IBM InfoSphere Information Server
Versión 11 Release 3

Guía de conectividad para IBM WebSphere ILOG JRules

SC43-1235-00

IBM InfoSphere Information Server
Versión 11 Release 3

Guía de conectividad para IBM WebSphere ILOG JRules

SC43-1235-00

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información del apartado “Avisos y marcas registradas” en la página 47.

Contenido

IBM WebSphere ILOG JRules

Con IBM® WebSphere ILOG JRules Connector, puede invocar los conjuntos de reglas de ILOG JRules desde un trabajo.

Nota: Los productos de la familia IBM WebSphere ILOG JRules Business Rules Management System (BRMS) han cambiado de nombre para denominarse IBM Operational Decision Manager. El componente al que accede el conector se denomina ahora IBM Decision Server. Los clientes y la documentación de InfoSphere Information Server no se han actualizado para reflejar este cambio.

Visión general

ILOG JRules Business Rules Management System (BRMS) permite a los clientes para externalizar reglas empresariales complejas de aplicaciones. Con la etapa ILOG JRules, puede invocar reglas empresariales complejas en el contexto de un trabajo.

La etapa proporciona trabajos paralelos de DataStage con acceso al motor de ILOG JRules. El motor de JRules se puede utilizar para realizar transformaciones de datos complejas en los registros del trabajo.

Si el conector está configurado para acceder a JRules en la modalidad de motor de núcleo, invoca los conjuntos de reglas que residen en los archivos de archivador de conjunto de reglas en el host del motor de IBM InfoSphere Information Server. Si el conector está configurado para acceder a JRules en modalidad gestionada localmente (J2SE) o gestionada remotamente (J2EE), invoca los conjuntos de reglas que persisten en el repositorio de reglas.

El conector ILOG JRules consta de un componente de tiempo de diseño y un componente de tiempo de ejecución.

El componente de tiempo de diseño consta del editor de etapas que recopila las entradas del usuario. También consta del asistente de configuración que se utiliza para configurar las propiedades de la etapa y las definiciones de esquema de enlace del conector según las definiciones de parámetros del conjunto de reglas especificado.

El componente de tiempo de ejecución del conector es responsable de invocar los conjuntos de reglas dentro del contexto del trabajo de DataStage. ILOG JRules acepta datos de entrada en forma de objetos Java y los datos de salida devueltos por ILOG JRules también consisten en objetos Java. La funcionalidad básica del componente de tiempo de ejecución es leer los registros de DataStage de entrada enviados por la etapa ascendente y convertirlos en objetos Java. A continuación, invoca el conjunto de reglas mediante los objetos Java instanciados y convierte los objetos Java devueltos por el conjunto de reglas en registros de DataStage. A continuación, estos registros se envían a la etapa descendente. Por consiguiente, la etapa ILOG JRules sólo puede operar en la modalidad de solicitud/respuesta, lo que significa que debe colocarse en el trabajo como una etapa intermedia y no como la primera o la última etapa. Es similar, por ejemplo, al uso de la etapa Transformer.

Modalidad de motor JRules

La modalidad de motor JRules especifica el tipo de interfaz que el conector utiliza para localizar el conjunto de reglas y configurar la etapa en tiempo de diseño y establecer la sesión de JRules e invocar el conjunto de reglas en tiempo de ejecución. Puede seleccionar el Motor de núcleo, XU RES J2SE, o XU RES J2EE como modalidad de motor JRules.

Motor de núcleo

En esta modalidad de motor, el conector accede al motor de JRules directamente a través del archivo JAR del motor. El conector apunta el motor de JRules al archivo de archivado de conjunto de reglas físico que contiene la definición del conjunto de reglas que se debe ejecutar.

Para utilizar esta modalidad, debe exportar el conjunto de reglas como un archivo de conjunto de reglas. El conjunto de reglas se exporta como un archivo JAR que debe copiarse en la máquina en la que se ejecuta el motor de IBM InfoSphere Information Server. La vía de acceso de este archivo JAR debe especificarse en la propiedad **Ubicación de conjunto de reglas** de la etapa.

Modalidad XU RES J2SE

En esta modalidad del motor, el motor de JRules se ejecuta como un servidor de ejecución de reglas gestionado localmente (RES) en el mismo proceso Java que el conector. Para invocar el conjunto de reglas, el conector apunta el motor de JRules a la definición del conjunto de reglas en el repositorio de conjunto de reglas.

El repositorio de conjunto de reglas puede ser un repositorio basado en archivos o un repositorio basado en bases de datos. Si el repositorio está basado en archivos, debe estar en la misma máquina en la que se ejecuta el motor de IBM InfoSphere Information Server. La configuración sobre si desea utilizar un repositorio basado en archivos o un repositorio basado en bases de datos se especifica en el archivo de configuración `ra.xml` de JRules. Para obtener más detalles sobre este archivo de configuración, consulte la documentación de JRules. Cuando se utiliza XU RES de Java 2 Standard Edition (J2SE), la propiedad de la etapa **Ubicación de conjunto de reglas** especifica la vía de acceso del conjunto de reglas en el repositorio.

Modalidad XU RES J2EE

En esta modalidad del motor, el motor de JRules se ejecuta como un RES gestionado de forma remota en el servidor J2EE (Java 2 Enterprise Edition). Para invocar el conjunto de reglas, el conector apunta el motor de JRules a la definición del conjunto de reglas en el repositorio de conjunto de reglas. La conexión con el repositorio de conjunto de reglas se configura cuando el RES se despliega en el servidor J2EE. Durante la ejecución del trabajo, el conector actúa como cliente EJB3 remoto y establece una conexión con el componente de sesión de regla EJB3 que debe desplegarse en el servidor J2EE en el que se ha desplegado Rule Execution Server. Cuando se utiliza XU RES J2EE, la propiedad de etapa de ubicación de conjunto de reglas especifica la vía de acceso del conjunto de reglas en el repositorio.

Nota: Para los tipos de operaciones por lotes, el motor de núcleo y XU RES J2SE generalmente proporcionan las modalidades para un mejor rendimiento que la modalidad XU RES J2EE porque el conector intercambia objetos Java con el motor de JRules que se ejecuta en la misma Java Virtual Machine que el conector. En la modalidad de motor XU RES J2EE, el conector accede al motor de JRules que se ejecuta en el servidor J2EE a través de la interfaz EJB remota y los objetos Java que

se intercambian se tienen que serializar y deserializar como parte del proceso que puede incurrir en una sobrecarga significativa.

Proceso de la modalidad de proceso por lotes y la modalidad de clave

Cada enlace de entrada de la etapa del conector de JRules corresponde a un parámetro IN o IN_OUT del conjunto de reglas especificado para la etapa. El conector convierte registros recuperados en los enlaces de entrada a los valores de Java y los pasa a JRules como los valores de los parámetros de conjunto de reglas IN e IN_OUT correspondientes. Un concepto clave asociado con la funcionalidad de tiempo de ejecución es el de ciclo de ejecución de reglas. Durante cada ciclo de ejecución de reglas, el conector lee uno o más registros de los enlaces de entrada, los convierte en objetos Java y los envía a JRules.

El número de registros recuperados de los enlaces de entrada durante cada ciclo de ejecución de la regla se rige por la modalidad de proceso por lotes y la modalidad de clave. Puede configurar la modalidad de proceso por lotes o la modalidad de clave de los registros de proceso. La estrategia que el conector utiliza para procesar los registros del primer enlace de entrada es diferente de la estrategia que utiliza para procesar los registros de los demás enlaces de entrada. El conector clasifica internamente los enlaces de entrada en enlace maestro y enlaces secundarios. El primer enlace de entrada es el enlace maestro y el resto de los enlaces de entrada son enlaces secundarios.

La información de este tema se aplica a escenarios en los que se cumplen las condiciones siguientes:

- Ninguno de los enlaces de entrada están marcados como enlaces de almacenamiento intermedio. Para obtener información acerca de la utilización de las modalidades de proceso por lotes y clave con enlaces de almacenamiento intermedio, consulte el tema *Enlaces de almacenamiento intermedio*.
- El tipo de XOM para los parámetros de conjunto de reglas es Java XOM. Para obtener información acerca de la utilización de las modalidades de proceso por lotes y clave con XOM dinámico, consulte el tema *Tipos de modelo de objeto de ejecución*.

Estrategia de proceso de registros

El conector explícitamente establece los valores para todos los parámetros IN e IN_OUT del conjunto de reglas antes de invocar el conjunto de reglas. En algunos casos, el valor que se establece para un parámetro puede ser el valor NULL. Sin embargo, el conector no invoca un conjunto de reglas sin antes establecer los valores de todos los parámetros IN e IN_OUT del conjunto de reglas.

Cuando no está disponible ningún registro en el enlace maestro, el conector garantiza que ningún registro está disponible en ninguno de los enlaces secundarios y el trabajo finaliza satisfactoriamente. Sin embargo, si hay más registros disponibles para algunos enlaces secundarios, el conector los rechaza si se ha definido el enlace de rechazo para dicho enlace secundario y la opción **Error de registro restante** está seleccionada para el enlace de rechazo. De lo contrario, el conector registra un error y el trabajo falla.

Si los registros no están disponibles en un enlace secundario, la acción que el conector realiza depende de si el parámetro de conjunto de reglas asociado con dicho enlace es de un tipo de matriz Java o no. Si es así, el conector crea una matriz vacía y la establece como el valor del parámetro de conjunto de reglas. De

lo contrario, el conector trata la condición como un error y rechaza los registros que recibe en todos los enlaces de entrada (incluido el enlace maestro), durante el ciclo de ejecución de reglas actual, pero sólo si se han definido los enlaces de rechazo para esos enlaces de entrada y se ha seleccionado la opción **Error de registro restante** para los enlaces de rechazo. De lo contrario, el conector registra un error y el trabajo falla. Las reglas adicionales que se aplican a la estrategia de proceso de registro dependen de la modalidad en la que la etapa está configurada para ejecutarse.

Modalidad de proceso por lotes

Para habilitar la modalidad de proceso por lotes, la propiedad **Habilitar proceso por lotes** de la etapa debe establecerse en **Sí** y la propiedad **Habilitar clave de proceso** de la etapa debe establecerse en **No**. Un valor entero no negativo *n* también se debe especificar para la propiedad **Tamaño de lote**. El valor predeterminado es 1. En la modalidad de proceso por lotes, para cada ciclo de ejecución de reglas, el conector lee todos los registros disponibles de cada uno de los enlaces de entrada, pero no más de *n* registros por enlace, donde *n* es el tamaño de lote especificado para la etapa. El tamaño de lote 0 significa ilimitado.

Cuando el conector recupera el número necesario de registros en todos los enlaces de entrada, los convierte en objetos Java que luego establece como valores de parámetros de conjunto de reglas e invoca el conjunto de reglas. Si el conjunto de reglas se ejecuta satisfactoriamente, el conector genera registros en los enlaces de salida basándose en los valores de parámetros de conjunto de reglas OUT e IN_OUT, y repite el proceso para la ejecución del siguiente conjunto de reglas. Si la ejecución del conjunto de reglas falla, el conector registra un error y el trabajo falla.

Considere un ejemplo de un trabajo donde la etapa ILOG JRules tiene dos enlaces de entrada, en que cada enlace sólo contiene una columna de tipo Integer. Digamos que los datos recibidos en los enlaces de entrada se especifican como se indica a continuación (los datos a la derecha llegan en primer lugar):

Enlace 1

19 17 16 15 14 12 11 10

Enlace 2

20 18 17 15 12 10 8

Digamos que la modalidad de proceso por lotes está habilitada y que el tamaño de lote es 2. Para estos datos hay cuatro ciclos de ejecución de reglas. Los datos enviados en cada ciclo de ejecución de reglas son los siguientes:

Ciclo de ejecución de reglas 1

Enlace 1: 11 10

Enlace 2: 10 8

Ciclo de ejecución de reglas 2

Enlace 1: 14 12

Enlace 2: 15 12

Ciclo de ejecución de reglas 3

Enlace 1: 16 15

Enlace 2: 18 17

Ciclo de ejecución de reglas 4

Enlace 1: 19 17

Enlace 2: 20

Modalidad de clave

Para habilitar la modalidad de clave, la propiedad **Habilitar proceso de clave** de la etapa debe establecerse en **Sí**. Debe seleccionar las columnas de enlace de entrada para servir como las columnas de clave utilizando la propiedad **Columna de clave [*n*]**, donde *n* es un entero positivo y representa el índice de la columna de clave. Los índices empiezan por 1 para la primera columna de clave seleccionada y aumentan a medida que se añaden más columnas. El nombre de cada columna de clave que se utilizará como columna de clave debe especificarse en la propiedad **Nombre de columna** de la etapa bajo la correspondiente propiedad **Columna de clave [*n*]**. Debe especificarse como mínimo una columna de clave, lo que significa que como mínimo la propiedad **Columna de clave[1]** debe estar presente y su valor de propiedad hijo **Nombre de columna** debe haberse establecido. Para añadir una **Columna de clave[2]**, pulse con el botón derecho del ratón en **Columna de clave[1]** y seleccione **Añadir valor de propiedad**. Repita el proceso para añadir más valores de **Columna de clave [*n*]**. Para establecer la propiedad **Nombre de columna** de la etapa para una **Columna de clave [*n*]** en concreto, es posible pulsar el botón **Seleccionar** que aparece a la derecha de **Nombre de columna**. Las columnas que se ofrecen para la selección son las columnas que existen en todos los enlaces de entrada.

Para cada columna de clave seleccionada, también es necesario especificar la dirección en la que los registros de los enlaces de entrada se ordenan respecto a los valores de dicha columna de clave determinada. Esto se especifica en la opción **Dirección de ordenación** bajo la correspondiente **Columna de clave [*n*]**. Los valores soportados son **Ascendente** y **Descendente**. El valor predeterminado es **Ascendente**. Cuando el conector compara dos valores de columna de clave para determinar su posición relativa respecto al resto, compara los valores de Java creados a partir de los valores de la columna. Para columnas de clave que contienen valores de caracteres (por ejemplo, para columnas VarChar), la comparación de valores se lleva a cabo de manera sensible a mayúsculas y minúsculas. Cuando el conector compara el valor NULL con un valor distinto de NULL, considera que el valor NULL es menor que el valor distinto de NULL. Para una columna de clave con posibilidad de nulos y el orden de clasificación ascendente, los registros con valores NULL para dicha columna de clave aparecerán antes que los registros con valores distintos de NULL para la misma columna de clave.

Cuando se ejecuta en modalidad de clave, el conector aplica automáticamente en tiempo de ejecución la ordenación de registros en sus enlaces de entrada en función de los nombres de columnas de claves y direcciones de clasificación especificadas en la etapa. Además, si la etapa del conector se configura para que se ejecute en modalidad de clave, en paralelo, en más de un nodo, y si el campo **Tipo de partición** en el separador **Particionamiento** para cualquier enlace de entrada se

establece en el valor predeterminado **(Automático)**, el conector aplica el particionamiento hash de los registros en dicho enlace de entrada y para la clave hash utiliza las columnas de clave especificadas en la etapa. De este modo se garantiza que los registros con los valores de columna de clave coincidentes son procesados por la misma instancia de la etapa (que se ejecuta en el mismo nodo).

