IBM InfoSphere DataStage and QualityStage
Versión 11 Release 3

Guía para la integración de código Java

SC43-1238-00

IBM InfoSphere DataStage and QualityStage
Versión 11 Release 3

Guía para la integración de código Java

SC43-1238-00

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 73.

Contenido

Capítulo 1. Integración de código Java (etapa Java Integration)

Cuando utilice IBM® InfoSphere DataStage para invocar código Java, puede elegir entre una colección de opciones de conectividad. Para la mayoría de los nuevos trabajos, utilice la etapa Java Integration, que ofrece mejor funcionalidad y rendimiento.

La etapa Java Integration puede utilizarse en las siguientes topologías:

- Como origen con uno o más enlaces de salida
- Como destino con uno o más enlaces de entrada y cero o más enlaces de rechazo
- Como transformador con uno o más enlaces de entrada, y uno o más enlaces de salida o de rechazo
- Como etapa Lookup con un enlace de salida de referencia

La etapa Java Integration es una de varias etapas diferentes que invoca código Java. Además de la etapa Java Integration, están disponibles las siguientes etapas:

- Etapa Java Client
- Etapa Java Transformer

Si desea integrar código Java en un trabajo de servidor, debe utilizar una de las etapas anteriores en lugar de la etapa Java Integration.

Si tiene trabajos que utilizan las etapas anteriores y desea utilizar la etapa Java Integration, utilice la Herramienta de migración de conectores para migrar trabajos para que utilicen conectores.

Capítulo 2. Escribir código Java para utilizar en los trabajos (de la etapa Java Integration)

Puede utilizar la etapa Java Integration para integrar su código en su diseño de trabajo escribiendo su código Java utilizando la API de la etapa Java Integration. La etapa Java Integration API define interfaces y clases para escribir código Java que se pueden invocar desde dentro de InfoSphere DataStage y los trabajos paralelos QualityStage.

Información relacionada:

API Java Pack
Pulse el enlace para ver la información de Javadoc para la API de Java Pack. Consulte también la información de Javadoc para la API de la etapa Java Integration.

Configurar el entorno de desarrollo (etapa Java Integration)

Debe configurar el entorno de desarrollo antes de crear el código Java.

Procedimiento

1. Instale Java 6 SDK.
2. Copie `ccjava-api.jar` en el espacio de trabajo, ya que necesitará el archivo para compilar el código Java. El archivo `ccjava-api.jar` está disponible en las ubicaciones siguientes:
   - `<ISDIR>/Server/DSComponents/bin`, si está utilizando una máquina de la capa de motor de Information Server.
   - `<ISDIR>/Clients/Samples/Connectors/JavaIntegration_Samples.zip`, si utiliza una máquina de la capa de motor de cliente de Information Server.

Instalación de documentos y ejemplos de API (etapa Java Integration)

Los documentos y ejemplos de la API de la etapa Java Integration se instalan en el sistema donde ha instalado la capa de cliente de Information Server.

Los dos siguientes documentos de la API están instalados en la ubicación `<ISInstall directory>\Clients\Samples\Conectores`:

- **JavaIntegration_API_Document.zip**: este archivo contiene el Javadoc para la API de la etapa Java Integration.
- **JavaIntegration_Samples.zip**: este archivo contiene los ejemplos de la API para la API de la etapa Java Integration.

Implementación de métodos abstractos de la clase Processor

El código Java debe implementar una subclase de la clase `Processor`. La clase `Processor` consta de métodos que se invocan mediante la etapa Java Integration. Cuando se inicia un trabajo que incluye la etapa Java Integration, la etapa crea una instancia de la clase `Processor` y llama a la lógica dentro de las implementaciones de `Processor`.

La clase Processor proporciona la siguiente lista de métodos a los que la etapa Java Integration puede llamar para interactuar con el código Java en tiempo de ejecución del trabajo o en tiempo de diseño.

- getCapabilities()
- validateConfiguration()
- getConfigurationErrors()
- getBeanForInput()
- getBeanForOutput()
- getAdditionalOutputColumns()
- initialize()
- process()
- terminate()
- getColumnMetadataForInput()
- getColumnMetadataForOutput()
- getUserPropertyDefinitions()

Como mínimo, el código Java debe implementar los dos siguientes métodos abstractos.

- public abstract boolean validateConfiguration(Configuration configuration, boolean isRuntime) throws Exception;
- public abstract void process() throws Exception;

El siguiente ejemplo muestra la implementación simple de la etapa Peek que imprime valores de columna de registro en el registro de trabajo que se pueden ver en el cliente del Director. El código presupone que hay un solo enlace de entrada.

```
package samples;

import com.ibm.is.cc.javastage.api.*;

public class SimplePeek extends Processor
{
  private InputLink m_inputLink;

  public boolean validateConfiguration(
    Configuration configuration, boolean isRuntime)throws Exception
  {
    if (configuration.getInputLinkCount() != 1)
    {
      // este código de ejemplo presupone que la etapa tiene 1 enlace de entrada.
      return false;
    }

    m_inputLink = configuration.getInputLink(0);

    return true;
  }

  public void process() throws Exception
  {
    do
    {
      InputRecord inputRecord = m_inputLink.readRecord();
      if (inputRecord == null)
      {
         // No hay más entradas. El código debe volver del método process ().
         break;
      }
```

```
      for (int i = 0; i < m_inputLink.getColumnCount(); i++)
      {
         Object value = inputRecord.getValue(i);
         Logger.information(value.toString());
      }
    }
    while (true);
  }
}
```

La etapa Java Integration llama al método `validateConfiguration()` para especificar la configuración actual (número y tipos de enlaces), y los valores para las propiedades de usuario. El código Java debe validar una configuración determinada y las propiedades de usuario, y devolver `false` para la etapa Java Integration si hay problemas con ellos. En el ejemplo anterior, puesto que este código presupone una etapa que tiene un solo enlace de entrada, éste comprueba el número de enlaces de entrada y devuelve `false` si la configuración de la etapa no cumple este requisito.

```
if (configuration.getInputLinkCount() != 1)
{
  // este código de ejemplo presupone que la etapa tiene 1 enlace de entrada.
  return false;
}
```

La interfaz `Configuration` define métodos que se utilizan para obtener la configuración de la etapa actual (número y tipos de enlaces) y los valores para las propiedades del usuario. El método `getInputLinkCount()` se utiliza para obtener el número de enlaces de entrada conectados a esta etapa.

Si la configuración de la etapa es aceptada por el código Java, guarda la referencia a un objeto para el proceso posterior de `InputLink` y devuelve `true` a la etapa Java Integration.

```
m_inputLink = configuration.getInputLink(0);
        return true;
      }
```

Después de que la configuración de la etapa se verifica mediante el código Java, puede interactuar con las etapas conectadas en su trabajo. El método `process()` es un punto de entrada para procesar registros desde el enlace de entrada o hacia el enlace de salida. Cuando una fila está disponible en cualquiera de los enlaces de entrada de la etapa (si hay alguno y cualquiera que sea el número de enlaces de salida), la etapa Java Integration llama a este método si el trabajo no finaliza. El código Java debe consumir todas las filas de los enlaces de entrada de la etapa.

Al llamar al método `readRecord()` de la interfaz `InputLink`, el código Java puede consumir una fila desde el enlace de entrada. Devuelve un objeto que implementa la interfaz `InputRecord`. La interfaz `InputRecord` define métodos que se utilizan para obtener datos de columna de un registro de fila consumido.

```
InputRecord inputRecord = m_inputLink.readRecord();
if (inputRecord == null)
{
  // No hay más entradas. El código debe volver del método process ().
  break;
}
```

Después de que su código Java consuma un registro de fila del enlace de entrada de la etapa, el código Java puede obtener los valores de registro de la columna

invocando el método `getValue(int columnIndex)` de la interfaz `InputRecord`. `getColumnCount()` en `InputLink` devuelve el número de columnas que existen en este enlace de entrada.

```
for (int i = 0; i < m_inputLink.getColumnCount(); i++)
{
  Object value = inputRecord.getValue(i);
```

Por último, cada valor de columna se graba en el registro de trabajo llamando al método `information()` de la clase `Logger`. La clase `Logger` permite al código Java grabar los datos en el registro de trabajo con los niveles de registro especificados. El código siguiente graba la representación de serie de cada valor de columna en un registro de trabajo.

```
Logger.information(value.toString());
```

Compilar el código Java (etapa Java Integration)

Después de crear el código Java, debe compilar el código Java y, opcionalmente, puede crear un archivo JAR para el despliegue.

Procedimiento

1. Compile el código Java con `ccjava-api.jar`, utilizando el mandato siguiente:
   ```
   javac -cp .;.\jars\ccjava-api.jar samples\SimplePeek.java
   ```
2. Cree un archivo jar para el despliegue, utilizando el mandato siguiente:
   ```
   jar -cvf .\jars\samples.jar samples\SimplePeek.class
   ```

Ejecución del código Java en el motor paralelo

El código Java es invocado por una o más instancias de máquina virtual Java en el entorno de motor paralelo de DataStage. El conductor es un proceso DataStage inicial que crea una instancia de máquina virtual Java individual y carga el código Java y otro código de etapa Connector basado en Java en el trabajo. Los procesadores en los nodos reproductor son bifurcados por cada una de las etapas del trabajo y son los procesos reales que se asocian con las etapas. Se crea una instancia de máquina virtual Java para cada proceso reproductor que está asociado con la etapa Java Integration donde se ejecuta el código Java.

Acceso a la configuración de la etapa (etapa Java Integration)

Una instancia de la interfaz `Configuration` define la configuración de la etapa actual, como el número y tipo de enlaces y los valores para las propiedades definidas por el usuario que se especifican en el trabajo.

Para ver los detalles de los métodos disponibles que proporciona la interfaz `Configuration`, consulte la información de Javadoc para la API de la etapa Java Integration.

Métodos para acceder a las configuraciones de enlaces

- `getDataChannel()`
- `getLinks()`
- `getInputLinks()`
- `getInputLinkCount()`
- `getInputLinks()`
- `getOutputLink()`

- getOutputLinkCount()
- getOutputLinks()
- getRejectOutputLink()
- getRejectLinkCount()
- getStreamOutputLink()
- getStreamOutputLinkCount()

Métodos para acceder a las configuraciones de nodos
- getNodeCount()
- getNodeNumber()

Método para acceder a las propiedades de etapa definidas por el usuario
- getUserProperties()

Métodos para acceder a los servicios de transferencia de datos
- getDataChannel()

Declaración de las capacidades del código Java (etapa Java Integration)

La clase Capabilities define las capacidades de su código Java encapsulando una lista de atributos y parámetros.

A continuación se proporciona una lista de los métodos disponibles que la clase Capabilities proporciona. Para obtener información detallada sobre los métodos, consulte la información de Javadoc para la documentación de la API de la etapa Java Integration.
- getMinimumInputLinkCount()
- getMaximumInputLinkCount()
- getMinimumOutputStreamLinkCount()
- getMaximumOutputStreamLinkCount()
- getMinimumRejectLinkCount()
- getMaximumRejectLinkCount()
- getColumnTransferBehavior()
- isWaveGenerator()
- isRunOnConductor()
- setMinumumInputLinkCount()
- setMaximumInputLinkCount()
- setMinimumOutputStreamLinkCount()
- setMaximumOutputStreamLinkCount()
- setMinimumRejectLinkCount()
- setMaximumRejectLinkCount()
- setIsWaveGenerator()
- setColumnTransferBehavior()
- setIsRunOnConductor()

Justo después de instanciar el código `Processor`, la etapa Java Integration invoca el método `getCapabilities()` en el código `Processor` para obtener su objeto `Capabilities` asociado para determinar si el código Java se puede ejecutar en el trabajo actual.

