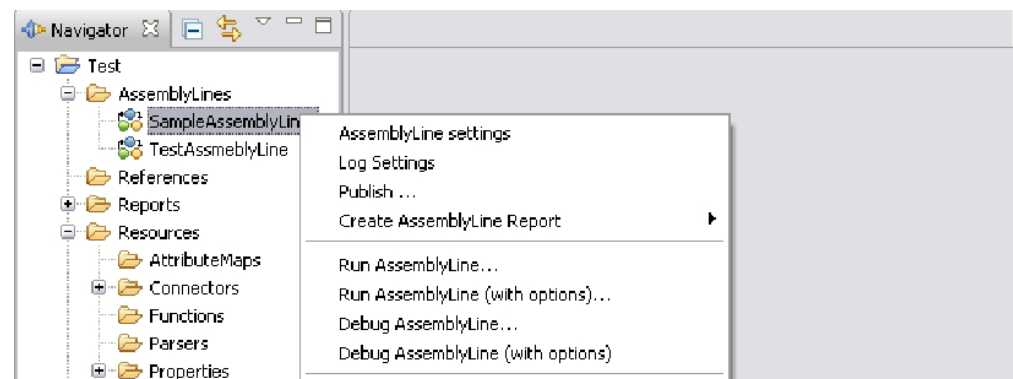


Figura 107. Ventana Asignaciones de teclas

La imagen anterior muestra cómo se llevan a cabo las asignaciones del teclado en la interfaz de usuario. En este ejemplo, se ha asignado **Alt+Mayúsculas+I** como atajo para el mandato Ejecutar informe. Cuando vuelva a abrir el menú en la línea de ensamblaje, verá esto reflejado en el menú.



Puede obtener una lista de todos los mandatos específicos de IBM Security Directory Integrator escribiendo el texto **security directory integrator** en el campo de búsqueda, que encontrará debajo del selector **Esquema**.

Capítulo 4. Características de depuración de IBM Security Directory Integrator

Una vez creada la solución de IBM Security Directory Integrator, generalmente utilizando el Editor de configuración (CE), existen varias maneras de probarla. Algunas de las características de depuración que proporciona IBM Security Directory Integrator se pueden utilizar desde dentro del Editor de configuración y algunas están basadas en scripts contenidos en el directorio de instalación de IBM Security Directory Integrator.

Las características de depuración de IBM Security Directory Integrator son el recinto de seguridad, la modalidad de simulación de línea de ensamblaje, el repetidor y el depurador.

El repetidor y el depurador forman parte del Editor de configuración; consulte “El repetidor y el depurador” en la página 162.

Recinto de seguridad

IBM Security Directory Integrator incluye una función de recinto de seguridad que le permite grabar la operación de uno o más conectores en una línea de ensamblaje para responder posteriormente sin que los orígenes de datos necesarios estén disponibles.

La función Recinto de seguridad utiliza el almacén del sistema. Consulte el apartado Capítulo 6, “Almacén del sistema”, en la página 211 para obtener más información.

Grabar un componente significa que la línea de ensamblaje va a interceptar cada llamada a la instancia de conector que el componente utiliza. Por ejemplo, si se graba un componente de conector, se grabarán todas las llamadas a sus métodos de instancia de conector tales como `selectEntries`, `getNextEntry`, etc. El resultado de cada una de estas llamadas se graba en la base de datos de recinto de seguridad configurada. La información grabada es el valor de retorno o la excepción generada por el conector.

Para ejecutar una sesión de prueba, cree una línea de ensamblaje y añada un conector del sistema de archivos que lea datos de un archivo. Añada un componente de script que realice un volcado de la entrada `work`. A continuación, compruebe el conector de sistema de archivos y seleccione **Ejecutar/Grabar** en el menú del botón Ejecutar. Después de hacer esto, puede mover el archivo que se acaba de leer y ejecutar la línea de ensamblaje en modalidad Reproducción. Debería ver la misma salida del registro aunque el conector de archivo normalmente termine anormalmente puesto que ya no se puede acceder al archivo.

Esta función puede ser muy útil cuando se proporciona material de soporte. Con frecuencia, el tiempo para reproducir el entorno para una línea de ensamblaje y el estado de los orígenes de datos para reproducir una condición puede ser bastante extenso. Con una base de datos de recinto de seguridad con una sesión grabada, una persona del servicio de soporte puede ejecutar la línea de ensamblaje sin tener acceso a todos los almacenes de datos que la línea de ensamblaje necesita. Además, la configuración de la línea de ensamblaje se puede modificar para imprimir más información si es necesario. El único cambio que no se puede realizar en la

configuración de la línea de ensamblaje es efectuar llamadas adicionales o reordenar las llamadas a componentes grabados. Esto produciría un error durante la reproducción puesto que las llamadas al conector no coincidirían con la llamada esperada siguiente al conector.

Antes de grabar o reproducir una línea de ensamblaje, debe indicar a IBM Security Directory Integrator dónde deben almacenarse los datos de grabación de la línea de ensamblaje. Esto se lleva a cabo en la ventana Recinto de seguridad, que se puede abrir seleccionando **Valores de línea de ensamblaje... > Valores de recinto de seguridad** en el editor de líneas de ensamblaje. En la parte superior de esta ventana hay un campo con la etiqueta **Base de datos** donde puede escribir la vía de acceso del directorio para que lo utilice el sistema.

El recurso de recinto de seguridad no está soportado en líneas de ensamblaje que contengan un conector en modalidad Servidor o un conector Iterador con Delta habilitado. El servidor terminará anormalmente la ejecución de la línea de ensamblaje si descubre esta situación.

Registro de entradas de la línea de ensamblaje

Grabar un componente significa que la línea de ensamblaje va a interceptar cada llamada a la instancia de conector que el componente utiliza. Por ejemplo, si se graba un componente de conector, se grabarán todas las llamadas a sus métodos de instancia de conector tales como `selectEntries`, `getNextEntry`, etc. El resultado de cada una de estas llamadas se graba en la base de datos de recinto de seguridad configurada. La información grabada es el valor de retorno o la excepción generada por el conector.

Para ejecutar una sesión de prueba, cree una línea de ensamblaje y añada un conector del sistema de archivos que lea datos de un archivo. Añada un componente de script que realice un volcado de la entrada `work`. A continuación, compruebe el conector de sistema de archivos y seleccione **Ejecutar/Grabar** en el menú del botón **Ejecutar**. Después de hacer esto, puede mover el archivo que se acaba de leer y ejecutar la línea de ensamblaje en modalidad Reproducción. Debería ver la misma salida del registro aunque el conector de archivo normalmente termine anormalmente puesto que ya no se puede acceder al archivo.

Reproducción del recinto de seguridad de los registros de la línea de ensamblaje

Cuando una línea de ensamblaje está en modalidad Recinto de seguridad, se dice que todos los conectores establecidos para la reproducción están en *modalidad virtual*. Esto significa que sus operaciones de interfaz de conector (por ejemplo, `getNext()`, `findEntry()`) en realidad no se llaman. En su lugar, estas operaciones se simulan durante la reproducción.

Para poder ejecutar una línea de ensamblaje en modalidad Reproducción, debe seleccionar los conectores que se van a ejecutar en modalidad virtual mediante el recuadro de selección **Reproducción habilitada** de la ventana de valores del **Recinto de seguridad** de la línea de ensamblaje.

Nota: No es necesario habilitar todos los conectores grabados para la reproducción. Puede habilitarlos para que accedan a los orígenes de datos activos, aunque es posible que esto afecte el resultado de la operación de reproducción.

Para ejecutar una línea de ensamblaje desde la línea de mandatos, inicie el servidor con el conmutador **-q2**. Una línea de ensamblaje en modalidad Recinto de

seguridad (incluida su entrada work inicial) procedente de un conjunto de datos grabado. Por ejemplo, si tiene un conector JMS (Java Messaging Service) en la línea de ensamblaje en modalidad Recinto de seguridad, el conector JMS recupera la entrada de los datos grabados anteriormente y realmente no se inicializa nunca.

Cuando se registra una línea de ensamblaje, el servidor crea una base de datos Derby en el directorio **Base de datos** especificado utilizando el nombre de la línea de ensamblaje como nombre de la base de datos. Esta base de datos contiene las tablas para cada conector de la línea de ensamblaje. Se pueden sustituir uno o más conectores virtuales de una línea de ensamblaje en modalidad Recinto de seguridad renombrando el conector grabado y añadiendo, a continuación, uno nuevo con su nombre original.

Modalidad de simulación de línea de ensamblaje

Las líneas de ensamblaje se pueden depurar sin que estén realmente intercambiando datos con sistemas conectados utilizando la modalidad de simulación de línea de ensamblaje. Cuando una línea de ensamblaje se ha iniciado en esta modalidad, todos los componentes de línea de ensamblaje que potencialmente podrían producir cambios en los sistemas de destino se omitirán.

Esta modalidad implica que la línea de ensamblaje se ejecuta normalmente, pero los conectores en modalidad Actualizar, Delta, Suprimir y Sólo adición no realizan la operación real.

Nota: Una línea de ensamblaje ejecutada de este modo sólo puede aproximarse a lo que sucedería en modalidad normal; en muchos casos el hecho de que un sistema conectado no reciba ninguna actualización en modalidad de simulación puede hacer que la lógica empresarial se comporte diferente e invalidar la utilidad de la simulación.

El estado de simulación no debería confundirse con el estado de un componente. El estado de un componente tiene una prioridad más alta que el estado de simulación del componente.

El estado del componente tiene dos valores – Habilitado o Inhabilitado. Si el componente está en estado Habilitado, se inicializará y su estado de simulación se seleccionará cuando se invoque la operación adecuada. Cuando el estado está establecido en Inhabilitado, el estado de simulación no se seleccionará puesto que no se invocará ninguna operación y no se invocará la inicialización del componente.

Los conectores y los componentes de función (FC) tienen otro estado denominado Pasivo. Cuando su estado se establece en Pasivo, únicamente se inicializarán. Sus operaciones sólo se pueden ejecutar mediante un script de usuario. Cuando se invoca su operación específica, se selecciona el estado de simulación.

Existen varias formas de iniciar en modalidad de simulación una línea de ensamblaje:

Desde el Editor de configuración:

Hay un recuadro de selección en el menú desplegable Modalidad de ejecución que se puede utilizar para habilitar e inhabilitar la simulación para la línea de ensamblaje. Debe elegir la modalidad de ejecución Ejecutar con opciones en el menú desplegable y seleccionar el recuadro de selección **Modalidad de simulación** en el recuadro de diálogo emergente Opciones

para activar la simulación de línea de ensamblaje durante la ejecución. El estado predeterminado de este recuadro de selección es que esté sin seleccionar.

Utilizando el mandato de inicio de servidor `ibmdisrv`:

Este mandato reconoce el conmutador `-M` e iniciará la línea de ensamblaje con la simulación activada si se proporciona este conmutador.

Utilizando la API:

Para que una línea de ensamblaje se ejecute en modalidad de simulación, debe establecer la propiedad `AssemblyLine.TCB_SIMULATE_MODE` en `true` en el objeto `TCB`. A continuación, este objeto debe proporcionarse al método `startAssemblyLine(String, TaskCallBlock)`. Si no se establece ninguna propiedad, de forma predeterminada se considera que su valor es `false`.

Para ejecutar una línea de ensamblaje en modalidad de simulación, se crea una nueva configuración. Es una configuración hija de la configuración de la línea de ensamblaje. Este objeto de configuración contiene parámetros que configuran la conexión con una línea de ensamblaje que se utilizará como línea de ensamblaje de proxy (`ProxyAL`). El objeto `SimulationConfig` tiene un método que crea o actualiza una plantilla de una línea de ensamblaje de proxy basada en los estados de los componentes de la línea de ensamblaje que se está simulando. El objeto `SimulateConfig` también contiene todos los complementos de software definidos para los componentes que están en estado de simulación `En script`. El nombre de cada complemento de software es igual que el nombre del componente en estado de simulación `En script`. También contiene el estado de simulación para cada componente.

Las líneas de ensamblaje que se van a ejecutar en modalidad de simulación se pueden configurar más detalladamente; los valores pertinentes se pueden configurar seleccionando **Valores de línea de ensamblaje > Valores de simulación** en la ventana Editor de líneas de ensamblaje.

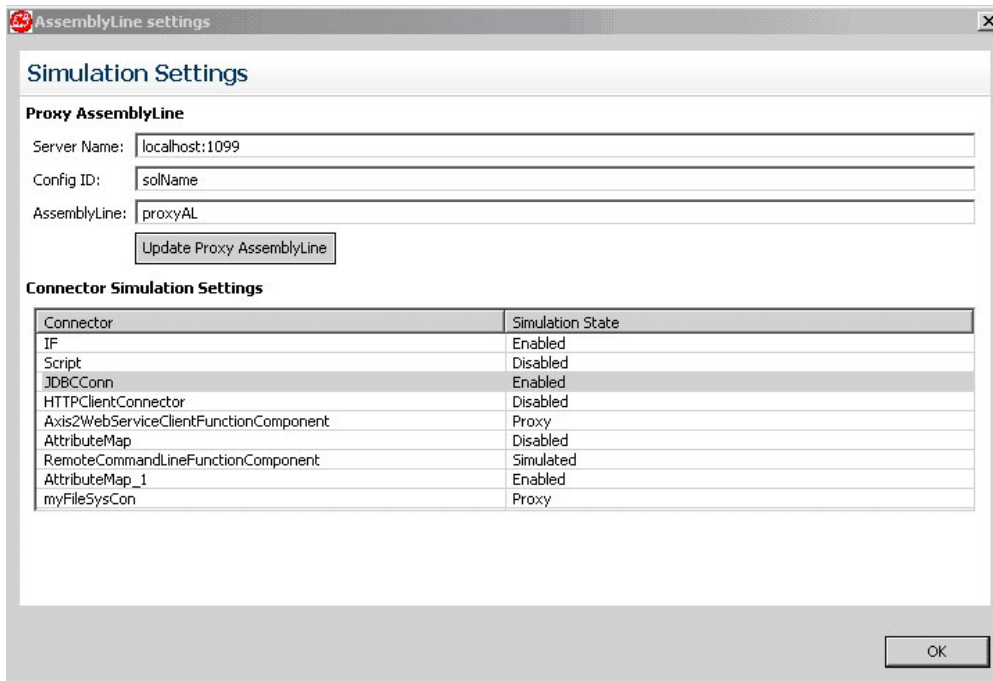


Figura 108. Ventana Valores de simulación

La configuración de los estados de simulación para cada componente se lleva a cabo utilizando el recuadro de diálogo Valores de simulación. Este recuadro de diálogo configura la línea de ensamblaje de proxy que utilizarán los componentes cuyo estado de simulación está establecido en *Línea de ensamblaje de proxy*. Si pulsa **Actualizar línea de ensamblaje de proxy**, el Editor de configuración creará una nueva línea de ensamblaje de proxy en el proyecto actual o actualizará una línea de ensamblaje de proxy existente. La línea de ensamblaje de proxy creada o actualizada se proporciona como una plantilla y su estructura se basa en la configuración que se ha realizado en el recuadro de diálogo Valores de simulación. Tenga en cuenta que en este proceso sólo se tiene en cuenta el nombre de la línea de ensamblaje de proxy. El nombre de servidor y el ID de configuración se tienen en cuenta durante la ejecución de la línea de ensamblaje simulada. Si ya existe una línea de ensamblaje con el nombre especificado en el recuadro de diálogo, sólo se añadirán las ramas nuevas y no se modificará ni eliminará ninguna rama anterior. Esto se debe a que algunas pueden contener una configuración específica del usuario. Los componentes individuales de la línea de ensamblaje se pueden establecer en uno de los estados siguientes:

Habilitado

Equivale a ejecutar este componente en la línea de ensamblaje en modalidad normal (es decir, el componente se ejecuta como lo haría normalmente).

Inhabilitado

Equivale a inhabilitar el componente (es decir, no se ejecutan operaciones, ni complementos de software ni nada más, excepto la inicialización para el componente).

Simulado

Generalmente, las estadísticas para todos los componentes que se describen a continuación contienen información para la operación que se hubiera completado si la línea de ensamblaje no estuviera simulando. Puesto que no se realiza ninguna operación crítica durante la simulación, no hay modo

de predecir cuál sería el resultado de la ejecución de una operación crítica (operación satisfactoria o de error), de modo que las estadísticas indicarán que la operación se ha completado satisfactoriamente.

- Conectores en modalidad Sólo adición: se ejecutan del modo normal, sólo se omite la llamada al método `connector.putEntry()` y la llamada al complemento de software `override_add`, potencialmente inseguras.
- Conectores en modalidad Actualizar: se ejecutan del modo normal, sólo se omiten la llamada al método `connector.modEntry()` y la llamada al complemento de software `override_modify`, potencialmente inseguras.
- Conectores en modalidad Suprimir: se ejecutan del modo normal, sólo se omite la llamada al método `connector.deleteEntry()` y la llamada al complemento de software `override_delete`, potencialmente inseguras.
- Conectores en modalidad Delta: se ejecutan del modo normal, pero debido a que este conector se basa en llamadas a los métodos previamente descritos, se simulará cuando se omitan los métodos y complementos de software anteriores.
- Conectores en modalidad Iterador con código delta: se ejecutan del modo normal; para estos componentes el cambio es similar a los conectores previamente descritos, es decir, se omiten las llamadas potencialmente inseguras a los métodos de la clase `BTree` (`putEntry`, `modEntry`, `deleteEntry`). Para `CDDeltaTaskComponent`, el estado de compromiso se altera temporalmente para inhabilitar el compromiso (“Sin compromiso automático”), pero el compromiso aún será posible para los usuarios que llamen explícitamente al método `CSDeltaTaskComponent#commitDeltaState()`.
- Componentes de función (FC): se ejecutan del modo normal, pero la invocación de los complementos de software “`before_functioncall`”, “`after_functioncall`” y “`no_reply`”, así como la llamada al método `function.perform()` están inhabilitadas. La invocación de estos complementos de software está inhabilitada porque están asociados a un objeto que se devuelve del método `perform`, que también está inhabilitado. Lo único que se realizará es un cambio en las estadísticas que indicará que el componente de función ha ejecutado satisfactoriamente esta operación.

Las correlaciones de entrada y salida también se ejecutarán pero la entrada real antes de la correlación de entrada será una entrada vacía (dado que esto podría hacer que se genere una excepción, es mejor inhabilitar cada correlación de atributos simple de la correlación de entrada basada en un atributo que se devuelve del componente de función o añadir una comprobación de la validez del atributo recuperado en la correlación de atributos avanzada; como alternativa, puede utilizar el mecanismo `NullBehaviour` para anular los errores potenciales).