El tamaño de lote especificado para la etapa en la propiedad **Tamaño de lote** juega un papel importante en la modalidad de clave. Si la propiedad **Habilitar proceso por lotes** se establece en **No** cuando **Habilitar proceso de clave** está establecida en **Sí**, el conector se ejecuta en modalidad de clave y supone que el tamaño de lote es 0 (ilimitado). Cuando el conector lee los registros de los enlaces de entrada en esta modalidad, además del tamaño de lote, también tiene en cuenta los valores de las columnas de clave presentes en los registros. Para cada ejecución de conjunto de reglas, el conector lee hasta *n* registros del enlace maestro donde *n* representa el tamaño de lote especificado para la etapa. Si el tamaño de lote *n* es diferente de 0, el conector lee hasta *n* registros del enlace, independientemente de los valores de clave en estos registros. Sin embargo, si el tamaño de lote es 0 (ilimitado), el conector lee todos los registros disponibles del enlace maestro que tiene valores de columna de clave coincidentes.

Después de leer los registros del enlace maestro, el conector continúa leyendo registros del resto de enlaces secundarios que tienen valores de columna de clave que coincidan con los valores de la columna de claves de cualquiera de los registros recuperados en el enlace maestro para el mismo ciclo de ejecución de reglas. El número de registros que el conector lee del enlace secundario depende de si el parámetro de conjunto de reglas para dicho enlace se basa en un tipo de matriz Java o no. Si el parámetro de conjunto de reglas se basa en un tipo de matriz, el conector lee todos los registros coincidentes disponibles y los utiliza para preparar el valor para el parámetro de conjunto de reglas. Si el parámetro de conjunto de reglas no se basa en un tipo de matriz, el conector lee únicamente un registro coincidente del enlace. A continuación, el conector invoca el conjunto de reglas.

A continuación, el conector comprueba si, para cualquiera de los parámetros no basados en tipos de matriz, existe otro registro coincidente en el enlace de entrada secundario correspondiente. Si existe tal registro, el conector lo utiliza para el valor del parámetro de conjunto de reglas e invoca el conjunto de reglas de nuevo. Cuando se vuelve a invocar el conjunto de reglas, el conector reutiliza el valor de la ejecución de conjunto de reglas anterior para cada parámetro de conjunto de reglas para el que no ha leído los registros coincidentes nuevos para la ejecución de conjunto de reglas actual. El conector repite este proceso hasta que no puede encontrar ningún registro coincidente nuevo más para ninguno de los parámetros de conjunto de reglas. A continuación, pasa a leer registros nuevos del enlace maestro en preparación de la siguiente ejecución de conjunto de reglas.

Considere un ejemplo de un trabajo en que la etapa ILOG JRules tiene dos enlaces de entrada y cada enlace tiene dos columnas. En Enlace 1, la primera columna es de tipo Integer y la segunda columna es de tipo VarChar(5). En Enlace 2, la primera columna es de tipo Integer y la segunda columna es de tipo Double. Digamos que los datos recibidos en los enlaces de entrada son como se indica a continuación (los datos a la derecha llegan en primer lugar):

Enlace 1

(16, Sam) (16, Sam) (16, Sam) (12, John) (12, John) (10, Rick) (10, Rick) (10, Rick)

Enlace 2

(16, 32.4) (12, 32.22) (12, 24.22) (12, 53.33) (12, 23.233) (10, 32.55) (10, 25.77)

Digamos que se habilita la modalidad de clave y que el tamaño de lote es 0. La primera columna de tipo Integer es la columna de clave y el orden de clasificación para la clave es **Ascendente**. Digamos que el conjunto de parámetros asociados con los enlaces de entrada es de tipos de matriz Java. Para estos datos hay tres ciclos de ejecución de regla. Los datos enviados en cada ciclo de ejecución de reglas son los siguientes:

Ciclo de ejecución de reglas 1

Enlace 1: (10, Rick) (10, Rick) (10, Rick)

Enlace 2: (10, 32.55) (10, 25.77)

Ciclo de ejecución de reglas 2

Enlace 1: (12, John) (12, John)

Enlace 2: (12, 32.22) (12, 24.22) (12, 53.33) (12, 23.233)

Ciclo de ejecución de reglas 3

Enlace 1: (16,Sam) (16,Sam) (16,Sam)

Enlace 2: (16, 32.4)

Si para algunos de los enlaces secundarios el conector detecta un registro con los valores de la columna de clave que no coinciden con los valores de la columna de clave en cualquiera de los registros maestros, y si no se recuperaron los registros con los valores de columna de clave coincidentes previamente en el mismo enlace para la misma ejecución de conjunto de reglas, el conector especifica una matriz vacía para el valor del parámetro de reglas para dicho enlace. La matriz vacía se especifica sólo si ese parámetro de conjunto de reglas es de un tipo de matriz Java y sólo si se basa en el orden de clasificación especificado para las claves de la columna, el conector determina que existe una posibilidad de que los valores de la columna de clave del registro detectado coincidan con los valores de la columna de clave en un registro que el conector todavía debe recuperar en el enlace maestro.

De lo contrario, el conector intenta rechazar los registros que ha causado el error de falta de coincidencia de claves. El conector compara los valores de columna de clave del registro detectado en el enlace secundario actual y los valores de la columna de clave de los registros que ya se han recuperado en el enlace maestro para la ejecución de conjunto de reglas actual. Al realizar esta comparación el conector tiene en cuenta la dirección de clasificación que se ha especificado para las columnas de clave. Si es posible que el siguiente registro en el enlace secundario actual tenga valores de la columna de clave que coinciden con los valores de la columna de clave de los registros que ya se han recuperado en el enlace maestro, el conector rechaza el registro que lee en el enlace secundario actual. De lo contrario, rechaza los registros que ha recuperado anteriormente en el resto de los enlaces de entrada, incluidos los registros del enlace de entrada maestro.

Considere un ejemplo de un trabajo donde la etapa ILOG JRules tiene tres enlaces de entrada. Digamos que la columna de clave en los tres enlaces es C1 de tipo Integer. El valor de la columna C1 en los tres enlaces es el siguiente (los datos a la derecha llegan en primer lugar):

Enlace 1 (Maestro): 5 5 5

Enlace 2: 5 5

Enlace 3: 6

Digamos que el tamaño de lote se establece en 0.

El comportamiento del conector bajo diversas modalidades operativas es el siguiente:

1. Los parámetros de conjunto de reglas para todos los enlaces de entrada son de tipo matriz y el orden de clasificación para la columna C1 es **Ascendente**. El conector invoca el conjunto de reglas y pasa tres registros con el valor de C1 de 5 para el primer parámetro de conjunto de reglas, dos registros con el valor de C1 de 5 para el segundo parámetro de conjunto de reglas y 0 registros para el tercer parámetro de conjunto de reglas. El conector determina según el orden de clasificación de la columna C1 que todavía es posible que el registro siguiente del enlace maestro tenga el valor de C1 de 6 y que presenta una coincidencia para el registro de enlace secundario actual y, por lo tanto, el registro en el enlace 3 no se ha rechazado.
2. El parámetro de conjunto de reglas para el enlace 3 no es de tipo matriz Java y el orden de clasificación es **Ascendente**. En este caso, el conector rechaza los registros con el valor de C1 de 5 que ya ha recuperado en el primer y segundo enlaces de entrada, porque los registros que potencialmente llegan a continuación en el tercer enlace de entrada después del registro con el valor de C1 de 6 sólo puede tener el valor de C1 de 6 o superior, por lo que ninguno de ellos puede coincidir con el valor de C1 de 5.
3. El parámetro de conjunto de reglas para el enlace 3 no es de tipo de matriz Java y el orden de clasificación es **Descendente**. En este caso, el conector rechaza el registro con el valor de C1 6 que ha recuperado en el tercer enlace de entrada, porque hay una posibilidad de que el siguiente registro en este enlace tenga el valor de C1 de 5 y, por lo tanto, coincida con los registros recuperados en los dos primeros enlaces de entrada.

Una vez que el conector rechaza los registros debido a la no coincidencia de clave, sigue tratando de preparar los valores de parámetro de conjunto de reglas para la ejecución de conjunto de reglas actual, leyendo registros del primer enlace de entrada a partir del cual se rechazaron los registros. Es el enlace maestro, en cuyo caso el conector empieza a preparar los valores de parámetro de conjunto de reglas desde cero, o el enlace secundario actual, en cuyo caso el conector sigue leyendo registros en dicho enlace para intentar hacerlos coincidir con los registros que ya ha recibido en el enlace maestro para la ejecución de conjunto de reglas actual.

Si un registro debe ser rechazado debido a una no coincidencia de clave, el conector puede hacerlo sólo si el registro que se rechaza proviene de un enlace de entrada para el que se ha definido el enlace de rechazo y sólo si la opción **Error de coincidencia de clave** está seleccionada en dicho enlace de rechazo. De lo contrario, el conector registra un error y el trabajo falla.

En la modalidad de proceso por lotes, el conector aplica el valor de tamaño de lote de 1 si el parámetro de conjunto de reglas para cualquiera de los enlaces de entrada no se basa en un tipo de matriz Java. En la modalidad de clave, el conector aplica el valor de tamaño de lote de 1 si el parámetro de conjunto de reglas para el enlace maestro no se basa en un tipo de matriz Java. Si en el tiempo de ejecución el conector debe sustituir el valor de **Tamaño de lote** especificado por el usuario y forzar el valor de 1, escribe un mensaje informativo en el registro de trabajo sobre este cambio.

Si los registros llegan a los enlaces de entrada de la etapa en oleadas de transacciones, el conector procesa los registros de cada oleada independientemente de los registros de otra oleada. Los registros de una oleada no se combinan con los registros de otra oleada para generar valores de parámetros de conjunto de reglas para una sola ejecución de conjunto de reglas.

Tipos de modelo de objeto de ejecución

El conector de ILOG JRules puede utilizar el Modelo de objetos de ejecución dinámico (XOM dinámico) o el Modelo de objetos de ejecución de Java (XOM Java). El conector no soporta parámetros de conjunto de reglas con una combinación de los tipos de XOM dinámico y XOM Java. Todos los parámetros del conjunto de reglas deben estar basados en el mismo tipo de XOM. Para asegurarse de esto, la propiedad **Tipo de XOM** está asociada con la etapa y no el enlace.

XOM Java

Para configurar la etapa para la modalidad de XOM Java, la propiedad **Tipo de XOM** de la etapa debe establecerse en **XOM Java**. Si el asistente de configuración se utiliza para importar parámetros de conjunto de reglas y configurar la etapa para un conjunto de reglas basado en XOM Java, el asistente establece automáticamente la propiedad **Tipo de XOM** en la etapa **XOM Java**.

Cuando la opción **XOM Java** está seleccionada, debe especificarse el valor de la propiedad de enlace **Clase de parámetro de conjunto de reglas**. El valor especificado representa el tipo de Java del parámetro de conjunto de reglas asociado a dicho enlace. Si el tipo especificado es uno de los tipos de Java básicos soportados por el conector, una sola columna en el enlace corresponde al parámetro de conjunto de reglas entero. De lo contrario, las columnas de los enlaces de entrada corresponden a los argumentos de los métodos de la clase Java de parámetro de conjunto de reglas y las columnas de los enlaces de salida corresponden a los valores de retorno de los métodos de la clase Java de parámetro de conjunto de reglas. Para obtener información sobre los tipos de Java básicos en el contexto del conector, consulte el tema *Tipos y métodos de Java básicos*.

XOM dinámico

El conector da soporte a invocar los conjuntos de reglas que se basan en XOM dinámico. En esta modalidad, los valores de parámetros de conjunto de reglas son documentos XML. Los documentos XML deben ser válidos respecto a los documentos de esquema XML que se han utilizado para definir XOM para el conjunto de reglas.

Para configurar la etapa para la modalidad de XOM dinámico, la propiedad **Tipo de XOM** de la etapa debe establecerse en **XOM dinámico**. Si el asistente de configuración se utiliza para importar parámetros de conjunto de reglas y

configurar la etapa para un conjunto de reglas basado en XOM dinámico, el asistente establece automáticamente la propiedad **Tipo de XOM** en la etapa en **XOM dinámico**.

Cuando la etapa se configura para XOM dinámico, el conector lee un registro de cada enlace de entrada, genera documentos XML a partir de esos registros, e invoca el conjunto de reglas y pasa los documentos XML generados como los valores para parámetros IN y IN_OUT del conjunto de reglas.

Cuando el conector lee un registro de un enlace de entrada, busca el campo en el registro que contiene los datos XML a partir de los cuales se va a crear el valor del documento XML para el parámetro de conjunto de reglas. El documento XML completo debe estar almacenado en un solo campo del registro. Los campos restantes del registro se pueden utilizar como valores de columnas de clave para correlacionar registros entre varios enlaces de entrada, o pueden ser campos que el conector propaga desde enlaces de entrada a enlaces de salida.

La columna del enlace de entrada que se debe utilizar para los documentos XML para los parámetros de conjunto de reglas debe identificarse de modo que el conector tenga acceso al campo en cada registro recuperado en ese enlace que contiene los datos XML reales que se utilizarán como valor de parámetro de conjunto de reglas. La columna identificada debe ser de un tipo de datos de DataStage basado en caracteres: Char, VarChar, LongVarChar, NChar, NVarChar y LongNVarChar. Si hay varias columnas de este tipo, sólo uno de ellos debe tener el valor de atributo **Descripción** que incluye la expresión `CC_JRules();`. El atributo **Descripción** está presente en el separador de columnas que es visible después de pulsar cualquiera de los enlaces de entrada o salida en la etapa. Si, después de comprobar estas condiciones, hay cero o más de una columna elegibles en el enlace, el trabajo falla con un error.

La modalidad de proceso por lotes no está soportada cuando el conector está configurado para invocar un conjunto de reglas basado en XOM dinámico. El tamaño de lote de 1 se utiliza implícitamente en este caso. El conector no da soporte a pasar varios documentos XML en un solo valor de parámetro de conjunto de reglas.

La modalidad de clave está soportada por el conector configurado para invocar el conjunto de reglas basado en XOM dinámico. En este caso, las columnas de clave especificadas se pueden utilizar para clasificar y particionar registros de entrada entre varios enlaces de entrada y varios nodos de proceso de la etapa, para garantizar que el conector está siempre invocando el conjunto de reglas con los valores del documento XML correlacionados en todos los parámetros IN e IN_OUT del conjunto de reglas.

Campos de DataStage y correlación de parámetros de conjunto de reglas

ILOG JRules Connector convierte los registros disponibles en el enlace de entrada en objetos Java. El conector debe comprender la correlación entre los campos de la etapa de conector y los parámetros de conjunto de reglas, con el fin de completar la conversión.

La correlación también se puede identificar automáticamente utilizando el asistente de configuración. En los apartados siguientes se describe cómo los campos de InfoSphere DataStage se correlacionan con los parámetros de conjunto de reglas.

XOM Java

Para XOM Java, la correlación entre los campos en un enlace y el parámetro de conjunto de reglas asociado a dicho enlace se determina en función de los atributos del campo **Nombre de columna** y **Tipo de SQL** y uno de los siguientes:

- Los métodos presentes en la clase de parámetro de conjunto de reglas
- El nombre del parámetro de conjunto de reglas y tipo de datos (si es del tipo de Java básico).