Al alterar temporalmente el método `getCapabilities()` en la clase `Processor`, el código Java puede personalizar los valores de las capacidades para resolver el código Java y pasarlo a la etapa Java Integration.

El siguiente ejemplo muestra que el código Java sólo acepta el caso de enlace de entrada único.

```
public Capabilities getCapabilities()
{
  Capabilities capabilities = new Capabilities();
  capabilities.setMinimumInputLinkCount(1);
  capabilities.setMaximumInputLinkCount(1);
  capabilities.setMaximumOutputStreamLinkCount(0);
  capabilities.setMaximumRejectLinkCount(0);
  return capabilities;
}
```

El trabajo finaliza si el diseño del trabajo actual no se ajusta a las capacidades especificadas.

El código siguiente proporciona la funcionalidad que es equivalente a la del primer ejemplo. La etapa Java Integration comparará el número de enlaces conectados a la etapa con los límites especificados por la implementación del método `getCapabilities()`. Si el número de enlaces está fuera de los límites especificados, la etapa Java Integration enviará un mensaje adecuado al registro de trabajo y terminará anormalmente el trabajo.

```
package samples;

import com.ibm.is.cc.javastage.api.*;

public class ReworkedSimplePeek extends Processor
{
  private InputLink m_inputLink;

  public Capabilities getCapabilities()
  {
    Capabilities capabilities = new Capabilities();
    capabilities.setMinimumInputLinkCount(1);
    capabilities.setMaximumInputLinkCount(1);
    capabilities.setMaximumOutputStreamLinkCount(0);
    capabilities.setMaximumRejectLinkCount(0);
    return capabilities;
  }

  public boolean validateConfiguration(
    Configuration configuration, boolean isRuntime)throws Exception
  {
    m_inputLink = configuration.getInputLink(0);
    return true;
  }

  public void process() throws Exception
  {
    do
{
      InputRecord inputRecord = m_inputLink.readRecord();
      if (inputRecord == null)
      {
```

```
            // No hay más entradas. El código debe volver del método process ().
            break;
         }

         for (int i = 0; i < m_inputLink.getColumnCount(); i++)
         {
            Object value = inputRecord.getValue(i);
            Logger.information(value.toString());
         }
      }
      while (true);
   }
}
```

Lectura de registros desde el enlace de entrada (etapa Java Integration)

La interfaz `InputLink` es una extensión de la interfaz `Link`. Define métodos que se utilizan para interactuar con el enlace de entrada de la etapa correspondiente. Las instancias de una intefaz `InputLink` están disponibles en el objeto `Configuration` que se proporciona como argumento del método `validateConfiguration()`.

Métodos proporcionados por la interfaz Link

- `getColumn()`
- `getColumnCount()`
- `getColumnMetadata()`
- `getLinkIndex()`
- `getUserProperties()`
- `subtractColumnList()`

Métodos proporcionados por la interfaz InputLink

- `GetAssociatedRejectLink()`
- `readRecord()`

Al llamar al método `readRecord()` de una interfaz `InputLink`, el código Java puede consumir una fila desde el enlace de entrada. Devuelve un objeto que implementa la interfaz `InputRecord`.

```
InputRecord inputRecord = m_inputLink.readRecord();
```

La interfaz `InputRecord` es una extensión de la interfaz `Record`. Define métodos que se utilizan para obtener datos de columna de un registro de fila consumido.

Métodos proporcionados por la interfaz InputRecord

- `getObject()`
- `getValue(String columnName)`
- `getValue(int columnIndex)`

En el ejemplo siguiente se muestra cómo recuperar el valor correspondiente a un índice de columna determinado "i" en este registro.

```
Object value = inputRecord.getValue(i);
```

También puede recuperar el valor especificando el nombre de la columna como en el ejemplo siguiente.

```
Object value = inputRecord.getValue("name");
```

Grabación de registros en el enlace de salida (etapa Java Integration)

Para grabar registros en un enlace de salida, el código Java debe crear instancia de un objeto `OutputRecord` utilizando el método `getOutputRecord()` de la interfaz `OutputLink`.

El ejemplo siguiente muestra las implementaciones de la etapa Transforme simple que convierte los textos de las series del registro de consumo en texto en mayúsculas, y luego lo graba en un enlace de salida.

```
package samples;

import com.ibm.is.cc.javastage.api.*;

public class ToUpperTransformer extends Processor
{
  private InputLink m_inputLink;
  private OutputLink m_outputLink;

  public Capabilities getCapabilities()
  {
    Capabilities capabilities = new Capabilities();
    // Establecer número mínimo de enlaces de entrada en 1
    capabilities.setMinimumInputLinkCount(1);
    // Establecer número máximo de enlaces de entrada en 1
    capabilities.setMaximumInputLinkCount(1);
    // Establecer número mínimo de enlaces de secuencia de salida en 1
    capabilities.setMinimumOutputStreamLinkCount(1);
    // Establecer número máximo de enlaces de secuencia de salida en 1
    capabilities.setMaximumOutputStreamLinkCount(1);
    // Establecer número máximo de enlaces de rechazo en 1
    capabilities.setMaximumRejectLinkCount(0);
    return capabilities;
  }

  public boolean validateConfiguration(
    Configuration configuration, boolean isRuntime)
    throws Exception
  {
     // Especificar configuraciones de enlaces actuales.
     m_inputLink = configuration.getInputLink(0);
     m_outputLink = configuration.getOutputLink(0);

     return true;
  }

  public void process() throws Exception
  {
    OutputRecord outputRecord = m_outputLink.getOutputRecord();

    do
    {
      InputRecord inputRecord = m_inputLink.readRecord();
      if (inputRecord == null)
      {
        // No hay más entradas
        break;
      }

      for (int i = 0; i < m_inputLink.getColumnCount(); i++)
      {
        Object value = inputRecord.getValue(columnIndex);
        if (value instanceof String)
        {
          String str = (String)value;
          value = str.toUpperCase();
```

```
        }
        outputRecord.setValue(i, value);
      }
      m_outputLink.writeRecord(outputRecord);
    }
    while (true);
  }
}
```

Para grabar registros en un enlace de salida, el código Java debe crear instancia de un objeto OutputRecord utilizando el método getOutputRecord() de la interfaz OutputLink.

```
OutputRecord outputRecord = m_outputLink.getOutputRecord();
```

Métodos proporcionados por la interfaz OutputRecord

- putObject()
- setValue(String columnName, Object value)
- setValue(int columnIndex, Object value)
- setValueAsString(String columnName, String value)
- setValueAsString(int columnIndex, String value)
- copyColumnsFromInputRecord(InputRecord inputRecord)
- getOfLink()

Después de crear una instancia del objeto OutputRecord, puede establecer el valor para cada columna utilizando el método setValue(String columnName, Object value) o setValue(int índiceColumna, Object value) de la interfaz OutputRecord.

En el ejemplo siguiente se muestra cómo establecer el valor correspondiente a un índice de columna determinado "i" en este registro.

```
outputRecord.setValue(i, value);
```

También puede establecer el valor indicando el nombre de columna de este modo:

```
outputRecord.setValue("name", value);
```

Finalmente, el código Java graba este registro de salida llamando al método writeRecord() de la interfaz OutputLink. La instancia de OutputLink está disponible en el objeto Configuración que se proporciona como argumento del método validateConfiguration().

```
m_outputLink.writeRecord(outputRecord);
```

Métodos proporcionados por la interfaz OutputLink

- getOutputRecord()
- getOutputRecord(InputRecord)
- getRejectRecord(InputRecord)
- writeRecord()
- writeRecord(RejectRecord)
- writeWaveMarker()
- isRcpEnabled()

Rechazo de registros (etapa Java Integration)

Es posible que desee rechazar los registros procedentes de un enlace de entrada porque el registro de entrada no cumpla con los requisitos de su código Java. En este caso, puede considerar la posibilidad de utilizar un enlace de rechazo en el diseño del trabajo y grabar los datos rechazados en este enlace.

Cuando el código Java necesite grabar el registro en un enlace de rechazo, el código Java debe llamar al método `getAssociatedRejectLink()` de la interfaz `InputLink` para obtener una instancia del `OutputLink` asociado con el enlace de entrada en el diseño del trabajo.

```
OutputLink m_rejectLink = m_inputLink.getAssociatedRejectLink();
```

A continuación verá una lista de los métodos proporcionados por una interfaz `RejectRecord`:

- `setErrorText()`
- `setErrorCode()`
- `getOfLink()`

De forma parecida al caso de la grabación de registros en un enlace de salida, el código Java debe crear una instancia de un objeto `RejectRecord` utilizando el método `getRejectRecord()` de la interfaz `OutputLink`. El código Java debe especificar el objeto `EntradaNivel` que va a rechazarse.

```
RejectRecord rejectRecord = m_rejectLink.getRejectRecord(inputRecord);
```

Un enlace de rechazo en su diseño de trabajo puede tener las columnas adicionales "ERRORTEXT" y "ERRORCODE". Su código Java puede establecer los valores para estas columnas adicionales utilizando los métodos `setErrorText()` y `setErrorCode()` de la interfaz `RejectRecord`.

```
rejectRecord.setErrorText("El campo de nombre contiene *");
rejectRecord.setErrorCode(123);
```

Finalmente, puede grabar este registro de rechazo en el enlace de rechazo correspondiente mediante el método `writeRecord(RejectRecord)` de la interfaz `OutputLink`.

```
m_rejectLink.writeRecord(rejectRecord);
```

.

Búsqueda de datos en la modalidad de búsqueda Dispersa

La etapa Java Integration da soporte a la búsqueda dispersa. Puede implementar la lógica de búsqueda en su código Java para leer el registro de entrada y grabar en el registro de salida de uno en uno.

Cuando el conector se utiliza para realizar una operación de búsqueda dispersa, el conector tiene un enlace de entrada, un enlace de salida y un enlace de rechazo opcional. El conector con un solo enlace de referencia de salida se convierte internamente para que contenga un enlace de entrada, un enlace de salida y, opcionalmente, un enlace de rechazo, si la etapa Lookup del trabajo contiene un enlace de rechazo. Debe añadir una implementación en el código Java para ejecutar en modalidad de búsqueda dispersa en el método `process()`.

Para identificar la modalidad de búsqueda en el método `process()`, llame al método `getSparseLookupMode()` para la interfaz Configuration de la siguiente manera:

```
SparseLookupMode m_sparseLookupMode = m_configuration.getSparseLookupMode()
```

El valor de retorno indica la modalidad de búsqueda que está configurada. En el panel Condiciones de la etapa Lookup, la columna Anomalía en la búsqueda indica el comportamiento esperado cuando la operación de búsqueda falla. El código Java debe contener la lógica para crear un bucle para leer repetidamente los datos del enlace de entrada, y grabar en el enlace de salida o el enlace de rechazo. El esquema del enlace de entrada y el enlace de rechazo es el mismo que el esquema del enlace de entrada para la etapa Lookup. El esquema del enlace de salida es el mismo que el esquema del enlace de referencia de salida para la etapa Java Integration. El código Java debe leer las columnas de clave en el enlace de entrada y grabar los registros de salida que tienen los datos de búsqueda.