- Los conectores en cualquier otra modalidad se ejecutan del modo habitual.

Proxy Los conectores en este estado de simulación iniciarán otra línea de ensamblaje (especificado en la pestaña Simulación) que anulará la ejecución potencialmente insegura de la operación. Esta línea de ensamblaje externa se comparte entre todos los componentes en esta modalidad de simulación. Se ejecutará con una operación diferente en la que el nombre coincide con el nombre del componente que se está simulando. Consulte el apartado “Flujo de trabajo de línea de ensamblaje de proxy” en la página 200.

En script

Un script definido por el usuario anulará la operación potencialmente insegura. Cada componente en esta modalidad de simulación tiene su propio complemento de software, a diferencia el estado Proxy en que una línea de ensamblaje se comparte entre los componentes. Consulte el apartado “Flujo de trabajo de script de simulación” en la página 201.

Dispone de la posibilidad de comprobar si la línea de ensamblaje está simulando y también de conmutar entre activar y desactivar la simulación utilizando los métodos siguientes:

- `boolean AssemblyLine#isSimulating()` Utilizado del modo siguiente desde un complemento de software: `task.isSimulating()`;
- `void AssemblyLine#setSimulating(boolean)` Utilizado del modo siguiente desde un complemento de software: `task.setSimulating(true)`;

Dispone de la posibilidad de comprobar el estado de cada componente y de establecerlo de forma dinámica utilizando los métodos siguientes:

- `String AssemblyLineComponent.getSimulatingState()` Utilizado del modo siguiente desde un complemento de software:
`ConnectorName.getSimulatingState()`;
- `void AssemblyLineComponent.setSimulatingState(String)` Utilizado del modo siguiente desde un complemento de software:
`ConnectorName.setSimulatingState("Proxy")`;

Nota: Tan solo los conectores y los componentes de función tienen el conjunto completo de estados de simulación que se ha descrito antes, el resto de componentes (y conectores en modalidad Servidor) sólo tienen los estados Habilitado e Inhabilitado.

IBM Security Directory Integrator considera algunos componentes de función seguros y su estado de simulación predeterminado será Habilitado, es decir, de forma predeterminada no simularán y se ejecutarán del modo habitual; se trata de todos los componentes de función que no pueden cambiar un sistema de destino porque simplemente no se conectan a ninguno. La lista de componentes de función seguros es la siguiente:

- FC CBE
- FC JavaToXML
- FC XMLToJava
- FC SDOToXML
- FC XMLToSDO
- FC de analizador
- FC MemQueue
- FC JavaToSOAP
- FC SOAPToJava
- FC WrapSOAP

Atención: Todavía puede producir cambios en sistemas de datos subyacentes utilizando código explícito; esto está fuera del alcance de la modalidad de simulación.

Los componentes de función (FC) restantes que no aparecen en la lista se consideran potencialmente inseguros y su estado de simulación predeterminado se establecerá en Simulado.

Flujo de trabajo de línea de ensamblaje de proxy

En el contexto de modalidad de simulación, la llamada a una línea de ensamblaje de proxy funciona de modo similar a como se utiliza el conector de línea de ensamblaje para controlar la línea de ensamblaje que haga elegido, sin embargo existen algunas diferencias significativas.

La llamada al complemento de software de alteración temporal de la operación específica está inhabilitada cuando el componente está en estado de simulación de proxy.

La tabla siguiente muestra los métodos que se llaman cuando el conector está en una de esas modalidades o cuando el componente es un componente de función.

Tabla 10. Invocación de método según la modalidad

Modalidad	Método
Conector: Sólo adición	putEntry
Conector: Actualizar	findEntry, modEntry, putEntry
Conector: Suprimir	findEntry, deleteEntry
Conector: Delta	findEntry, modEntry, putEntry, deleteEntry
Conector: Iterador	selectEntries, getNextEntry
Conector: CallReply	queryReply
Conector: Buscar	findEntry
Componente de función	perform

Cuando llega el momento de llamar a un método específico y el componente está en estado de simulación de proxy, la llamada a este método se delegará a la línea de ensamblaje de proxy. Cuando la línea de ensamblaje de proxy se inicia, se le pasa una entrada op con los atributos siguientes:

- \$operation – este atributo contiene el nombre de la operación que debe ejecutarse. Cuando se llama a la línea de ensamblaje de proxy en lugar de un método específico de un componente, el valor de este atributo será igual que el nombre del componente.
- \$method – este atributo contiene el nombre del método que se llamaría normalmente, pero puesto que el componente está en estado de simulación de proxy y los componentes en algunas modalidades (por ejemplo, Actualizar) ejecutan varios métodos antes de llevar a cabo realmente la modificación (es decir, findEntry), la línea de ensamblaje de proxy debe reconocer el método que debe implementarse. El atributo \$method sólo indica a la línea de ensamblaje de proxy qué método debe ejecutarse en su lugar para que la línea de ensamblaje de proxy pueda manejar la operación correctamente.
- search – este atributo está disponible cuando el atributo \$method es findEntry. Su valor es un objeto de tipo SearchCriteria y representa los criterios de búsqueda definidos por el usuario. Por ejemplo, si el componente está en estado de simulación de proxy y su modalidad es Actualizar, deben definirse criterios de búsqueda para poder actualizar la entrada correcta en el sistema de destino. Puesto que el estado de simulación es Proxy, la operación de búsqueda real que tiene lugar antes de la operación de modificación delega su ejecución a la línea

de ensamblaje de proxy. A continuación, la línea de ensamblaje de proxy utiliza estos criterios de búsqueda para una simulación de búsqueda correcta.

- *current* – este atributo sólo está disponible cuando el atributo *\$method* es *modEntry*. Su valor es un objeto de entrada que representa la entrada que se encuentra en el sistema de destino antes de que se produzca la modificación real. Es la entrada que devuelve el método *findEntry*, que se ejecuta antes del método *modEntry*.

Se pasa una entrada *work* inicial (IWE) a la línea de ensamblaje de proxy cuando se llama. Cuando *\$method* es *findEntry*, *selectEntry* o *getNextEntry*, la IWE es una copia de la entrada *work* de la línea de ensamblaje que realiza la llamada. En los demás casos, la IWE es la entrada que se recupera del procedimiento de correlación de salida (también conocido como entrada *conn*). En concreto, para la operación *deleteEntry*, la IWE es la entrada que se recupera de la operación *findEntry* anterior.

Una vez realizada la ejecución de la línea de ensamblaje de proxy y *\$method* es *findEntry*, la entrada resultante se selecciona para el atributo *conn*. Si está disponible, se supone que este atributo contiene todas las entradas encontradas de la operación *findEntry* y en función de su valor, se llamará a los complementos de software adecuados, es decir, *no_match* (ninguna coincidencia) y *multiple_match* (varias coincidencias). Si no se encuentra ningún atributo con el nombre *conn*, la entrada resultante de la ejecución de la línea de ensamblaje de proxy se trata como la entrada que encuentra el *\$method* *findEntry*. La entrada recuperada de la línea de ensamblaje de proxy que sustituye al *\$method* *selectEntries* se fusiona automáticamente con la entrada *work* de la línea de ensamblaje que realiza la llamada. La entrada recuperada de la línea de ensamblaje de proxy que simula los métodos que esperan una entrada (es decir, *findEntry*, *getNextEntry*, *queryReply* y *perform*) se envía a la correlación de entrada definida. Para todos los demás métodos que no se espera que devuelvan un resultado, la entrada de la línea de ensamblaje de proxy se ignora.

Flujo de trabajo de script de simulación

Cada componente en el que el estado de simulación se ha establecido en En script tiene un Script de simulación (SS) definido en la pestaña Simular.

A continuación se proporciona una lista de objetos que se muestran para utilizarse directamente desde un SS:

- *work* – la entrada *work*.
- *conn* – la entrada recuperada de la operación de búsqueda o directamente desde la correlación de salida o nulo.
- *resEntry* – la entrada que se utiliza como resultado si la operación que se simula necesita que se devuelva un resultado; de lo contrario, si el resultado no se utilizará, es nulo.
- *current* – la entrada que está en el sistema de destino que se modificará o nulo.
- *search* – el objeto *SearchCriteria* definido por el usuario o nulo.
- *method* – un objeto *String* que contiene el nombre del método que se altera temporalmente mediante el SS.

La tabla siguiente explica los métodos que se simulan y muestra cuándo se invoca el SS.

Tabla 11. Invocación de método según la modalidad

Modalidad	Método
Conector: Sólo adición	putEntry
Conector: Actualizar	modEntry o putEntry si no se encuentra la entrada
Conector: Suprimir	deleteEntry
Conector: Delta	modEntry, putEntry o deleteEntry (según la lógica interna)
Conector: Iterador	getNextEntry
Conector: CallReply	queryReply
Conector: Buscar	findEntry
Componente de función	perform

Si un conector está en modalidad Servidor con un estado de simulación En script, todavía necesitará recibir una solicitud de un cliente. Se llamará al SS cuando deba enviarse la respuesta.

La operación de búsqueda interna que se ejecuta para conectores en modalidad Actualizar, Suprimir y Delta se realizará internamente y no permitirá alterar temporalmente desde un SS como hace para la línea de ensamblaje de proxy.

Los conectores en estado de simulación no ejecutarán el complemento de software de alteración temporal de la operación, si existe alguno.

Capítulo 5. EasyETL

La perspectiva EasyETL del Editor de configuración es un modo altamente especializado de abordar las configuraciones de IBM Security Directory Integrator, dedicado a darle rápidamente los medios para ejecutar proyectos relativos a simples tareas de extracción, transformación y carga de datos en algún tipo de base de datos.

La perspectiva EasyETL muestra proyectos EasyETL y permite ejecutar, abrir y crear proyectos ETL nuevos. La perspectiva ETL se puede abrir seleccionando **Ventana > Abrir perspectiva** y luego seleccionando la perspectiva EasyETL.

Para iniciar el Editor de configuración con esta perspectiva, añada `-perspective com.ibm.tdi.rcp.perspective.etl` a la línea de mandatos del Editor de configuración (`ibmditk`).

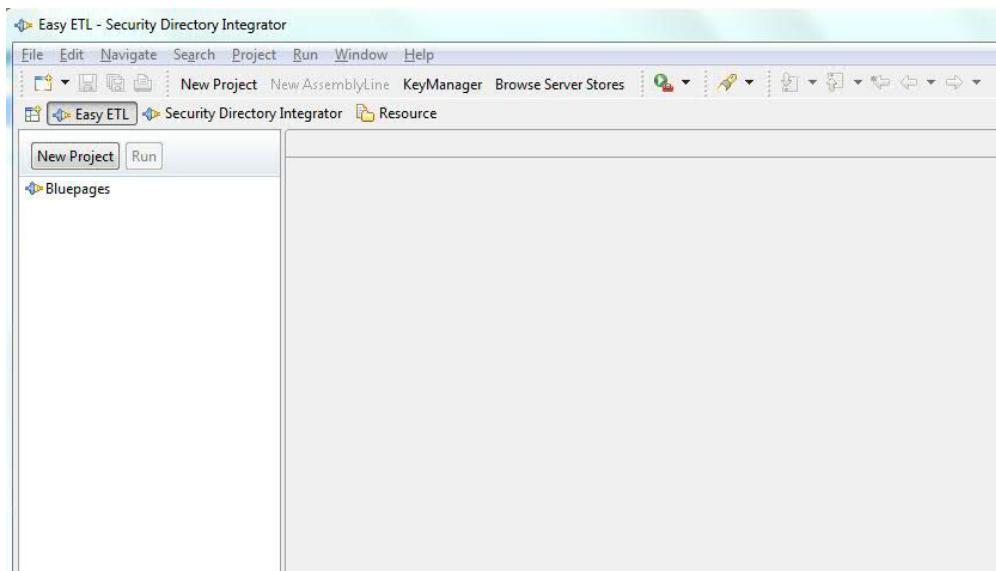


Figura 109. Ventana principal de EasyETL

Utilice el botón **New Project** para crear un nuevo proyecto EasyETL. Un proyecto EasyETL es un proyecto normal de IBM Security Directory Integrator con una sola línea de ensamblaje y dos conectores. Haga una doble pulsación en el proyecto o seleccione el proyecto y pulse la tecla Intro para abrir el editor ETL.

El menú contextual de un proyecto ETL tiene los elementos siguientes:

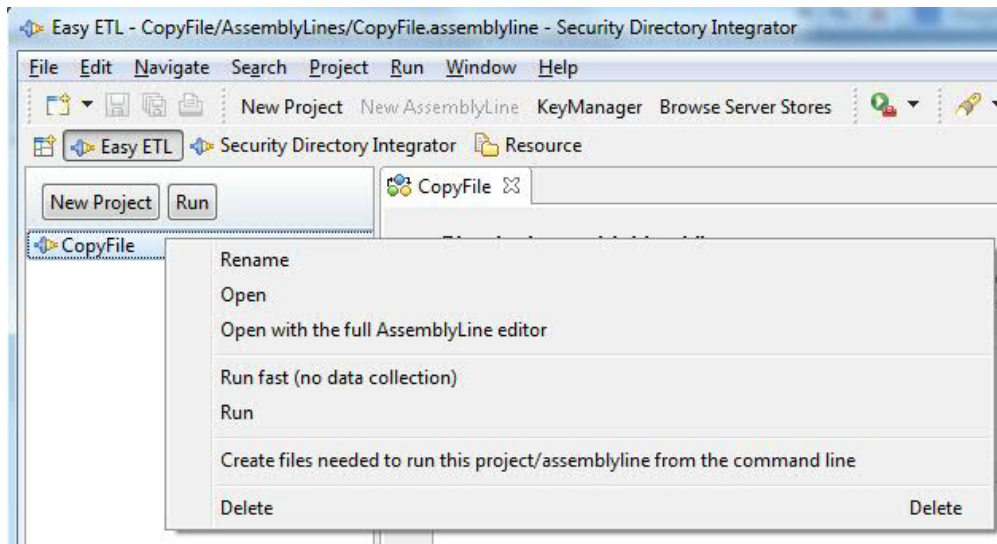


Figura 110. Menú contextual de proyecto EasyETL

- **Abrir:** abre el proyecto en el editor.
- **Abrir con editor de línea de ensamblaje completo:** abre el proyecto en el editor de línea de ensamblaje avanzado.
- **Ejecución rápida:** ejecuta el proyecto sin recopilar datos de la línea de ensamblaje.
- **Ejecutar:** ejecuta el proyecto y muestra los datos recopilados en la vista del recopilador de datos.
- **Crear archivos ... :** genera los archivos necesarios para ejecutar el proyecto desde la línea de mandatos.
- **Renombrar:** renombra el proyecto.
- **Suprimir:** suprime el proyecto.

El editor EasyETL muestra los dos conectores con una tabla que muestra la correlación entre los dos conectores. La pantalla inicial muestra una selección vacía para ambos conectores tal como se muestra en la imagen siguiente:

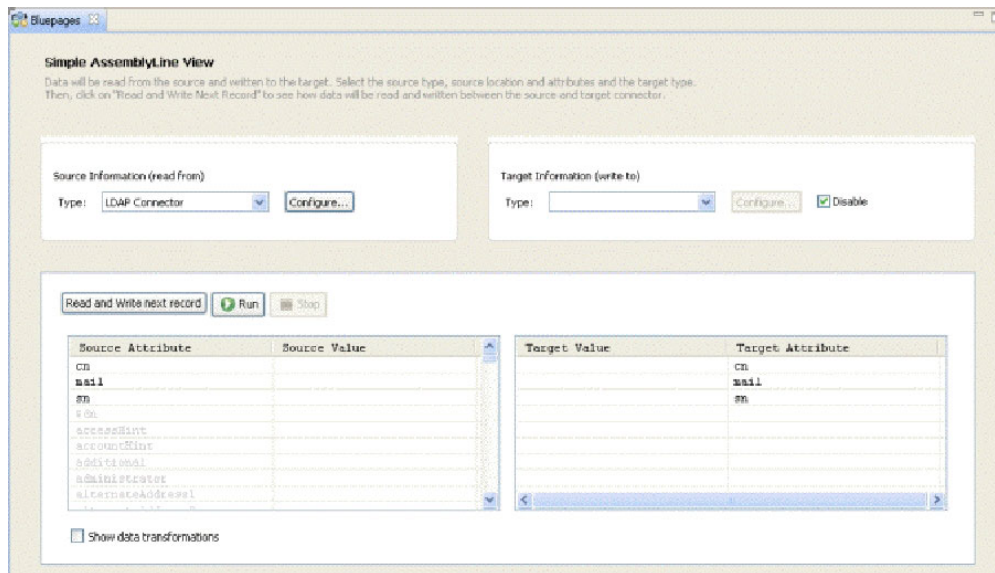


Figura 111. Ventana de proyecto inicial EasyETL

A continuación, se suele empezar seleccionando el conector de origen. Existen cuatro opciones de tipo desplegable:

- Conector del sistema de archivos
- Conector LDAP
- Conector de base de datos (JDBC)
- Seleccionar conector...

Las tres primeras son selecciones rápidas, ya que son las que se utilizan con más frecuencia. La última opción, **Seleccionar conector...** hace que aparezca el diálogo de selección de conector estándar donde puede escoger el conector que desee. Con todo, la lista del conector se limita a los que implementan la modalidad del conector de origen (Iterador) y destino (Sólo adición).

Una vez que se seleccione el conector, puede configurarlo. El diálogo de configuración contiene los formularios para configurar el conector y el analizador en el caso de que convenga. Además, la pantalla *Delta* se encuentra disponible para el conector de origen.