Para obtener más información sobre los tipos de Java básicos, consulte el tema *Tipos y métodos de Java básicos*. Si el parámetro de conjunto de reglas conforme a las condiciones especificadas en la sección *Correlación automática*, la correlación se realiza automáticamente basándose en el atributo **Nombre de columna** del campo. De lo contrario, la correlación se realiza utilizando una especificación de correlación proporcionada en el atributo **Descripción** del campo.

Correlación automática

ILOG JRules Connector puede identificar automáticamente la correlación entre los campos de enlace y el parámetro de conjunto de reglas si se cumple alguna de las condiciones siguientes:

1. Si el parámetro de conjunto de reglas de uno de los tipos de Java básicos soportados por el conector, el **Nombre de columna** debe corresponderse con el nombre del parámetro de conjunto de reglas. El **Tipo de SQL** del campo debe ser compatible con el tipo Java básico del parámetro de conjunto de reglas. Para obtener más información sobre la correlación o compatibilidad de tipos de datos, consulte los temas *Correlación de tipo de DataStage a tipo de Java* y *Correlación de tipo de Java a tipo de DataStage.* En este caso, el campo está directamente correlacionado con el parámetro de conjunto de reglas.
2. Si el parámetro de conjunto de reglas no es de un tipo de Java básico y el enlace es un enlace de entrada, la clase de parámetro de conjunto de reglas debe tener un método denominado **`setFieldName()`**, donde **`FieldName`** es el **Nombre de columna** del campo. La comparación del **Nombre de columna** y el nombre del método no es sensible a mayúsculas y minúsculas. El método no debe tener ningún tipo de retorno y debe tener un solo argumento y el **Tipo de SQL** del campo en el enlace de entrada debe ser compatible con el tipo de datos del argumento que espera el método. Por ejemplo, el campo **`firstName`** de tipo **`VarChar(10)`** en un enlace de entrada se correlaciona con el primer y único argumento del método **`public void setFirstName(String name)`**. En este caso, el campo de entrada está correlacionado con el argumento que espera el método.
3. Si el parámetro de conjunto de reglas no es de un tipo de Java básico y el enlace es un enlace de salida, entonces la clase de parámetro de conjunto de reglas debe tener un método denominado **`getFieldName()`**, donde **`FieldName`** es el **Nombre de columna** del campo. La comparación del **Nombre de columna** y el nombre del método no es sensible a mayúsculas y minúsculas. Un caso especial es cuando el método es del tipo de retorno **`boolean`** o **`java.lang.Boolean`**, en ese caso se espera que el método se denomine **`isFieldName()`**. El valor de retorno del método debe ser compatible con el **Tipo de SQL** del campo presente en el enlace de salida. En este caso, el campo se correlaciona con el valor de retorno del método.

Si hay un campo en el enlace de entrada para el que no hay ningún método en la clase de parámetro de conjunto de reglas conforme a la segunda condición (y el atributo **Descripción** está vacío), los valores de campo no se pasan a ILOG JRules.

De forma similar, si hay un campo en el enlace de salida para el que no hay ningún método en la clase de parámetro de conjunto de reglas conforme a la tercera condición (y el atributo **Descripción** está vacío), el valor del campo no se establece por ILOG JRules Connector.

Correlación utilizando la especificación de Descripción

Si el parámetro de conjunto de reglas no sigue las condiciones de correlación automática, ILOG JRules Connector proporciona un mecanismo para especificar el método que se va a invocar para establecer u obtener cada valor de campo en los registros de entrada y salida de InfoSphere DataStage. Esto se especifica en el atributo **Descripción** de los campos del esquema en el separador **Columnas** del editor de etapas. Para cada campo en el esquema del enlace, la **Descripción** contiene información sobre el método que se va a invocar en la clase Java asociada con el enlace, para establecer, o para obtener ese valor de campo. Si no hay información presente en el atributo **Descripción** para ninguno de los campos, se supone que el parámetro de conjunto de reglas se ajusta a la especificación mencionada en la sección *Correlación automática*.

Para las columnas del enlace de entrada, el valor de su atributo **Descripción** consta de una expresión concatenada más, donde cada expresión está en el formato `CC_JRules(methodName, parameterIndex);`. `methodName` especifica el nombre del método que se va a invocar en el objeto de parámetro de conjunto de reglas para esta columna. El valor de `parameterIndex` es un entero mayor o igual que 1 que especifica el índice del parámetro (para el método) con el que se correlaciona la columna.

Por ejemplo, un objeto Java tiene un método `setFullName(String firstName, String lastName)` y el registro de DataStage contiene dos campos, *firstName* y *lastName*. En este caso, el atributo **Descripción** para *firstName* es `CC_JRules(setFullName,1);` y para *lastName* es `CC_JRules(setFullName,2);`. El valor del campo *firstName* de un registro de DataStage se utiliza para el primer argumento de método `setFullName` y el valor del campo *lastName* del mismo registro se utiliza para el segundo argumento.

methodName también puede ser un constructor. Si la columna *firstName* en el ejemplo se utiliza también para el primer argumento del constructor para la clase `Cliente`, el atributo **Descripción** para *firstName* es `CC_JRules(setFullName,1);CC_JRules(Customer,1);`.

En el caso de los enlaces de salida, se llama al método para obtener el valor del campo del objeto Java. Por lo tanto, el campo no espera ningún parámetro. Como resultado, para los campos de los enlaces de salida, el índice del parámetro no es necesario. La sintaxis del atributo **Descripción** para los campos en el enlace de salida es: `CC_JRules(methodName);`. Para cada campo, debe haber sólo un método al que se pueda llamar. Si hay más de un método, ILOG JRules Connector registra un error y el trabajo falla.

Cuando una sola columna en el enlace representa el valor del parámetro del conjunto de reglas entero, el atributo **Descripción** para la columna tiene el valor `CC_JRules();`. Esto ocurre así cuando el parámetro de conjunto de reglas es uno de los tipos de Java básicos soportados por el conector.

XOM dinámico

Si un campo de registro de InfoSphere DataStage contiene un documento XML que se debe enviar al conjunto de reglas basado en XOM dinámico, el atributo **Descripción** para el campo puede contener el valor de `CC_JRules();`. Cuando hay varios campos en el registro de InfoSphere DataStage, el valor es necesario para especificar cuál de ellos contiene el documento XML.

Si hay un único campo en el registro, se supone automáticamente que ese campo contiene el documento XML y el atributo **Descripción** del campo puede dejarse vacío. El campo que contiene el documento XML debe ser de cualquiera de los tipos de datos basados en caracteres de InfoSphere DataStage, tales como NChar, VarChar, NVarChar, LongVarChar o LongNVarChar.

Estrategias de propagación de campo

ILOG JRules Connector da soporte al paso a través de campos no coincidentes de los enlaces de entrada a los enlaces de salida. Los campos no coincidentes son campos de enlace de entrada que no son utilizados por el conector para invocar el conjunto de reglas. Cuando se habilita la propagación de campos no coincidentes, los campos no coincidentes se propagan a los enlaces de salida asociados con los enlaces de entrada.

El conector establece automáticamente la asociación entre cada enlace de entrada que tiene los campos no coincidentes y los enlaces de salida a los que esos campos deben propagarse. Para cada campo no coincidente en un enlace de entrada y para cada enlace de salida asociado con dicho enlace de entrada, el conector comprueba si el enlace de salida contiene el campo con el mismo nombre que el campo no coincidente.

- Si lo hace y si el campo de salida no está correlacionado con ningún campo en XOM, los valores de los campos de la columna de entrada se propagan y establecen como valores de campo de la columna de salida.
- Si no es así, el conector comprueba si el recuadro de selección **Propagación de columnas de tiempo de ejecución** está seleccionado para dicho enlace de salida. Si se selecciona, el conector añade la definición de campo de entrada a ese esquema de enlace de salida y propaga los valores de campo mientras se ejecuta el trabajo. Si el recuadro de selección no está seleccionado, la propagación de la definición de campo de entrada y los valores no tienen lugar.

Puede utilizar una de las estrategias siguientes para propagar los campos no coincidentes:

Coincidencia basada en el orden de los registros

Cuando se selecciona esta estrategia, el conector propaga los campos no coincidentes del primer enlace de entrada al primer enlace de salida, del segundo enlace de entrada al segundo enlace de salida, etc. Si hay más enlaces de entrada que enlaces de salida, la propagación de campos no se realiza para los enlaces de entrada para los cuales no hay enlaces de salida correspondientes. Un enlace de entrada se puede asociar con un solo enlace de salida. De forma similar, un enlace de salida sólo puede asociarse con un enlace de entrada.

Cuando se utiliza esta estrategia, el número de registros en el enlace de entrada que se han utilizado para invocar el conjunto de reglas debe coincidir con el número de registros producidos en el enlace de salida asociado después de ejecutar

el conjunto de reglas. Si los números no coinciden, el conector registra un error y el trabajo falla.

Coincidencia basada en el valor de la columna de clave de propagación

Cuando se utiliza esta estrategia, el conector propaga los campos no coincidentes de un enlace de entrada a un enlace de salida con la misma columna de clave de propagación. La columna de clave de propagación se especifica para cada enlace de entrada por separado, a través de la propiedad de enlace **Columna de clave de propagación**.

El conector comprueba cada enlace de entrada para verificar si hay una columna marcada como columna de clave de propagación. Si se encuentran enlaces de entrada con la columna marcada como columna de clave de propagación y si los enlaces tienen campos que no se utilizan para la ejecución del conjunto de reglas, los campos son elegibles para la propagación de campos.

Una columna de clave de propagación no puede ser una columna que sea elegible para la propagación de campos al mismo tiempo. Si se produce este escenario, el trabajo falla con un error. El conector inspecciona los enlaces de entrada y, para cada enlace de entrada que tiene la columna de clave de propagación, así como las columnas elegibles para la propagación, el conector realiza los pasos siguientes:

- Inspecciona todos los enlaces de salida que no están ya asociados con alguno de los enlaces de entrada anteriores.
- El enlace de salida que incluye la columna con el mismo nombre y el tipo de datos que la columna de clave de propagación del enlace de entrada se asocia con ese enlace de entrada.

Con esta estrategia, un enlace de entrada se puede asociar con varios enlaces de salida, pero un enlace de salida sólo se puede asociar con un enlace de entrada.

Cuando el conector lee los registros en un enlace de entrada con la columna de clave de propagación habilitada, comprueba el valor de la columna de clave de propagación del registro. Si el valor coincide con el valor de otro registro recuperado en el mismo enlace para la ejecución del conjunto de reglas actual, el trabajo falla con un error.

Para cada registro que se envía a un enlace de salida, el campo de registro se rellena desde el valor del parámetro OUT o IN_OUT del conjunto de reglas. A continuación, el conector comprueba si el enlace de salida se asocia con un enlace de entrada para la propagación de campo. Si está asociado, el conector busca el registro de entrada que se recupera del enlace de entrada para la ejecución del conjunto de reglas actual y que tiene el mismo valor de la columna de clave de propagación que el registro de salida. Si un registro así existe, el conector copia los valores de los campos propagados desde el registro de entrada al registro de salida. Si tal registro de entrada no existe, los campos propagados en el registro de salida permanecen sin establecer.

Considere un ejemplo de un trabajo donde la etapa ILOG JRules tiene dos enlaces de entrada y dos enlaces de salida.

Los enlaces contienen las columnas de la siguiente manera:

Enlaces de entrada

Enlace 1: (id, balance)

Enlace 2: (id, name, address)

Enlaces de salida

Enlace 1: (name, cust_type, address)

Enlace 2: (name, outstanding, address)

Si la propagación de campo se habilita utilizando la estrategia de la columna de clave de propagación y la columna **name** se especifica como columna de clave propagación para el enlace 2, el conector detecta que la columna **name** también está definida en los dos enlaces de salida. Si la columna **address** no se está correlacionando con el conjunto de reglas, tanto en el enlace de entrada como el de salida, los datos de la columna **address** del enlace de entrada 2 se envía a la columna **address** de ambos enlaces de salida. La correlación se realiza basándose en el campo **name**. Por ejemplo, si el enlace de entrada 2 tiene los datos siguientes:

1, "John", "123, Las Vegas Dr, Las Vegas, NV"

2, "Paul", "abc Drive, New York"

Los registros en los enlaces de salida se muestran como se indica a continuación:

Enlace 1

"John", "Gold", "123, Las Vegas Dr, Las Vegas, NV"

"Paul", "Silver", "abc Drive, New York"

Enlace 2

"John", "244.33", "123, Las Vegas Dr, Las Vegas, NV"

"Paul", "435.6445", "abc Drive, New York"

"Manish", 4363.453", ""

Los datos de la columna de entrada **address** se correlaciona con la columna de salida **address** utilizando el valor de la columna **name**. El valor del campo **name** en el tercer registro en el enlace de salida 2 no coincide con el valor del campo **name** en cualquiera de los registros en el enlace de entrada 2, por lo que el valor del campo **address** en el registro de salida permanece sin establecer.

Coincidencia según los identificadores de objeto Java

Cuando se utiliza esta estrategia, el conector propaga campos no coincidentes de un registro de entrada al registro de salida que corresponde al mismo objeto Java. Para cada enlace de entrada, el conector comprueba el tipo de Java del parámetro de conjunto de reglas especificado para el enlace. Si este tipo no es uno de los tipos de Java básicos soportados por el conector, éste toma nota de las columnas en el enlace que son elegibles para la propagación de campo, lo que significa que no se utilizan para producir los valores de parámetros de conjunto de reglas.

Para cada enlace de entrada que tiene algunas columnas (campos) elegibles para la propagación, el conector establece internamente la asociación entre el enlace de entrada y los enlaces de salida del modo siguiente:

- Inspecciona todos los enlaces de salida que no están ya asociados con alguno de los enlaces de entrada anteriores.
- Los enlaces de salida cuyos parámetros de conjunto de reglas son de los tipos de Java compatible con el especificado para el enlace de entrada se asocian con el enlace de entrada. Dos tipos de Java son compatibles en este contexto si son idénticos o si uno de ellos se deriva del otro. Los tipos de matriz Java se tratan como tipos no de matriz al comprobar la compatibilidad. Por ejemplo, si un tipo de Java es ClassA y el otro es de tipo ClassB[], los dos tipos de Java son compatibles si ClassA y ClassB son idénticos o bien si uno se deriva del otro.

Con esta estrategia de propagación de campo vigente, un enlace de entrada se puede asociar con varios enlaces de salida, pero un enlace de salida sólo se puede asociar con un enlace de entrada.

Para cada registro que el conector envía a un enlace de salida, primero llena los campos de registro del valor del parámetro OUT o IN_OUT correspondiente del conjunto de reglas y, a continuación, comprueba si el enlace de salida se asocia con un enlace de entrada para los efectos de propagación de campo. Si está asociado, el conector busca el registro recuperado del enlace de entrada para la ejecución de conjunto de reglas actual y que corresponde al mismo objeto Java que el registro de salida. Si existe un registro así, el conector copia los valores de los campos propagados desde el registro de entrada al registro de salida. Si tal registro no existe, los campos propagados en el registro de salida permanecen sin establecer.

Configuración de vía de acceso de clases Java

El conector ILOG JRules se ejecuta en el host de nivel de motor de IBM InfoSphere Information Server. ILOG JRules Connector es un conector basado en Java y requiere acceso a diversos recursos de ILOG JRules tales como archivos JAR de ILOG JRules, clases XOM Execution Object Model de Java clases y los archivos de configuración para funcionar correctamente.