Para identificar las columnas de clave en el enlace de entrada, utilice el método `isKey()` para la interfaz ColumnMetadata de la siguiente manera:

```
List<ColumnMetadata> inputColumns = m_inputLink.getColumnMetadata();
ColumnMetadata cm = inputColumns.get(columnIndex);
boolean isKey = cm.isKey();
```

Después de que el código Java graba los registros de salida, la etapa Java Integration y la infraestructura añaden o copian las columnas adicionales de acuerdo con la configuración de correlación en la etapa Lookup para que el esquema del enlace de salida sea el mismo que el esquema del enlace de salida de la etapa Lookup.

Cuando implemente la lógica para la búsqueda dispersa, asegúrese de que el código de usuario llame al método `writeRecord()` para OutputLink o RejectLink, antes de llamar al método `InputLink.readRecord()` para captar el siguiente registro de entrada. La infraestructura también graba en la columna para salida de búsqueda dispersa junto con el conector. La infraestructura de conector retiene el cursor para el registro de entrada que el código de usuario lee y cuando se llama al método `InputLink.readRecord()`, cambia la posición del cursor a un registro diferente. Para grabar el registro de salida correctamente, se debe llamar al método `writeRecord()` antes que al método `InputLink.readRecord()` para el siguiente registro.

No es obligatorio implementar un comportamiento esperado para la búsqueda dispersa en caso de anomalía en la búsqueda. Sin embargo, es una buena práctica implementar un comportamiento esperado, ya que el código de usuario puede comprobar el valor de retorno para los métodos `getSparseLookupMode()` y procesar los registros que han fallado. El siguiente es el comportamiento esperado para cada modalidad:

Tabla 1. Comportamiento esperado para la búsqueda dispersa en caso de anomalía en la búsqueda

Modalidad	**Comportamiento esperado**
Continuar	La operación de búsqueda continúa después de que los datos vacíos se envían a columnas de los datos consultados. Puede implementar código Java para pasar registros vacíos o registros sin valores al método `writeRecord()`.

Tabla 1. Comportamiento esperado para la búsqueda dispersa en caso de anomalía en la búsqueda (continuación)

Modalidad	Comportamiento esperado
Descartar	Descartar los registros que no coinciden con los registros de la tabla de búsqueda. Cuando el código hace la siguiente llamada al método `readRecord()`, el registro de entrada se descarta automáticamente. No hay nada que implementar en el código Java.
Finalización anómala	El trabajo finaliza cuando la búsqueda falla. El código Java emite una excepción cuando el trabajo finaliza.
Rechazar	Los registros que no coinciden con los registros de la tabla de búsqueda son rechazados. Los registros de entrada se pasan al método `RejectLink.writeRecord()`.

El siguiente ejemplo muestra el método `process()` y la implementación del comportamiento esperado para la búsqueda dispersa en caso de anomalía en la búsqueda:

```
public void process() throws Exception
{

    if (m_sparseLookupMode == Configuration.SparseLookupMode.NOT_SPARSE)
    {
         // implementación normal para process()
         // ...
         // ...
    }
    else
    {
         // implementación de búsqueda dispersa
         do
         {
            InputRecord inputRecord = m_inputLink.readRecord();
            if (inputRecord == null)
            {
               // No hay más entradas
               return;
            }

            OutputRecord outputRecord = m_outputLink.getOutputRecord();
            OutputRecord emptyRecord = m_outputLink.getOutputRecord();

            // -- realizar la búsqueda y rellenar el registro de salida
            boolean fLookupFail = lookup(inputRecord, outputRecord);

            if (!fLookupFail)
            {
               // búsqueda satisfactoria - poner el registro de salida rellenado
               // como desee
               m_outputLink.writeRecord(outputRecord);
            }
            else
            {
               switch(m_sparseLookupMode)
               {
               case Configuration.SparseLookupMode.CONTINUE:
                  // modalidad "Continuar" - enviar el registro vacío
                  m_outputLink.writeRecord(emptyRecord);
```

```
                    break;
                case Configuration.SparseLookupMode.DROP:
                    // modalidad "Descartar" - no hacer nada y leer el siguiente
                    // registro.
                    break;
                case Configuration.SparseLookupMode.FAIL:
                    // modalidad "Finalización anómala" - emitir excepción para
                    // abortar el trabajo
                    throw new ConnectorException(9998, "Lookup failed.");
                case Configuration.SparseLookupMode.REJECT:
                    // modalidad "Rechazar" - copiar registro de entrada en enlace
                    // de rechazo
                    RejectRecord rejectRecord = m_rejectLink.getRejectRecord
(inputRecord);
                    m_rejectLink.writeRecord(rejectRecord);
                    break;
                }
            }
        }
        while (true);
}
}
```

En el código de ejemplo, los usuarios implementan su propia lógica de búsqueda en la función `lookup()` consultando los datos de entrada en el argumento `inputRecord` y llenan el argumento `outputReocrd` con los datos extraídos. El valor de retorno indica si la búsqueda ha sido satisfactoria o no.

La modalidad de búsqueda dispersa se ejecuta con la topología de enlace especial. En la modalidad de búsqueda dispersa, la etapa Java Integration no impone la comprobación de capacidad en el número de enlaces de entrada y de rechazo.

Tipos de datos (etapa Java Integration)

InfoSphere DataStage da soporte a un conjunto de tipos de datos que son diferentes de los tipos de datos de Java.

En un enlace de salida, en el que columnas de DataStage se establecen a partir de los tipos de datos Java que son producidos por la etapa Java Integration, la etapa Java Integration convierte los tipos de datos Java en tipos de datos de InfoSphere DataStage. En cambio, en un enlace de entrada, donde las propiedades o las columnas de Java Bean se establecen a partir de las columnas de DataStage, los tipos de datos de InfoSphere DataStage se convierten en tipos de datos Java.

Conversiones de tipos de datos de DataStage a tipos de datos Java (etapa Java Integration)

En un enlace de entrada en el que se han establecido tipos de Java a partir de las columnas de DataStage consumidas por la etapa Java Integration, la etapa Java Integration convierte los tipos de datos de InfoSphere DataStage en tipos de datos Java.

La siguiente tabla muestra las normas de correlación entre los tipos de datos InfoSphere DataStage y los tipos de datos Java.

Tabla 2. Los tipos de datos de InfoSphere DataStage y sus equivalentes tipos de datos de Java

Tipos de datos de InfoSphere DataStage	Tipos de datos de Java
BigInt	java.math.BigInteger

Tabla 2. Los tipos de datos de InfoSphere DataStage y sus equivalentes tipos de datos de Java (continuación)

Tipos de datos de InfoSphere DataStage	Tipos de datos de Java
Binario	byte[]
Bit	int/java.lang.Integer o boolean/ java.lang.Boolean **Nota:** boolean/java.lang.Boolean sólo es aplicable para bean de Java o el caso UDF
Char	java.lang.String
Fecha	java.sql.Date
Decimal	java.math.BigDecimal
Doble	double/java.lang.Double
Flotante	float/java.lang.Float
Entero	long/java.lang.Long
LongNVarChar	java.lang.String
LongVarBinary	byte[]
LongVarChar	java.lang.String
NChar	java.lang.String
Numérico	java.math.BigDecimal
NVarChar	java.lang.String
Real	java.lang.Float
SmallInt	java.lang.Integer
Hora	java.sql.Time
Indicación de fecha y hora	java.sql.Timestamp
TinyInt	java.lang.Short
VarBinary	byte[]
VarChar	java.lang.String

Conversiones de tipos de datos de Java a tipos de datos DataStage (etapa Java Integration)

En un enlace de salida, en el que columnas de DataStage se establecen a partir de los tipos de Java que son producidos por la etapa Java Integration, la etapa Java Integration convierte los tipos de datos Java en tipos de datos de InfoSphere DataStage.

De igual manera, una vez importados los metadatos a través de la etapa Java Integration, los tipos de datos Java se convierten en tipos de datos de InfoSphere DataStage. La siguiente tabla muestra las normas de correlación entre los tipos de datos InfoSphere DataStage y los tipos de datos Java.

Tabla 3. Los tipos de datos Java y sus tipos de datos de InfoSphere DataStage equivalentes

Tipo de datos Java	Tipo de datos de InfoSphere DataStage
boolean/java.lang.Boolean **Nota:** sólo aplicable para bean Java o el caso UDF	Bit
short/java.lang.Short	TinyInt

Tabla 3. Los tipos de datos Java y sus tipos de datos de InfoSphere DataStage equivalentes (continuación)

Tipo de datos Java	Tipo de datos de InfoSphere DataStage
int/java.lang.Integer	SmallInt o Bit
long/java.lang.Long	Entero
java.math.BigInteger	BigInt
float/java.lang.Float	Real o flotante
double/java.lang.Double	Doble
byte[]	Binary, VarBinary o LongVarBinary
java.lang.String	Char, VarChar, NVarChar, LongNVarChar, LongVarChar o NChar
java.sql.Date	Fecha
java.sql.Time	Hora
java.sql.Timestamp	Indicación de fecha y hora
java.math.BigDecimal	Decimal o Numérico

Tiempo con resolución de microsegundos (etapa Java Integration)

El tipo de datos de tiempo de DataStage Parallel Engine puede tener una resolución de microsegundos, pero `java.sql.Time` sólo tiene una resolución de milisegundos. Como resultado, los 3 últimos dígitos de microsegundos se truncan. El truncamiento también se produce cuando se propagan columnas de un enlace de entrada a un enlace de salida. Para evitar el truncamiento, la API de la etapa Java Integration proporciona una clase `com.ibm.is.cc.javastage.api.TimeMicroseconds` que conserva el valor de tiempo para la especificación de microsegundos.

La clase `com.ibm.is.cc.javastage.api.TimeMicroseconds` hereda `java.sql.Time` y tiene resolución de microsegundos, así como de hora, minuto y segundo. Los microsegundos no se truncan al propagar columnas de un enlace de entrada a un enlace de salida. Si desea grabar los datos con resolución de microsegundos para el enlace de salida, utilice esta clase.

Para obtener más información sobre la clase `TimeMicroseconds`, consulte la información de Javadoc para la documentación de la API de la etapa Java Integration.

Recuperación de metadatos de columna en el enlace (etapa Java Integration)

La interfaz `ColumnMetadata` define métodos que se utilizan para obtener los metadatos, como nombre de columna, tipo de datos y longitud asociado a cada columna en el enlace. Los metadatos de columnas que están asociados al enlace Information Server DataStage® se pueden recuperar invocando el método `Link.getColumnMetadata()`, `Link.getColumn(int)` y `Link.getColumn(String)`.