Si selecciona el conector LDAP, verá la ventana siguiente:

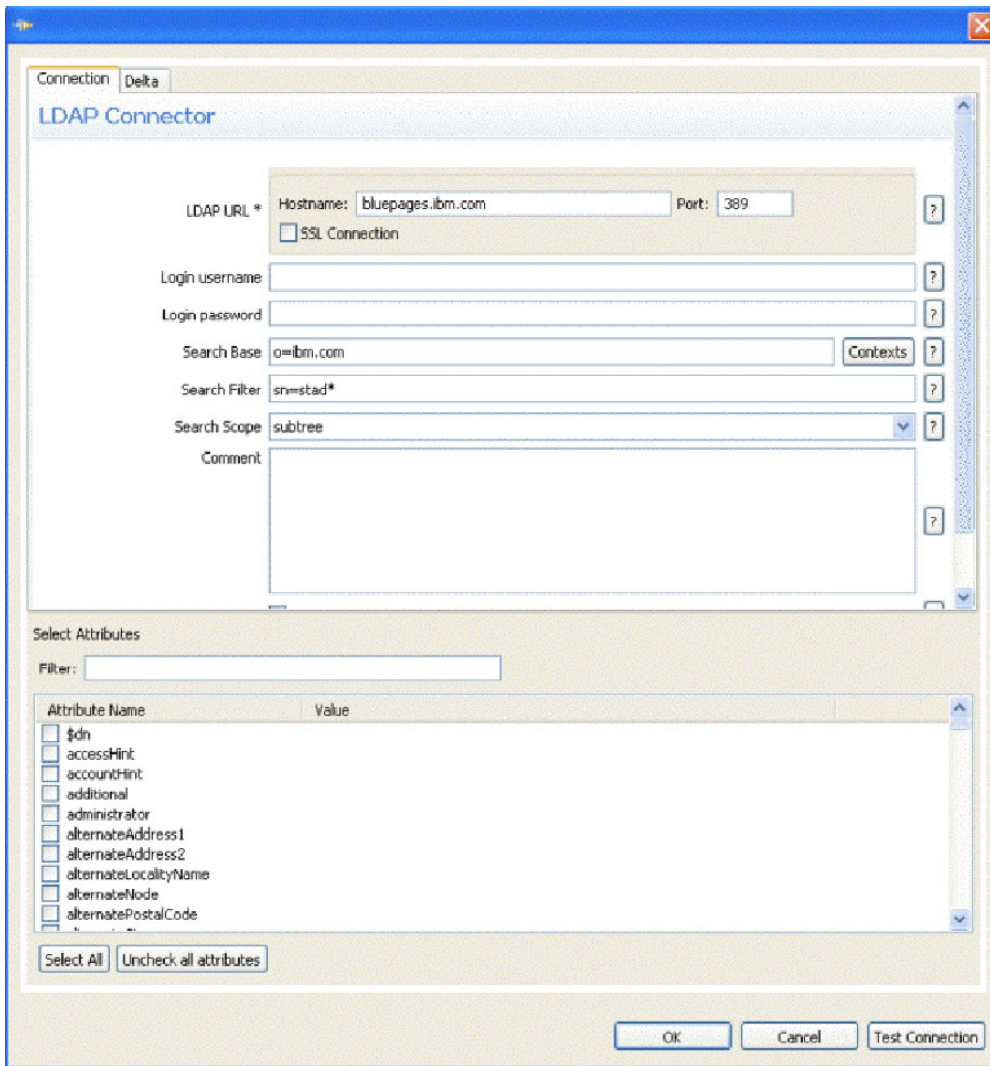


Figura 112. El conector LDAP en Easy ETL

Después de descubrir los atributos en el conector, puede seleccionar los atributos que desee leer del conector. Para el conector de destino, sólo está presente la lista de elementos de esquema, pues la correlación se basa en los atributos del conector de entrada.

Una vez que el esquema y los atributos se encuentran disponibles, se mostrarán en la columna de atributos de origen.

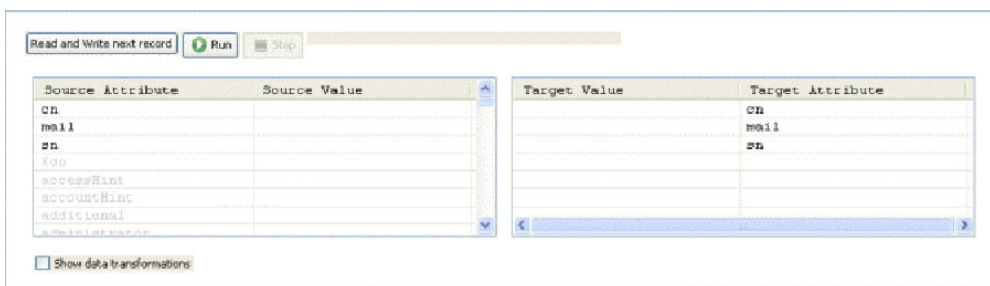


Figura 113. Correlación de entrada/salida

Los elementos que aparecen atenuados son atributos que no se han correlacionado. Pulse el botón derecho del ratón y seleccione **Correlacionar atributo** para correlacionar el atributo. También puede realizar una doble pulsación o pulsar la tecla Intro para correlacionar la selección actual. En la columna del atributo de destino, puede pulsar y seleccionar otro nombre de atributo de salida. En cambio, puede hacer lo mismo en un atributo correlacionado para devolverlo a la lista de atributos sin correlacionar.

Para personalizar la correlación entre los dos atributos, seleccione la casilla de verificación **Mostrar transformaciones**. De este modo, se añadirá una nueva tabla entre las tablas de origen y destino.

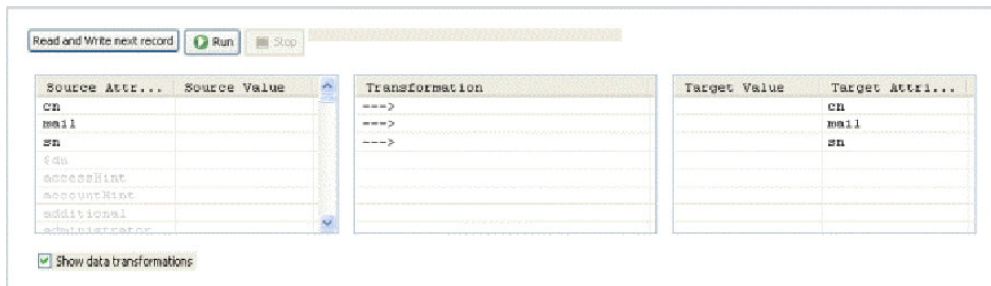


Figura 114. Correlación de entrada/salida con transformaciones

La tabla de transformación que se muestra contiene flechas que denotan copias de especificador textual entre dos atributos. Realice una doble pulsación en un elemento de transformación para abrir el editor de JavaScript para esa correlación.

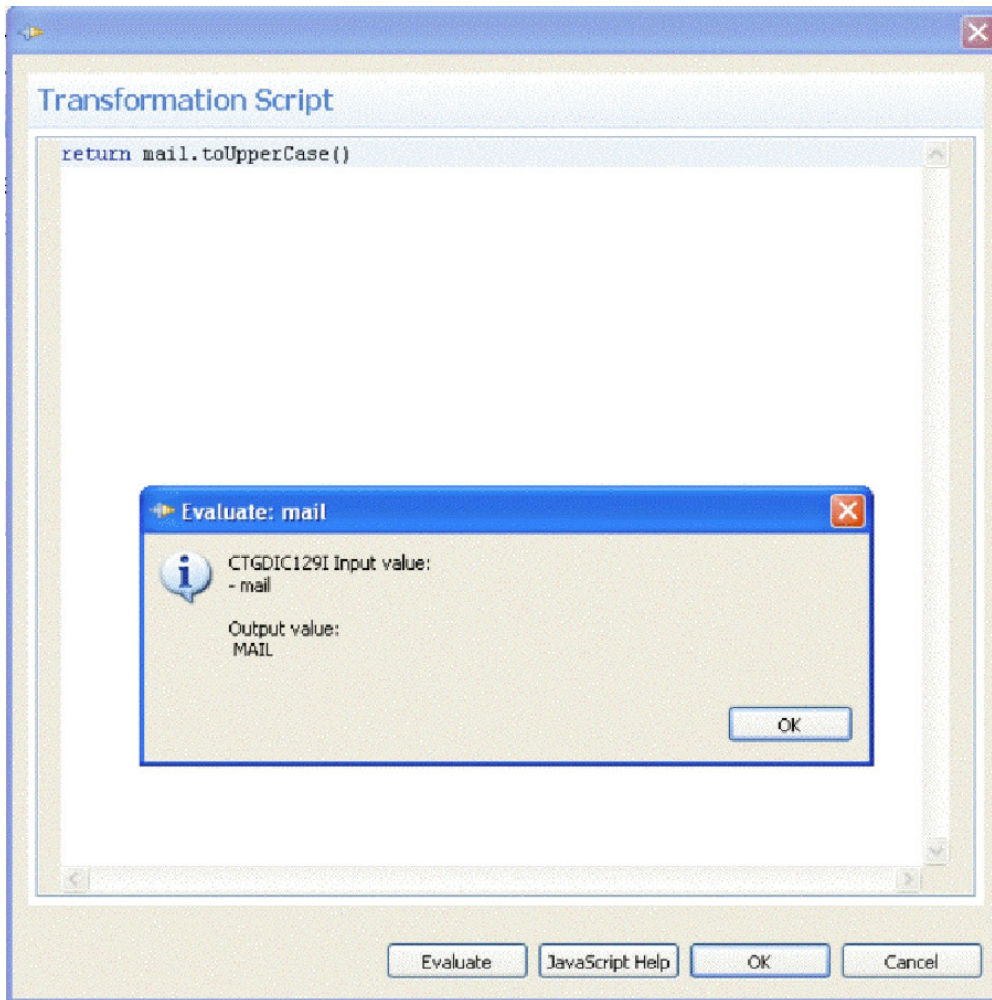


Figura 115. Script de transformación

Especifique el script que llevará a cabo la transformación personalizada del valor. Tenga en cuenta que todos los atributos correlacionados en el conector de origen se encuentran disponibles como beans de nivel superior. Esto significa que puede consultar directamente `cn` en lugar de utilizar la notación `work.cn`. El editor también reconoce la clase Java basándose en lo que ha leído la acción *Leer y escribir registro siguiente*. La última lectura de entrada también se utiliza cuando se prueba el script con el botón **Evaluar** de modo que la evaluación del script pueda probarse con datos reales tal como se muestra en la imagen anterior. El mensaje muestra el valor de entrada y el resultado del script de transformación (salida). El botón **Ayuda de JavaScript** es una forma rápida de acceder a la página de ayuda de JavaScript.

El conector de destino tiene una casilla de selección denominada **Inhabilitar**. Cuando se selecciona, esta casilla inhabilita el conector de salida y realiza un vuelco de la entrada (después de las transformaciones personalizadas) en el archivo de registro de la consola.

Una vez que se han configurado los conectores, puede ejecutar la línea de ensamblaje para que se finalice o pasando por cada registro cada vez por la línea de ensamblaje. Al pasar por la línea de ensamblaje, la tabla reflejará la última entrada leída y grabada en el conector de destino. Al pulsar el botón **Ejecutar**, la línea de ensamblaje se ejecutará de forma continuada hasta que se finalice o se

pulse el botón **Detener**. Si se pulsa el botón de parada durante la ejecución, la línea de ensamblaje se interrumpe y se devuelve el control al usuario. Si se pulsa el botón de parada mientras el usuario tiene el control, la línea de ensamblaje finalizará. Una vez que la línea de ensamblaje finalice, mostrará un diálogo de finalización con algunas estadísticas sobre la ejecución:

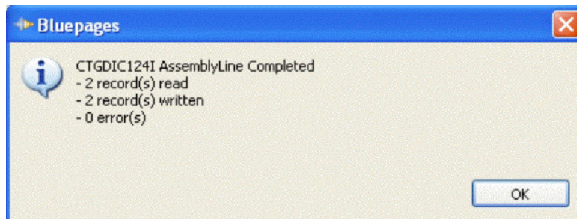


Figura 116. Diálogo Terminación

Capítulo 6. Almacén del sistema

El almacén del sistema aborda las diversas necesidades de IBM Security Directory Integrator respecto al almacenamiento persistente y de forma predeterminada utiliza el sistema de gestión de bases de datos relacionales de Apache Derby (conocido anteriormente como IBM Cloudscape) como tecnología de almacenamiento básica.

También pueden utilizarse otras bases de datos relacionales, como IBM DB2, para el almacén del sistema. El almacén del sistema puede ser compartido por varias instancias de servidores de IBM Security Directory Integrator si la base de datos Derby se ejecuta en la modalidad en red o si se utiliza un sistema de bases de datos relacionales de varios usuarios. Si Derby se ejecuta de forma integrada en un servidor de IBM Security Directory Integrator, no se puede compartir simultáneamente con otros servidores.

El almacén del sistema implementa tres tipos de almacenes permanentes para los componentes de IBM Security Directory Integrator:

- “Almacén de propiedades de usuario” en la página 212
- “Almacén delta” en la página 212
- Tablas de recinto de seguridad

Cada almacén dispone de su propio conjunto de funciones y un comportamiento predefinido, una interfaz invocable a la que los usuarios pueden acceder desde sus scripts, por ejemplo, para mantener sus propios datos e información de estado.

Puede configurar los “Valores de almacén del sistema” en la página 182 para un proyecto seleccionando el proyecto en el navegador IBM Security Directory Integrator y seleccionando **Registro de soluciones y valores**. A continuación, seleccione la pestaña **Almacén del sistema**.

Puede configurar un conector JDBC para que acceda directamente a cualquiera de las tablas del almacén del sistema, aunque debe evitarse modificar los datos de estas tablas ya que esto puede hacer que la solución no funcione correctamente.

Atención: Si está ejecutando Derby de forma integrada en IBM Security Directory Integrator, en lugar de ejecutarlo en la modalidad de red como servidor, debe **Cerrar** de nuevo la base de datos antes de probar o ejecutar la configuración. Debido a que el Editor de configuración inicia una instancia separada del servidor, que se ejecuta en su propia Máquina virtual Java, el almacén del sistema no está disponible para este servidor. Al cerrar la ventana de detalles del almacén del sistema también se cierra la conexión con la base de datos.

Nota: aunque la función de Cajón de arena también utiliza la tecnología de almacén del sistema, se especifica un nuevo directorio de base de datos para cada línea de ensamblaje.

Almacén de propiedades de usuario

El almacén de propiedades de usuario es una tabla del almacén del sistema que se utiliza para mantener los objetos Java serializados asociados con un valor clave. Aquí es donde se guardan los parámetros y las propiedades de componentes permanentes como, por ejemplo, el **Almacén de estado de iterador** y los datos que almacena el usuario.

Por ejemplo, cuando establece el parámetro **Almacén de estado de iterador** para el conector de detección de cambios de Active Directory, está especificando el valor clave que el conector utiliza para guardar y restaurar el estado del iterador. Si desea que la iteración comience por la primera (o la última) entrada modificada, simplemente suprime la entrada Almacén de estado de repetidor en el almacén de propiedades del usuario; es decir, pulse **Suprimir** junto al parámetro.

Puede pasar sus propios objetos a permanentes con las siguientes llamadas a sistema:

system.setPersistentObject(valorClave,obj)

Guarda el objeto **obj** en el almacén de propiedades del usuario utilizando el *valorClave* especificado. El objeto se devuelve si se guarda correctamente, de lo contrario, la función devuelve **null**.

system.getPersistentObject(valorClave)

Devuelve el objeto con el *valorClave* especificado del almacén de propiedades del usuario. Si no se encuentra *valorClave*, la función devuelve **NULL**.

system.deletePersistentObject(valorClave)

Suprime el objeto con el *valorClave* especificado en el almacén de propiedades del usuario. Esta función devuelve el objeto que se ha suprimido o **NULL** si no se ha encontrado *valorClave*.

Estos métodos acceden al almacén de propiedades del usuario por omisión.

No obstante, puede crear y utilizar sus propios almacenes utilizando la *StoreFactory* (Fábrica de almacén).

Si visualiza el almacén de propiedades del usuario desde la ventana del almacén del sistema, tenga en cuenta que tiene la definición de tabla siguiente:

Clave La clave exclusiva (512 caracteres)

Entrada

El objeto asociado a la clave.

Nota: cualquier objeto que ha de permanecer en el almacén de propiedades del usuario debe ser serializable.

Almacén delta

El almacén delta se encuentra en la carpeta **Tablas delta** del navegador del almacén del sistema. Cada tabla representa un valor del parámetro **Almacén delta** (en la pestaña **Delta** de un iterador). Hay varias clases y métodos para trabajar directamente con el almacén delta, aunque no se recomienda. Para obtener más información acerca de la función delta, consulte el apartado Capítulo 7, "Deltas", en la página 217.

Métodos de la fábrica de almacén

A continuación se muestran unos ejemplos de los métodos que se pueden utilizar con la fábrica de almacén:

```
public static PropertyStore getDefaultPropertyStore () throws Exception;  
Devuelve el almacén de propiedades predeterminado.
```

```
public static PropertyStore getPropertyStore ( String table ) throws  
Exception;
```

Devuelve el almacén de propiedades identificado por nombre. Sólo una instancia de un nombre determinado está presente cada vez.

@param name

El nombre del almacén de propiedades.

@return

El objeto de almacén de propiedades asociado al nombre.

```
public static String getSystemDatabaseURL ();
```

Devuelve el URL de JDBC del almacén del sistema.

```
public static Connection getConnection () throws Exception;
```

Devuelve un objeto de conexión a la base de datos predeterminada.

```
public static Connection getConnection ( String database ) throws  
Exception;
```

Devuelve un objeto de conexión a la base de datos especificada con AutoCommit establecido en TRUE.

@param database

El nombre de la base de datos.

```
public static Connection getConnection ( String database, boolean  
autoCommit ) throws Exception;
```

Devuelve un objeto de conexión a la base de datos especificada.

@param database

El nombre de la base de datos.

@param autocommit

El distintivo AutoCommit.

@return

Un objeto de conexión a la base de datos especificada.

```
public static boolean dropTable ( Connection connection, String table );
```

Borra una tabla de la base de datos asociada a la conexión.

@param connection

El objeto de conexión que se ha obtenido mediante getConnection().

@param table

La tabla que se va a borrar.

```
public static void verifyTable ( Connection connection, String table,  
Vector sql ) throws Exception;
```

Verifica que pueda accederse a una tabla de la base de datos.

@param connection

El objeto de conexión que se ha obtenido mediante getConnection(). Si su valor es null, se obtiene una conexión con la tabla predeterminada.

@param table
El nombre de la tabla que se ha de verificar.

@param sql
Un vector de las sentencias SQL para crear la tabla si no existe.

public static Exception dropTable (String tableName);
Borra una tabla de la base de datos predeterminada.

@param tableName
El nombre de la tabla que se va a borrar.

public static byte[] serializeObject (Object obj) throws Exception;
Serializa un objeto en una matriz de bytes.

@param obj
El objeto que se va a serializar.