Debe especificar los recursos que necesita el conector en la vía de acceso de clases Java para el conector. Puede especificar los recursos necesarios en la propiedad **Vía de acceso de clases** en la página de la etapa de conector ILOG JRules.

Puede definir una variable de entorno de proyecto de InfoSphere DataStage para listar todos los recursos necesarios y, a continuación, añadirla al trabajo como parámetro de trabajo especificado como el valor de la propiedad **Vía de acceso de clases**. Cuando los recursos se especifican en la propiedad **Vía de acceso de clases**, están disponibles para el conector durante la ejecución del trabajo, así como durante la sesión del asistente de configuración.

No se recomienda el uso de la vía de acceso de clases Java del sistema para listar los recursos necesarios. Si se especifica en la vía de acceso de clases del sistema, los recursos pueden interferir con otros procesos Java que se ejecutan en la misma máquina. Tampoco es conveniente añadir recursos a la vía de acceso de clases del sistema de forma incremental porque requiere reiniciar el motor de IBM InfoSphere DataStage para reconocer las nuevas entradas que se añaden a la vía de acceso de clases del sistema. Si el motor es un sistema Windows, entonces se requiere un rearranque. Un ejemplo en que la vía de acceso de clases del sistema tiene que

actualizarse periódicamente es añadir archivos Java XOM JAR que el conector puede utilizar para invocar nuevos conjuntos de reglas.

Si no es conveniente añadir los recursos necesarios en la propiedad **Vía de acceso de clases** de cada etapa ILOG JRules, puede realizar una de las acciones siguientes en lugar de establecer el valor de vía de acceso de clases del sistema:

- Para la ejecución en tiempo de ejecución del trabajo, defina la variable de entorno CLASSPATH en el nivel de proyecto de IBM InfoSphere DataStage y obtenga una lista de los recursos necesarios en el valor predeterminado para esta variable de entorno. El valor de CLASSPATH especificado es posteriormente recogido por cada trabajo en dicho proyecto de IBM InfoSphere DataStage. Para sustituir el valor predeterminado especificado en el nivel de proyecto, los trabajos pueden importar esta variable de entorno como un parámetro de trabajo y establecer su valor que el valor de vía de acceso de clases de conector utilizará para ese trabajo concreto.
- Para el asistente de configuración, copie los recursos necesarios en *IS_HOME*/ASBNode/lib/java, donde *IS_HOME* representa el directorio de inicio de la instalación de IBM InfoSphere Information Server en el host de nivel de motor de IBM InfoSphere Information Server.

  Nota: No puede añadir recursos a la vía de acceso de clases del sistema para las sesiones del asistente de configuración porque estas sesiones se ejecutan bajo el control de servicio del Agente ASB. El servicio del Agente ASB se inicializa con la vía de acceso de clases personalizada interna que no incluye los recursos especificados en el valor de vía de acceso de clases del sistema.

Según la modalidad de motor seleccionada, debe tener los archivos JAR JRules especificados en la vía de acceso de clases durante la ejecución del trabajo, así como para su uso con el asistente de configuración.

El asistente de configuración se ejecuta bajo el proceso de servicio del Agente ASB en el host del motor de IBM InfoSphere Information Server. Cuando el asistente carga las clases Java XOM desde el archivo JAR Java XOM especificado en la propiedad **Vía de acceso de clases**, dicho archivo JAR queda bloqueado durante la sesión del asistente. El asistente carga las clases XOM en cada sesión de forma independiente, por lo tanto, si es necesario, generalmente es posible sobrescribir el archivo JAR XOM entre las sesiones del asistente, por ejemplo para poner una versión más reciente del archivo JAR en lugar del anterior. Sin embargo, en algunas plataformas el archivo JAR puede permanecer bloqueado incluso después de que el asistente finalice la sesión, porque el proceso del Agente ASB bajo el cual se han cargado las clases aún está en ejecución. En estos casos, incluso si no es posible suprimir los archivos JAR bloqueados o mover otro JAR para sustituir el JAR bloqueado, quizás aún sea posible copiar otro JAR sobre el JAR bloqueado. Si esto tampoco funciona, otra opción es colocar el nuevo JAR en una nueva ubicación y especificar dicha ubicación en lugar de la anterior en la propiedad **Vía de acceso de clases**. Como última opción, el proceso del Agente ASB se puede reiniciar.

Modalidad de motor de núcleo

Debe tener el archivo JAR *JRULES_HOME*/executionserver/lib/jrules-engine.jar en la vía de acceso de clases durante la expresión de la ejecución del trabajo, donde *JRULES_HOME* representa el directorio inicial del producto ILOG JRules en el host del motor de IBM InfoSphere Information Server.

Este archivo JAR no es necesario para las sesiones del asistente de configuración.

Cuando el conjunto de reglas para el que el conector está configurado está basado en Java XOM, las clases XOM de los parámetros de conjunto de reglas también deben incluirse en la vía de acceso de clases. Esto se aplica a la ejecución del trabajo, así como las sesiones del asistente de configuración.

Cuando el conector está configurado para la modalidad de motor de núcleo, no requiere acceso a los archivos de configuración.

Modalidad XU RES J2SE

Los siguientes archivos JAR JRules tienen que estar presentes en la vía de acceso de clases durante la ejecución del trabajo, así como para utilizar el asistente de configuración:

JRULES_HOME/executionserver/lib/jrules-engine.jar

JRULES_HOME/executionserver/lib/jrules-res-session-java.jar

JRULES_HOME/executionserver/lib/jrules-res-execution.jar

JRULES_HOME/executionserver/lib/sam.jar

JRULES_HOME/executionserver/lib/log4j-1.2.8.jar

JRULES_HOME/executionserver/lib/j2ee_connector-1_5-fr.jar

Donde *JRULES_HOME* representa el directorio de inicio del producto ILOG JRules en el host del motor de IBM InfoSphere Information Server.

Cuando el conjunto de reglas para el que el conector está configurado está basado en Java XOM, las clases XOM de los parámetros de conjunto de reglas también deben incluirse en la vía de acceso de clases. Esto se aplica a la ejecución del trabajo, así como las sesiones del asistente de configuración.

Si el repositorio de conjunto de reglas está basado en JDBC, la implementación del controlador JDBC necesaria para que el motor de JRules acceda al repositorio debe estar presente en la vía de acceso de clases. Por ejemplo, si el repositorio de conjunto de reglas se implementa como una base de datos DB2, el controlador JDBC `db2jcc4.jar` disponible con la instalación del producto DB2 puede seleccionarse para acceder al repositorio.

El tipo de repositorio de conjunto de reglas y la información de conexión de repositorio se almacenan en el archivo de configuración `ra.xml` de JRules. El directorio con este archivo debe estar listado en la vía de acceso de clases para el conector. La instalación de JRules viene con el archivo `ra.xml` de ejemplo en el directorio *JRULES_HOME*/executionserver/bin, que se puede personalizar para acceder a un repositorio de conjunto de reglas en particular. Para obtener más información sobre el archivo de configuración `ra.xml`, consulte la documentación del producto ILOG JRules.

Modalidad XU RES J2EE

La conexión con el repositorio de conjunto de reglas se configura cuando Rule Execution Server se despliega en el servidor J2EE.

Durante la ejecución del trabajo, el conector actúa como cliente EJB3 remoto y establece una conexión con el componente de sesión EJB3 que debe desplegarse en el servidor J2EE en el que se despliega Rule Execution Server. Esta modalidad de

motor necesita que los complementos Java EE de ILOG JRules estén instalados y configurados correctamente en el entorno de JRules. Los siguientes archivos JAR JRules deben estar presentes en la vía de acceso de clases para la ejecución del trabajo, donde JRULES_HOME representa el directorio inicial del producto de ILOG JRules en el host del motor de IBM InfoSphere Information Server.

JRULES_HOME/executionserver/lib/jrules-engine.jar

JRULES_HOME/executionserver/lib/jrules-res-session-java.jar

JRULES_HOME/executionserver/lib/jrules-res-execution.jar

JRULES_HOME/executionserver/lib/sam.jar

JRULES_HOME/executionserver/lib/log4j-1.2.8.jar

JRULES_HOME/executionserver/lib/j2ee_connector-1_5-fr.jar

Además de los archivos JAR de JRules, debe tener los archivos JAR de cliente J2EE necesarios para establecer una conexión con el componente EJB3 en el servidor J2EE.

Cuando el motor de JRules se ejecuta en WebSphere Application Server v7 y el conector se conecta al mismo mediante el WebSphere Application Client v7 instalado en la máquina del nivel de motor, debe añadir los siguientes archivos JAR a la vía de acceso de clases:

- *WAS_HOME*/runtimes/com.ibm.ws.ejb.thinclient_7.0.0.jar
- *WAS_HOME*/runtimes/com.ibm.ws.webservices.thinclient_7.0.0.jar

Donde *WAS_HOME* es el directorio de inicio de WebSphere Application Client 7 en la máquina del nivel de motor de IBM InfoSphere Information Server.

Nota: Debe seleccionar la opción **Clientes ligeros autónomos y adaptadores de recursos** durante la instalación de WebSphere Application Client para que estos dos archivos JAR se incluyan en la instalación.

Los archivos JAR de apéndice de EJB3 deben crearse e incluirse en la lista de la vía de acceso de clases. Este archivo JAR de apéndice se genera ejecutando el script `createEJBStubs` ubicado en el directorio `INICIO_WAS/bin`, donde `WAS_HOME` es el directorio inicial de la instalación de WebSphere Application Server v7 en el que se despliega Rule Execution Server. La extensión de archivo para este script varía entre sistemas operativos. Por ejemplo, en Windows, es `.bat` y en Linux es `.sh`. El script debe ejecutarse una vez en el archivo `jrules-res-session-ejb3-WAS7.jar` que se encuentra bajo el directorio *JRULES_HOME*`/executionserver/applicationservers/WebSphere7`. La opción `-newfile` debe especificar la ubicación del archivo de apéndice generado. Por ejemplo:

```
WAS_HOME/bin/createEJBStubs.sh JRULES_HOME/executionserver/
applicationservers/WebSphere7/jrules-res-session-ejb3-WAS7.jar -newfile
JRULES_HOME/executionserver/lib/jrules-res-session-ejb3-WAS7_stub.jar
```

El archivo de apéndice EJB3 generado debe copiarse en el host del motor de IBM InfoSphere Information Server e incluirse en la vía de acceso de clases para la ejecución del trabajo.

Cuando el conjunto de reglas para el que el conector está configurado está basado en Java XOM, las clases XOM de los parámetros de conjunto de reglas también deben incluirse en la vía de acceso de clases. Esto se aplica a la ejecución del trabajo, así como las sesiones del asistente de configuración. En esta modalidad de motor, las clases Java XOM deben implementar la interfaz `java.io.Serializable`.

De lo contrario, el trabajo fallará con excepciones de ordenación CORBA ya que los valores de parámetro de conjunto de reglas no son transportados adecuadamente a través de la conexión EJB.

El conector también requiere acceso al archivo de configuración `jndi.properties`. Este archivo contiene las propiedades necesarias para crear el contexto inicial (objeto `java.naming.InitialContext`) para establecer la conexión con el componente sesión de reglas EJB3 en el servidor J2EE. Un archivo de ejemplo `jndi.properties` se proporciona con la instalación de JRules bajo el directorio *JRULES_HOME*`/executionserver/samples/j2eerulesession/websphere7`. Puede personalizarlo y añadir el directorio que lo contiene a la vía de acceso de clases para el conector.

También debe apuntar con el conector a los archivos `sas.client.props`, `ssl.client.props`, y `soap.client.props`, que contienen información adicional para establecer una conexión con WebSphere Application Server desde la instalación de WebSphere Application Client utilizada por el conector. La instalación de WebSphere Application Client tiene archivos de ejemplo que se pueden personalizar y configurar para conectarse a una instancia de WebSphere Application Server concreta. Estos archivos de ejemplo se almacenan en el directorio *WAS_HOME*`/properties`.

Especifique las opciones siguientes en la propiedad **Valores de JVM** bajo la sección **Propiedades de Java** en las propiedades de la etapa:

```
-Dcom.ibm.CORBA.ConfigURL=file:WAS_HOME/properties/sas.client.props
-Dcom.ibm.SSL.ConfigURL=file:WAS_HOME/properties/ssl.client.props
-Dcom.ibm.SOAP.ConfigURL=file:WAS_HOME/properties/soap.client.props
-Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager
-Djava.util.logging.configureByServer=true
```

Donde *WAS_HOME* es el directorio de inicio de WebSphere Application Client 7 en la máquina de nivel de motor de IBM InfoSphere Information Server.

Nota: El asistente de configuración nunca accede al Rule Execution Server de JRules desplegado en WebSphere Application Server. Si la modalidad de motor J2EE RES XU se selecciona cuando se inicia el asistente de configuración, o si esta modalidad de motor se selecciona en la sesión del asistente de configuración, el asistente accede al motor de JRules como Rule Execution Server de JRules gestionado localmente. Los requisitos en este caso son, por tanto, los mismos que para la modalidad de motor J2SE RES XU.

Generación de código Java mediante el asistente de configuración

Cuando el asistente de configuración se utiliza para generar código Java basado en las columnas definidas en los enlaces de la etapa, el conector no requiere acceso a ningún recurso distinto del que proporciona automáticamente IBM InfoSphere Information Server y el entorno de ejecución Java (JRE) en el que se ejecuta el conector.

Diseño de un trabajo de InfoSphere DataStage con el conector ILOG JRules

Debe crear un trabajo de IBM InfoSphere DataStage y configurarlo para invocar un conjunto de reglas JRules específico.

Procedimiento

1. Crear un trabajo.
2. Configurar la etapa de conector ILOG.
3. Configurar las propiedades de enlace de entrada y salida.
4. (Opcional)Configurar la propiedad de enlace de rechazo.
5. (Opcional) Configurar enlaces de almacenamiento intermedio.
6. Compilar el trabajo.

Creación de un trabajo

Debe crear un trabajo paralelo de IBM InfoSphere DataStage con la etapa ILOG JRules para invocar un conjunto de reglas JRules específico.

Procedimiento

1. En el cliente de Designer, seleccione **Archivo** > **Nuevo**.
2. Seleccione el icono **Trabajo paralelo** y pulse **Aceptar**.
3. En el lienzo Trabajo paralelo, defina las etapas para acceder a los datos de origen y de destino, tales como datos de archivo o de base de datos.
4. En el área de la paleta del cliente de Designer, pulse **Tiempo real**.
5. Seleccione el icono **Conector de ILOG JRules** de la etapa y arrastre la etapa al trabajo abierto. Coloque la etapa entre las etapas de origen y de destino.
6. Enlace las distintas etapas.
7. (Opcional) Renombre los enlaces y las etapas.
8. Seleccione **Archivo** > **Guardar** para guardar el trabajo.

Configuración de la etapa para invocar el conjunto de reglas

Cuando se crea un trabajo de etapa ILOG JRules, debe configurar las propiedades de la etapa.