Métodos proporcionados por la interfaz ColumnMetadata

- `getDataElementName()`
- `getDerivation()`

- getDescription()
- getDisplaySize()
- getIndex()
- getName()
- getNativeType()
- getPrecision()
- getScale()
- getSQLType()
- getSQLTypeName()
- getType()
- hasMicrosecondResolution()
- isKey()
- isNullable()
- isSigned()
- isUnicode()

En el siguiente ejemplo se recupera el nombre y el tipo de SQL de la primera columna en el enlace.

```
ColumnMetadata columnMetadata = m_inputLink.getColumn(0);
String columnName = columnMetadata.getName();
int sqlType = columnMetadata.getSQLType();
```

Utilización de propiedades definidas por el usuario (etapa Java Integration)

Puede utilizar el código Java para definir propiedades personalizadas y utilizar estos valores de propiedades en el código Java.

En el momento del diseño de trabajo, el editor de la etapa Java Integration llama al método `getUserPropertyDefinitions()` en la clase `Processor` para obtener una lista de definiciones de propiedades definidas por el usuario y, a continuación, mostrar las propiedades en el panel del editor para permitir a los usuarios especificar el valor de serie para cada propiedad.

El siguiente ejemplo muestra la implementación de ejemplo del método `getUserPropertyDefinitions()`.

```
public List<propertyDefinition> getUserPropertyDefinitions()
{
    List<PropertyDefinition> list = new ArrayList<PropertyDefinition>();
    propList.add(new PropertyDefinition
  ("NumOfRecords",
  "10",
    "Número de registros",
    "Especifica el número de registros que se deben generar.",
    PropertyDefinition.Scope.STAGE));

    propList.add(new PropertyDefinition
  ("WaveCount",
      "5",
      "Recuento de oleadas",
     "Especifica el número de registros que se deben procesar
    antes de grabar marcador de fin de oleada." ,
      PropertyDefinition.Scope.OUTPUT_LINK_ONLY));
    return list;
 }
```

El método `getUserProperties()` de la interfaz `Configuration` devuelve un conjunto de propiedades definidas por el usuario que se forma como pareja clave=par. El método `getUserProperties()` de la interfaz `Link` devuelve un conjunto de propiedades definidas por el usuario que se forma como pareja clave=par.

El código siguiente obtiene el valor de la propiedad **NumOfRecords** que se establece como propiedades de etapa, y la propiedad **WaveCount**, que se establece como propiedades de enlace.

```
public boolean validateConfiguration(
  Configuration configuration, boolean isRuntime) throws Exception
{
  // Especificar configuraciones de enlaces actuales.
  m_outputLink = configuration.getOutputLink(0);

  Properties userStageProperties = configuration.getUserProperties();
  String propValue;
  // Obtener el valor de la propiedad de usuario "NumOfRecords".
  // Si no se especifica, utilice el valor 10 predeterminado.
  // El número mínimo de NumOfRecords es 0.
  // El número máximo de NumOfRecords es 100.
  // Si el valor especificado está fuera del rango, se devuelve un error.
  propValue = userStageProperties.getProperty("NumOfRecords");
  if (propValue != null)
  {
    m_numOfRecords = Integer.valueOf(propValue);
  }
  if (m_numOfRecords < 0 || m_numOfRecords > 100)
  {
    m_errors.add("Configure el valor NumOfRecords entre 1 y 100.");
  }

  // Obtener el valor de la propiedad de usuario "WaveCount".
  // Si no se especifica, utilice el valor 5 predeterminado.
  // El número mínimo de WaveCount es 0.
  // El número máximo de WaveCount es 100.
  // Si el valor especificado está fuera del rango, se devuelve un error.
  Properties userLinkProperties = m_outputLink.getUserProperties();
  propValue = userLinkProperties.getProperty("WaveCount");
  if (propValue != null)
  {
    m_waveCount = Integer.valueOf(propValue);
  }
  if (m_waveCount < 0 || m_waveCount > 100)
  {
    m_errors.add("Configure el valor de waveCount entre 1 y 100.");
  }
```

Propagación de columnas en tiempo de ejecución (etapa Java Integration)

InfoSphere DataStage le permite definir parte del esquema y especificar que si el trabajo encuentra columnas adicionales que no estén definidas en los metadatos cuando se ejecute realmente, se adopten dichas columnas adicionales y se propaguen a través del resto del trabajo. Esto se conoce como propagación de columnas de tiempo de ejecución (RCP).

Puede habilitar la propagación de columnas en tiempo de ejecución para un proyecto y establecer enlaces individuales en el separador **Columnas** de la página de salida. Para habilitar la propagación de columnas en tiempo de ejecución, seleccione el recuadro de selección **Propagación de columnas de tiempo de ejecución**.

Cómo añadir columnas extra al enlace de salida

Cuando la propagación de columnas en tiempo de ejecución se habilita en un enlace de salida, el código de usuario puede añadir columnas adicionales a este enlace de salida. Las columnas adicionales se pueden especificar mediante el método `Processor.getAdditionalOutputColumns()`, al que se llama para cada enlace de salida cuyo RCP esté habilitado.

El siguiente ejemplo muestra cómo añadir las columnas denominadas "charCol" e "intCol" a un enlace de salida determinado.

```
public List<ColumnMetadata> getAdditionalOutputColumns
Link outputLink, List<Link> inputLinks,
Properties stageProperties)
{
List<ColumnMetadata> addtionalColumns = new
ArrayList<ColumnMetadata>();
ColumnMetadata charCol = new ColumnMetadataImpl
("charCol",ColumnMetadata.SQL_TYPE_CHAR);
addtionalColumns.add(charCol);
ColumnMetadata intCol = new ColumnMetadataImpl
("intCol",ColumnMetadata.SQL_TYPE_INTEGER);
addtionalColumns.add(intCol);
return addtionalColumns;
}
```

Si desea añadir columnas que se encuentran en el enlace de entrada 0, pero no en el enlace de salida, puede utilizar el método del programa de ayuda denominado `subtractColumnList()` de la interfaz `InputLink` como a continuación.

```
public List<ColumnMetadata> getAdditionalOutputColumns(Link outputLink,
List<Link> inputLinks, Properties stageProperties)
{
return inputLinks.get(0).subtractColumnList(outputLink);^M
}
```

Como alternativa, puede utilizar JavaBean para especificar definiciones de columna en los enlaces de salida en tiempo de ejecución. Si su código de usuario utiliza JavaBean en los enlaces de salida cuyo RCP está habilitado, y tanto al propiedad **Clase JavaBean** como la propiedad **Correlación de columnas** están vacías, la etapa Java Integration crea automáticamente columnas a partir de las propiedades de bean en una clase JavaBean determinada, y las añade a un enlace de salida.

La tabla siguiente resume el tipo de SQL creado por la etapa Java Integration. Puede sobrescribir el tipo SQL de columnas añadiendo manualmente la definición de columna correspondiente en el enlace de salida.

Tipo Java	Tipo de SQL
byte[]	Binario
boolean/java.lang.Boolean	Bit
short/java.lang.Short	TinyInt
int/java.lang.Integer	SmallInt
double/java.lang.Double	Doble
float/java.lang.Float	Flotante
long/java.lang.Long	Entero
java.lang.String	VarChar
java.math.BigInteger	BigInt

Tipo Java	Tipo de SQL
java.math.BigDecimal	Decimal
java.sql.Date	Fecha
java.sql.Time	Hora
com.ibm.is.cc.javastage.api.TimeMicroseconds.class	Hora
java.sql.Timestamp	Indicación de fecha y hora

Si la etapa tiene un solo enlace de entrada y uno o varios enlaces de salida, la etapa Java Integration añade automáticamente todas las columnas del enlace de entrada que no están presentes en el enlace de salida cuyo RCP está habilitado.

Transferencia de los datos de columna de la entrada a la salida

La API de la etapa Java Integration proporciona la funcionalidad para consultar metadatos de columna dinámicamente en tiempo de ejecución y acceder a los datos. Necesita crear código para leer los datos de las columnas propagadas en el enlace de entrada, y grabarlos en el enlace de salida para la que el RCP esté habilitado.

La etapa Java Integration también proporciona la funcionalidad para transferir automáticamente los datos de la columna de un enlace de entrada al enlace de salida si la etapa tiene un único enlace de entrada y uno o varios enlaces de salida. En este caso, el código de usuario no es necesario para transferir los datos de las columnas propagadas en el enlace de entrada. Para obtener más información, consulte la clase `com.ibm.is.cc.javastage.api.ColumnTransferBehavior` en la información de Javadoc para la API de la etapa Java Integration.

Ejecución del código Java en el nodo conductor

La etapa Java Integration da soporte a la ejecución de código Java en el nodo conductor durante la inicialización y la finalización de las etapas.

Para ejecutar el código Java en el conductor, el código Java llama al método `setIsRunOnConductor` de la clase `Capabilities`. Durante la inicialización de la etapa, la clase `Processor` llama a los siguientes métodos:

1. `getCapabilities()`
2. `validateConfiguration()`
3. `initialize()`

Durante la inicialización, algunos de los métodos adicionales de la clase `Processor` son invocados en los siguientes escenarios:

- Si el método `validateConfiguration` devuelve false, se llama al método `getConfigurationErrors`.
- Si la propagación de columnas en tiempo de ejecución está habilitada en un enlace de entrada y el método `getBeanForOutput` se ha implementado para devolver el objeto JavaBean, se llama a los siguientes métodos de la clase `Processor` durante la inicialización:
  1. `getCapabilities()`
  2. `validateConfiguration()`
  3. `getBeanForOutput()`
  4. `getAdditionalOutputColumns()`

- Si la capacidad que ejecuta el código Java en el nodo conductor y la propagación de columnas en tiempo de ejecución están habilitadas ambas en al menos un enlace de salida asociado al objeto JavaBean, se llama a los siguientes métodos de la clase `Processor`, en el orden especificado, durante la inicialización. El método `validateConfiguration` se invoca dos veces en el nodo conductor.
  1. `getCapabilities()`
  2. `validateConfiguration()`
  3. `getBeanForOutput()`
  4. `getAdditionalOutputColumns()`
  5. `validateConfiguration()`
  6. `initialize()`

Durante la finalización de una etapa, se llama al método `terminate()` de la clase `Processor`.

El siguiente ejemplo de código Java se utiliza para establecer la capacidad para ejecutar los métodos de la clase `Processor` en el nodo conductor para la inicialización y finalización de la etapa.

```
public Capabilities getCapabilities()
{
   Capabilities capabilities = new Capabilities();
   capabilities.setIsRunOnConductor(true);
   return capabilities;
}
```

El código Java comprueba el valor de retorno del método `getNodeNumber` de la clase `Configuration` para identificar si el método se invoca en el nodo conductor o los nodos reproductor. Si el valor de retorno es `-1` el método se ejecuta en el nodo conductor. Si el valor de retorno es cualquier otro número, por ejemplo 0 o 1, el método se invoca en los nodos reproductor.