@return
La matriz de bytes que contiene el objeto serializado.

public static Object deserializeObject (byte[] array) throws Exception;
Deserializa una matriz de bytes en un objeto Java.

@param array
La matriz de bytes con el objeto Java serializado.

@return
El objeto Java que se ha renovado.

Métodos del almacén de propiedades

A continuación se muestran unos ejemplos de los métodos que se pueden utilizar con el almacén de propiedades:

public Object setProperty (String key, Object obj) throws Exception;
Añade o actualiza un valor del almacén de propiedades. Si se efectúa una actualización, se devuelve el valor antiguo.

@param key
El identificador exclusivo.

@param obj
El valor.

@return
El valor antiguo en caso de una actualización.

public Object getProperty (String key) throws Exception;
Devuelve un valor del almacén de propiedades.

@param key
El identificador exclusivo.

@return
El valor del almacén o el valor NULL si no se encuentra.

public Object removeProperty (String key) throws Exception;
Elimina un valor del almacén de propiedades.

@param key
El identificador exclusivo que se ha de suprimir.

@return
El valor antiguo o el valor **null** si la clave no está en la tabla.

Métodos de UserFunctions (objeto system)

La clase UserFunctions (por ejemplo, el objeto system) tiene métodos adicionales definidos para obtener o establecer objetos en el almacén de propiedades del sistema:

public Object getPersistentObject (String key) throws Exception;

Este método recupera un objeto con nombre del almacén de propiedades predeterminado del sistema.

@param key

La clave exclusiva.

public Object setPersistentObject (String key, Object value) throws Exception;

Este método almacena un objeto con nombre en el almacén de propiedades predeterminado del sistema.

@param key

La clave exclusiva.

@param value

El objeto que se ha de almacenar (debe ser un objeto Java serializable).

@return

El objeto antiguo, si existe.

public Object removePersistentObject (String key) throws Exception;

Este método elimina un objeto con nombre del almacén de propiedades predeterminado del sistema.

@param key

La clave exclusiva.

@return

El objeto antiguo, si existe.

Capítulo 7. Deltas

La función Motor Delta se encuentra disponible para los conectores de la modalidad de iterador. Si se habilita desde la pestaña Delta del repetidor, la función Motor delta utiliza el almacén del sistema para tomar una instantánea de los datos que se iteran. Cada entrada leída correctamente se compara con la base de datos de la instantánea llamada Almacén Delta para comprobar qué ha cambiado. Según las diferencias entre la entrada de lectura y la entrada almacenada en el almacén Delta, el motor Delta crea una nueva entrada llamada Entrada Delta. Esta entrada se etiqueta con códigos de operación Delta especiales para indicar lo que ha cambiado y cómo.

La modalidad Delta es una modalidad de conector que permite a un conector "entender" y utilizar los códigos de operación delta. Un conector de esta modalidad utiliza los códigos de operación delta de una entrada Delta recibida para determinar el tipo de cambio que debe aplicarse al sistema conectado. La modalidad Delta da soporte a todos los tipos de modificaciones (adición, modificación y supresión). Esta modalidad se utiliza para facilitar la sincronización entre los diferentes sistemas (por ejemplo, la sincronización de dos servidores LDAP en diferentes máquinas).

Atención: El motor delta introduce un repositorio local subyacente para almacenar instantáneas de datos a fin de calcular los cambios realizados durante el proceso de sincronización. El origen de datos que se está explorando en busca de cambios se convierte en el maestro en una relación maestro/esclavo y, de este modo, resulta vital que todos los cambios realizados en el esclavo (por ejemplo el almacén delta) se realicen utilizando mecanismo delta, nunca manipulando directamente la tabla de base de datos subyacente. De lo contrario, la información de la instantánea delta que IBM Security Directory Integrator mantiene queda inconsistente y el motor delta da error.

Funciones Delta

Las funciones Delta de IBM Security Directory Integrator son las siguientes:

Entrada Delta

Se trata de un objeto de entrada habitual que se etiqueta con códigos de operación delta especiales. Estos códigos describen el tipo de cambio (*add*, *modify*, *delete* o *unchanged*) y pueden asignarse en niveles diferentes (entrada, atributo o valor de atributo).

Componentes que crean entradas Delta

- Motor Delta: detecta cambios en el origen de datos. Esto es de utilidad cuando el origen de datos en sí no ofrece acceso cómodo a los cambios (por ejemplo, mecanismo de registro o notificación de cambios). Los cambios se detectan comparando el estado actual del origen de datos con una instantánea histórica. La instantánea se guarda en un repositorio denominado "Almacén Delta". Físicamente, el almacén Delta consta de un número de tablas Delta ubicadas en el almacén del sistema.

La próxima vez que se lean los datos, se compararán con los ya almacenados en el almacén Delta. Una vez que se haya calculado la información de los cambios,

el conector creará y devolverá una entrada Delta. Esta entrada Delta puede utilizarse a continuación para transferir los cambios detectados a otros sistemas conectados mediante conectores en las modalidades *Actualización*, *Sólo adición*, *Supresión* o *Delta*.

- Conectores de detección de cambios: son los conectores LDAP, el conector de detección de cambios RDBMS y el conector de detección de cambios Domino.
- LDIF y analizador DSMLv2: en las operaciones de lectura y escritura, LDIF y el analizador DSMLv2 permiten el etiquetado Delta a nivel de entrada, atributo y valor de atributo.

Componentes que consumen entradas Delta

- Modalidad Delta: esta modalidad de conector facilita soluciones de sincronización entre sistemas. Puede utilizarse para aplicar todos los cambios a un sistema conectado. La modalidad Delta es la única que requiere (y utiliza) entradas Delta. Esta modalidad detecta tipos de cambios mediante los códigos de operación delta de una entrada.
- Actualizar conectores con cambios de calcular cambios: los conectores en modalidad de actualización con el parámetro "Calcular cambios" activado no desencadenan la lógica de "Calcular cambios" si la entrada recibida se ha etiquetado como delta.

Entrada Delta

Una entrada Delta es una entrada que posee todas las características y funciones de una entrada normal. Además de esto, la entrada Delta también contiene códigos de operación Delta. Indican el tipo de cambio aplicado a la entrada (adición, supresión, modificación o sin cambios). Los códigos de operación delta pueden adjuntarse a entradas, atributos y valores para reflejar sus cambios.

Visión general

El proceso de asignar códigos de operación delta se denomina codificación delta y los códigos delta se conocen como códigos de operación y etiquetas delta. En pocas palabras: la entrada Delta es, en realidad, una entrada normal codificada en Delta. A continuación, un ejemplo de una entrada normal y una entrada Delta.

Entrada normal:

```
{
  "#type": "generic",
  "#count": 3,
  "UserName":
    "#type": "replace",
    "#count": 1,"tanders",
  "FullName":
    "#type": "replace",
    "#count": 1,"Teddy Anderson",
  "id":
    "#type": "replace",
    "#count": 1,"66"
}
```

Entrada Delta:

```
{
  "#type": "modify",
  "#count": 3,
  "@delta.old": "{
  "UserName": "manders",
  "FullName": "Mary Anderson",
```



```

    "id": "66"
  },
  "UserName": [
    "#type": "modify",
    "#count": 2,
    "tanders",
    "manders"
  ],
  "FullName": [
    "#type": "replace",
    "#count": 1,
    "Mary Anderson"
  ],
  "id":
    "#type": "unchanged",
    "#count": 1, "66"
}

```

El proceso de evaluación de códigos delta es descendente y siguiendo el orden Nivel de entrada → Nivel de atributo → Nivel de valor de atributo. La operación en el nivel superior toma prioridad.

Si una operación de entrada es *suprimir* todas las demás pestañas delta se ignorarán. Si es *reemplazar*, *modificar* o *añadir*, la evaluación continuará con las etiquetas Delta de atributos.

Si se etiqueta un atributo como *suprimir*, *añadir* o *reemplazar*, se ignorarán las etiquetas delta de sus valores. Sólo si se etiqueta un atributo como *modificar*, se tendrán en cuenta las etiquetas delta de los valores de atributo. Estas etiquetas delta indican que los valores de atributo pueden tener diferentes códigos de operación (por ejemplo, algunos de ellos se añaden y otros se suprimen).

Las etiquetas delta de valor de atributo tienen el significado siguiente en el contexto de la codificación Delta:

- *añadir* – el valor se añade a la lista de valores del atributo especificado;
- *suprimir* – el valor se elimina de la lista de valores del atributo especificado;
- *reemplazar* – el valor se reemplaza; ésta es la etiqueta delta por defecto.

Las entradas Delta son creadas por los componentes siguientes:

1. Conectores en modalidad de iterador compatibles con Delta;
2. Conectores de detección de cambios;
3. Analizador LDIF y DSMLv2 durante lectura.

Son consumidos por los componentes siguientes:

1. Conectores en modalidad Delta;
2. Analizador LDIF y DSMLv2 durante la grabación;
3. Conectores en modalidad de actualización.

Obtención y establecimiento de códigos de operación mediante Script

La codificación Delta se soporta a nivel de entrada, atributo y valor de atributo. A continuación, se explica cómo obtener/establecer la operación de entrada mediante script:

```

var entryOper = work.getOperation(); //obtener operaciones de entrada como serie (p.ej. 'add')
var entryOp = work.getOp();         //obtener operaciones de entrada como caracteres (p.ej. 'a')

work.setOperation("modify");       //establecer operación de entrada
work.setOp ('m');

```

Si desea establecer/obtener distintivos Delta para un atributo, puede hacerlo con el código siguiente:

```

var attr = work.getAttribute("sn"); // obtener objeto de atributo

var attrOper = attr.getOperation(); // obtener operación delta como serie (p.ej. 'replace')
var attrOp = attr.getOp();         // obtener operación delta como carácter (p.ej. 'r')

attr.setOperation("replace");     // establecer operación delta de atributo
attr.setOp('r');

```

Las etiquetas delta a nivel de valor de atributo pueden establecerse/obtenerse mediante este script:

```

var attr = work.getAttribute("sn"); // obtener objeto de atributo

var valOper = attr.getValueOperation(0); // obtener operación delta de valor para primer valor
var valOp = attr.getValueOp(0);

attr.setValueOperation(1, "add"); // establecer operación delta de valor para segundo valor
attr.setValueOp(1, 'a');

```

Producción de entradas Delta

Las entradas Delta pueden generarse mediante la función Delta o un analizador de conectores adecuado en modalidad Iterador, o mediante los conectores de detección de cambios de IBM Security Directory Integrator.

En concreto, las entradas etiquetadas Delta son devueltas por los componentes siguientes:

- Conector de detección de cambios de Active Directory
- Conector de detección de cambios de Domino
- Conector de registro de cambios de IBM Security Directory Server
- Conector de detección de cambios RDBMS
- Conector de detección de cambios de Sun Directory
- Conector de registro de cambios de LDAP para z/OS

Nota: El sistema operativo z/OS no está soportado en IBM Security Directory Integrator a partir de la versión 7.2.

- Analizador DSMLv2
- Analizador LDIF

Función Delta para modalidad Iterador

Los conectores en modalidad Iterador pueden generar entradas Delta. Esta función utiliza el motor Delta y el almacén Delta para detectar cambios.

Motor delta

El motor Delta permite leer un origen de datos y detectar cambios realizados anteriormente. De este modo, puede detectar entradas nuevas, entradas modificadas e incluso entradas suprimidas. En el caso de determinados orígenes de datos (como por ejemplo archivos LDIF y servidores LDAP), IBM Security

Directory Integrator incluso puede detectar si los atributos y valores de las entradas han sido modificados. Puede configurar los valores delta en los conectores sólo en la modalidad Iterador.

El mecanismo delta sabe si se han añadido, modificado o suprimido entradas o atributos manteniendo una copia local de cada entrada en un almacén permanente, que forma parte del almacén del sistema. Este repositorio local se llama *Almacén Delta* y consta de *tablas Delta*. Cada vez que se ejecuta la línea de ensamblaje, el motor delta compara los datos que se están iterando con su copia de la tabla delta. Cuando se detecta un cambio, el conector devuelve una *entrada Delta*.

Nota: No modifique las tablas del almacén Delta de forma manual. De lo contrario, la información de la instantánea delta queda inconsistente y el motor delta da error.

Nota: En versiones anteriores de IBM Security Directory Integrator V6.1, las instantáneas que se escribían en el almacén delta durante el proceso del motor delta se comprometían de inmediato. Como resultado de ello, el motor delta consideraba una entrada modificada como manejada aunque el proceso de la sección Flujo de la línea de ensamblaje hubiera fallado. Esta limitación se solventa utilizando el parámetro *Comprometer* en la pestaña Delta del conector. El valor de este parámetro controla cuando el motor delta compromete las instantáneas que se toman de los datos entrantes en el almacén del sistema.

Nombre de atributo exclusivo

Para que el mecanismo delta pueda identificar de forma exclusiva cada entrada, debe especificar un atributo exclusivo que se utilizará como clave delta. Los valores de este atributo deben ser exclusivos en el origen de datos utilizado. Puede especificar la clave Delta en la pestaña Delta del conector especificando o seleccionando un nombre de atributo en el parámetro *Nombre de atributo exclusivo*. Este atributo se debe encontrar en la correlación de entrada del iterador y puede ser un atributo leído del sistema conectado o un atributo calculado (mediante script en *Correlación de atributos*).

También puede especificar varios atributos separándolos con un signo más (+):
Apellido+Nombre+FechaNacimiento

Al menos uno de los atributos especificados en el parámetro *Nombre de atributo exclusivo* debe contener un valor. Cuando se especifican varios atributos, los valores de cadena se concatenan en una cadena que, a continuación, se convierte en el identificador delta exclusivo. Los atributos que no tienen valores (por ejemplo, cuyo valor es NULL) se ignoran cuando se crea la clave delta para una entrada.

Almacén delta

El almacén Delta se encuentra físicamente en el almacén del sistema. Consta de una tabla del sistema Delta y una o varias tablas Delta. Cada tabla Delta se utiliza para el almacén Delta de un conector de iterador diferente compatible con Delta.

Aunque se puede acceder a las tablas del almacén Delta con el conector JDBC y el conector del almacén del sistema, no es recomendable cambiarlas sin comprender perfectamente el modo en que el motor Delta estructura y gestiona dichas tablas.

Estructura de la tabla Delta

Cada tabla Delta (DT) contiene información sobre cada entrada procesada por el motor Delta de un conector determinado. Una tabla del sistema delta (DS) mantiene una lista de todas las tablas Delta actualmente utilizadas por el almacén Delta.

- Tabla del sistema Delta - La tabla del sistema delta (DS) contiene información sobre cada tabla delta (DT) del almacén del sistema. La finalidad de la DS es mantener el contador de secuencias para cada DT. La estructura de la DS es la siguiente:

Tabla 12. Estructura de la tabla del sistema delta

Columna	Tipo	Descripción
ID	Varchar	Identificador de la DT (nombre)
SequenceID	Int	El ID de secuencia de la última ejecución
Version	Int	La versión de la DS (1)

- Tabla Delta - Cada conector que solicita un almacén delta tiene que especificar un identificador delta exclusivo para asociarlo al conector. Este identificador también se utiliza como el nombre de la tabla delta del almacén del sistema. La estructura de la tabla delta es la siguiente:

Tabla 13. Estructura de la tabla Delta

Columna	Tipo	Descripción
ID	Varchar	Valor exclusivo que identifica una entrada
SequenceID	Int	Número de secuencia de la entrada
Entrada	Long Varbinary	El objeto Entry serializado

Proceso delta

Habida cuenta de la estructura anterior del almacén Delta, el número de secuencia se utiliza para determinar las entradas que ya no forman parte del conjunto de datos de origen. Cada vez que se ejecuta una línea de ensamblaje, el número de secuencia de la tabla Delta utilizada en particular por el conector es leída desde la tabla del sistema Delta. A continuación, se incrementa y dicho valor incrementado será utilizado posteriormente para marcar las entradas actualizadas durante la ejecución de toda la línea de ensamblaje.

El motor Delta procesa el trabajo de dos modos.

1. Lectura → Búsqueda → Comparación → Actualización → Establecimiento de SequenceID actual
 - a. El iterador lee entradas en el origen de datos de entrada.
 - b. El proceso Delta busca la entrada correspondiente en la tabla Delta mediante el valor de atributo exclusivo.
 - c. Si se encuentra una coincidencia, el proceso Delta compara cada atributo (y sus valores) para determinar si se ha modificado la entrada. Basándose en el resultado de esta comparación, el motor Delta devuelve la entrada Delta etiquetada con los códigos de operación pertinentes: *modify* *unchanged*:
 - Entrada Modify – la entrada leída y la entrada correspondiente de la tabla Delta se consideran diferentes; la entrada se actualiza en la tabla Delta

- Entrada Unchanged – la entrada que se ha leído y la entrada correspondiente de la tabla Delta se consideran iguales.
- d. Si no se encuentra ninguna coincidencia en la tabla Delta, la entrada se trata como nueva:
- Entrada Add – la entrada se añade a la tabla Delta.
- e. En ambos casos, el valor de número de secuencia de la tabla Delta se actualiza con el número de secuencia utilizado para la ejecución de la línea de ensamblaje actual.
2. Compruebe la existencia de datos con (SequenceID < SequenceID actual) → Marcar como suprimido

Una vez que el iterador alcance el final de los datos, el motor Delta pasará de nuevo por la tabla Delta en busca de las entradas a las que no se ha accedido durante la primera revisión. Estas entradas se reconocen fácilmente porque su número de secuencia no se ha actualizado con el número de secuencia actual. Por lo tanto, cualquier entrada de la tabla Delta con un número de secuencia inferior al número de secuencia actual se considera como entrada suprimida y se devuelve como tal.

Nota: Esta revisión sucede sólo cuando se completa correctamente la iteración por los datos de entrada. Si por algún motivo se produce un error durante dicha iteración, no se etiquetará ninguna entrada como suprimida ni será devuelta por la línea de ensamblaje ni eliminada de la tabla Delta. Esto no afectará al origen de datos original y la siguiente vez que se ejecute la línea de ensamblaje correctamente, las entradas suprimidas se procesarán correctamente.