Procedimiento

1. En el lienzo paralelo, realice una doble pulsación en el icono de etapa de conector ILOG JRules.
2. En el separador **Propiedades**, especifique un valor para el campo Modalidad de motor.
3. Especifique un valor para el campo **Ubicación del conjunto de reglas**.
4. Para habilitar el proceso por lotes, seleccione **Sí** como valor para el campo **Habilitar proceso por lotes**.
5. Para habilitar el proceso de clave, seleccione **Sí** como valor para el campo **Habilitar proceso de clave**.
6. Especifique un valor para el campo **Tipo de XOM**.
7. Para propagar los campos sin coincidencia de los enlaces de entrada a los enlaces de salida, seleccione **Sí** como valor para la opción **Propagar campos no coincidentes**.

8. Especifique un valor para **Estrategia de propagación de campo**, dependiendo de cómo desea que el conector asocie los enlaces de entrada y de salida al propagar los campos de enlace de entrada no coincidentes.
9. En la sección **Propiedades de Java**, establezca el valor del campo **Vía de acceso de clases** para incluir directorios y archivadores con definiciones de clase Java y los archivos de configuración que necesita la etapa.
10. (Opcional) Especifique un valor para el campo **Tamaño de almacenamiento dinámico**.
11. (Opcional) Especifique un valor para el campo **Opciones de JVM**.
12. Pulse **Aceptar** para guardar.

Configuración de propiedades de enlace de entrada y salida

La etapa ILOG JRules que invoca el conjunto de reglas JRules seleccionado contiene enlaces de entrada y de salida. La etapa ILOG JRules da soporte a varios enlaces de entrada y de salida. En la modalidad de clave, los registros de entrada deben ordenarse y correlacionarse con una clave común. Existe un enlace de entrada maestro y uno o varios enlaces de entrada secundarios.

Procedimiento

1. Abra la etapa del conector ILOG JRules.
2. Abra el separador **Propiedades** del enlace de entrada.
3. En el campo **Nombre de parámetro de conjunto de reglas**, especifique el nombre del parámetro de conjunto de reglas tal y como se define en la signatura del conjunto de reglas.
4. En el campo **Clase de parámetro de conjunto de reglas**, especifique la clase XOM Java para el parámetro de conjunto de reglas que está asociado con el enlace. Si la clase XOM Java es una matriz, el nombre de parámetro debe finalizar con []. Si la clase XOM Java es una XOM dinámica, este campo debe dejarse vacío.
5. Si la opción **Propagar campos no coincidentes** está habilitada en las propiedades de la etapa y la opción **Estrategia de propagación de campo** se establece en **Coincidencia utilizando valores de clave presentes en registros**, en el campo **Columna de clave de propagación**, especifique la columna clave de propagación que se debe utilizar.
6. (Opcional) Para convertir el enlace de entrada como enlace de almacenamiento intermedio, establezca el campo **Enlace de almacenamiento intermedio** en **Sí**.
7. Pulse **Aceptar** para guardar.

Enlaces de etapa del conector y asociación de parámetros de conjunto de reglas

Los enlaces de una etapa ILOG JRules deben estar correlacionados con los nombres de parámetro de conjunto de reglas. Si el conjunto de reglas está basado en Java XOM, la clase Java completa del parámetro de conjunto de reglas también debe especificarse para cada enlace. Si el parámetro es de tipo matriz, el nombre de clase debe finalizar con [].

ILOG JRules Connector lee registros de un enlace de entrada y los convierte en objetos Java de la clase con la que se correlaciona el enlace. Si el parámetro es de tipo matriz, todos los objetos derivados del mismo enlace de entrada se almacenan en una matriz. Toda la matriz se establece como valor de parámetro.

Si el parámetro no es de tipo matriz, los registros de entrada individuales se convierten en objetos Java y se establecen como valores de parámetro. Si un

parámetro asociado con el enlace secundario no es una matriz y varias filas del enlace secundario se unen en una sola fila del enlace primario, se envía un objeto a la vez a ILOG JRules en las filas del enlace secundario. La misma fila del enlace maestro se envía a ILOG JRules para cada invocación del conjunto de reglas ILOG JRules.

Para producir registros en los enlaces de salida, diseñe las reglas para añadir objetos Java a las matrices o variables Java asociadas con el conjunto de parámetros de salida. Si los enlaces de entrada y de salida se correlacionan con clases diferentes, las reglas empresariales deben crear los objetos de salida. Sin embargo, si la misma clase Java se correlaciona con los enlaces de entrada y de salida, puede pasar el objeto del parámetro de entrada a los parámetros de salida. En este escenario, los objetos de salida no se crean explícitamente mediante las reglas, sino que se pasan desde la entrada.

Enlaces de almacenamiento intermedio

Si un enlace de entrada está marcado como un enlace de almacenamiento intermedio, el conector lee los registros de este enlace sólo una vez y reutiliza los registros para cada ejecución de conjunto de reglas durante la duración del trabajo. Puede configurar un enlace de entrada en el trabajo del conector ILOG JRules como un enlace de almacenamiento intermedio

Cuando se configura el enlace de almacenamiento intermedio, los registros capturados en el enlace de almacenamiento intermedio se utilizan para generar valores de parámetro de conjunto de reglas fijos para utilizarlos en múltiples ejecuciones del conjunto de reglas.

Si los registros llegan a la etapa en oleadas de transacciones, el conector lee los registros de los enlaces de almacenamiento intermedio por separado en cada una de las oleadas y los reutiliza entre ejecuciones de conjunto de reglas en la misma oleada.

Si no hay registros en el enlace de almacenamiento intermedio, se lleva a cabo una acción según el tipo del parámetro de conjunto de reglas asociado con el enlace de almacenamiento intermedio. Si el parámetro de conjunto de reglas se basa en un tipo de Java de matriz, el conector crea una matriz vacía y la utiliza para el valor del parámetro de conjunto de reglas entre ejecuciones del conjunto de reglas. De lo contrario, el conector registra un error y el trabajo falla.

Cuando el parámetro de conjunto de reglas asociado con el enlace de almacenamiento intermedio no se basa en un tipo de Java de matriz, o si la etapa se ha configurado para un conjunto de reglas basado en Dynamic XOM, el conector espera encontrar un registro único en el enlace de almacenamiento intermedio y utilizarlo para producir dicho valor de parámetro de conjunto de reglas. Si varios registros están disponibles en el enlace, el conector envía los registros que siguen al primer registro a un enlace de rechazo, si se ha definido para dicho enlace de entrada y la opción **Error de registro restante** está habilitada. De lo contrario, el conector registra un mensaje de error y el trabajo falla.

Si un enlace secundario se configura como enlace de almacenamiento intermedio, independientemente de la modalidad en la que el conector se ejecuta y del tamaño de lote y columnas de clave especificados para la etapa, el conector lee todos los registros disponibles en dicho enlace intermedio.

Si el enlace maestro está configurado como un enlace de almacenamiento intermedio, el conector lee primero todos los registros disponibles en el enlace

maestro. A continuación, lee los registros del resto de enlaces secundarios que no están configurados como enlaces de almacenamiento intermedio del modo siguiente:

- Si el conector se ejecuta en modalidad de proceso por lotes y la modalidad de clave está inhabilitada, el conector lee todos los registros disponibles en el enlace secundario, pero no más de *n* registros, donde *n* es el tamaño de lote especificado (0 para ilimitado). Después de ejecutar el conjunto de reglas, el conector continúa leyendo registros en este enlace para la ejecución del conjunto de reglas.
- Si el conector se ejecuta en modalidad de clave, el valor de tamaño de lote se ignora y el conector lee todos los registros en el enlace secundario y espera que todos ellos tengan valores de columna de clave que coincidan con los valores de columna de clave de uno de los registros recuperados en el enlace de almacenamiento intermedio maestro. Una sola ejecución de conjunto de reglas se lleva a cabo en el trabajo, o en la oleada si los registros están llegando en oleadas de transacciones. Los registros cuyos valores de columna de clave no coinciden con los valores de columna de clave de cualquiera de los registros maestros se rechazan si el enlace de rechazo se ha definido para dicho enlace secundario y la opción **Error de no coincidencia de clave** está seleccionado para ese enlace de rechazo. De lo contrario, el conector registra un error y el trabajo falla.

Manejo del valor nulo

ILOG JRules Connector tiene un mecanismo para manejar valores nulos en los campos de los registros en los enlaces de entrada y salida y los valores nulos de Java en los valores de parámetro de conjunto de reglas.

Enlaces de entrada

Cuando un campo de un registro de entrada que se utiliza para producir el valor del parámetro de conjunto de reglas IN o IN_OUT es un valor nulo, el valor nulo se correlaciona con el valor nulo Java en el parámetro de conjunto de reglas, si dicha correlación es posible. Si la correlación no es posible, el trabajo falla con un error si no se ha configurado un enlace de rechazo. Si se ha configurado un enlace de rechazo para el enlace de entrada y la opción **Error de entrada nula**, el registro se rechaza.

No es posible correlacionar el valor de entrada nulo con el valor nulo de Java en el parámetro de conjunto de reglas en los casos siguientes:

- El conjunto de reglas se basa en Dynamic XOM. En este caso, el campo que representa el valor del documento XML del parámetro de conjunto de reglas no se puede establecer en nulo.
- El campo se correlaciona con el parámetro de conjunto de reglas de un tipo Java primitivo como, por ejemplo, byte, short, int, long, float, double, boolean o char. No puede especificarse el valor nulo para un tipo Java primitivo.
- El campo se correlaciona con un argumento de un método en la clase Java de los parámetros de conjunto de reglas, y ese argumento es de un tipo Java primitivo. Una vez más, el valor nulo no puede especificarse para un tipo Java primitivo.

Enlaces de salida

Cuando se encuentra un valor nulo de Java mientras se inspeccionaba el valor del parámetro de conjunto de reglas OUT o IN_OUT asociado con el enlace de salida,

el valor nulo de Java se correlaciona con el valor nulo del campo de registro de salida correspondiente, si la correlación es posible. Si la correlación no es posible, el trabajo falla con un error.

No es posible correlacionar el valor nulo de Java con el valor nulo en el campo de registro de salida correspondiente en los casos siguientes:

- El valor del parámetro de conjunto de reglas es nulo y se correlaciona con una sola columna en el enlace de salida que no tiene posibilidades de nulos. Una columna no tiene posibilidades de nulos si tiene el atributo **Con posibilidades de nulos** establecido en **No**.
- El valor del parámetro de conjunto de reglas reglas es nulo y se correlaciona con varias columnas en el enlace de salida, de las cuales algunas tienen posibilidades de nulos y otras no. En este caso, el conector no puede invocar métodos en el parámetro de salida para inicializar los valores de campo basándose en los valores de retorno de método.
- El valor del parámetro de conjunto de reglas no es nulo pero un método invocado en el objeto Java del parámetro de salida devuelve un valor nulo, y la columna con la que se correlaciona el valor de retorno no tiene posibilidades de nulos.

Configuración del enlace de rechazo

Puede configurar enlaces de rechazo en la etapa ILOG JRules para aceptar los registros de entrada que el conector ha descubierto que son erróneos. Cuando el conector envía los registros erróneos a los enlaces de rechazo definidos para la etapa, el trabajo continúa ejecutándose para el resto de registros de entrada y puede verificar los registros rechazados para determinar la causa del error. Si un enlace de rechazo no está configurado para un enlace de entrada en el que el conector ha detectado registros erróneos, el conector registra un error y el trabajo falla.

Procedimiento

1. Defina un enlace de salida para la etapa ILOG JRules
2. Pulse con el botón derecho del ratón en el enlace de salida y, a continuación, seleccione el recuadro de selección **Convertir en rechazo** para convertir el enlace de salida en un enlace de rechazo.
3. Abra el editor de etapa y seleccione el separador **Rechazar**.
4. Seleccione las opciones relevantes en la lista **Rechazar filas según las seleccionadas** de la lista para definir la condición de rechazo.
5. Seleccione **ERRORCODE** o **ERRORCODE**, o ambos, en la sección **Añadir a fila de rechazo** para especificar que el código de error y la información de texto de error deben incluirse en los registros rechazados para identificar la razón por la que los registros se han rechazado.
6. En el campo **Rechazar de enlace**, seleccione el enlace de entrada.
7. En el campo **Terminar anormalmente si**, especifique el valor límite para la condición.
8. En el campo **Terminar anormalmente después** , especifique el límite superior para los enlaces de rechazo.
9. Pulse **Aceptar** para guardar.

Compilación del trabajo

Cuando acabe de diseñar un trabajo, compílelo.

Procedimiento

1. En el cliente del Diseñador, abra el trabajo que desee compilar.
2. En la barra de herramientas, pulse **Compilar**.
3. Si se visualizan mensajes de error, edite el trabajo para resolver los errores. Pulse **Compilar** después de resolver los errores para volver a compilar el trabajo. Después de compilar el trabajo satisfactoriamente, puede ejecutarlo, ya sea desde IBM InfoSphere DataStage and QualityStage Designer o desde IBM InfoSphere DataStage and QualityStage Director.

Utilización del asistente de configuración

ILOG JRules Connector incluye un asistente de configuración que se puede utilizar para simplificar la configuración de propiedades y los esquemas de enlace en la etapa para invocar el conjunto de reglas de JRules cuando se ejecuta el trabajo. El asistente de configuración también se puede utilizar para generar automáticamente código Java basado en los esquemas de enlace y valores de propiedad definidos en la etapa.

Configuración de la etapa mediante el asistente

ILOG JRules Connector incluye un asistente que puede utilizar para configurar propiedades y los esquemas de enlace en la etapa para invocar el conjunto de reglas de JRules seleccionado cuando se ejecuta el trabajo.

Acerca de esta tarea

Cuando configure la etapa para invocar un conjunto de reglas de JRules específico, el número de enlaces de entrada debe coincidir con el número de parámetros de conjunto de reglas IN e IN_OUT, y el número de enlaces de salida debe coincidir con el número de parámetros OUT e IN_OUT. Los enlaces de rechazo en la etapa están excluidos de esta fórmula y el asistente de configuración los ignora. Si sabe el número y tipo de parámetros de conjunto de reglas, puede definir el número correcto de enlaces de los tipos correctos en la etapa antes de invocar el asistente. Si la información no está disponible, aún puede iniciar el asistente y utilizarlo para localizar el conjunto de reglas de interés y averiguar el número y tipo de parámetros definidos para el mismo. Sin embargo, si hay enlaces que faltan, tendrá que cancelar el asistente y añadir manualmente los enlaces necesarios para que el asistente pueda completar la configuración de la etapa porque el asistente no puede añadir automáticamente enlaces a la etapa.

Procedimiento

1. Efectúe una doble pulsación en la etapa ILOG JRules.
2. Pulse **Configurar**.
3. En la página **Acción del asistente y valores del archivo de registro**, seleccione **Importar parámetros de conjunto de reglas**.
4. (Opcional) Especifique la vía de acceso del archivo de registro en el campo **Vía de acceso de archivo de registro** para almacenar la información de depuración y, a continuación, pulse **Siguiente**.
5. En la página **Modalidad de motor y filtro de conjunto de reglas**, especifique la modalidad del motor y la expresión de filtro de conjunto de reglas y, a continuación, pulse **Siguiente**.
6. Configure la pantalla posterior dependiendo de la modalidad de motor seleccionada.