En el siguiente ejemplo, la clase `Processor` completa procesos de inicialización y finalización en el nodo conductor y los nodos reproductor.

```
private int m_nodeID = -1;
public boolean validateConfiguration(Configuration configuration,
boolean isRuntime) throws Exception
{
...
m_nodeID = configuration.getNodeNumber();
...
return true;
}

public void initialize() throws Exception
{
if (m_nodeID == -1) // En el Conductor
   {
     // Realizar cualquier proceso de inicialización en el nodo conductor
   }
   else
   {
     // Realizar cualquier proceso de inicialización en cada nodo reproductor
   }
}

public void terminate(boolean isAborted) throws Exception
{
   if (m_nodeID == -1)  // En el Conductor
   {
```

```
    // Realizar cualquier proceso de finalización en el nodo conductor
   }
   else
   {
   // Realizar cualquier proceso de finalización en cada nodo reproductor
   }
}
```

Transferencia de datos desde el nodo conductor a los nodos reproductor

La etapa Java Integration proporciona al código Java la interfaz `DataChannel` para transferir datos desde el nodo conductor a los nodos reproductor.

La interfaz `DataChannel` proporciona los siguientes métodos:

- `sendTo()`
- `receiveFrom()`

Para obtener una instancia de la interfaz `DataChannel`, el código Java llama al método `getDataChannel` de la clase `Configuration`. Para enviar datos desde el nodo conductor, el código Java llama al método `sendTo`. Para recibir datos en cada nodo reproductor, el código Java llama al método `receiveFrom`.

Para habilitar la capacidad de servicio de transferencia de datos durante los procesos de inicialización y finalización de la etapa, el código Java llama al método `setIsRunOnConductor` de la clase `Capabilities`. El método `getDataChannel` devuelve null si la capacidad no está habilitada.

En el siguiente código de ejemplo, el código Java pasa datos de serie desde el nodo conductor a los nodos reproductor.

```
private int m_nodeID = -1;

public boolean validateConfiguration(Configuration configuration,
boolean isRuntime) throws Exception
{
   ...
   m_nodeID = configuration.getNodeNumber();
   if (isRuntime == true)
   {
      // Obtener canal de datos
      DataChannel channel = configuration.getDataChannel();
      if (m_nodeID == -1) // En el nodo Conductor
      {
         // Enviar datos al nodo reproductor
         String object = new String("*** test data ***");
         Logger.information(object + " is sent from Conductor to Player. ");
         channel.sendTo(DataChannel.NODE_PLAYER, object);
      }
      else  // En el nodo reproductor
      {
         // Recibir datos del nodo conductor
         String object = (String) channel.receiveFrom(DataChannel.NODE_CONDUCTOR);
         Logger.information(object + "is received on Player(" +  m_nodeID + ").");
      }
   }
   ...
   return true;
}
```

Cuando se ejecuta el código de ejemplo, se anotan los siguientes mensajes en el registro de trabajo de InfoSphere DataStage:

```
Java_Stage: *** datos de prueba *** establecidos en el nodo Conductor.
Java_Stage,0: *** datos de prueba *** recibidos en Reproductor(0).
Java_Stage,1: *** datos de prueba *** recibidos en Reproductor(1).
```

Registro de mensajes con la etapa Java Integration

Puede utilizar la clase Logger (com.ibm.is.cc.javastage.api.Logger) para registrar mensajes de tiempo de ejecución y de tiempo de diseño para el trabajo.

En tiempo de ejecución, se graban mensajes de error, de aviso e informativos en el registro de trabajo. Por ejemplo, el código puede grabar mensajes informativos llamando a `Logger.information()`. El código de tiempo de ejecución de la etapa Java Integration también registra mensajes en el registro de trabajo.

Los mensajes de tiempo de diseño se almacenan en el registro del Servicio de acceso de conector, que se puede ver en la consola web de InfoSphere Information Server. Por ejemplo, el código puede depurar mensajes llamando a `Logger.debug()`. El código de tiempo de diseño de la etapa Java Integration también registra mensajes en el registro del Servicio de acceso de conector.

La clase Logger proporciona los siguientes métodos:

- `debug(int messageNumber, String message)`
- `debug(String message)`
- `fatal(String message)`
- `getComponentID()`
- `information(int messageNumber, String message)`
- `information(String message)`
- `isDebugEnabled ()`
- `setComponentID(String compID)`
- `warning(int messageNumber, String message)`
- `warning(String message)`

Para obtener más información sobre la clase Logger, consulte el Javadoc para la API de la etapa Java Integration.

Formato del ID de mensaje para la clase Logger

La clase Logger (com.ibm.is.cc.javastage.api.Logger) registra los mensajes en el formato `IIS-`*`IDcategoría-IDcomponente-númeroMensaje`*.

IIS es el prefijo para InfoSphere Information Server. *IDcategoría* es la categoría del componente que genera el mensaje. CONN es el *IDcategoría* para los conectores, incluida la etapa Java Integration. *IDcomponente* es el componente que emite el mensaje. *númeroMensaje* es un entero positivo para el mensaje.

La siguiente tabla contiene una lista de tipos de suceso y los ID de mensaje predefinidos de la etapa Java Integration.

Tabla 4. Lista de tipos de suceso y de los ID de mensaje predefinidos de la etapa Java Integration

Método	Tipo de suceso	ID de mensaje
Logger.debug(String message)	Informativo	IIS-CONN-JAVA-00011
Logger.information(String message)	Informativo	IIS-CONN-JAVA-00012
Logger.warning(String message)	Aviso	IIS-CONN-JAVA-00013
Logger.fatal(String message)	Muy grave	IIS-CONN-JAVA-00015

Registro de mensajes con identificadores (ID) personalizados

Si la clase Logger se invoca en tiempo de ejecución, los mensajes se anotan en el registro del cliente del Director. Si la clase Logger se invoca en tiempo de diseño, los mensajes se anotan en el registro del Servicio de acceso de conector. En tiempo de ejecución, puede registrar un mensaje con el ID de mensaje que se define en la etapa Java Integration o puede registrar un mensaje con un ID de mensaje personalizado que se define en el código Java.

Puede utilizar los siguientes métodos de la clase Logger en su código Java para anotar mensajes con IDs de mensaje personalizados en el registro del cliente del Director:

- `setComponentID(String compID)`
- `debug(int messageNumber, String message)`
- `information(int messageNumber, String message)`
- `warning(int messageNumber, String message)`

Para obtener más información sobre la clase Logger, consulte la información de Javadoc para la API de la etapa Java Integration.

En el siguiente ejemplo se registra un mensaje informativo `Trabajo completado satisfactoriamente` con el ID de mensaje IIS-CONN-MYCONN-00123.

```
Logger.setComponentID("MYCONN");
Logger.information(123, "Trabajo completado satisfactoriamente");
```

Registro de mensajes de depuración

Puede utilizar código Java para registrar mensajes de depuración cuando la variable de entorno JAVASTAGE_API_DEBUG está establecida en 1.

Puede utilizar los siguientes métodos de la clase `Logger` en el código Java para registrar mensajes de depuración:

- `debug(String message)`
- `debug(int messageNumber, String message)`
- `isDebugEnabled()`
- `setComponentID(String compID)`

El siguiente ejemplo registra los mensajes de depuración:

```
debug test 001
debug test 002
 ...
debug test 999
```

```
if (Logger.isDebugEnabled())
{
   Logger.debug("prueba de depuración 001");
   Logger.debug("prueba de depuración 002");
   ...
   Logger.debug("prueba de depuración 999");
}
```

El método `Logger.isDebugEnabled()` devuelve true si la variable de entorno JAVASTAGE_API_DEBUG está establecida en 1. Para evitar llamar al método `Logger.debug` innecesariamente, asegúrese de que el método `Logger.isDebugEnabled()` devuelve true.

Terminación de un trabajo desde el código Java

Puede generar un objeto de clase Exception, o un objeto de clase ConnectorException, o llamar al método Logger.fatal() para detener un trabajo antes de que finalice de forma natural.

Generación de un objeto de clase Exception (etapa Java Integration)

Puede utilizar la clase `Exception (java.lang.Exception)` o su objeto de subclase de su implementación de Processor para detener un trabajo.

El siguiente ejemplo muestra código Java que realiza un cálculo básico para obtener un precio. Si el valor establecido para `count` es cero, el código Java genera la excepción `java.lang.ArithmeticException` y registra el mensaje: `Dividir por cero`.

```
public void process() throws Exception
{
   int count = 0;
   int total = 1000;
   ...
   int price = total / count;
}
```

El código Java también puede construir un objeto `ArithmeticException` con un mensaje que usted especifique y generarlo para terminar un trabajo.

```
public void process() throws Exception
{
   ...
   if (result < 0)
   {
      throw new ArithmeticException("Se ha producido una excepción.
                                     result=" + result);
   }
}
```

Generación de un objeto de clase ConnectorException (etapa Java Integration)

Puede utilizar el objeto de clase ConnectorException(com.ibm.is.cc.javastage.api.ConnectorException) desde la implementación de Processor para detener un trabajo que está en ejecución o para finalizar una operación de tiempo de diseño con un mensaje.

Utilice la clase ConnectorException para registrar mensajes que no incluyen la información de rastreo de la pila y también para asignar los ID de mensaje a dichos mensajes.

En el siguiente ejemplo, el código Java construye y genera un objeto ConnectorException después de llamar al método Logger.setComponentID(). La infraestructura de conector captura la excepción, registra el mensaje de error Se ha producido una excepción. result=-1 y detiene el trabajo. El ID de mensaje es IIS-CONN-MYCONN-00999.

```
public class MyConnector extends Processor
{
   public MyConnector()
   {
     super();
     Logger.setComponentID("MYCONN");
   }


   public void process() throws Exception
   {
      ...
      if (result < 0)
      {
         throw new ConnectorException(999, " Se ha producido
una excepción. result=" + result);
      }
   }
   ...
}
```

Invocación del método Logger.fatal()

Puede terminar un trabajo llamando al método com.ibm.is.cc.javastage.api.Logger.fatal() de su implementación de Processor.

Cuando el código Java llama al método Logger.fatal(), el trabajo no devuelve el control al código Java. El método Processor.terminate(), que detiene el trabajo, no se puede invocar. Esto puede ocasionar un problema al reiniciar el trabajo.

Si su código Java requiere un proceso de finalización, utilice el objeto ConnectorException o cualquier otro objeto Exception, ya que el método Processor.terminate() se invoca con el argumento isAborted establecido en true.

El siguiente ejemplo muestra código Java que llama al método Logger.fatal(). El trabajo finaliza y el mensaje Mi trabajo termina anormalmente se anota en el registro de trabajo con el ID de mensaje IIS-CONN-JAVA-00015.

```
Logger.fatal("Mi trabajo termina anormalmente.");
```

Utilización de JavaBeans (etapa Java Integration)

La API de la etapa Java Integration admite JavaBeans que permiten a la etapa Java Integration acceder al código Java existente.

La etapa de Java Integration presupone los siguientes convenios de JavaBeans:

- La clase debe tener un constructor predeterminado público (sin argumento).
- La clase debe método de obtención y establecimiento para cada propiedad.

La etapa Java Integration admite los siguientes tipos de propiedad de JavaBeans:

- Boolean/java.lang.Boolean
- byte[]
- short/java.lang.Short
- int/java.lang.Integer
- long/java.lang.Long
- float/java.lang.Float
- double/java.lang.Double
- java.lang.String
- java.math.BigInteger
- java.math.BigDecimal
- java.sql.Time
- java.sql.Timestamp
- java.sql.Date

El ejemplo siguiente muestra cómo se utiliza la clase `samples.InputBean` para un enlace de entrada y la clase `samples.OutputBean` para un enlace de salida.