Bloqueo de fila

Este parámetro se encuentra disponible en la pestaña Delta de los conectores del iterador y la configuración del componente de función Delta. Permite establecer el nivel de aislamiento de transacción utilizado por la conexión establecida para la base de datos del almacén Delta. Al establecer un nivel de aislamiento mayor, se reducen las anomalías de transacción conocidas como 'lecturas incorrectas', 'lecturas no repetibles' y 'lecturas fantasma' mediante los bloqueos de fila y tabla. Este parámetro tiene los valores siguientes:

READ_UNCOMMITTED

Corresponde a `java.sql.Connection.TRANSACTION_READ_UNCOMMITTED`; indica que se pueden producir lecturas incorrectas, no repetibles y fantasma. Este nivel permite que otra transacción lea una fila modificada por una transacción antes de que se hayan confirmado los cambios en dicha fila (una "lectura incorrecta"). Si se retrotrae cualquiera de estos cambios, la segunda transacción habrá recuperado una fila no válida.

READ_COMMITTED

Corresponde a `java.sql.Connection.TRANSACTION_READ_COMMITTED`; indica que se evitan las lecturas incorrectas y que se pueden producir lecturas no repetibles y fantasma. Este nivel sólo prohíbe que una transacción lea una fila cuyos cambios no han sido confirmados.

REPEATABLE_READ

Corresponde a `java.sql.Connection.TRANSACTION_REPEATABLE_READ`; indica que se evitan las lecturas incorrectas y las no repetibles. Se pueden producir lecturas fantasma. Este nivel prohíbe que una transacción lea una fila cuyos cambios no se han confirmado. También prohíbe la situación en la que una transacción lee una fila, una segunda transacción altera la fila y

la primera transacción vuelve a leer la fila, con lo cual se obtienen valores diferentes la segunda vez (una "lectura no repetible").

SERIALIZABLE

Corresponde a `java.sql.Connection.TRANSACTION_READ_SERIALIZABLE`; indica que se evitan lecturas incorrectas, no repetibles y fantasma. Este nivel incluye las prohibiciones en `TRANSACTION_REPEATABLE_READ` así como la situación en la que una transacción lee todas las filas conformes con la condición `WHERE`, una segunda transacción inserta una fila que cumple con dicha condición `WHERE` y la primera transacción vuelve a leer la misma condición, con lo cual se recupera la fila "fantasma" en la segunda lectura. Ésta suele ser la opción más lenta aunque la más segura y el valor por defecto del parámetro **Bloqueo de fila**.

Para más información sobre los niveles de aislamiento de transacción, consulte la documentación en línea de la interfaz `java.sql.Connection`: <http://docs.oracle.com/javase/1.6.0/docs/api/java/sql/Connection.html>.

Cada servidor de bases de datos define un nivel de aislamiento de transacción predeterminado. El valor predeterminado para Apache Derby, Oracle y Microsoft SQL Server es `TRANSACTION_READ_COMMITTED`. Sin embargo, el valor predeterminado del parámetro **Bloqueo de fila** de `SERIALIZABLE` alterará temporalmente esto al utilizar un componente Delta (es decir, la función Delta de los conectores de iterador o el componente de la función delta).

Es posible que algunos servidores de bases de datos no den soporte a todos los niveles de aislamiento de transacción; por lo tanto, consulte la documentación específica de base de datos para obtener información precisa sobre los niveles de aislamiento de transacción soportados.

Nota: Los niveles de aislamiento de transacción son mantenidos por el propio servidor de bases de datos para cada conexión establecida para la base de datos. Por lo tanto, cuando un componente Delta (con el **nivel de aislamiento de transacción** establecido en `REPEATABLE_READ` o `SERIALIZABLE` y el parámetro **Comprometer** establecido en `Al` cerrarse el conector inicia su transacción, todas las demás consultas que intenten modificar los mismos datos será bloqueadas. Esto significa que otros componentes que deben modificar los mismos datos tendrán que esperar hasta que el primer componente confirme su transacción al finalizarse. A causa de esta espera, es posible que se exceda el tiempo de espera de las consultas SQL emitidas y que los datos no se modifiquen.

Además, cuando un componente tiene el parámetro **Comprometer** establecido en `Sin compromiso automático`, deberá confirmar las transacciones de forma manual de modo que otros componentes no tengan que esperar de forma indefinida para llevar a cabo una modificación.

Detectar o ignorar cambios sólo en atributos específicos

Los parámetros **Lista de atributos** y **Modalidad de detección de cambios** configuran la capacidad del motor Delta de detectar cambios sólo en atributos específicos en lugar de en todos los atributos recibidos.

El parámetro **Lista de atributos** es una lista de atributos separados por comas que se verán afectados por la **modalidad de detección de cambios**. Este parámetro **Modalidad de detección de cambios** especifica el modo en que se gestionarán los cambios de estos atributos. Tiene tres valores:

IGNORE_ATTRIBUTES

(“Ignorar cambios de los atributos siguientes”) – Todos los cambios en cada atributo especificado en el parámetro **Lista de atributos** serán ignorados durante el proceso de cálculo de cambios.

DETECT_ATTRIBUTES

(“Detectar cambios de los atributos siguientes”) – Esta opción tiene el efecto contrario – los únicos cambios detectados se encontrarán en los atributos enumerados en el parámetro **Lista de atributos**.

DETECT_ALL

(“Utilizar todos los atributos para la detección de cambios”) – Se instruye al motor Delta que detecte cambios en todos los atributos. Cuando se selecciona esta opción, el parámetro **Lista de atributos** se inhabilita, ya que no es necesaria ninguna lista de atributos afectados.

Ejemplo de caso de uso

Al utilizar el motor Delta, en ocasiones las entradas recibidas contienen atributos que no se consideran importantes y se desean ignorar. En tales casos, estos atributos no deben afectar al resultado del cálculo Delta, ya que cuando varias entradas se diferencian únicamente mediante dichos atributos, se pueden llevar a cabo actualizaciones innecesarias de la tabla del almacén Delta.

La solución en este caso es utilizar los parámetros **Lista de atributos** y **Modalidad de detección de cambios**

A continuación, un escenario de ejemplo en el que dos líneas de ensamblaje reciben entradas de registro de cambios de dos réplicas de un servidor LDAP y dichos cambios se aplican a un almacén Delta. Para ilustrar esto, utilizaremos el ejemplo siguiente de entradas de registro de cambios:

Entrada 1:

Atributos de entrada:

```
targetdn (replace): 'cn=Niki,o=IBM,c=us'
changetime (replace): '20071015094646'
$dn (replace): 'changenumber=78955,cn=changelog'
ibm-changeInitiatorsName (replace): 'CN=ROOT'
changenumber (replace): '78955'
objectclass (replace): 'top' 'changelogentry' 'ibm-changelog'
changetype (replace): 'modify'
cn (replace): 'Niki' 'Niky'
changes (replace): 'replace: cn
cn: Niki
cn: Niky
-
```

Entrada 2:

Atributos de entrada:

```
targetdn (replace): 'cn=Niki,o=IBM,c=us'
changetime (replace): '20071015094817'
$dn (replace): 'changenumber=10076,cn=changelog'
ibm-changeInitiatorsName (replace): 'CN=ROOT'
changenumber (replace): '10076'
objectclass (replace): 'top' 'changelogentry' 'ibm-changelog'
changetype (replace): 'modify'
cn (replace): 'Niki' 'Nikolai'
changes (replace): 'replace: cn
```



```

cn: Niki
cn: Nikolai
-
,

```

Entry3:

```

Atributos de entrada:
targetdn (replace): 'cn=Niki,o=IBM,c=us'
changetime (replace): '20071037454817'
$dn (replace): 'changenumber=112,cn=changelog'
ibm-changeInitiatorsName (replace): 'CN=ADMIN'
changenumber (replace): '112'
objectclass (replace): 'top' 'changelogentry' 'ibm-changelog'
changetype (replace): 'modify'
cn (replace): 'Niki' 'Nikolai'
changes (replace): 'replace: cn
cn: Niki
cn: Nikolai
-
,

```

Los atributos modificados se marcan en **negrita** y los atributos que pueden ignorarse, en *cursiva*. Los atributos ignorados (como changenumber, changetime, etc.) no se tendrán en cuenta al comparar la entrada recibida con la entrada almacenada. Por lo tanto, estos atributos deben aparecer enumerados en el parámetro **Lista de atributos**. Con el fin de especificar que deseamos ignorarlos, el parámetro **Modalidad de detección de cambios** debe establecerse en Ignorar cambios de los atributos siguientes.

Éste es el flujo de trabajo cuando las líneas de ensamblaje reciben las entradas:

1. Cuando AL1 recibe Entry1, se devolverá como *modify* y se guardará en la tabla del almacén Delta.
2. Cuando AL2 recibe Entry2, sus atributos changetime, \$dn, bm-changeInitiatorsName y changenumber se modifican pero no serán ignorados. Sin embargo, los atributos cn y changes también son modificados y, por tanto, la entrada Delta resultante será etiquetada como *modify* y se guardará en la tabla del almacén Delta.
3. Cuando AL2 recibe Entry3, sus atributos changetime, \$dn, bm-changeInitiatorsName y changenumber se modifican pero no serán ignorados. El resto de atributos son iguales; en consecuencia, la entrada Delta resultante será etiquetada como *unchanged* y se devolverá a la línea de ensamblaje (sólo si se marca el parámetro **Return unchanged**) o se omitirá. La entrada Delta devuelta será idéntica a la Entrada3 recibida. En este caso, el almacén Delta no se actualiza. Si no se utilizan los parámetros Lista de atributos y Modalidad de detección de cambios, la última Entrada3 se etiquetará como *modify* y guardará en el almacén Delta.

Conectores de detección de cambios

Estos componentes aprovechan la información del sistema conectado para detectar cambios y se utilizan en modalidad Iterador o Servidor, en función del conector. Por ejemplo, la modalidad Iterador se utiliza para muchos conectores de detección de cambios, como por ejemplo LDAP, RDBMS, Active Directory y Notes/Domino. Están diseñados para comportarse de un modo común, así como para proporcionar las mismas etiquetas de parámetro para valores comunes.

Los conectores de detección de cambios de IBM Security Directory Integrator son los siguientes:

- Registro de cambios de IBM Security Directory Server

- Detección de cambios de AD (Active Directory)
- Detección de cambios de Domino
- Detección de cambios de Sun Directory (openLDAP, SunOne, iPlanet, etc.)
- Detección de cambios de RDBMS DB2, Oracle, SQL Server, etc.)
- Registro de cambios LDAP de z/OS

Nota: El sistema operativo z/OS no está soportado en IBM Security Directory Integrator a partir de la versión 7.2.

La función de motor delta informa de los cambios específicos hasta los valores individuales de atributos. La codificación Delta en el nivel de valor de atributo también se encuentra disponible al realizar el análisis con analizadores LDIF o DSMLv2. El analizador de LDIF es utilizado internamente por el conector de registro de cambios de IBM Security Directory Server, el conector de detección de cambios de Sun Directory y el conector de registro de cambios de LDAP para z/OS; por lo tanto, estos conectores también permiten la codificación delta a nivel de valor de atributo. El resto de conectores de detección de cambios se limitan a informar simplemente de si se añade, se modifica o se suprime una entrada completa. Para obtener más información sobre el soporte de codificación Delta de un componente determinado, consulte la descripción específica de dicho componente en *Referencia*.

En algunos casos, las líneas de ensamblaje de larga ejecución deben procesar las mismas entradas más de una vez. Estas entradas tendrán una clave delta duplicada y harán que la línea de ensamblaje lance una excepción. Si desea permitir la duplicación de claves delta, seleccione la casilla de verificación **Permitir claves Delta suplicadas** en la pestaña Delta del iterador. Esto significa que las entradas duplicadas pueden ser gestionadas por líneas de ensamblaje que tienen conectores de iterador compatibles con Delta o conectores de detección de cambios y conectores de modalidad Delta.

Nota: Existe la posibilidad de tener, por ejemplo, una línea de ensamblaje con varios conectores de modalidad Delta y de registro de cambios. En este caso, si el conector de modalidad Delta apunta al mismo sistema subyacente que el conector de registro de cambios, la operación delta podría volver a desencadenar el registro de cambios. Dado que no se puede diferenciar entre los cambios que se acaban de recibir y los que ha desencadenado el motor delta, debe tener consideración con atención el escenario para no entrar en un bucle infinito.

La información delta calculada por el motor Delta se almacena en el objeto Entrada de trabajo y, en función del componente o característica de detección de cambios utilizada, puede ser utilizada como código de operación de *nivel de entrada*, *nivel de atributo* o *nivel de valor de atributo*.

Consumo de entradas Delta

Las entradas Delta de la línea de ensamblaje pueden ser modificadas ("consumidas") por conectores en la modalidad Delta y conectores en la modalidad Actualización dentro de la lógica de cálculo de cambios.

La modalidad Delta se encuentra disponible en los conectores siguientes:

- Conector JDBC
- Conector DSMLv2 SOAP
- Conector JNDI

- Conector LDAP

Conectores de modalidad Delta

La modalidad Delta está diseñada para simplificar la aplicación de cambios en los datos y, para ello, proporciona modificaciones incrementales en el sistema conectado, en función de los códigos de operación delta. Las modificaciones incrementales implican sólo grabar los valores específicos que se han modificado.

En primer lugar, la modalidad delta maneja todos los tipos de deltas: adiciones, modificaciones y supresiones. La modalidad Delta requiere recibir una entrada Delta para funcionar. Por lo tanto, al utilizar un conector en modalidad Delta, debe combinarse con componentes que produzca entradas Delta; éstos son los conectores de iterador compatibles con Delta, los conectores de detección de cambios o el conector que utiliza un LDIF o un analizador DSMLv2. Por ejemplo, puede sincronizar dos sistemas con el uso de sólo dos conectores: un conector de detección de cambios en la sección Canal de información para seleccionar los cambios y un segundo conector en modalidad Delta para aplicar estos cambios a un sistema de destino.

Además, la modalidad Delta aplicará la información delta al nivel más inferior soportado por el propio sistema de destino. Esto se lleva a cabo comprobando primero la interfaz del conector para ver a qué nivel de modificación incremental da soporte el origen de datos. Si trabaja con un directorio LDAP, la modalidad delta realizará las adiciones y supresiones del valor de atributo. En el contexto de un RDBMS (JDBC) tradicional, realizar una supresión y luego una adición de un valor de columna no tiene sentido, por lo tanto esto se maneja como una sustitución de un valor de ese atributo.

Además, las modificaciones incrementales son gestionadas de forma automática por la modalidad Delta en el caso de los orígenes de datos que dan soporte a esta funcionalidad. Si el origen de datos ofrece llamadas optimizadas para manejar modificaciones incrementales y éstas las admite la interfaz del conector, entonces la modalidad delta las utilizará. Por otro lado, si el sistema conectado no ofrece mecanismos de actualización delta "inteligentes", la modalidad delta los simulará en la medida en que sea posible, realizando búsquedas de actualización previa (como la modalidad Actualizar), cálculos de cambios y la aplicación subsiguiente de los cambios detectados.

Nota: Los únicos conectores que dan soporte a las modificaciones incrementales son los conectores LDAP, ya que los directorios LDAP ofrecen esta funcionalidad.

Las funciones delta de IBM Security Directory Integrator están diseñadas para facilitar la soluciones de sincronización mediante no sólo orígenes de datos que proporcionan un mecanismo de detección de cambios, sino también mediante archivos sin formato, por ejemplo. En el diagrama siguiente se muestra una solución de sincronización de estas características utilizando conectores en una modalidad de iterador y Delta. El conector Iterador lee las entradas de un origen de datos. Las entradas leídas se comparan con las almacenadas en el almacén Delta de la iteración anterior. El resultado de la comparación es la asignación de la entrada Delta con códigos de operación delta que definen los cambios realizados (adición/supresión/modificación). A continuación, el conector utiliza la entrada delta en la modalidad delta para aplicar los cambios detectados en un sistema de destino.

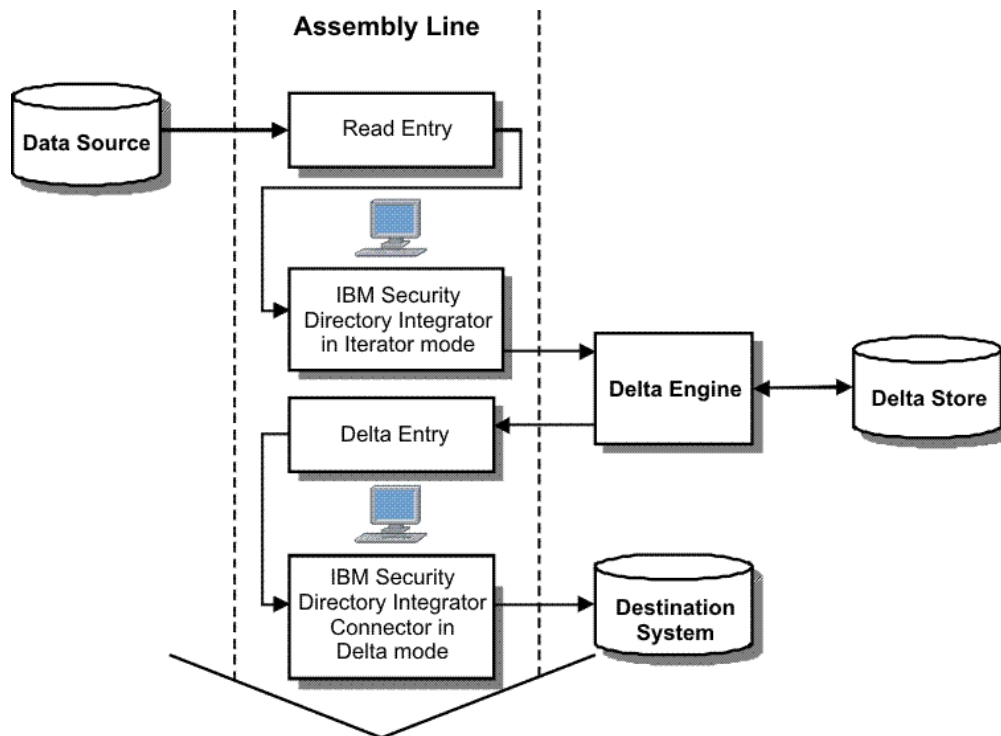


Figura 117. Línea de ensamblaje de sincronización mediante la funcionalidad Delta

El resultado de la ejecución de la línea de ensamblaje es que los datos contenidos en el origen de datos se sincronizan con los del sistema de destino.

Modalidad de actualización y entradas Delta

Los conectores de la modalidad Actualizar tienen una característica de control de cambios adicional llamada “Calcular cambios”. Esta característica puede adaptarse para funcionar con entradas Delta.