- Si ha seleccionado **Motor de núcleo** como modalidad de motor, se muestra la **página de selección de archivado de conjunto de reglas**. Seleccione el recuadro de selección **Incluir métodos heredados** para especificar si se deben procesar métodos que las clases Java del parámetro de conjunto de reglas heredan de sus clases de ancestro. Especifique el archivo de archivado de conjunto de reglas para el que se va a configurar la etapa.
- Si ha seleccionado **J2SE RES XU** o **J2EE RES XU** como modalidad de motor, se muestra la página **Selección de conjunto de reglas desplegadas**. Seleccione el recuadro de selección **Incluir métodos heredados** para especificar si se deben procesar métodos que las clases Java del parámetro de conjunto de reglas heredan de sus clases de ancestro. Especifique el conjunto de reglas desplegadas para el que se va a configurar la etapa.

7. En la página **Selección de parámetros de conjunto de reglas**, configure la modalidad de importación y los parámetros de conjunto de reglas y los métodos de clase Java correspondientes para los que se van a configurar definiciones de esquema para los enlaces definidos en la etapa.
8. En la página **Correlación de esquema para los enlaces y parámetros de conjunto de reglas**, defina las asociaciones entre los parámetros de conjunto de reglas y los enlaces de la etapa.
9. En la página **Importar parámetros del conjunto de reglas - Vista previa**, revise el informe y el resumen de resultados esperados y pulse **Finalizar**.

Generación de código Java para la etapa utilizando el asistente

Puede utilizar el asistente para generar código Java basado en las definiciones de columna de los enlaces definidos en la etapa.

Procedimiento

1. Efectúe una doble pulsación en el conector de ILOG JRules.
2. Pulse **Configurar**.
3. En la página **Acción del asistente y valores del archivo de registro**, seleccione el valor de **Generar código Java** en el campo **Acción de asistente**.
4. En la página **Valores de código Java**, en el campo **Vía de acceso de directorio de código fuente**, especifique la vía de acceso del directorio completa en el host del motor en el que se grabarán los archivos `.java` generados.
5. En el campo **Formato de salida**, especifique el formato del tipo Java para el que se va a generar el código.
6. En el campo **Prefijo de variable de miembro**, especifique el prefijo que debe utilizarse para los nombres de variable de miembro privado generado.
7. Seleccione el recuadro de selección **Sustituir archivos existentes** para especificar si se debe grabar encima de los archivos `.java` que tienen el mismo nombre y ubicación.
8. Seleccione el recuadro de selección **Excluir métodos individuales** para especificar si se debe permitir la exclusión manual de los métodos individuales de la lista de métodos notificados, de forma que el código no se generará para estos métodos.
9. En la página **Correlación de código fuente para tipos Java**, inspeccione los métodos Java para los que se generará el código. Si, en el paso anterior, se ha seleccionado la opción **Excluir métodos individuales**, seleccione los métodos para los que no desea que se genere el código. Pulse **Siguiente**.
10. En la página **Generar código Java - Vista previa**, revise las acciones que realizará el asistente y, a continuación, pulse **Finalizar**.

Resultados

Los archivos se escriben en los subdirectorios del directorio de código fuente especificado, determinado por las especificaciones de paquete en los nombres de tipo Java. Si el paquete no se especifica para algunos de los tipos, el archivo de código fuente `.java` para ese tipo se genera en el directorio de código fuente especificado. Por ejemplo, si el directorio de código fuente se especifica como `/projects/javaxom` y el nombre de tipo Java es `com.example.import.Test`, se creará el archivo `Test.java` en el directorio `/projects/javaxom/com/example/import`. Si el directorio existe, el asistente lo vuelve a utilizar. Si se detecta un problema durante la creación del directorio o al grabar el archivo `.java` en el directorio, aparece un mensaje de error.

Archivo de configuración de asistente

El archivo de configuración del asistente proporciona la opción de especificar los valores predeterminados visualizados por el asistente de configuración cuando se inicia desde la etapa de conector.

Finalidad

En algunos casos, los valores predeterminados para los valores del asistente de configuración podrían no ser convenientes respecto al entorno actual. Por ejemplo, si el asistente se lanza principalmente para generar código Java para las definiciones de columna de los enlaces, **Generar código Java** es una opción predeterminada que es mejor que **Importar parámetros de conjunto de reglas** en la página **Acción del asistente y valores del archivo de registro**.

En algunos casos, el valor predeterminado puede ser específico del entorno del sistema actual. Por ejemplo, si el asistente se utiliza principalmente para configurar la etapa según los archivos de archivado de conjunto de reglas que, a su vez, siempre se despliegan en el mismo directorio en el host del motor de InfoSphere Information Server, la vía de acceso a ese directorio puede ser un buen candidato para utilizarlo como valor predeterminado para el valor de **Expresión de filtro de conjunto de reglas**.

Los valores predeterminados para los valores del asistente pueden especificarse como propiedades en el archivo de configuración del asistente `ccjrules.wizard.properties`. Este archivo no se crea de forma predeterminada cuando el tipo de etapa de conector está instalado. Puede crear y copiar el archivo de configuración del asistente en el directorio *IS_HOME*`/ASBNode/lib/java` en el host del motor de InfoSphere Information Server, donde *IS_HOME* es el directorio inicial de InfoSphere Information Server. Como alternativa, puede incluir el directorio en el valor de la propiedad **Vía de acceso de clases** en la etapa desde la que se inicia el asistente.

Un contenido de ejemplo para el archivo `ccjrules.wizard.properties` se muestra en la sección de ejemplo. El contenido del ejemplo incluye todos los valores del asistente para el que puede especificarse el valor predeterminado. Las líneas que asignan valores predeterminados a estos valores están inhabilitadas de forma predeterminada. Para habilitar un valor concreto, debe eliminarse el carácter # al principio de la línea.

Por ejemplo, para especificar que el directorio `C:/temp/javadoc` se utiliza como el directorio predeterminado en el que el asistente debe generar archivos de código fuente `.java`, el carácter inicial # debe ser eliminado de la línea:

```
# source_code_directory = C:/temp/javacode
```

Cada vez que el contenido de la configuración del asistente se cambia, debe cerrar e iniciar el asistente de nuevo para utilizar los nuevos valores.

Ejemplo

Puede copiar la muestra de código dada a continuación para crear su propio archivo de configuración del asistente:

```
#-------------------------------------------------------------------------------
# Acción predeterminada del asistente:
# import_ruleset - Importar parámetros de conjunto de reglas
# generate_code - Generar código Java
#-------------------------------------------------------------------------------
# wizard_action = import_ruleset
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Registro de ubicación de archivo. Utilice la barra inclinada como separador
# en todas las plataformas, incluida Windows.
#-------------------------------------------------------------------------------
# log_file_path = C:/temp/ccjrules.log
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Sobrescribir distintivo de archivo de registro. Los valores permitidos son:
# 0 - No sobrescribir. Añadir al registro si existe, de lo contrario, crearlo.
# 1 - Sobrescribir el archivo si existe, de lo contrario, crearlo
#-------------------------------------------------------------------------------
# overwrite_log_file = 0
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Modo de motor predeterminado. Si el valor se especifica aquí, tiene prioridad
# a través del valor de propiedad del modo de motor en la etapa desde la que
# se invocó el asistente. Los valores permitidos son:
# core_engine - Motor de núcleo
# j2se_res_xu - XU RES J2SE
# j2ee_res_xu - XU RES J2EE
#-------------------------------------------------------------------------------
# engine_mode = core_engine
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Expresión de filtro predeterminada para localizar conjuntos de reglas
# Si se especifica, se utiliza el valor a menos que se especifique un valor en
# la propiedad de ubicación de conjunto de reglas en la etapa desde la que se
# invocó el asistente. En ese caso, el valor de ubicación de conjunto de reglas
# se utiliza como expresión de filtro predeterminada en el asistente. Utilice
# la barra inclinada como separador de archivos en todas las plataformas,
 incluida Windows.
#-------------------------------------------------------------------------------
# filter_expression = C:/temp
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Preferencia predeterminada para visualizar todas las versiones de rulpapp y
# ruleset. Los valores permitidos son:
# 0 - Mostrar todas las versiones de ruleapp y ruleset
# 1 - Mostrar sólo las versiones de ruleapp y ruleset más recientes
#-------------------------------------------------------------------------------
# show_all_versions = 0
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
```

```
# Preferencia predeterminada para visualizar métodos heredados para las clases
# de parámetro de conjunto de reglas. Los valores permitidos son:
# 0 - No incluir métodos heredados
# 1 - Incluir métodos heredados
#-------------------------------------------------------------------------------
# include_inherited_methods = 0
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Modalidad de importación predeterminada para preservar columnas existentes
# en el enlace al importar parámetros de conjunto de reglas.
# Los valores permitidos son:

# merge - Utilizar las columnas existentes para la correlación
# replace - Sustituir todas las columnas existentes
#-------------------------------------------------------------------------------
# import_mode = merge
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Directorio en el que se generarán archivos .java. Utilice la barra inclinada
# como separador de archivos en todas las plataformas, incluida Windows.
#-------------------------------------------------------------------------------
# source_code_directory = C:/temp/javacode
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Formato de salida para el código fuente generado. El valor permitido es:
# class - clase Java
# inteface - interfaz Java
#-------------------------------------------------------------------------------
# output_format = class
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Prefijo para variables de miembro en el código generado
#-------------------------------------------------------------------------------
# member_variable_prefix = m_
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Preferencia predeterminada para sustituir los archivos .java existentes al
# generar el código .java.
# 0 - No sustituir los archivos existentes
# 1 - Sustituir los archivos existentes
#-------------------------------------------------------------------------------
# replace_existing_files = 0
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Preferencia predeterminada para excluir métodos individuales al generar el
# código .java.
# 0 - No habilitar la exclusión de métodos individuales
# 1 - Habilitar los recuadros de selección para utilizarlos para excluir
# métodos individuales
#-------------------------------------------------------------------------------
# exclude_individual_methods = 0
#-------------------------------------------------------------------------------
```

Tipos y métodos de Java básicos

El tipo de Java básico en el contexto de las ILOG JRules Connector es un tipo de Java que puede correlacionarse directamente con una columna de enlace.

Tipos de Java básicos

Los tipos de Java básicos son:

- Tipos primitivos: int, short, long, double, float, boolean, byte, char
- Clases de derivador para tipos primitivos: java.lang.Integer, java.lang.Short, java.lang.Long, java.lang.Double, java.lang.Float, java.lang.Boolean, java.lang.Byte, java.lang.Character
- Tipo de serie: java.lang.String
- Tipos de fecha/hora: java.util.Date, java.util.Calendar, java.sql.Date, java.sql.Time, java.sql.Timestamp
- Tipos numéricos: java.math.BigInteger, java.math.BigDecimal

Cuando un parámetro de conjunto de reglas es un parámetro XML o un parámetro Java de tipo de Java básico, el parámetro se correlaciona directamente con una columna de enlace del modo siguiente:

- Si el parámetro está correlacionado con una columna en un enlace de entrada, ese parámetro debe ser un parámetro IN o IN_OUT. La columna representa el valor que debe pasarse al parámetro cuando se invoca el conjunto de reglas.
- Si el parámetro está correlacionado con una columna en un enlace de salida, ese parámetro debe ser un parámetro OUT o IN_OUT. La columna representa el valor devuelto por el conjunto de reglas para el parámetro, cuando se invoca el conjunto de reglas.

Cuando el parámetro de conjunto de reglas no se correlaciona directamente con una columna de enlace, las construcciones de conjunto de reglas que se correlacionan con las columnas de enlace son los valores de retorno y argumentos de métodos básicos de la clase Java asociada con el parámetro de conjunto de reglas.

Métodos de Java básicos

Un método de Java básico satisface una de las condiciones siguientes:

- Es un método constructor o un método que no devuelve un valor (método vacío) y todos sus argumentos son de tipos de Java básicos. Los argumentos del método se correlacionan con la columna del enlace de entrada asociado con el parámetro de conjunto de reglas en cuya clase se ha definido el método. El parámetro de conjunto de reglas debe ser un parámetro IN o IN_OUT. Los valores de columnas se pasan a los argumentos del método al invocar el método en el objeto de parámetro de conjunto de reglas respectivo antes de invocar el conjunto de reglas y pasándole el parámetro.
- Es un método que no contiene ningún argumento y su valor de retorno es un tipo de Java básico. El valor de retorno del método se correlaciona con una columna en el enlace de salida asociado con el parámetro de reglas en cuyo tipo de clase se ha definido el método. El parámetro de conjunto de reglas debe ser un parámetro OUT o IN_OUT. La columna representa el valor de retorno que se debe recuperar del método después de invocar el conjunto de reglas que contiene el parámetro e invocar el método en el objeto de parámetro de conjunto de reglas.

El asistente de configuración maneja los parámetros de conjunto de reglas asociados con los tipos de matriz Java como si fueran de los tipos no de matriz correspondientes. Cuando se ejecuta el trabajo, la etapa normalmente recopila varios registros en el enlace para preparar el valor del parámetro de conjunto de reglas. Los registros de datos en el enlace asociado con ese parámetro de conjunto

de reglas corresponden a objetos Java que se almacenan en la matriz, que en su conjunto corresponde al valor del parámetro de conjunto de reglas. Sin embargo, si un parámetro de conjunto de reglas es de los tipos de matriz Java char[] y byte[], corresponde a un solo valor de campo de un solo registro en el enlace, tal como se indica a continuación :

- Si la columna es del tipo de datos de caracteres como VarChar o NVarChar, los caracteres del valor de campo de texto se correlacionan con los elementos char de la matriz char[] para el parámetro de conjunto de reglas.
- Si la columna es de un tipo de datos binario, como por ejemplo VarBinary, los bytes de los valores de campo binario correspondientes se correlacionan con los elementos de byte de la matriz byte[].

Correlación de tipos de DataStage con tipos de Java

El asistente correlaciona definiciones de tipo de columna de InfoSphere DataStage en el enlace con los tipos de Java utilizados para las variables de miembro privado, los valores de retorno de método y los argumentos de método cuando genera código Java.