JavaBeansTransformer.java

```
package samples;

import com.ibm.is.cc.javastage.api.*;

public class JavaBeansTransformer extends Processor
{
  private InputLink m_inputLink;
  private OutputLink m_outputLink;
  private OutputLink m_rejectLink;

  public Capabilities getCapabilities()
  {
    Capabilities capabilities = new Capabilities();
    // Establecer número mínimo de enlaces de entrada en 1
    capabilities.setMinimumInputLinkCount(1);
    // Establecer número máximo de enlaces de entrada en 1
    capabilities.setMaximumInputLinkCount(1);
    // Establecer número mínimo de enlaces de secuencia de salida en 1
    capabilities.setMinimumOutputStreamLinkCount(1);
    // Establecer número máximo de enlaces de secuencia de salida en 1
    capabilities.setMaximumOutputStreamLinkCount(1);
    // Establecer número máximo de enlaces de rechazo en 1
    capabilities.setMaximumRejectLinkCount(1);
    // Establecer is Wave Generator en false
    capabilities.setIsWaveGenerator(false);
    return capabilities;
  }

  public boolean validateConfiguration(
    Configuration configuration, boolean isRuntime)
    throws Exception
  {
    // Especificar configuraciones de enlaces actuales.
    m_inputLink = configuration.getInputLink(0);
    m_outputLink = configuration.getOutputLink(0);
    if (configuration.getRejectLinkCount() == 1)
    {
      m_rejectLink = m_inputLink.getAssociatedRejectLink();
    }

    return true;
  }
```

```
public void process() throws Exception
{
  OutputRecord outputRecord = m_outputLink.getOutputRecord();

  // Bucle hasta que no haya más datos de entrada
  do
  {
    InputRecord record = m_inputLink.readRecord();
    if (record == null)
    {
      // Fin de los datos
      break;
    }

    // Obtener el objeto de la fila de entrada.
    InputBean inputBean = (InputBean) record.getObject();

    // Obtener el valor de la columna de nombre del enlace de entrada.
    // Si el valor contiene el carácter "*", marcar distintivo de rechazo
    // y enviar el registro
    // al enlace de rechazo en el procesamiento posterior.
    boolean fReject = false;
    String name = inputBean.getFirstName();
    if ((name == null) || (name.indexOf('*') >= 0))
    {
      fReject = true;
    }

    if (!fReject)
    {
      // Enviar registro a salida
      OutputBean outputBean = new OutputBean();
      outputBean.setEmpno(inputBean.getEmpno());
      outputBean.setFirstName(inputBean.getFirstName().toUpperCase());
      outputBean.setLastName(inputBean.getLastName().toUpperCase());
      outputBean.setHireDate(inputBean.getHireDate());
      outputBean.setEdLevel(inputBean.getEdLevel());
      outputBean.setSalary(inputBean.getSalary());
      outputBean.setBonus(inputBean.getBonus());
      outputBean.setLastUpdate(inputBean.getLastUpdate());
      outputRecord.putObject(outputBean);
      m_outputLink.writeRecord(outputRecord);
    }
    else if (m_rejectLink != null)
    {
      // Rechazar registro. Esto transfiere la fila al enlace de rechazo.
      // El mismo tipo de reenvío también es posible para los enlaces de
      // secuencia normales.
      RejectRecord rejectRecord = m_rejectLink.getRejectRecord(record);

      // El registro de rechazo puede contener más columnas "ERRORTEXT" y
      // "ERRORCODE". El campo aparecerá como columnas en los registros de
      // de salida rechazados.
      rejectRecord.setErrorText("El campo de nombre contiene *");
      rejectRecord.setErrorCode(123);
      m_rejectLink.writeRecord(rejectRecord);
    }
  }
  while (true);
}

public Class<InputBean> getBeanForInput(Link inputLink)
{
  return InputBean.class;
}
```

```
  public Class<OutputBean> getBeanForOutput(Link outputLink)
  {
    return OutputBean.class;
  }
}
```

InputBean.java

```
package samples;

import java.sql.Date;
import java.sql.Time;

public class InputBean
{
  private long      m_empno;
  private String    m_firstname;
  private String    m_lastname;
  private Date      m_hiredate;
  private int       m_edlevel;
  private Double    m_salary;
  private double    m_bonus;
  private Time      m_lastupdate;

  /**
   * Capta el valor del campo empno.
   *
   * Valor largo @return del campo empno
   */
  public long getEmpno()
  {
    return m_empno;
  }

  /**
   * Establecer el valor del campo empno.
   *
   * Valor @param empno del campo empno.
   */
  public void setEmpno(long empno)
  {
    m_empno = empno;
  }

  /**
   * Capta el valor del campo firstname.
   *
   * Valor @return String del campo firstname
   */
  public String getFirstName()
  {
    return m_firstname;
  }

  /**
   * Establecer el valor del campo firstname.
   *
   * Valor @param firstname del campo firstname.
   */
  public void setFirstName(String firstname)
  {
    m_firstname = firstname;
  }

  /**
   * Capta el valor del campo lastname.
   *
```

```
 * Valor @return String del campo lastname
 */
public String getLastName()
{
  return m_lastname;
}

/**
 * Establecer el valor del campo lastname.
 *
 * Valor @param lastname del campo lastname.
 */
public void setLastName(String lastname)
{
  m_lastname = lastname;
}

/**
 * Captura del valor del campo hiredate.
 *
 * Valor @return Date del campo hiredate
 */
public Date getHireDate()
{
  return m_hiredate;
}

/**
 * Establecer el valor del campo hiredate.
 *
 * Valor @param hiredate del campo hiredate.
 */
public void setHireDate(Date hiredate)
{
  m_hiredate = hiredate;
}

/**
 * Capta el valor del campo edlevel.
 *
 * Valor @return int del campo edlevel
 */
public int getEdLevel()
{
  return m_edlevel;
}

/**
 * Establecer el valor del campo edlevel.
 *
 * Valor @param edlevel del campo edlevel.
 */
public void setEdLevel(int edlevel)
{
  m_edlevel = edlevel;
}

/**
 * Capta el valor del campo salary.
 *
 * Valor @return Double del campo salary
 */
public Double getSalary()
{
  return m_salary;
}
```

```
  /**
   * Establecer el valor del campo salary.
   *
   * Valor @param del campo salary.
   */
  public void setSalary(Double salary)
  {
    m_salary = salary;
  }

  /**
   * Capta el valor el campo bonus.
   *
   * Valor @return double del campo bonus
   */
  public double getBonus()
  {
    return m_bonus;
  }

  /**
   * Establecer el valor del campo bonus.
   *
   * Valor @param bonus del campo bonus.
   */
  public void setBonus(double bonus)
  {
    m_bonus = bonus;
  }

  /**
   * Capta el valor del campo lastupdate.
   *
   * Valor @return del campo lastupdate
   */
  public Time getLastUpdate()
  {
    return m_lastupdate;
  }

  /**
   * Establecer el valor del campo lastupdate.
   *
   * Valor @param lastupdate del campo lastupdate.
   */
  public void setLastUpdate(Time lastupdate)
  {
    m_lastupdate = lastupdate;
  }
}
```

OutputBean.java

```
package samples;

import java.sql.Date;
import java.sql.Time;

public class OutputBean
{
  private long      m_empno;
  private String    m_firstname;
  private String    m_lastname;
  private Date      m_hiredate;
  private int       m_edlevel;
  private Double    m_salary;
  private double    m_bonus;
```

```
private double    m_income;
private Time      m_lastupdate;

/**
 * Capta el valor del campo empno.
 *
 * Valor largo @return del campo empno
 */
public long getEmpno()
{
  return m_empno;
}

/**
 * Establecer el valor del campo empno.
 *
 * Valor @param empno del campo empno.
 */
public void setEmpno(long empno)
{
  m_empno = empno;
}

/**
 * Capta el valor del campo firstname.
 *
 * Valor @return String del campo firstname
 */
public String getFirstName()
{
  return m_firstname;
}

/**
 * Establecer el valor del campo firstname.
 *
 * Valor @param firstname del campo firstname.
 */
public void setFirstName(String firstname)
{
  m_firstname = firstname;
}

/**
 * Capta el valor del campo lastname.
 *
 * Valor @return String del campo lastname
 */
public String getLastName()
{
  return m_lastname;
}

/**
 * Establecer el valor del campo lastname.
 *
 * Valor @param lastname del campo lastname.
 */
public void setLastName(String lastname)
{
  m_lastname = lastname;
}

/**
 * Captura del valor del campo hiredate.
 *
 * Valor @return Date del campo hiredate
```

```
 */
public Date getHireDate()
{
  return m_hiredate;
}

/**
 * Establecer el valor del campo hiredate.
 *
 * Valor @param hiredate del campo hiredate.
 */
public void setHireDate(Date hiredate)
{
  m_hiredate = hiredate;
}

/**
 * Capta el valor del campo edlevel.
 *
 * Valor @return int del campo edlevel
 */
public int getEdLevel()
{
  return m_edlevel;
}

/**
 * Establecer el valor del campo edlevel.
 *
 * Valor @param edlevel del campo edlevel.
 */
public void setEdLevel(int edlevel)
{
  m_edlevel = edlevel;
}

/**
 * Capta el valor del campo salary.
 *
 * Valor @return Double del campo salary
 */
public Double getSalary()
{
  return m_salary;
}

/**
 * Establecer el valor del campo salary.
 *
 * Valor @param del campo salary.
 */
public void setSalary(Double salary)
{
  m_salary = salary;
}

/**
 * Capta el valor el campo bonus.
 *
 * Valor @return double del campo bonus
 */
public double getBonus()
{
  return m_bonus;
}

/**
```

```
   * Establecer el valor del campo bonus.
   *
   * Valor @param bonus del campo bonus.
   */
  public void setBonus(double bonus)
  {
    m_bonus = bonus;
  }

  /**
   * Capta el valor del campo lastupdate.
   *
   * Valor @return del campo lastupdate
   */
  public Time getLastUpdate()
  {
    return m_lastupdate;
  }

  /**
   * Establecer el valor del campo lastupdate.
   *
   * Valor @param lastupdate del campo lastupdate.
   */
  public void setLastUpdate(Time lastupdate)
  {
    m_lastupdate = lastupdate;
  }
}
```

Si el código Java utiliza JavaBeans como representaciones de registros en el enlace, debe sustituir los métodos **getBeanForInput()** y **getBeanforOutput()** de la clase `Processor`. El código Java debe devolver `java.lang.Class` de la clase JavaBeans correspondiente a cada enlace. La etapa Java Integration invocará este método en el momento de la inicialización.

```
public Class<InputBean> getBeanForInput(Link inputLink)
{
  return InputBean.class;
}

public Class<OutputBean> getBeanForOutput(Link outputLink)
{
  return OutputBean.class;
```

La clase `JavaBeans` correspondiente a un enlace de entrada puede crear una instancia de la etapa Java Integration, y el código Java puede obtener esta instancia llamando al método **getObject()** de la interfaz `InputLink` .

```
InputBean inputBean = (InputBean) record.getObject();


String name = inputBean.getFirstName();
      :
      :
```

Antes de que su código Java grabe un registro en un enlace de salida que esté asociado con JavaBeans, el código Java debe crear una instancia de un objeto JavaBeans para el enlace de salida y establecer el valor de cada una de las propiedades del bean. El tipo de objeto JavaBeans debe coincidir con el tipo especificado por el método **getBeanForOutput()**.

```
 // Enviar registro a salida
 OutputBean outputBean = new OutputBean();
 outputBean.setEmpno(inputBean.getEmpno());
 outputBean.setFirstName(inputBean.getFirstName().toUpperCase());
```

```
outputBean.setLastName(inputBean.getLastName().toUpperCase());
outputBean.setHireDate(inputBean.getHireDate());
outputBean.setEdLevel(inputBean.getEdLevel());
outputBean.setSalary(inputBean.getSalary());
outputBean.setBonus(inputBean.getBonus());
outputBean.setLastUpdate(inputBean.getLastUpdate());
```

Finalmente, el código debe llamar al método **putObject(Object)** de la interfaz OutputRecord para establecer este objeto JavaBeans para el registro de salida.

```
outputRecord.putObject(outputBean);
```

Función definida por el usuario (UDF) - etapa Java Integration

La etapa Java Integration admite la ejecución de las funciones definidas por el usuario existentes que utilizan JavaBeans o tipos primitivos (que admite la etapa Java) en su interfaz de llamada.