La característica “Calcular cambios” puede habilitarse en un conector de modalidad Actualizar seleccionando una casilla de verificación con dicho nombre en el panel Detalle de conector. Cuando se activa, el conector comparará su entrada con la entrada real en el sistema conectado. Si las dos entradas son iguales, el conector no llevará a cabo ninguna actualización. En consecuencia, la opción “Calcular cambios” permite al conector en modalidad Actualizar omitir las actualizaciones no necesarias. Esto resulta de utilidad para los orígenes de datos con operaciones de actualización relativamente pesadas.

Con todo, cuando un conector en modalidad Actualizar y con el parámetro **Calcular cambios** habilitado recibe una entrada Delta, la lógica de “Calcular cambios” no se desencadenará. Utilizará los códigos de operación Delta asignados a la entrada Delta para aplicar las actualizaciones necesarias al sistema conectado.

Ejemplos

Ejemplo Delta simple

Las instrucciones siguientes muestran cómo utilizar algunas de las características Delta tratadas anteriormente.

Configure un conector del sistema de archivos con la función de motor delta habilitada. Póngalo en iteración sobre un documento XML sencillo que pueda modificar fácilmente en un editor de texto. Por ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocRoot>
  <Entry>
    <Telephone>
      <ValueTag>111-1111</ValueTag>
      <ValueTag>222-2222</ValueTag>
      <ValueTag>333-3333</ValueTag>
    </Telephone>
    <Birthdate>1958-12-24</Birthdate>
    <Title>Full-Time SDI Specialist</Title>
    <uid>jdoe</uid>
    <FullName>John Doe</FullName>
  </Entry>
</DocRoot>
```

Asegúrese de utilizar el carácter de correlación de atributos especial de correlación de todo, el asterisco (*). Este es el único atributo necesario en la correlación para asegurar que todos los atributos devueltos se correlacionan en el objeto de entrada de trabajo.

Ahora añada un componente de script con el código siguiente:

```
// Obtener los nombres de todos los atributos de trabajo como una matriz de series
var attName = work.getAttributeNames();

// Imprimir el código de operación delta de nivel de entrada
task.logmsg(" Entry ( " +
work.getString( "FullName" ) + " ) : " +
work.getOperation() );

// Repetir en bucle a través de todos los atributos del trabajo
for (i = 0; i < attName.length; i++) {

// Tomar un atributo e imprimir el código de operación de nivel de atributo
att = work.getAttribute( attName[ i ] );
task.logmsg(" Att ( " + attName[i] + " ) : " + att.getOperation() );

// Ahora repetir un bucle por todos los valores de atributo e imprimir sus códigos
// de operación
for (j = 0; j < att.size(); j++) {
task.logmsg( " Val ( " +
att.getValue( j ) + " ) : " +
att.getValueOperation( j ) );
}
}
```

La primera vez que ejecuta esta línea de ensamblaje, el código del componente de script creará esta salida de registro:

```
12:46:31 Entry (John Doe) : add
12:46:31 Att ( Telephone) : replace
12:46:31 Val (111-1111) :
12:46:31 Val (222-2222) :
12:46:31 Val (333-3333) :
12:46:31 Att ( Birthdate) : replace
12:46:31 Val (1958-12-24) :
12:46:31 Att ( Title) : replace
12:46:31 Val (Full-Time SDI Specialist) :
12:46:31 Att ( uid) : replace
12:46:31 Val (jdoe) :
12:46:31 Att ( FullName) : replace
12:46:31 Val (John Doe) :
```

Dado que esta entrada no se ha encontrado en el almacén delta anterior vacío, se marca a nivel de entrada como new (nueva). Además, cada uno de sus atributos tiene un código replace (sustituir), que significa que todos los valores han cambiado (lo que tiene sentido puesto que la función delta nos indica que estos datos son nuevos).

Efectúe los siguientes cambios en su archivo XML:

1. Cambie el valor del último número de teléfono, Telephone, a 333-4444.
2. Suprima Birthdate (fecha de nacimiento).
3. Añada un nuevo atributo Address (dirección).

La configuración resultante debería ser:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocRoot>
  <Entry>
    <Telephone>
      <ValueTag>111-1111</ValueTag>
      <ValueTag>222-2222</ValueTag>
      <ValueTag>333-4444</ValueTag>
    </Telephone>
    <Title>Full-Time SDI Specialist</Title>
    <uid>jdoe</uid>
    <FullName>John Doe</FullName>
    <Address>123 Willowby Lane</Address>
  </Entry>
</DocRoot>
```

Ejecute de nuevo la línea de ensamblaje. Esta vez la salida del registro se parecerá a esta:

```
13:53:22 Entry (John Doe) : modify
13:53:22   Att ( Telephone) : modify
13:53:22     Val (111-1111) : unchanged
13:53:22     Val (222-2222) : unchanged
13:53:22     Val (333-4444) : add
13:53:22     Val (333-3333) : delete
13:53:22   Att ( Birthdate) : delete
13:53:22     Val (1958-12-24) : delete
13:53:22   Att ( uid) : unchanged
13:53:22     Val (jdoe) : unchanged
13:53:22   Att ( Title) : unchanged
13:53:22     Val (Full-Time SDI Specialist) : unchanged
13:53:22   Att ( Address) : add
13:53:22     Val (123 Willowby Lane) : add
13:53:22   Att ( FullName) : unchanged
13:53:22     Val (John Doe) : unchanged
```

Ahora la entrada está marcada como *modify* (modificar) y los atributos reflejan las modificaciones en cada uno de ellos. Como puede ver, el atributo Birthdate (Fecha de nacimiento) está marcado como *delete* (suprimir) y Address (Dirección) como *add* (añadir). Éste es el motivo por el que se ha utilizado el carácter especial de correlación de todo para la correlación de entrada. Si sólo hubiese correlacionado los atributos que existían en la primera versión de este documento XML, no habríamos podido recuperar la dirección cuando apareció en la entrada.

Observe especialmente las dos últimas entradas de valor bajo el atributo Telephone (Número de teléfono), marcado como *modify* (modificar). El cambio en uno de estos valores del atributo ha producido dos elementos delta: un valor *delete* y otro *add*.

Para crear una línea de ensamblaje de sincronización de datos en versiones anteriores de IBM Security Directory Integrator, tenía que crear un script para

gestionar el control de flujo. Aunque puede recibir *adiciones, modificaciones y supresiones* del componente o función de cambio, un conector sólo podía establecerse en una de las dos modalidades de salida necesarias: Actualizar o suprimir.

Por consiguiente, o bien tenía dos conectores que apuntaban al mismo sistema de destino y colocaba el script en el complemento de software Antes de Execute de cada uno para pasar por alto la entrada si el código de operación no coincidía con la modalidad del componente, o bien podía tener un único conector (en modalidad Actualizar o Suprimir) en el estado *pasivo* y controlar su ejecución desde el código de script en el que comprobaba el código de la operación. Y esto implicaba que, aun cuando supiera qué había cambiado en el caso de una entrada modificada, el conector en modalidad Actualizar leía los datos originales antes de escribir los cambios de nuevo en el origen de datos. Esto puede conllevar un tráfico de red o de origen de datos no deseable cuando sólo está cambiando un único valor en un grupo de varios valores relacionado con atributos que contienen miles de valores.

Otros ejemplos

Vaya al directorio `directorio_raíz/examples/` de la instalación de IBM Security Directory Integrator.

- En el subdirectorio `deltas` encontrará una configuración simple de IBM Security Directory Integrator que demuestra la funcionalidad Delta. Esta demostración se ejecuta en sistemas MS Windows solamente (Windows 2000), pues utiliza la base de datos MS Access y el puente JDBC:ODBC. Si utiliza otra plataforma, deberá crear su propia base de datos y configurar los valores de JDBC de los conectores de esta base de datos.
- En el subdirectorio `delta_tagging` encontrará un ejemplo que demuestra cómo convertir una entrada etiquetada a nivel de valor en una entrada normal y una entrada normal en una entrada Delta.

Capítulo 8. Panel de control de IBM Security Directory Integrator

Utilice el Panel de control de IBM Security Directory Integrator para instalar, configurar, desplegar y supervisar soluciones de integración de datos.

También puede utilizar Panel de control para crear y configurar soluciones EasyETL (ETL son las siglas de Extraer, Transformar y Cargar) para integraciones de datos simples. Una solución EasyETL contiene una sola línea de ensamblaje con un conector de origen y de destino.

Información básica sobre el Panel de control de IBM Security Directory Integrator

La aplicación Panel de control, basada en la web, se ha desarrollado utilizando la infraestructura Open Service Gateway Initiative (OSGI). Se utiliza para configurar y gestionar soluciones de integración de datos en un servidor a través de la interfaz RESTful (Representational State Transfer, transferencia de estados de representación). En el Panel de control, podrá:

- Cargar soluciones de integración de datos existentes e instalarlas como una solución normal o guardarlas como una plantilla.
- Configurar soluciones de integración de datos para reflejar los cambios que son específicos de su entorno local.
- Añadir planificaciones para ejecutar líneas de ensamblaje a determinadas horas.
- Crear líneas de ensamblaje simples con un solo conector de origen y un solo conector de destino.
- Examinar el contenido del conector seleccionado.
- Desplegar, supervisar y auditar los procesos de las soluciones de integración de datos.
- Ver detalles del servidor, registros de servidor y datos de almacenamiento del sistema.
- Filtrar datos del archivo de registro estableciendo criterios de filtrado para extraer únicamente la información necesaria.
- Ver el historial de ejecución de la línea de ensamblaje en forma de gráfico.
- Crear informes que contengan el estado de ejecución de la línea de ensamblaje que se envían automáticamente a los usuarios por correo electrónico.

Ventajas del Panel de control de IBM Security Directory Integrator

Las ventajas son:

- Facilita un despliegue más rápido y sencillo de soluciones de integración de datos
- Crea y configura integraciones sencillas sin utilizar el Editor de configuración
- Genera informes planificados por correo electrónico del historial de la línea de ensamblaje, que se envían mediante la función Informe de ejecución del Panel de control. Puede utilizar el informe para resolver los requisitos de alta disponibilidad/migración tras error del despliegue.
- Facilita la integración con el entorno de desarrollo de Eclipse.

- Asegura una gestión efectiva y eficaz de las ejecuciones de líneas de ensamblaje

Acceso a la aplicación Panel de control

Puede acceder a la aplicación web Panel de control desde el navegador o desde el Editor de configuración.

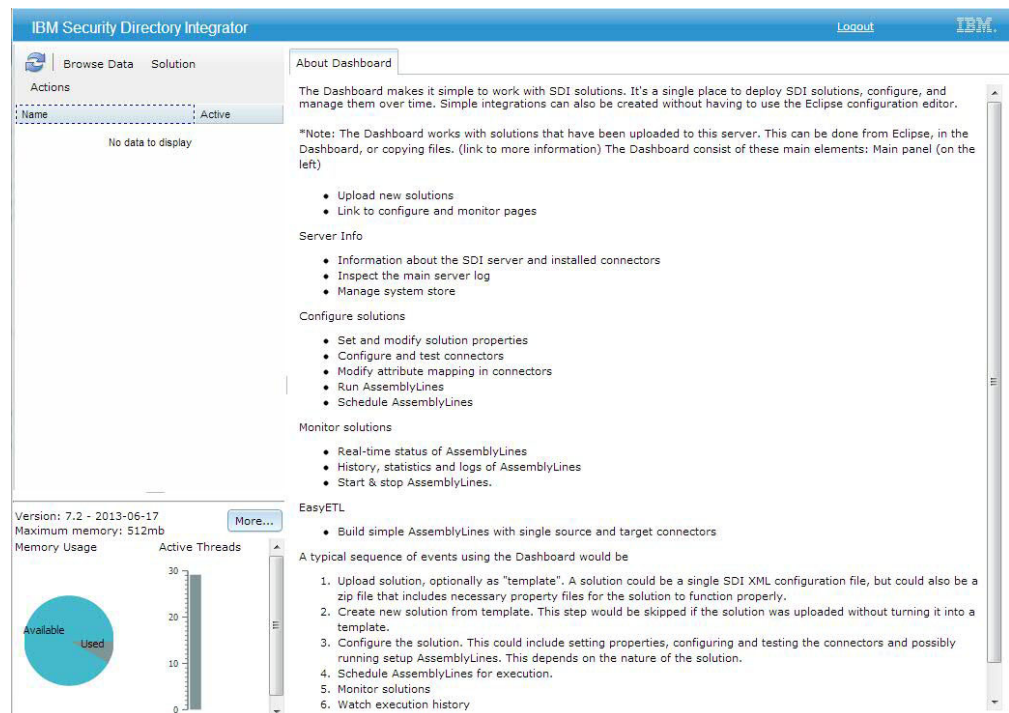
Antes de empezar

Inicie el servidor de IBM Security Directory Integrator antes de acceder al Panel de control.

Apertura desde un navegador

Procedimiento

Desde un navegador, acceda a `https://<nombre de host>:<puerto_api_rest>/dashboard`. Se abrirá la página de inicio del Panel de control.



Nota: De forma predeterminada, puede acceder al Panel de control en el host local. Para autenticar a un usuario, configure el archivo de propiedades de IBM Security Directory Integrator para añadir un nombre de usuario y contraseña de LDAP para el acceso local y remoto.

Apertura desde el menú Inicio de Windows

Procedimiento

En el menú **Inicio** de Windows, seleccione **Inicio > Todos los programas > IBM Security Directory Integrator v7.2 > Abrir Panel de control**. Se abrirá la página de inicio del Panel de control de IBM Security Directory Integrator.

Tabla 14. Opciones de menú del Panel de control de IBM Security Directory Integrator

Opción de menú		Descripción
Examinar datos		Utilice esta opción para configurar el conector seleccionado y examinar su contenido.
Solución	Supervisor	Utilice esta opción para realizar un seguimiento del progreso de ejecución de la línea de ensamblaje y ver el historial de ejecución de la solución.
	Iniciar	Utilice esta opción para iniciar la solución seleccionada.
	Detener	Utilice esta opción para terminar una solución en ejecución.
	Configurar	Utilice esta opción para abrir la solución seleccionada para modificarla.
	Suprimir	Utilice esta opción para suprimir la solución seleccionada del servidor.
	Desbloquear solución	Utilice esta opción para desbloquear una solución.
Acciones	Examinar datos	Utilice esta opción para configurar el conector seleccionado y examinar su contenido.
	Mostrar registro del sistema	Utilice esta opción para ver el contenido del registro del sistema (ibmdi.log).
	Crear solución	Utilice esta opción para elegir una plantilla y crear una solución de integración de datos.
	Cargar solución	Utilice esta opción para cargar una solución existente en el servidor de IBM Security Directory Integrator.
	Mostrar detalles del servidor	Utilice esta opción para ver información sobre el servidor de IBM Security Directory Integrator, tal como la versión, componentes instalados y demás información del servidor.

Configuración de Internet Explorer para el acceso remoto

Añada los valores de configuración obligatorios para acceder al Panel de control desde un sistema remoto en un navegador Internet Explorer, donde esté habilitado Internet Explorer Enhanced Security Configuration (IE ESC).

De forma predeterminada, IE ESC bloquea todos los scripts que se ejecutan en una página web. El panel de control carga varios scripts antes de mostrar nada en la página web. Por lo tanto, un navegador Internet Explorer habilitado para IESC mostrará una página en blanco al abrir el Panel de control. Para acceder a la página, debe añadir los sitios que alojan el Panel de control a la lista segura de Internet Explorer.

1. Desde el menú de **Internet Explorer**, pulse **Herramientas > Opciones de Internet**.
2. Pulse la pestaña **Seguridad**.
3. Pulse **Sitios de confianza**.

4. Pulse **Sitios**.
5. En el campo **Añadir este sitio web a la zona**, especifique el URL del Panel de control. Por ejemplo: <https://mydashboard.com/dashboard/>.
6. Pulse **Añadir**.
7. Pulse **Cerrar** y, a continuación, **Aceptar** para cerrar la página y guardar la configuración.
8. Reinicie el navegador Internet Explorer.
9. Acceda al Panel de control en el navegador Internet Explorer.

Carga de una solución de integración de datos

Utilice la ventana Cargar soluciones de Panel de control para cargar una solución de integración de datos existente e instalarla en el servidor de IBM Security Directory Integrator. También puede guardar como plantilla la solución actualizada.

Acerca de esta tarea

Puede utilizar el Panel de control para cargar una solución existente de integración de datos de IBM Security Directory Integrator, que es un archivo de configuración XML, para instalarla en el servidor. Puede modificar la solución con las propiedades de requisitos antes del despliegue. También puede cargar la solución existente para guardarla como plantilla. La plantilla guardada se puede utilizar para crear muchas instancias de la misma integración.

Procedimiento

1. En el panel de navegación de la ventana Panel de control, pulse **Acciones > Cargar soluciones**.
2. Defina los siguientes valores en la ventana Cargar soluciones.

Opción	Descripción
Examinar	Examine la solución existente que desea cargar.
Cargar como plantilla	Guarda como plantilla la solución cargada Nota: La plantilla guardada está ahora instalada en el servidor de IBM Security Directory Integrator (directorio de configuración).
Sobrescribir solución existente	Sobrescribe las soluciones o las plantillas instaladas existentes
Nombre de la solución	Especifica el nombre de la solución cargada. Nota: También podrá ver los nombres de las soluciones instaladas actualmente y las soluciones de plantilla en la ventana Cargar soluciones.

3. Pulse **Aceptar**.

Creación de una solución de integración de datos

Utilice la ventana Crear soluciones de Panel de control para crear una solución de integración de datos utilizando diferentes opciones.

Procedimiento

1. En el panel de navegación de la ventana Panel de control, pulse **Acciones > Crear soluciones**.
2. Seleccione una de las siguientes opciones en la ventana Crear soluciones.

Opción	Descripción
Solución instalada	La solución se crea basándose en una solución ya instalada en el servidor de IBM Security Directory Integrator.
Plantilla	La solución se crea basándose en una solución existente, que se carga como plantilla.
Valor predeterminado	Se crea una solución EasyETL que tiene una única línea de ensamblaje con dos conectores.

3. Escriba el nombre de la solución en el campo **Nombre de solución**.
4. Pulse **Aceptar**.

Configuración de soluciones

El Panel de control consta de varios editores para configurar los componentes de su solución de integración de datos.

En el Panel de control, podrá modificar la solución seleccionada con los valores apropiados que se ajusten a sus necesidades antes del despliegue. Para favorecer configuraciones más rápidas, puede limitar la cantidad de información visible mientras configura una solución. También añadir nuevas páginas de configuración para incluir propiedades que sean específicas de la solución.