Durante la ejecución, el conector correlaciona, según sea necesario, las definiciones de tipo de columna de InfoSphere DataStage presentes en los enlaces de entrada con los tipos de Java, a fin de inicializar valores para los parámetros IN y IN_OUT asociados con dichos enlaces. La tabla siguiente describe cómo la definición de tipo de InfoSphere DataStage está correlacionada con el tipo de Java:

Tabla 1. Correlación del tipo de InfoSphere DataStage con el tipo de Java

Tipo de SQL DataStage	Longitud	Escala	Ampliado	Tipo de Java
TinyInt	(n/a)	(n/a)	Vacío	byte, java.lang.Byte
TinyInt	(n/a)	(n/a)	No firmado	short, java.lang.Short
SmallInt	(n/a)	(n/a)	Vacío	short, java.lang.Short
SmallInt	(n/a)	(n/a)	No firmado	int, java.lang.Integer
Integer	(n/a)	(n/a)	Vacío	int, java.lang.Integer
Integer	(n/a)	(n/a)	No firmado	long, java.lang.Long
BigInt	(n/a)	(n/a)	Vacío	long, java.lang.Long
BigInt	(n/a)	(n/a)	No firmado	java.math.BigInteger
Bit	(n/a)	(n/a)	(n/a)	boolean, java.lang.Boolean
Float	(n/a)	(n/a)	(n/a)	float, java.lang.Float
Double	(n/a)	(n/a)	(n/a)	double, java.lang.Double
Decimal	Cualquiera	Cual-quiera	(n/a)	java.math.BigDecimal
Char, NChar	1	(n/a)	Cualquiera	char, java.lang.Character
Char, NChar	n > 1	(n/a)	Cualquiera	char[]
Char, NChar	Vacío	(n/a)	Cualquiera	char[]
VarChar, NVarChar, LongVarChar, LongNVarChar	Cualquiera	(n/a)	Cualquiera	java.lang.String
Binary	1	(n/a)	(n/a)	byte, java.lang.Byte
Binary	n > 1	(n/a)	(n/a)	byte[]

Tabla 1. Correlación del tipo de InfoSphere DataStage con el tipo de Java (continuación)

Tipo de SQL DataStage	Longitud	Escala	Ampliado	Tipo de Java
Binary	Vacío	(n/a)	(n/a)	byte[]
VarBinary	Cualquiera	(n/a)	(n/a)	byte[]
Hora	(n/a)	(n/a)	Vacío	java.sql.Time
Hora	(n/a)	(n/a)	Microsegundos	java.sql.Timestamp
Fecha	(n/a)	(n/a)	(n/a)	java.sql.Date
Indicación de fecha y hora	(n/a)	(n/a)	Vacío	java.util.Date
Indicación de fecha y hora	(n/a)	(n/a)	Microsegundos	java.sql.Timestamp

Algunas de las definiciones de tipo de InfoSphere DataStage se pueden correlacionar con más de un tipo de Java. En estos casos, el valor especificado para el atributo **Con posibilidades de nulos** determina cuál de los dos tipos de Java se correlaciona. Si el valor de **Con posibilidades de nulos** está establecido en **No**, el tipo de Java primitivo está seleccionado. Si el valor de **Con posibilidades de nulos** está establecido en **Sí** o **Desconocido**, el tipo de derivador de objeto está seleccionado. Por ejemplo, si una columna de enlace del tipo TinyInt de InfoSphere DataStage con el atributo **Firmado** establecido en **Sí** se correlaciona con el tipo de Java primitivo `byte`, si el atributo **Con posibilidades de nulos** se establece en **No** y se correlaciona con el tipo de derivador Java `java.lang.Byte` si su atributo **Con posibilidades de nulos** se ha establecido en **Sí**.

Los tipos de datos basados en caracteres de InfoSphere DataStage Char, NChar, VarChar, LongVarChar, NVarChar y LongNVarChar también se pueden correlacionar con los tipos de Java `boolean` y `java.lang.Boolean`. El valor de texto "true" (no sensible a mayúsculas y minúsculas) se correlaciona con el valor booleano *true*. El resto de los valores se correlacionan con el valor booleano *false*.

Correlación de tipo de Java con tipo de DataStage

El asistente de configuración correlaciona tipos de Java utilizados por los parámetros de conjunto de reglas con las definiciones de columna correspondientes en los enlaces de la etapa.

Durante la ejecución, el conector correlaciona, según sea necesario, los tipos de Java en los parámetros de conjunto de reglas OUT e IN_OUT asociados a enlaces de salida con definiciones de tipo de InfoSphere DataStage para las columnas en los enlaces.

Tabla 2. Correlación de tipo de Java con tipo de InfoSphere DataStage

Tipo de Java	Tipo de SQL DataStage	Longitud	Escala	Ampliado
short, java.lang.Short	SmallInt	(n/a)	(n/a)	Vacío
int, java.lang.Integer	Integer	(n/a)	(n/a)	Vacío
long, java.lang.Long	BigInt	(n/a)	(n/a)	Vacío
float, java.lang.Float	Float	(n/a)	(n/a)	(n/a)
double, java.lang.Double	Double	(n/a)	(n/a)	(n/a)

Tabla 2. Correlación de tipo de Java con tipo de InfoSphere DataStage (continuación)

Tipo de Java	Tipo de SQL DataStage	Longitud	Escala	Ampliado
boolean, java.lang.Boolean	Bit	(n/a)	(n/a)	(n/a)
byte, java.lang.Byte	TinyInt	(n/a)	(n/a)	Vacío
byte[], java.lang.Byte[]	VarBinary	Vacío	(n/a)	(n/a)
char, java.lang.Character	Char	1	(n/a)	Unicode
char[], java.lang.Character[]	VarChar	Vacío	(n/a)	Unicode
java.lang.String	VarChar	Vacío	(n/a)	Unicode
java.util.Date, java.util.Calendar	Indicación de fecha y hora	(n/a)	(n/a)	Microsegundos
java.sql.Date	Fecha	(n/a)	(n/a)	(n/a)
java.sql.Time	Hora	(n/a)	(n/a)	Microsegundos
java.sql.Timestamp	Indicación de fecha y hora	(n/a)	(n/a)	Microsegundos
java.math.BigInteger	BigInt	(n/a)	(n/a)	Vacío
java.math.BigDecimal	Decimal	255	127	(n/a)

Las columnas de enlace que corresponden a tipos de Java int, short, long, double, float, boolean, byte y char se configuran con el atributo **Con posibilidades de nulos** establecido en **No**. Las columnas de los tipos de Java restantes, incluido los tipos char[] y byte[], están configuradas con el atributo **Con posibilidades de nulo** establecido en **Sí**.

Los tipos de Java `boolean` y `java.lang.Boolean` también se pueden correlacionar con los tipos de datos basados en caracteres de InfoSphere DataStage Char, NChar, VarChar, LongVarChar, NVarChar y LongNVarChar. El valor booleano true se correlaciona con el valor de texto *true*, y el valor booleano false se correlaciona con el valor de texto *false*.

Apéndice A. Accesibilidad de los productos

Puede obtener información sobre el estado de accesibilidad de los productos de IBM.

Los módulos de producto y las interfaces de usuario de IBM InfoSphere Information Server no son totalmente accesibles.

Para obtener información sobre el estado de accesibilidad de los productos de IBM, consulte la información de accesibilidad de productos de IBM en http://www.ibm.com/able/product_accessibility/index.html.

Documentación sobre accesibilidad

Se proporciona documentación accesible para los productos en IBM Knowledge Center. IBM Knowledge Center presenta la documentación en formato XHTML 1.0, que se puede ver en la mayoría de navegadores web. Dado que IBM Knowledge Center utiliza XHTML, puede establecer preferencias de visualización en el navegador. Esto también le permite utilizar lectores de pantalla y otras tecnologías de asistencia para acceder a la documentación.

La documentación que está en IBM Knowledge Center se proporciona en archivos PDF, que no son totalmente accesibles.

IBM y la accesibilidad

Consulte el sitio web IBM Human Ability and Accessibility Center para obtener más información sobre el compromiso de IBM con la accesibilidad.

Apéndice B. Lectura de la sintaxis de la línea de mandatos

En esta documentación se utilizan caracteres especiales para definir la sintaxis de la línea de mandatos.

Los siguientes caracteres especiales definen la sintaxis de línea de mandatos:

[ ] Identifica un argumento opcional. Se necesitan los argumentos que no están entre delimitadores.

... Indica que puede especificar varios valores para el argumento anterior.

| Indica mutuamente información exclusiva. Puede utilizar el argumento a la izquierda del separador o el argumento a la derecha del separador. No puede utilizar los dos argumentos en un solo mandato.

{ } Delimita un conjunto de argumentos mutuamente excluyentes cuando se requiere el uso de uno de los argumentos. Si los argumentos son opcionales, se muestran entre corchetes ([ ]).

Nota:

- El número máximo de caracteres de argumento es de 256.
- Los valores de argumento que incluyan espacios en blanco deben ir entre comillas dobles o simples.

Por ejemplo:

wsetsrc*[-S server] [-l label] [-n name] origen*

El argumento *source* es el único argumento obligatorio para el mandato **wsetsrc**. Los corchetes que encierran el resto de los argumentos indican que son opcionales.

wlsac *[-l | -f format] [key...] profile*

En este ejemplo, los argumentos de formato -l y -f son mutuamente excluyentes y opcionales. El argumento *profile* es obligatorio. El argumento *key* es opcional. Los puntos suspensivos (...) que siguen al argumento *key* indican que puede especificar varios nombres de clave.

wrb -import *{rule_pack | rule_set}...*

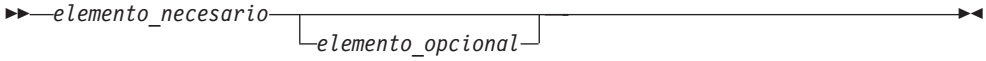
En este ejemplo, los argumentos rule_pack y rule_set se excluyen mutuamente, pero debe especificarse uno de ellos. Además, los puntos suspensivos (...) indican que puede especificar varios paquetes de reglas o conjuntos de reglas.

Apéndice C. Cómo leer los diagramas de sintaxis

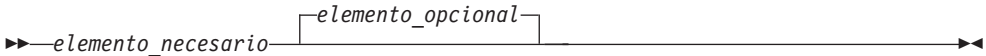
Las reglas siguientes se aplican a los diagramas de sintaxis que se utilizan en esta documentación:

- Lea los diagramas de sintaxis de izquierda a derecha y de arriba abajo, siguiendo el recorrido de la línea. Se utilizan las convenciones siguientes:
  - El símbolo >>--- indica el comienzo de un diagrama de sintaxis.
  - El símbolo ---> indica que el diagrama de sintaxis continúa en la línea siguiente.
  - El símbolo >--- indica que el diagrama de sintaxis viene de la línea anterior.
  - El símbolo --->< indica el final del diagrama de sintaxis.
- Los elementos obligatorios aparecen en la línea horizontal (la trayectoria principal).

- Los elementos opcionales aparecen debajo de la trayectoria principal.

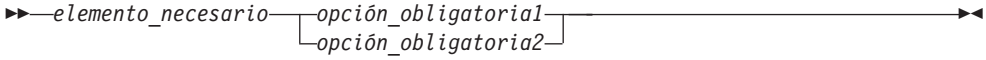


  Si aparece un elemento opcional sobre la línea principal, dicho elemento no tendrá efecto sobre el elemento de sintaxis y sólo se utilizará para facilitar la lectura.

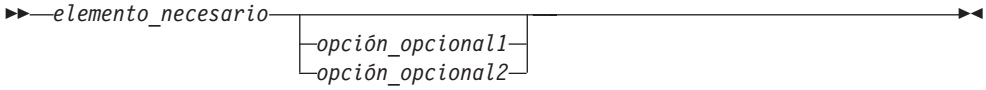


- Si se puede elegir entre dos o más elementos, éstos aparecerán apilados verticalmente.

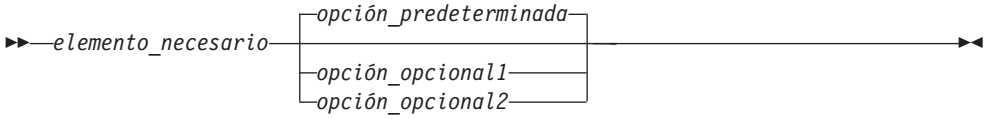
  Si se debe elegir uno de los elementos, un elemento de la pila aparece en la línea principal.



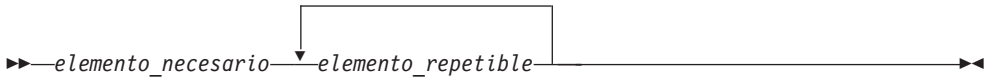
  Si la elección de uno de los elementos es opcional, toda la pila aparecerá por debajo de la línea principal.



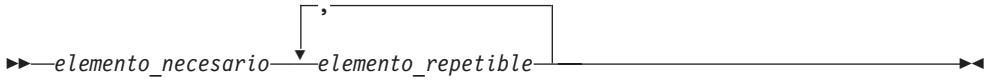
  Si uno de los elementos es el predeterminado, aparecerá por encima de la línea principal y las opciones restantes se mostrarán por debajo.



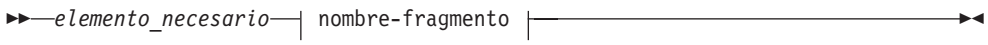
- Una flecha que vuelve hacia la izquierda, sobre la línea principal, indica un elemento que se puede repetir.



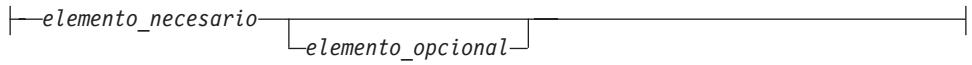
  Si la flecha de repetición contiene una coma, los elementos repetidos se deben separar mediante una coma.



  Una flecha de repetición sobre una pila indica que los elementos de la pila se pueden repetir.
- A veces, un diagrama se debe dividir en fragmentos. El fragmento de sintaxis se muestra por separado del diagrama de sintaxis principal, pero el contenido del fragmento se debe leer como si formara parte de la línea principal del diagrama.



  Nombre-fragmento:



- Las palabras clave, y sus abreviaturas mínimas si las hay, aparecen en mayúsculas. Se deben escribir exactamente tal como se muestran.
- Las variables aparecen en letras minúsculas en cursiva (por ejemplo, ***nombre-columna***). Representan nombres o valores proporcionados por el usuario.
- Separe las palabras clave y los parámetros con un espacio como mínimo si no se muestra ningún signo de puntuación en el diagrama.
- Entre los signos de puntuación, paréntesis, operadores aritméticos y otros símbolos exactamente como se muestran en el diagrama.
- Los pies de página se muestran con un número entre paréntesis, por ejemplo, (1).

Apéndice D. Cómo ponerse en contacto con IBM

Puede ponerse en contacto con IBM para obtener soporte al cliente, servicios de software, información sobre productos e información general. También puede facilitar comentarios a IBM sobre los productos y la documentación.

En la tabla siguiente se listan los recursos para soporte al cliente, servicios de software, formación e información sobre productos y soluciones.

Tabla 3. Recursos de IBM

Recurso	**Descripción y ubicación**
Portal de soporte de IBM	Puede personalizar la información de soporte eligiendo los productos y los temas que le interesen en www.ibm.com/support/entry/portal/Software/Information_Management/InfoSphere_Information_Server
Servicios de software	Puede encontrar información sobre servicios de software, de tecnologías de la información y de consultoría de negocio en el sitio de soluciones, en www.ibm.com/businesssolutions/
Mi IBM	Puede gestionar enlaces a sitios web de IBM y a información que satisfaga sus necesidades específicas de soporte técnico creando una cuenta en el sitio Mi IBM en www.ibm.com/account/
Formación y certificación	Puede obtener información sobre formación técnica y servicios de educación diseñados para personas, empresas y organizaciones públicas, a fin de adquirir, mantener y optimizar sus habilidades de TI en http://www.ibm.com/training
Representantes de IBM	Puede contactar con un representante de IBM para obtener información sobre soluciones en www.ibm.com/connect/ibm/us/en/

Apéndice E. Acceso a la documentación del producto

La documentación se proporciona en diversos formatos: en el IBM Knowledge Center en línea, en un centro de información opcional instalado localmente y como manuales PDF. Puede acceder a la ayuda en línea o instalada localmente directamente desde las interfaces de cliente del producto.

IBM Knowledge Center es el mejor lugar para encontrar la información más actualizada de InfoSphere Information Server. IBM Knowledge Center contiene ayuda para la mayoría de las interfaces del producto, así como documentación completa para todos los módulos de producto de la suite. Puede abrir IBM Knowledge Center desde el producto instalado o desde un navegador web.