Para ejecutar el las funciones definidas por el usuario existentes, el número de parámetros debe coincidir con el número de enlaces de entrada, y el bean del valor de retorno debe correlacionarse con un enlace de salida. Una excepción a esta regla es que cuando la etapa Java Integration se utiliza como destino, no importa si la función definida por el usuario devuelve un valor o no.

El código de ejemplo siguiente muestra una función definida por el usuario que combinará dos registros de entrada y grabará el resultado en un enlace de salida:

```
public class UserDefinedFunction
{

   /**
    * Pasa el tipo primitivo doble y un bean como argumentos UDF y
* devuelve un bean.
    *
    * @param commission commission
    * Objeto @param input {@link InputBean}.
    * Objeto @return output {@link UDFOutputBean}.
    */
   public UDFOutputBean AnnualIncome(double commission, InputBean input)
   {UDFOutputBean output = new UDFOutputBean();

      output.setEmpno(input.getEmpno());
      output.setFirstName(input.getFirstName().toUpperCase());
      output.setLastName(input.getLastName().toUpperCase());

      double total = commission +
      input.getSalary().doubleValue() +
      input.getBonus();
      output.setIncome(total);

      return output;
   }

}
```

Capítulo 3. Diseñar trabajos (etapa Java Integration)

Puede utilizar la etapa Java Integration para integrar código Java y desarrollar trabajos para leer, grabar y cargar datos.

Procedimiento

1. Añada una etapa Java Integration a un trabajo.
   a. Opcional: Migre un trabajo de Java Pack de herencia a la etapa Java Integration
2. Para configurar la etapa Java Integration como origen para recuperar datos de código Java:
   a. Configure la etapa Java Integration como origen.
   b. Configure definiciones de columna.
   c. Correlacione el índice de enlaces con los nombres de enlaces.
3. Para configurar la etapa Java Integration como destino para pasar datos al código Java:
   a. Configure la etapa Java Integration como un destino.
   b. Configure definiciones de columna.
   c. Correlacione el índice de enlaces con los nombres de enlaces.
4. Para configurar la etapa Java Integration como un transformador para transformar datos en el enlace de entrada, y escribir en el enlace de salida:
   a. Configure la etapa Java Integration como un transformador.
   b. Configure definiciones de columna.
   c. Correlacione el índice de enlaces con los nombres de enlaces.
5. Busque datos utilizando los enlaces de referencia.
6. Compile y ejecute el trabajo.

Añadir una etapa Java Integration a un trabajo

Utilice el cliente InfoSphere DataStage and QualityStage Designer para añadir una etapa Java Integration a un trabajo.

Procedimiento

1. Desde el Cliente del Diseñador, seleccione en el menú **Archivo** > **Nuevo**.
2. En la ventana Nuevo, seleccione el icono **Trabajo paralelo** y pulse **Aceptar**.
3. En la parte izquierda del cliente Designer del menú Paleta, seleccione la categoría **Tiempo real**.
4. Ubique **Java Integration** en la lista.
5. Arrastre el icono de la etapa **Java Integration** al lienzo de diseño de trabajos.
6. Conecte los enlaces de entrada y de salida a la etapa **Java Integration**.
7. Efectúe una doble pulsación en el icono de la etapa **Java Integration** y seleccione el separador **Etapa** para entrar o modificar los atributos siguientes:
   - **Nombre de etapa:** modifique el nombre predeterminado de la **etapa**. Puede introducir hasta 255 caracteres. Como alternativa, puede modificar el nombre de la etapa en el lienzo de diseño de trabajos.
   - **Descripción:** especifique una descripción opcional de la etapa.
8. Pulse **Guardar**.

Qué hacer a continuación

Defina las propiedades para la etapa Java Integration.

Integración de datos de código Java (etapa Java Integration)

Para leer datos de su código Java utilizando la etapa Java Integration necesita integrar un código Java admitido por la etapa Java Integration y configurar la etapa Java Integration para que procese los datos como un origen. Como origen, el conector extrae o lee datos de su código Java.

En la figura siguiente se muestra un ejemplo de la utilización de la etapa Java Integration para leer datos. En este caso, la etapa Java Integration **Java_Stage_0** lee datos y los escribe en los archivos **Peek_1** y **Peek_2**. Cuando configure la etapa Java Integration para que lea datos, puede crear 1 o más enlaces de salida **Peek_1** y **Peek_2**, que aparece en la figura posterior transfiriendo filas de **Java_Stage_0** a **Peek_1** y **Peek_2**.

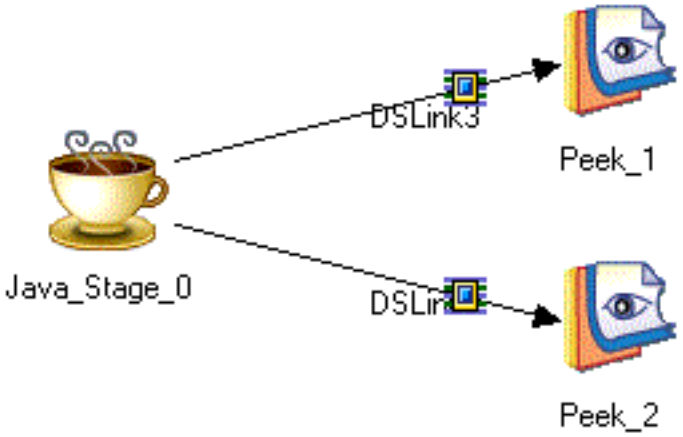


Figura 1. Ejemplo de lectura de datos

Configurar la etapa Java Integration como origen

Puede configurar la etapa Java Integration para procesar datos como origen para uno o más enlaces de salida.

Procedimiento

1. En el lienzo de diseño de trabajos, efectúe una doble pulsación en el icono **Etapa Java Integration**.
2. Seleccione el separador **Salida** y seleccione el enlace de entrada que desee editar desde la lista desplegable **Nombre de entrada (de la etapa en sentido ascendente)**. Al editar el enlace de salida, está configurando la etapa Java Integration como origen.
3. Especifique una descripción opcional del enlace de salida en el separador **General**.
4. Opcional: Pulse **Configurar** para configurar las propiedades adicionales. Según el código de usuario, se visualiza el **Editor de propiedad personalizado**, **Editor de correlaciones de columnas** o el **Importador de metadatos de columna**.
   a. Se necesita especificar valores para las propiedades. Si el código de usuario expone propiedades definidas por el usuario utilizando el método

`Processor.getUserPropertyDefinitions()`, aparece la ventana **Editor de propiedad personalizada**, en la que puede especificar el valor de cada propiedad.

b. Si el código de usuario utiliza JavaBeans en su interfaz (mediante la implementación de los métodos `Processor.getBeanForInput()` y `Processor.getBeanForOutput()`), aparece el **Editor de correlaciones de columnas** de la ventana. Complete el esquema de columna de DataStage basándose en las propiedades de los JavaBeans y luego correlacione las propiedades JavaBeans con las columnas de DataStage. Si crea un trabajo donde los enlaces no contienen columnas, entonces el **Editor de correlaciones de columnas** inicial contiene tablas vacías. Pulse **Examinar objetos** para lanzar la ventana **Seleccionar bean Propiedades** y seleccione las propiedades del bean o los argumentos de función definida por el usuario (UDF) que se vayan a importar en el **Editor de correlaciones de columnas** y, a continuación, pulse **Aceptar**. Pulse **Finalizar**.También puede correlacionar las propiedades JavaBean con las columnas de DataStage existentes, en lugar de crear nuevas columnas.

c. Si su código de usuario implementa los métodos `Processor.getColumnMetadataForInput()` y `Processor.getColumnMetadataForOutput()` en lugar de los métodos `Processor.getBeanForInput()` y `Processor.getBeanForOutput()`, aparece la ventana **Importador de metadatos de columna**. En la ventana **Importador de metadatos de columna**, seleccione los metadatos de la columna para completar el esquema de la columna DataStage a partir de una lista de instancias de `ColumnMetadata` que son devueltas por los métodos `Processor.getColumnMetadataForInput()` y `Processor.getColumnMetadataForOutput()` para cada enlace. Pulse **Examinar objetos** y seleccione los metadatos de la columna que vayan a completarse en el trabajo. Pulse **Finalizar**.

5. Especifique los detalles necesarios en el separador **Propiedades** y el separador **Avanzado**.
6. Pulse **Aceptar** para guardar la configuración de la conexión que haya especificado.

Configurar definiciones de columna (Java Integration stage)

Puede configurar definiciones de columna para un enlace y también personalizar la cuadrícula de columnas, guardar definiciones de columna para utilizarlas posteriormente y cargar definiciones de columna predefinidas del repositorio.

Procedimiento

1. En el lienzo de diseño de trabajos, efectúe una doble pulsación en el icono **Etapa Java Integration**.
2. Seleccione el separador **Salida** y seleccione el enlace de entrada que desee editar desde la lista desplegable **Nombre de entrada (de la etapa en sentido ascendente)**.
3. En el separador **Columnas**, modifique la cuadrícula de columnas para especificar los metadatos que desee definir.
   a. Pulse con el botón derecho del ratón en la cuadrícula y seleccione **Propiedades** en el menú.
   b. En la ventana Propiedades de cuadrícula, seleccione las propiedades que desee visualizar y el orden en que desee que se visualicen. A continuación, pulse **Aceptar**.
4. Especifique definiciones de columna para la tabla utilizando uno de los métodos siguientes:

Opción	Descripción
Método 1	1. En la columna **Nombre de la columna**, efectúe una doble pulsación dentro de la celda apropiada y escriba un nombre de columna. 2. Para cada celda de la fila, efectúe una doble pulsación dentro de la celda y seleccione las opciones que desee. 3. En la columna **Descripción**, efectúe una doble pulsación dentro de la celda apropiada y escriba una descripción.
Método 2	1. Pulse con el botón derecho del ratón en la cuadrícula y seleccione **Editar fila** en el menú. 2. En la ventana Editar metadatos de columna, especifique los metadatos de la columna.