Solo las soluciones de integración de datos instaladas se mostrarán en el panel de navegación de la ventana Panel de control. Cada solución contiene componentes, como líneas de ensamblaje y conectores asociados. Estos componentes se muestran como una estructura en árbol en el Editor de soluciones. En el Editor de soluciones, podrá instalar, crear, configurar, desplegar y supervisar sus soluciones mediante los siguientes editores:

- Editor de línea de ensamblaje - Este editor se utiliza para crear y configurar planificaciones en las líneas de ensamblaje y ejecutar la línea de ensamblaje con su salida en el visor de registros.
- Editor de conector - Este editor se utiliza para configurar los conectores de la línea de ensamblaje.
- Editor de EasyETL – Este editor se utiliza para configurar una solución EasyETL, que contiene una única línea de ensamblaje con dos conectores.
- Editor de supervisión – Este editor se utiliza para supervisar la información del rendimiento actual y anterior de su solución.

El Editor de soluciones tiene las siguientes opciones de menú:

Opción	Descripción
Editar interfaz de la solución	Utilice esta opción para seleccionar los componentes de la solución, que están visibles para su edición.

Opción	Descripción
Línea de ensamblaje nueva	Utilice esta opción para crear una solución EasyETL y añadirla a la solución seleccionada.
Renombrar línea de ensamblaje	Utilice esta opción para renombrar la línea de ensamblaje seleccionada.
Suprimir línea de ensamblaje	Utilice esta opción para suprimir líneas de ensamblaje.
Crear informe de ejecución	Utilice esta opción para crear un informe de ejecución para enviar por correo electrónico sobre el estado de las ejecuciones de las líneas de ensamblaje.

Añadir una descripción de la solución

Utilice el Editor de soluciones del Panel de control para añadir información importante sobre la solución de integración de datos.

Procedimiento

1. En el panel de navegación de la ventana Panel de control, seleccione la solución.
2. Pulse **Solución > Configurar**.
3. En el Editor de soluciones, seleccione **Descripción de la solución** y pulse **Editar descripción**.
4. Escriba la descripción en el área de texto.
5. Pulse **Cerrar**.
6. Pulse **Guardar**.

Configuración de una planificación de línea de ensamblaje

Utilice el Editor de líneas de ensamblaje para crear o configurar el planificador de Panel de control para ejecutar líneas de ensamblaje de su solución de integración de datos en la hora especificada.

Creación de una planificación

Procedimiento

1. Seleccione la línea de ensamblaje en la estructura en árbol.
2. En el Editor de soluciones, pulse la pestaña **Planificación**.
3. Pulse **Crear planificación**.
4. Defina los siguientes valores:

Tabla 15. Opciones de planificación

Valor	Opción	Descripción
Planificación	Se inicia automáticamente cuando se inicia la solución	Utilice esta opción para ejecutar la línea de ensamblaje en función de la opción de ejecución seleccionada.
	No iniciar si ya se está ejecutando	
	Terminar planificación si la línea de ensamblaje falla	

Tabla 15. Opciones de planificación (continuación)

Valor	Opción	Descripción
Mes	Todos los meses	Utilice esta opción para seleccionar los meses de ejecución de la planificación.
	Seleccionar meses	
Día	Todos los días	Utilice esta opción para seleccionar los días de ejecución de la planificación.
	Días de la semana	
	Seleccionar días	
Horas/Minutos/Segundos	Horas	Utilice esta opción para seleccionar la temporización de ejecución de la planificación.
	Minutos	
	Segundos	
Compartir registro		Utilice esta opción para especificar si el registro entre el planificador y las líneas de ensamblaje se va a compartir o no.

5. Pulse **Guardar**.

Eliminación de una planificación

Procedimiento

1. Seleccione la línea de ensamblaje en la estructura en árbol.
2. En el Editor de soluciones, pulse la pestaña **Planificación**.
3. Pulse **Suprimir planificación**.
4. Pulse **Guardar**.

Ejecución y detención del planificador de Panel de control

Procedimiento

1. Seleccione la línea de ensamblaje en la estructura en árbol.
2. En el Editor de soluciones, pulse la pestaña **Ejecutar**.
3. Para ejecutar la línea de ensamblaje, pulse **Ejecutar**. La salida se muestra en el visor de registros.
4. Para detener la línea de ensamblaje, pulse **Detener**.

Configuración de un conector

Utilice el Editor de conectores para configurar los detalles del conector, como la información de correlación de los atributos y los detalles de la conexión, para ajustarse a sus requisitos de configuración.

Modificación de los detalles de conexión

Procedimiento

1. Seleccione el conector de la estructura en árbol de la solución.
2. Pulse la pestaña **Conector**.
3. Para ver solo los campos importantes, pulse **Menos**.
4. Para cambiar el tipo de conector, pulse **Elegir componente** y seleccione el conector en la lista **Seleccionar conector**.
5. Modifique los valores de conexión en función de los requisitos.
6. Para probar la conexión al origen de datos, pulse **Probar conexión**.
7. Pulse **Guardar**.

Modificación de una correlación de atributos

Procedimiento

1. Seleccione el conector de la estructura en árbol de la solución.
2. Pulse la pestaña **Correlación de atributos**.
3. Para añadir un atributo;
 - a. Pulse **Añadir**.
 - b. Seleccione un atributo de trabajo en la lista o escriba un nombre nuevo en el texto relleno.
 - c. Pulse **Aceptar**.
4. Para leer los datos del conector y mostrarlos en la correlación de atributos, pulse **Leer siguiente**.
5. Para cerrar la conexión al origen de datos, pulse **Cerrar conexión**.
6. Para modificar el atributo seleccionado, pulse **Acciones**.
 - a. Para editar la asignación del atributo seleccionado, pulse **Editar atributo** y modifique el siguiente tipo de asignación:
 - **Asignación simple**: puede asignar un atributo de la lista de atributos de origen.
 - **Asignación mediante script**: puede escribir un script al atributo en el área de texto.
 - b. Pulse **Cerrar**.
 - c. Para correlacionar el atributo seleccionado, pulse **Correlacionar atributo**.
 - d. Para anular la correlación del atributo seleccionado, pulse **Anular correlación de atributo**.
7. Pulse **Guardar**.

EasyETL de Panel de control

La función EasyETL de Panel de control se puede utilizar para crear y configurar soluciones de integración de datos simples.

La función EasyETL de Panel de control proporciona una interfaz de usuario para crear y configurar líneas de ensamblaje simples con un origen único y un conector de destino. Una solución EasyETL puede:

- Programarse en el Planificador de Panel de control
- Guardarse como una plantilla
- Incluirse en informes de ejecución
- Implementarse y supervisarse en el supervisor de soluciones del Panel de control

Configuración de soluciones EasyETL

Utilice el Editor de EasyETL para configurar y añadir planificaciones a una solución EasyETL.

Procedimiento

1. En el panel de navegación de la ventana Panel de control, seleccione la solución EasyETL.
2. Pulse **Solución > Configurar**.
O también,
 - a. Pulse con el botón derecho del ratón sobre la solución seleccionada.

- b. En el menú, pulse **Configurar**.
3. En el Editor de soluciones, seleccione la solución EasyETL.
4. Añada una descripción a la solución. Consulte el tema “Añadir una descripción de la solución” en la página 238 para obtener más detalles.
5. En el Editor de EasyETL Editor, seleccione la solución y pulse **Configurar**.
6. En la lista **Conector**, seleccione un componente para el conector de origen y los conectores de destino.
7. Especifique los detalles de configuración de los conectores seleccionados en el formulario.
8. Seleccione un analizador en la lista **Analizador**.

Nota: La lista **Analizador** solo está activa cuando el conector seleccionado requiere o puede utilizar un analizador.

Para establecer la conexión y examinar los datos:

- a. Para establecer una conexión y leer los primeros 25 registros del conector, pulse **Conectar**.
- b. Para ver los siguientes 25 registros, pulse **Siguiente**.
- c. Para alternar la vista de registros de datos para mostrar un registro cada vez, pulse **Alternar vista**.
- d. Para ver los siguientes registros en la vista alternada, pulse **Siguiente**.
- e. Para ver solo los atributos seleccionados de la tabla:
 - 1) Pulse el icono **Configurar opciones** de la barra de menús.
 - 2) En la ventana Configurar opciones, seleccione los atributos.
 - 3) Para cambiar el orden de los atributos, pulse en las flechas hacia arriba o abajo.
 - 4) Pulse **Aceptar**.
9. Pulse **Atrás** para volver al Editor de EasyETL.
10. Para ver los atributos de los conectores configurados, pulse **Descubrir**.
11. Para correlacionar los atributos de origen seleccionados con los atributos de destino, pulse **Correlacionar**. Se creará el siguiente tipo de correlación en base a la selección del atributo de origen:
 - Si solo hay un atributo seleccionado en la vista de origen y también en la vista de destino, se creará una correlación uno-a-uno entre los dos atributos.
 - Si se seleccionan varios atributos en la vista de origen y no se selecciona ninguno en la vista de destino, se creará una correlación uno-a-uno para cada atributo. Si no hay ningún atributo en la vista de destino, se crearán.
 - Si se seleccionan varios atributos en la vista de origen y solo se ha seleccionado un atributo en la vista de destino, realice una de las siguientes tareas:
 - Pulse **Copiar** para copiar atributos de destino en sus atributos de destino equivalentes.
 - Pulse **Fusionar** para fusionar todos los valores del origen en un atributo del destino.
 - Pulse **Concatenar** para concatenar los valores del atributo de origen como un único valor en el atributo de destino.
12. Pulse **Guardar**.

Configuración de servidor

Utilice la ventana Información del servidor para ver y modificar la configuración del servidor.

Puede ver las siguientes propiedades del servidor y configurar los comportamientos en la ventana Información del servidor:

- Especificar el número máximo de archivos de registro que se deben crear y mantener
- Iniciar y detener Tombstone Manager
- Definir valores del servidor LDAP para autenticar usuarios a fin de acceder a la aplicación Panel de control
- Ver información de servidor, tal como la versión de IBM Security Directory Integrator con la fecha de compilación, el nombre de host, la dirección IP, la hora de arranque del servidor, el nombre del sistema operativo y el ID de servidor
- Ver los conectores y analizadores instalados en el servidor de IBM Security Directory Integrator
- Ver el contenido de diversos almacenes del sistema que son utilizados por el servidor de IBM Security Directory Integrator

En la esquina inferior izquierda de la ventana Panel de control, podrá ver la información sobre la versión de la aplicación. También podrá supervisar las siguientes medidas para obtener una instantánea del funcionamiento de la aplicación:

- Gráfico circular del uso de la memoria
- Recuento de hebras activas del proceso de servidor

Configuración de valores de registro

Utilice la pestaña Registro y lápidas para habilitar o inhabilitar el registrador predeterminado para especificar el número máximo de archivos de registro para guardar.

Procedimiento

1. En la ventana Panel de control, haga clic en **Acciones > Mostrar detalles del servidor** o en **Más**.
2. Pulse la pestaña **Registro y lápidas**.
3. Defina los siguientes valores:

Opción	Descripción
Registrador predeterminado	Habilita el registrador predeterminado Nota: Utilice el registrador predeterminado, que se adjunta a cada línea de ensamblaje que se ejecuta, si desea archivos de registro individuales para las líneas de ensamblaje.
Número máximo de archivos de registro	Especifica el número máximo de archivos de registro que el registrador puede guardar.

4. Pulse **Actualizar**.

Configuración de las lápidas

Utilice la ficha Registro y lápidas para iniciar el gestor de lápidas del panel de control, que crea registros de lápidas.

Acerca de esta tarea

El gestor de lápidas del Panel de control crea registros de lápidas para cada línea de ensamblaje según finalizan. Un registro de lápida contiene estadísticas, como el número de entradas leídas por los iteradores, el número de registros omitidos, el número de registros actualizados, etc. Las estadísticas se pueden utilizar para mostrar gráficamente la carga de trabajo a lo largo del tiempo.

Procedimiento

1. En la ventana Panel de control, haga clic en **Acciones > Mostrar detalles del servidor** o en **Más**.
2. Para generar registros de lápida, haga clic en **Iniciar gestor de lápidas**.

Nota: Para detener el gestor de lápidas, reinicie el servidor de IBM Security Directory Integrator.

Configuración de los valores de seguridad de Panel de control

Utilice la pestaña Seguridad y conexión para configurar el servidor LDAP con el fin de realizar la autenticación de usuarios de las conexiones locales y remotas.

Procedimiento

1. En la ventana Panel de control, haga clic en **Acciones > Mostrar detalles del servidor** o en **Más**.
2. Pulse la pestaña **Seguridad y conexión**.
3. Defina los siguientes valores:

Opción	Descripción
Local	Especifica las conexiones de un host local
Remoto	Especifica las conexiones de un host no local
Nombre de host LDAP	Especifica el nombre del servidor LDAP
Base de búsqueda LDAP	Especifica el nombre de base de búsqueda LDAP para localizar usuarios.
DN de grupo LDAP	Especifica el nombre DN de grupo LDAP para comprobar la pertenencia de los usuarios.

4. Pulse **Actualizar**.

Visualización de los componentes instalados

Utilice la pestaña Componentes instalados para ver los componentes, tales como conectores y analizadores, que están instalados en el servidor de IBM Security Directory Integrator.

Procedimiento

1. En la ventana Panel de control, haga clic en **Acciones > Mostrar detalles del servidor** o en **Más**.
2. Para ver la lista de componentes, pulse la pestaña **Componentes instalados**.

Visualización de los datos de almacenamiento del sistema

Utilice la pestaña Almacenes de servidor para ver el contenido de diversos almacenes del sistema que son utilizados por el servidor de IBM Security Directory Integrator.

Procedimiento

1. En la ventana Panel de control, haga clic en **Acciones > Mostrar detalles del servidor** o en **Más**.
2. Para ver la lista de almacenes de servidor, pulse la pestaña **Almacenes de servidor**.

Informes de ejecución de Panel de control

Utilice la función Informe de ejecución de Panel de control para crear informes automáticos por correo electrónico del historial de ejecución de las líneas de ensamblaje.

Le permite crear una línea de ensamblaje que se ejecute en función de las planificaciones establecidas para generar informes automáticos por correo electrónico de las líneas de ensamblaje supervisadas. La línea de ensamblaje del informe de ejecución utiliza la base de datos de lápidas de IBM Security Directory Integrator para determinar si las líneas de ensamblaje se han ejecutado desde la última vez que se ejecutó el informe de ejecución.

El informe por correo electrónico contiene información sobre el historial de ejecución e indica si las líneas de ensamblaje supervisadas se han ejecutado correctamente o no. Este informe se envía periódicamente por correo al destinatario especificado.

Creación de informes de ejecución

Utilice la ventana Crear informe de ejecución para crear una línea de ensamblaje de informe de ejecución. Esta línea de ensamblaje sirve para generar informes por correo electrónico automáticos. Los informes por correo electrónico contienen información sobre el historial de ejecución de las líneas de ensamblaje supervisadas y se envían de forma periódica a los destinatarios especificados.

Creación y planificación de un informe de ejecución

Procedimiento

1. En el Editor de soluciones, pulse **Acciones > Crear informe de ejecución**.
2. En la ventana Crear informe de ejecución, escriba un nombre para la línea de ensamblaje del informe de ejecución en el campo **Nombre**.
3. Pulse **Aceptar**.
4. En el Editor de líneas de ensamblaje, defina los siguientes valores:

Tabla 16. Opciones de planificación de RunReport

Valor	Opción	Descripción
Planificación	Se inicia automáticamente cuando se inicia la solución	Ejecuta la línea de ensamblaje en función de la opción de ejecución seleccionada.
	No iniciar si ya se está ejecutando	
	Terminar planificación si la línea de ensamblaje falla	

Tabla 16. Opciones de planificación de RunReport (continuación)

Valor	Opción	Descripción
Mes	Todos los meses	Especifica los meses de ejecución de la planificación
	Mes(es) seleccionado(s)	
Día	Todos los días	Especifica los días de ejecución de la planificación
	Días de la semana	
	Día(s) seleccionado(s)	
Horas/Minutos/Segundos	Horas	Especifica la temporización de la ejecución de la planificación
	Minutos	
	Segundos	
Asunto (éxito)		Especifica la línea de asunto del correo electrónico de ejecución correcta de las líneas de ensamblaje especificadas. Nota: El informe generado utiliza la línea del asunto en función de si las líneas de ensamblaje supervisadas se han ejecutado correctamente o no desde el último informe generado.
Asunto (error)		Especifica la línea de asunto del correo electrónico de ejecución con errores de las líneas de ensamblaje especificadas.
Dirección del remitente		Especifica la dirección de correo electrónico del remitente.
Dirección de destinatario		Especifica la dirección de correo electrónico del destinatario. Nota: Especifique una lista delimitada por comas o puntos y coma para indicar varias direcciones de correo electrónico.
Host SMTP		Parámetro de host SMTP que acepta conexiones SMTP y transfiere el correo al destinatario.
Supervisar líneas de ensamblaje		Lista separada por comas de los nombres de línea de ensamblaje para supervisar
Compartir registro		Especifica si el registro entre el planificador y las líneas de ensamblaje se va a compartir o no

5. Pulse **Guardar**.

Eliminación de una planificación

Procedimiento

1. Seleccione la línea de ensamblaje del informe de ejecución de la estructura en árbol.
2. En el Editor de soluciones, pulse la pestaña **Planificación**.
3. Pulse **Suprimir planificación**.
4. Pulse **Guardar**.

Ejecución y detención del planificador de informes de ejecución

Procedimiento

1. Seleccione la línea de ensamblaje del informe de ejecución de la estructura en árbol.
2. En el Editor de soluciones, pulse la pestaña **Ejecutar**.
3. Para ejecutar la línea de ensamblaje, pulse **Ejecutar**. La salida se muestra en el visor de registros.
4. Para detener la línea de ensamblaje, pulse **Detener**.

Configuración y examen de los datos del conector

Utilice la página Examinar datos del Editor de soluciones para configurar un conector, examinar orígenes de datos del conector o crear una solución de integración de datos.

Procedimiento

1. En el panel de navegación de la ventana Panel de control, haga clic en **Examinar datos**.
2. En la ventana Examinar datos, en la lista **Conector**, seleccione un conector para configurar y examinar su contenido.
3. Seleccione un analizador en la lista **Analizador**.

Nota: La lista **Analizador** solo está activa cuando el conector seleccionado requiere o puede utilizar un analizador.