Cómo acceder a IBM Knowledge Center

Existen varias maneras de acceder a la documentación en línea:

- Pulse el enlace **Ayuda** en la parte superior derecha de la interfaz de cliente.
- Pulse la tecla F1. Normalmente, la tecla F1 abre el tema que describe el contexto actual de la interfaz de cliente.

  Nota: La tecla F1 no funciona en clientes web.
- Escriba la dirección en un navegador web, por ejemplo, cuando no tenga iniciada una sesión en el producto.

  Escriba la siguiente dirección para acceder a todas las versiones de la documentación de InfoSphere Information Server:

  `http://www.ibm.com/support/knowledgecenter/SSZJPZ/`

  Si desea acceder a un tema concreto, especifique el número de versión con el identificador de producto, el nombre del plug-in de documentación y la vía de acceso al tema en el URL. Por ejemplo, el URL para la versión 11.3 de este tema es el siguiente. (El símbolo ⇒ indica una continuación de línea):

  ```
  http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/⇒
  com.ibm.swg.im.iis.common.doc/common/accessingiidoc.html
  ```

  Consejo:

  El Knowledge Center tiene también un URL corto:

  `http://ibm.biz/knowctr`

  Para especificar un URL corto a una página de producto, versión o tema específico, utilice un carácter de almohadilla (#) entre el URL corto y el identificador de producto. Por ejemplo, el URL corto a toda la documentación de InfoSphere Information Server es el siguiente URL:

  `http://ibm.biz/knowctr#SSZJPZ/`

  Y el URL corto al tema anterior para crear un URL ligeramente más corto es el siguiente URL (El símbolo ⇒ indica una continuación de línea):

  ```
  http://ibm.biz/knowctr#SSZJPZ_11.3.0/com.ibm.swg.im.iis.common.doc/⇒
  common/accessingiidoc.html
  ```

Cambiar los enlaces de ayuda para que hagan referencia a la documentación instalada localmente

IBM Knowledge Center contiene la versión más actualizada de la documentación. Sin embargo, puede instalar una versión local de la documentación como un centro de información y configurar los enlaces de ayuda para que apunten a él. Un centro de información local es útil si su empresa no proporciona acceso a Internet.

Siga las instrucciones de instalación que vienen con el paquete de instalación del centro de información para instalarlo en el sistema que elija. Después de instalar e iniciar el centro de información, puede utilizar el mandato **iisAdmin** en el sistema de la capa de servicios para cambiar la ubicación de la documentación a la que hacen referencia la tecla F1 y los enlaces de ayuda del producto. (El símbolo ⇒ indica una continuación de línea):

Windows

```
vía_instalación_IS\ASBServer\bin\iisAdmin.bat -set -key ⇒
com.ibm.iis.infocenter.url -value http://<host>:<puerto>/help/topic/
```

AIX Linux

```
vía_instalación_IS/ASBServer/bin/iisAdmin.sh -set -key ⇒
com.ibm.iis.infocenter.url -value http://<host>:<puerto>/help/topic/
```

Donde `<host>` es el nombre del sistema donde está instalado el centro de información y `<puerto>` es el número de puerto para el centro de información. El número de puerto predeterminado es 8888. Por ejemplo, en un sistema llamado `server1.example.com` que utilice el puerto predeterminado, el valor del URL sería `http://server1.example.com:8888/help/topic/`.

Obtener la documentación en PDF y en copia impresa

- Los manuales en archivos PDF están disponibles en línea y puede accederse a ellos desde este documento de soporte: https://www.ibm.com/support/docview.wss?uid=swg27008803&wv=1.
- También puede solicitar publicaciones de IBM en formato impreso en línea o a través de su representante local de IBM. Para solicitar publicaciones en línea, vaya al Centro de Publicaciones de IBM en http://www.ibm.com/e-business/linkweb/publications/servlet/pbi.wss.

Apéndice F. Cómo aportar comentarios sobre la documentación del producto

Puede aportar valiosos comentarios en relación a la documentación de IBM.

Sus comentarios ayudarán a IBM a ofrecer información de calidad. Puede utilizar cualquiera de los métodos siguientes para enviar sus comentarios:

- Para proporcionar un comentario acerca de un tema del IBM Knowledge Center que está alojado en el sitio web de IBM, inicie la sesión y pulse el botón **Añadir comentario** en la parte inferior del tema. Los comentarios enviados de esta manera serán visibles para todos los usuarios.
- Para enviar un comentario acerca de un tema del IBM Knowledge Center a IBM y que ningún otro usuario pueda ver, inicie la sesión y pulse en el enlace **Comentarios** en la parte inferior del IBM Knowledge Center.
- Envíe sus comentarios utilizando el formulario de comentarios del lector que encontrará en www.ibm.com/software/awdtools/rcf/.
- Envíe sus comentarios por correo electrónico a comments@us.ibm.com. Incluya el nombre y el número de versión del producto, así como el nombre y el número de pieza de la información (si es pertinente). Si su comentario es sobre un texto específico, incluya la ubicación del texto (por ejemplo, un título, un número de tabla o un número de página).

Avisos y marcas registradas

Esta información ha sido desarrollada para productos y servicios ofrecidos en los Estados Unidos. Este material puede estar disponible en IBM en otros idiomas. Sin embargo, es posible que deba tener una copia del producto o de la versión del producto en ese idioma para poder acceder al mismo.

Avisos

Es posible que IBM no ofrezca en otros países los productos, servicios o características que se describen en este documento. Póngase en contacto con el representante local de IBM para obtener información acerca de los productos y servicios que actualmente están disponibles en su localidad. Cualquier referencia a un producto, programa o servicio de IBM no implica ni establece que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patente pendientes de aprobación que cubran temas tratados en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a la siguiente dirección:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 EE.UU.

Para realizar consultas relativas a la información de 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 en el que las disposiciones en él expuestas sean incompatibles con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍAS DE NINGÚN TIPO, NI EXPLÍCITAS NI IMPLÍCITAS, INCLUYENDO, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO INFRACCIÓN, COMERCIALIZACIÓN E IDONEIDAD PARA UN PROPÓSITO DETERMINADO. Algunas legislaciones no contemplan la declaración de limitación de garantías, ni implícitas ni explícitas, en determinadas transacciones, por lo que cabe la posibilidad de que esta declaración no se aplique en su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. La información incluida en este documento está sujeta a cambios periódicos, que se

incorporarán en nuevas ediciones de la publicación. IBM puede realizar en cualquier momento mejoras o cambios en los productos o programas descritos en esta publicación sin previo aviso.

Cualquier referencia hecha en esta información a sitios web que no sean de IBM se proporciona únicamente para su comodidad y no debe considerarse en modo alguno como una aprobación de dichos sitios web. Los materiales de estos sitios web no forman parte de los materiales de este producto de IBM y el uso que haga de estos sitios web es de la entera responsabilidad del usuario.

IBM puede utilizar o distribuir cualquier información que se le facilite de la manera que considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciatarios de este programa que deseen obtener información acerca del mismo con el fin de: (i) intercambiar la información entre los programas creados independientemente y otros programas (incluido éste) y (ii) utilizar mutuamente la información que se ha intercambiado, deben ponerse en contacto con:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 EE.UU.

Esta información puede estar disponible, sujeta a los términos y condiciones pertinentes, incluido en algunos casos el pago de una cantidad determinada.

IBM proporciona el programa bajo licencia descrito en este documento, y todo el material bajo licencia disponible para el mismo, bajo los términos del Acuerdo de cliente de IBM, el Acuerdo acuerdo internacional de licencia de programa de IBM o cualquier otro acuerdo equivalente entre las partes.

Los datos de rendimiento incluidos en este documento se determinaron en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse efectuado en sistemas a nivel de desarrollo, y no existe ninguna garantía de que dichas mediciones sean las mismas en sistemas de disponibilidad general. Además, es posible que algunas mediciones se hayan estimado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables a su entorno específico.

La información relacionada con productos no de IBM se ha obtenido de los suministradores de dichos productos, de sus anuncios publicados o de otras fuentes de información pública disponibles. IBM no ha probado dichos productos y no puede confirmar la precisión del rendimiento, la compatibilidad ni ninguna otra afirmación relacionada con productos que no son de IBM. Las consultas acerca de las prestaciones de los productos que no son de IBM deben dirigirse a los suministradores de tales productos.

Todas las declaraciones relativas a la dirección o intención futura de IBM están sujetas a cambios o anulación sin previo aviso y representan únicamente metas y objetivos.

Esta información se suministra sólo con fines de planificación. La presente información esta sujeta a cambios antes de que los productos que en ella se describen estén disponibles.

Esta información contiene ejemplos de datos e informes utilizados en las operaciones de negocios diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con los nombres y direcciones utilizados por una empresa real es totalmente casual.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en idioma de origen, que ilustra las técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier forma, sin pagar a IBM, con la finalidad de desarrollar, utilizar, comercializar o distribuir programas de aplicación que se ajusten a la interfaz de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas de ejemplo. Estos ejemplos no se han probado bajo todas las condiciones posibles. Por lo tanto, IBM no puede garantizar ni dar por sentada la fiabilidad, capacidad de servicio o funcionamiento de esos programas. Los programas de ejemplo se suministran "TAL CUAL", sin garantía de ninguna clase. IBM no se hará responsable de los daños que puedan derivarse del uso de los programas de ejemplo.

Cada copia, parcial o completa, de estos programas de ejemplo o cualquier trabajo derivado, debe incluir un aviso de copyright como el siguiente:

© (el nombre de su empresa) (año). Partes de este código provienen 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 copia software, es posible que las fotografías y las ilustraciones en color no aparezcan.

Consideraciones sobre la política de privacidad

Los productos de software de IBM, incluidas las soluciones de software como servicio, (“Ofertas de software”), pueden utilizar cookies u otras tecnologías para recopilar información sobre el uso de productos, para ayudar a mejorar la experiencia del usuario final, para personalizar las interacciones con el usuario final o para otros fines. En muchos casos, las Ofertas de software no recopilan información de identificación personal. Algunas de nuestras Ofertas de software pueden ayudarle a recopilar información de identificación personal. Si esta Oferta de software utiliza cookies para recopilar información de identificación personal, la información específica sobre el uso de cookies por parte de esta oferta se expone más abajo.

Dependiendo de las configuraciones desplegadas, esta Oferta de software puede utilizar cookies de sesión o persistentes. Si un producto o componente no está en la lista, ese producto o componente no utiliza cookies.

Tabla 4. Uso de cookies de los productos y componentes de InfoSphere Information Server

Módulo de producto	Componente o característica	Tipo de cookie que se utiliza	Recopilar estos datos	Finalidad de los datos	Inhabilitación de las cookies
Cualquiera (parte de la instalación de InfoSphere Information Server)	Consola web de InfoSphere Information Server	• Sesión • Persistente	Nombre de usuario	• Gestión de sesiones • Autenticación	No se pueden inhabilitar

Tabla 4. Uso de cookies de los productos y componentes de InfoSphere Information Server (continuación)

Módulo de producto	Componente o característica	Tipo de cookie que se utiliza	Recopilar estos datos	Finalidad de los datos	Inhabilitación de las cookies
Cualquiera (parte de la instalación de InfoSphere Information Server)	InfoSphere Metadata Asset Manager	• Sesión • Persistente	Ninguna información de identificación personal	• Gestión de sesiones • Autenticación • Usabilidad de usuario mejorada • Configuración de inicio de sesión único	No se pueden inhabilitar
InfoSphere DataStage	Etapa Big Data File	• Sesión • Persistente	• Nombre de usuario • Firma digital • ID de sesión	• Gestión de sesiones • Autenticación • Configuración de inicio de sesión único	No se pueden inhabilitar
InfoSphere DataStage	Etapa XML	Sesión	Identificadores internos	• Gestión de sesiones • Autenticación	No se pueden inhabilitar
InfoSphere DataStage	Consola de operaciones de IBM InfoSphere DataStage and QualityStage	Sesión	Ninguna información de identificación personal	• Gestión de sesiones • Autenticación	No se pueden inhabilitar
InfoSphere Data Click	Consola web de InfoSphere Information Server	• Sesión • Persistente	Nombre de usuario	• Gestión de sesiones • Autenticación	No se pueden inhabilitar
InfoSphere Data Quality Console		Sesión	Ninguna información de identificación personal	• Gestión de sesiones • Autenticación • Configuración de inicio de sesión único	No se pueden inhabilitar
InfoSphere QualityStage Standardization Rules Designer	Consola web de InfoSphere Information Server	• Sesión • Persistente	Nombre de usuario	• Gestión de sesiones • Autenticación	No se pueden inhabilitar
InfoSphere Information Governance Catalog		• Sesión • Persistente	• Nombre de usuario • Identificadores internos • Estado del árbol	• Gestión de sesiones • Autenticación • Configuración de inicio de sesión único	No se pueden inhabilitar
InfoSphere Information Analyzer	Etapa Reglas de datos en el cliente del Diseñador de InfoSphere DataStage and QualityStage	Sesión	ID de sesión	Gestión de sesiones	No se pueden inhabilitar

Si las configuraciones desplegadas para esta Oferta de software le ofrecen como cliente la posibilidad de recopilar información de identificación personal de los usuarios finales mediante cookies y otras tecnologías, debe buscar asesoramiento jurídico sobre la legislación aplicable a dicha recopilación de datos, incluidos los requisitos de notificación y consentimiento.

Para obtener más información sobre el uso de diversas tecnologías, incluidas las cookies, para estos fines, consulte la Política de privacidad de IBM en http://www.ibm.com/privacy, la sección "Cookies, balizas web y otras tecnologías" de la Declaración de privacidad en línea de IBM en http://www.ibm.com/privacy/details y la "Declaración de privacidad de productos de software y software como servicio de IBM" (en inglés) en http://www.ibm.com/software/info/product-privacy.

Marcas registradas

IBM, el logotipo de IBM e ibm.com son marcas comerciales o marcas registradas de International Business Machines Corp., registradas en muchas jurisdicciones en todo el mundo. Otros nombres de productos y servicios pueden ser marcas registradas de IBM u otras empresas. Encontrará una lista actual de las marcas registradas de IBM en el sitio web www.ibm.com/legal/copytrade.shtml.

Los términos siguientes son marcas comerciales o marcas registradas de otras empresas:

Adobe es una marca registrada de Adobe Systems Incorporated en los Estados Unidos y/o en otros países.

Intel e Itanium son marcas comerciales o marcas registradas de Intel Corporation o sus filiales en los Estados Unidos y otros países.

Linux es una marca registrada de Linus Torvalds en los Estados Unidos y/ en otros países.

Microsoft, Windows y Windows NT son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

UNIX es una marca registrada de The Open Group en los Estados Unidos y en otros países.

Java™ y todas las marcas registradas y logotipos basados en Java son marcas comerciales o marcas registradas de Oracle y/o sus filiales.

El Servicio de correos de Estados Unidos (United States Postal Service) es propietario de las siguientes marcas registradas: CASS, CASS Certified, DPV, LACS^{Link}, ZIP, ZIP + 4, ZIP Code, Post Office, Postal Service, USPS y United States Postal Service. IBM Corporation es un licenciatario no exclusivo de DPV y LACS^{Link} del Servicio de correos de Estados Unidos.

Otros nombres de empresas, productos y servicios pueden ser marcas comerciales o marcas de servicio de terceros.

Índice

Impreso en España

SC43-1235-00