4. Configure los detalles del conector en el formulario situado en el lado derecho de la ventana.
5. Para crear una solución para el conector seleccionado, haga clic en **Crear integración**.
6. Para establecer una conexión y leer los primeros 25 registros del conector, pulse **Conectar**.
7. Para ver los siguientes 25 registros, pulse **Siguiente**.
8. Para alternar la vista de registros de datos para mostrar un registro cada vez, pulse **Alternar vista**.
9. Para ver los siguientes registros en la vista alternada, pulse **Siguiente**.
10. Para ver solo los atributos seleccionados de la tabla:
 - a. Pulse el icono **Configurar opciones** de la ventana Examinar datos.
 - b. En la ventana Configurar opciones, seleccione los atributos.
 - c. Para cambiar el orden de los atributos, pulse en las flechas hacia arriba o abajo.
 - d. Pulse **Aceptar**.

Supervisor de soluciones

Utilice el Editor de supervisión de Panel de control para realizar un seguimiento del progreso de ejecución de la línea de ensamblaje y ver el historial de ejecución de la solución.

Puede ver las líneas de ensamblaje de la solución seleccionada en el Editor de supervisión, que presenta la información del rendimiento anterior y actual de las líneas de ensamblaje. Podrá supervisar las siguientes actividades:

- Supervisión del estado en tiempo real de las líneas de ensamblaje, que incluye:
 - Hora de inicio de la ejecución de la línea de ensamblaje
 - Hora de finalización de la ejecución de la línea de ensamblaje
 - Hora planificada para la siguiente ejecución de la línea de ensamblaje
 - Número de objetos de datos procesados en la línea de ensamblaje
- Historial de ejecución de las líneas de ensamblaje, mostrado gráficamente como carga de trabajo a lo largo del tiempo.
- Archivos de registro para registrar los resultados y los problemas de ejecución de cada línea de ensamblaje.
- Registros de lápidas de todas las líneas de ensamblaje de la solución.
- Archivo de registro del servidor (`ibmdi.log`)

Inicio y detención de las líneas de ensamblaje

Utilice el Editor de supervisión de Panel de control para iniciar y detener las líneas de ensamblaje.

Procedimiento

1. En el Panel de control, seleccione la solución instalada.
2. Pulse **Solución > Supervisor**.
O también,
 - a. Pulse con el botón derecho del ratón sobre la solución seleccionada.
 - b. En el menú, pulse **Supervisor**.
3. En el Editor de supervisión, seleccione la línea de ensamblaje para realizar las siguientes acciones:
 - Para iniciar una línea de ensamblaje, pulse **Iniciar**.
 - Para detener una línea de ensamblaje, pulse **Detener**.

Los detalles relativos a la ejecución se muestran en el editor. El icono de color verde con el nombre de la línea de ensamblaje indica que la línea está activa.

Visualización del historial de ejecución de la línea de ensamblaje

Utilice el Editor de supervisión de Panel de control para ver el historial de ejecución de la línea de ensamblaje en forma de gráfico.

Procedimiento

1. En el Editor de supervisión, seleccione la línea de ensamblaje.
2. Pulse **Historial** o efectúe una doble pulsación en la línea de ensamblaje seleccionada. Los detalles de ejecución de la línea de ensamblaje se muestran en una vista de gráfico como una carga de trabajo a lo largo del tiempo.
3. Seleccione uno de los siguientes contadores del gráfico:

- Obtener
 - Errores
 - Añadir
 - Suprimir
 - Buscar
 - Modificar
4. Pulse el nodo del gráfico para ver el archivo de registro de una determinada ejecución. También podrá ver las estadísticas de esa ejecución como ayuda contextual.

Visualización de registros de lápida

Utilice la pestaña Lápidas del Editor de supervisión de Panel de control para ver los registros de lápidas que se crean al finalizar la ejecución de las líneas de ensamblaje.

Procedimiento

1. Vaya al Editor de supervisión de Panel de control.
2. Para ver las lápidas registradas, pulse la pestaña **Lápidas**.

Nota: Solo podrá ver los registros de lápidas cuando se el Gestor de lápidas esté en ejecución. Para obtener más información, consulte el tema “Configuración de las lápidas” en la página 242.

Visualización de archivos de registro

Utilice la pestaña Archivos de registro del Editor de supervisión del Panel de control para ver los archivos de registro creados para la ejecución de cada línea de ensamblaje. Los archivos de registro se utilizan para analizar problemas de forma rápida y sencilla y solucionar cualquier problema.

Procedimiento

1. En el Editor de supervisión, pulse la pestaña **Archivos de registro**. Los archivos de registro de ejecución de todas las líneas de ensamblaje se muestran como una estructura en árbol.
2. Para ver el contenido del archivo de registro, selecciónelo en la estructura en árbol y efectúe una doble pulsación sobre él.
3. Para buscar información concreta en el contenido del registro, escriba el texto que desee buscar en el campo **Buscar**.
4. Seleccione la casilla de verificación **Incluir origen** del origen del mensaje o el componente que registró el mensaje.
5. Pulse **Opciones** para configurar los siguientes detalles:

Opción	Descripción
Tamaño de la página	
Tipos de mensaje	Seleccione alguno de los siguientes tipos de mensaje para incluir en el archivo de registro: <ul style="list-style-type: none"> • INFO • WARN • ERROR • DEBUG

Opción	Descripción
Opciones de visualización	Seleccione opciones para mostrar la fecha, la hora, el origen del mensaje o los números de línea en el archivo de registro: <ul style="list-style-type: none">• Mostrar fecha• Mostrar hora• Incluir origen• Mostrar números de línea

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en EE.UU. Es posible que IBM no ofrezca en otros países los productos, servicios o características que se describen en este documento. Consulte al representante local de IBM para obtener información acerca de los productos y servicios que actualmente están disponibles en su localidad. Las referencias a productos, programas o servicios de IBM no pretenden afirmar ni dar a entender que únicamente puedan utilizarse dichos productos, programas o servicios de IBM. En su lugar, puede utilizarse cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes pendientes de aprobación que cubran alguno de los temas tratados en este documento. La entrega de este documento no le confiere ninguna licencia sobre dichas patentes. Puede enviar sus consultas sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 EE.UU.

Para consultas sobre licencias referentes a información sobre el juego de caracteres de doble byte (DBCS), póngase en contacto con el departamento de Propiedad intelectual de IBM de su país o envíe las consultas, por escrito, a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokio 103-8510, Japón

El párrafo siguiente no se aplica en el Reino Unido ni en ningún otro país donde estas disposiciones no sean coherentes con la legislación local:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O ADECUACIÓN PARA UN FIN DETERMINADO.

Algunos estados no permiten la renuncia a garantías explícitas o implícitas en determinadas transacciones. Por lo tanto, es posible que estas indicaciones no sean de aplicación en su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; esos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede efectuar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento y sin previo aviso.

Cualquier referencia hecha en esta publicación a sitios web que no sean de IBM se proporciona solamente para la comodidad del usuario y no representa un aval de esos sitios web. La información de esos sitios web no forma parte de la información del presente producto de IBM y la utilización de esos sitios web se realiza bajo la propia responsabilidad del usuario.

IBM puede utilizar o distribuir la información que se le proporcione del modo que estime apropiado sin incurrir por ello en ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información acerca de él con la finalidad de: (i) intercambiar información entre programas creados independientemente y otros programas (incluido el presente programa) y (ii) utilizar mutuamente la información que se ha intercambiado, deben ponerse en contacto con:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 EE.UU.

Esta información puede estar disponible, sujeta a las condiciones y los términos apropiados, incluido en ciertos casos el pago de una cuota.

IBM proporciona el programa bajo licencia descrito en este documento y todo el material bajo licencia disponible para el mismo según las condiciones del contrato de cliente IBM, del contrato internacional de programas bajo licencia IBM o cualquier contrato equivalente entre ambas partes.

Cualquier dato sobre rendimiento que contiene este documento se ha determinado en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar de forma significativa. Es posible que se hayan realizado algunas mediciones en sistemas a nivel de desarrollo y no existe ninguna garantía que dichas mediciones sean iguales en los sistemas de disponibilidad general. Asimismo, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información relacionada con productos que no son de IBM procede de los proveedores de dichos productos, sus anuncios publicados u otras fuentes de disponibilidad general. IBM no ha comprobado estos productos y no puede confirmar la precisión del rendimiento, compatibilidad o cualquier otra afirmación relacionada con productos que no son de IBM. Las consultas sobre las posibilidades de los productos que no son de IBM se deben dirigir a los proveedores de dichos productos.

Todas las declaraciones relativas a las intenciones futuras de IBM están sujetas a cambios o cancelaciones sin previo aviso y únicamente representan planes y objetivos.

Todos los precios de IBM mostrados son precios de venta al público sugeridos por IBM, son actuales y están sujetos a cambios sin previo aviso. Los precios para distribuidores pueden variar.

Esta información es únicamente para propósitos de planificación. La información que contiene este documento puede cambiar antes de que los productos descritos estén disponibles.

Esta información contiene ejemplos de datos e informes empleados en operaciones empresariales diarias. A fin de ofrecer una ilustración lo más completa posible, los ejemplos incluyen los nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con los nombres y las direcciones empleados por una empresa real es totalmente casual.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustran las técnicas de programación en diferentes plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier modo sin abonar nada a IBM, para fines de desarrollo, uso, comercialización o distribución de programas de aplicación que se ajusten a la interfaz del programa de aplicación de la plataforma operativa para la que se han escrito estos programas de ejemplo. Estos ejemplos no se han comprobado detenidamente bajo todas las condiciones. Por lo tanto, IBM no puede garantizar ni implicar la fiabilidad, la capacidad de servicio ni el funcionamiento de estos programas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier modo, sin pago alguno a IBM, con fines de desarrollo, uso, comercialización o distribución de programas de aplicación que se ajusten a las interfaces de programa de aplicación de IBM.

Cada copia o cualquier porción de estos programas de ejemplo o cualquier trabajo derivativo debe incluir un aviso de copyright, tal como se indica a continuación:

© (nombre de su empresa) (año). Partes de este código se derivan de programas de ejemplo de IBM Corp. © Copyright IBM Corp. _escriba el año o años_. Reservados todos los derechos.

Si está viendo esta información en formato de copia software, es posible que no aparezcan las fotografías y las ilustraciones a color.

Marcas registradas

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corp., registradas en numerosas jurisdicciones de todo el mundo. Otros nombres de servicios y productos pueden ser marcas registradas de IBM u otras empresas. Encontrará una lista actualizada de marcas registradas de IBM en la sección "Información sobre copyright y marcas registradas" del sitio web www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript y todas las marcas registradas basadas en Adobe son marcas registradas de Adobe Systems Incorporated en EE.UU. o en otros países.

IT Infrastructure Library es una marca registrada de la CCTA (Central Computer and Telecommunications Agency) que actualmente forma parte de la OGC (Office of Government Commerce).

Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Intel Centrino, el logotipo de Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium y Pentium son marcas registradas de Intel Corporation o de sus empresas filiales en los Estados Unidos o en otros países.

Linux es una marca registrada de Linus Torvalds en Estados Unidos o en otros países.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en Estados Unidos y/o en otros países.

ITIL es una marca registrada y una marca registrada comunitaria de OGC (Office of Government Commerce), y está registrada en la Oficina de Patentes y Marcas de Estados Unidos.

UNIX es una marca registrada de Open Group en Estados Unidos y en otros países.



Java y todas las marcas registradas y logotipos basados en Java son marcas registradas de Oracle o de sus empresas filiales.

Cell Broadband Engine es una marca comercial de Sony Computer Entertainment, Inc. en los Estados Unidos o en otros países y se utiliza bajo licencia.

Linear Tape-Open, LTO, el logotipo de LTO, Ultrium y el logotipo de Ultrium son marcas registradas de HP, IBM Corp. y Quantum en Estados Unidos y en otros países.

Índice

A

Abrir panel de control 234
accesibilidad ix
acciones en objetos 190
Acumulador 73
Agrupación 132
agrupación de líneas de ensamblaje 15
almacén de propiedades 88
Almacén de propiedades de usuario 211, 212
almacén del sistema 182
Almacén del sistema 211
almacén delta 212
Almacén delta 211
ámbito 79
AMC 185
Analizador 32, 125, 138
Añadir servidor 93
aplicación delta 21
árbol de proyecto 85
archivos de configuración 86
asistente 147
Asistentes 147
Atributo 46, 50
AttMap 26

B

base de búsqueda 140
bibliotecas 185
bibliotecas de Java 185
BOM 33
Buscar 9

C

Calcular cambios 229
Cambiar la herencia 189
Cargar 203
CE 83
clases Java personalizadas 33
codificación de caracteres 33
código de operación Delta 218
coloración de sintaxis 179
Comp. de función 24
compartimiento de proyecto 173
complemento de software 124
Complementos de software 34, 39, 68
Complementos de software de línea de ensamblaje 68
complementos de software de servidor 68, 71
Componente de la línea de ensamblaje 71
Componente de script 24, 68
Componentes de flujo 29
Componentes de rama 29
Comportamiento ante un valor nulo 26
comprobación de la sintaxis 180
concepto de delta 217

Conector 122
 Biblioteca de conectores 4
 línea de ensamblaje 4
Conectores de detección de cambios 226
Config 86
Configuración de conector 125, 239
Conn 46
contribución 89
correlación avanzada 50
Correlación de atributos 26, 63, 115, 120, 121, 123, 239
Correlación de atributos avanzada 75
Correlación de atributos de entrada 120
Correlación de atributos de salida 121
Creación de instancias de clases 34
Creación de instancias de una clase Java 80
Creación de instancias en tiempo de ejecución 34
Creación de un conector 122
creador de proyectos 87, 176, 188
Crear informes de ejecución 244
Criterio de enlace 10
Criterio de enlace avanzado 22
Criterio de enlace simple 21
Criterios de enlace 9, 11, 21, 126
Current 46
cursos ix
CVS 172, 173, 175

D

Definición de agrupación de conectores 132
Delta 17, 130, 217
Deltas 217
depuración 162
Depuración 169, 193
depuración de un servidor 169
depurador 162
Depurador 159
Desarrollo de líneas de ensamblaje 157
detección de errores graves 39
detección delta 17
determinación de problemas x
directorio de instalación 83
directorio de soluciones 83
directorio de trabajo 83
directorio Runtime 85

E

editor de conectores 122
Editor de conectores 122
Editor de configuración 83
editor de expresiones 95
editor de formularios 142
Editor de líneas de ensamblaje 97
Editor rápido 107
Editores externos 181

Ejecución de líneas de ensamblaje 157
Ejecutar línea de ensamblaje 114
Ejecutar líneas de ensamblaje 159
ejemplos de deltas 229
enlaces de teclas 190
entorno de ejecución 65
entrada 50
Entrada 46
entrada Delta 217, 218, 220, 229
Entrada Delta 217
entrada work inicial 8, 74
entradas Delta 220
Epílogo 34
espacio de trabajo 90
esquema 123
Esquema 115
establecer parámetros 75
ETL 203
evaluación de variables 180
Explorador de datos 135
Explorador de datos de secuencia 138
Explorador de datos genérico 137
Explorador de datos JDBC 139
Explorador de datos LDAP 140
expresiones 95
Expresiones 40, 43, 45
Extraer/transformar/cargar 203

F

fase de respuesta 14, 16
FC 24
fechas 81
Flujo 39
Flujo de línea de ensamblaje 34
formación ix
formulario de configuración 155
formularios 142
Función 24
funciones Delta 217

H

Herencia 133, 189
Herencia del conector 133

I

IBM
 Soporte de software x
 Support Assistant x
Importar configuración 147
importar configuración de servidor 147
Informes de la línea de ensamblaje 157
inhabilitar 74
Interfaz de la solución 185
Interfaz de usuario 90
Iterador 7, 8
Iteradores 7
IWE 8, 74

J

JavaScript 49, 177
JDBC 139
juego de caracteres 33

L

LDAP 140
LE 1
línea de ensamblaje 1, 24, 26, 39, 159
Línea de ensamblaje de estado 185
líneas de ensamblaje, agrupación 15
lógica personalizada 49

M

mejoras en la edición de JavaScript 177
modalidad Actualizar 11
Modalidad Actualizar 10, 229
modalidad Buscar 9
Modalidad Buscar 9
Modalidad CallReply 13, 14
modalidad de simulación 195
modalidad del conector 6
modalidad Delta 21
Modalidad Delta 17, 228
modalidad Iterador 7, 8, 220
modalidad Servidor 14, 15, 16
Modalidad Sólo adición 10
modalidad Suprimir 12, 13
modelo de datos 50
Modelo de interfaz de usuario 89
modelo de proyecto 83
motor delta 17

N

Nuevo asistente de componente 150

O

objeto de entrada 46, 52
objetos jerárquicos 52
objetos permanentes 212
Opciones de ejecución 169
Opciones de línea de ensamblaje 100
operaciones 72

P

página de inicio de Panel de control 234
parámetros 142
pestaña Conexión 125
pestaña Delta 130
Pestaña Errores de conexión 128
planificar informe de ejecución 244
Preferencias 181, 190
Preferencias del CE 181
Primitivas 76
Prólogo 34
Propiedades 88
propiedades de componente 65
Propiedades de ejecución 65
propiedades de servidor 188

Propiedades del proyecto 170
Proyecto 83, 85
proyecto compartido 175
punto de ampliación 89

R

rama 31
Recinto de seguridad 193, 194
Reconexión 128
Registro de entradas de la línea de ensamblaje 194
regla 180
repetidor 162
Repetidor 159
repetidor de datos 162
representación de datos 76
Reproducción del recinto de seguridad 194
resolución de problemas x

S

salida 31
script de inicialización 142
script de sucesos 142
Scripts 49, 63
SDI, proyecto 85
secciones del formulario 155
selección de analizador 125
Selección del servidor 170
Servidor predeterminado 170
Servidor SDI 84
Servidores 16
solución de SDI 85
Soporte de equipo 172
Suprimir 12, 13
Suprimir servidor 93
sustitución de propiedades 88

T

TCB 72
TCB (bloque de llamada de tareas) 72
terminación de código 177
tipo de datos char 77
tipos de datos JavaScript 76

U

UI 90

V

Valor 50
valores binarios 80
valores de coma flotante 81
valores de fecha 81
valores del almacén del sistema de configuración 182
valores nulos 26
ventana Aplicación 90
ventana principal 90
vista Problemas 176
Vista Servidores 93

vista Servidores SDI 84

W

Work 46



Impreso en España

SC11-7821-03