5. Para compartir metadatos entre varias columnas, seleccione las columnas cuyos metadatos desee compartir.
   a. Pulse con el botón derecho del ratón y seleccione **Propagar valores**.
   b. En la ventana Propagar valores de columnas, seleccione las propiedades que desee que las columnas seleccionadas compartan.
6. Para guardar las definiciones de columna como una definición de tabla en el repositorio, pulse **Guardar**.
   a. Especifique la información apropiada en la ventana Guardar definición de tabla y luego pulse **Aceptar**.
   b. En la ventana Guardar definición de tabla como, seleccione la carpeta en la que desee guardar la definición de tabla y luego pulse **Guardar**.
7. Para cargar definiciones de columna del repositorio, pulse **Cargar**.
   a. En la ventana Definiciones de tabla, seleccione la definición de tabla que desee cargar y luego pulse **Aceptar**.
   b. En la ventana Seleccionar columnas, utilice los botones de flecha para mover columnas de la lista **Columnas disponibles** a la lista **Columnas seleccionadas**. Pulse **Aceptar**.

Asociar índices de enlaces con enlaces (etapa Java Integration)

Puede asociar índices de enlace con enlaces cuando haya varios enlaces de salida.

Procedimiento

1. En el lienzo de diseño de trabajos, efectúe una doble pulsación en el icono de enlace de salida (que se conecta con el icono de la **Etapa Java Integration**) para el que desee cambiar el orden.
2. Seleccione el separador **Orden de los enlaces**. Este separador le permite especificar cómo los enlaces de salida corresponden a etiquetas de enlace numérico. La etiqueta de enlace numérico corresponde al índice del enlace al que se ha accedido a través de la API de la etapa Java Integration. Para reorganizar los enlaces, seleccione un enlace de salida necesarios y pulse el botón de flecha arriba o botón de flecha abajo.
3. Pulse **Aceptar** para guardar los detalles de correlación.

Cómo pasar datos al código Java (etapa Java Integration)

Para pasar datos al código Java utilizando la etapa Java Integration necesita integrar un código Java admitido por la etapa Java Integration y configurar la etapa Java Integration para que procese los datos como un destino.

En la figura siguiente se muestra un ejemplo de la utilización de la etapa Java Integration para pasar datos al código Java. En este caso, el generador de filas lee datos de los archivos **Row_Generator_5** y **Row_Generator_7** y, a continuación, la etapa Java Integration **Java_Stag** inserta, actualiza o suprime datos según convenga.

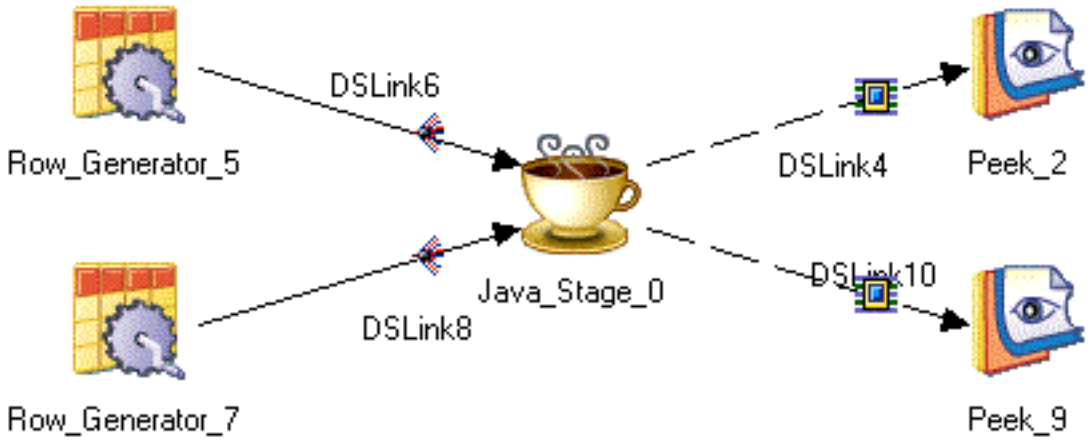


Figura 2. Ejemplo de grabación de datos

Configurar la etapa Java Integration como destino

Puede configurar la etapa Java Integration como destino para escribir datos. La etapa Java Integration como destino puede tener 1 o más enlaces de entrada, y 0 o más enlaces de rechazo.

Procedimiento

1. En el lienzo de diseño de trabajos, efectúe una doble pulsación en el icono **Etapa Java Integration**.
2. Seleccione el separador **Entrada** y seleccione el enlace de entrada que desee editar desde la lista desplegable **Nombre de entrada (de la etapa en sentido ascendente)**. Al editar el enlace de entrada, está configurando la etapa Java Integration como destino.
3. Especifique una descripción opcional del enlace de entrada en el separador **General**.
4. Opcional: Pulse **Configurar** para configurar las propiedades adicionales. Según el código de usuario, se visualiza el **Editor de propiedad personalizado**, **Editor de correlaciones de columnas** o el **Importador de metadatos de columna**.
   a. Se necesita especificar valores para las propiedades. Si el código de usuario expone propiedades definidas por el usuario utilizando el método `Processor.getUserPropertyDefinitions()`, aparece la ventana **Editor de propiedad personalizada**, en la que puede especificar el valor de cada propiedad.

b. Si el código de usuario utiliza JavaBeans en su interfaz (mediante la implementación de los métodos `Processor.getBeanForInput()` y `Processor.getBeanForOutput()`), aparece el **Editor de correlaciones de columnas** de la ventana. Complete el esquema de columna de DataStage basándose en las propiedades de los JavaBeans y luego correlacione las propiedades JavaBeans con las columnas de DataStage. Si crea un trabajo donde los enlaces no contienen columnas, entonces el **Editor de correlaciones de columnas** inicial contiene tablas vacías. Pulse **Examinar objetos** para lanzar la ventana **Seleccionar bean Propiedades** y seleccione las propiedades del bean o los argumentos de función definida por el usuario (UDF) que se vayan a importar en el **Editor de correlaciones de columnas** y, a continuación, pulse **Aceptar**. Pulse **Finalizar**.También puede correlacionar las propiedades JavaBean con las columnas de DataStage existentes, en lugar de crear nuevas columnas.

c. Si su código de usuario implementa los métodos `Processor.getColumnMetadataForInput()` y `Processor.getColumnMetadataForOutput()` en lugar de los métodos `Processor.getBeanForInput()` y `Processor.getBeanForOutput()`, aparece la ventana **Importador de metadatos de columna**. En la ventana **Importador de metadatos de columna**, seleccione los metadatos de la columna para completar el esquema de la columna DataStage a partir de una lista de instancias de `ColumnMetadata` que son devueltas por los métodos `Processor.getColumnMetadataForInput()` y `Processor.getColumnMetadataForOutput()` para cada enlace. Pulse **Examinar objetos** y seleccione los metadatos de la columna que vayan a completarse en el trabajo. Pulse **Finalizar**.

5. Especifique los detalles necesarios en el separador **Propiedades** y el separador **Avanzado**.
6. Pulse **Aceptar** para guardar la configuración que haya especificado.

Configurar definiciones de columna (Java Integration stage)

Puede configurar definiciones de columna para un enlace y también personalizar la cuadrícula de columnas, guardar definiciones de columna para utilizarlas posteriormente y cargar definiciones de columna predefinidas del repositorio.

Procedimiento

1. En el lienzo de diseño de trabajos, efectúe una doble pulsación en el icono **Etapa Java Integration**.
2. Seleccione el separador **Entrada** y seleccione el enlace de entrada que desee editar desde la lista desplegable **Nombre de entrada (de la etapa en sentido ascendente)**.
3. En el separador **Columnas**, modifique la cuadrícula de columnas para especificar los metadatos que desee definir.
   a. Pulse con el botón derecho del ratón en la cuadrícula y seleccione **Propiedades** en el menú.
   b. En la ventana Propiedades de cuadrícula, seleccione las propiedades que desee visualizar y el orden en que desee que se visualicen. A continuación, pulse **Aceptar**.
4. Especifique definiciones de columna para la tabla utilizando uno de los métodos siguientes:

Opción	Descripción
Método 1	1. En la columna **Nombre de la columna**, efectúe una doble pulsación dentro de la celda apropiada y escriba un nombre de columna. 2. Para cada celda de la fila, efectúe una doble pulsación dentro de la celda y seleccione las opciones que desee. 3. En la columna **Descripción**, efectúe una doble pulsación dentro de la celda apropiada y escriba una descripción.
Método 2	1. Pulse con el botón derecho del ratón en la cuadrícula y seleccione **Editar fila** en el menú. 2. En la ventana Editar metadatos de columna, especifique los metadatos de la columna.

5. Para compartir metadatos entre varias columnas, seleccione las columnas cuyos metadatos desee compartir.
   a. Pulse con el botón derecho del ratón y seleccione **Propagar valores**.
   b. En la ventana Propagar valores de columnas, seleccione las propiedades que desee que las columnas seleccionadas compartan.
6. Para guardar las definiciones de columna como una definición de tabla en el repositorio, pulse **Guardar**.
   a. Especifique la información apropiada en la ventana Guardar definición de tabla y luego pulse **Aceptar**.
   b. En la ventana Guardar definición de tabla como, seleccione la carpeta en la que desee guardar la definición de tabla y luego pulse **Guardar**.
7. Para cargar definiciones de columna del repositorio, pulse **Cargar**.
   a. En la ventana Definiciones de tabla, seleccione la definición de tabla que desee cargar y luego pulse **Aceptar**.
   b. En la ventana Seleccionar columnas, utilice los botones de flecha para mover columnas de la lista **Columnas disponibles** a la lista **Columnas seleccionadas**. Pulse **Aceptar**.

Asociar índices de enlaces con enlaces (etapa Java Integration)

Puede asociar índices de enlace con enlaces cuando haya varios enlaces de entrada.

Procedimiento

1. En el lienzo de diseño de trabajos, efectúe una doble pulsación en el icono de enlace de entrada (que se conecta con el icono de la **etapa Java de integración**) para el que desee cambiar el orden.
2. Seleccione el separador **Orden de los enlaces**. Este separador le permite especificar cómo los enlaces de salida corresponden a etiquetas de enlace numérico. La etiqueta de enlace numérico corresponde al índice del enlace al que se ha accedido a través de la API de la etapa Java Integration. Para reorganizar los enlaces, seleccione un enlace de salida necesarios y pulse el botón de flecha arriba o botón de flecha abajo.
3. Pulse **Aceptar** para guardar los detalles de correlación.

Transformación de datos (etapa Java Integration)

Para transformar datos utilizando la etapa Java Integration, necesita integrar un código Java que es admitido por la etapa Java Integration y configurar la etapa Java Integration para transformar datos en el enlace de entrada, y escribir resultados en el enlace de salida.

En la figura siguiente se muestra un ejemplo de la utilización de la etapa Java Integration como un transformador. En este caso, el generador de filas lee datos de **Row_Generator_5** y luego la etapa Java Integration **Java_Stage_0** escribe los datos transformados en **Peek_1** y en el enlace de rechazo **Peek_2**.

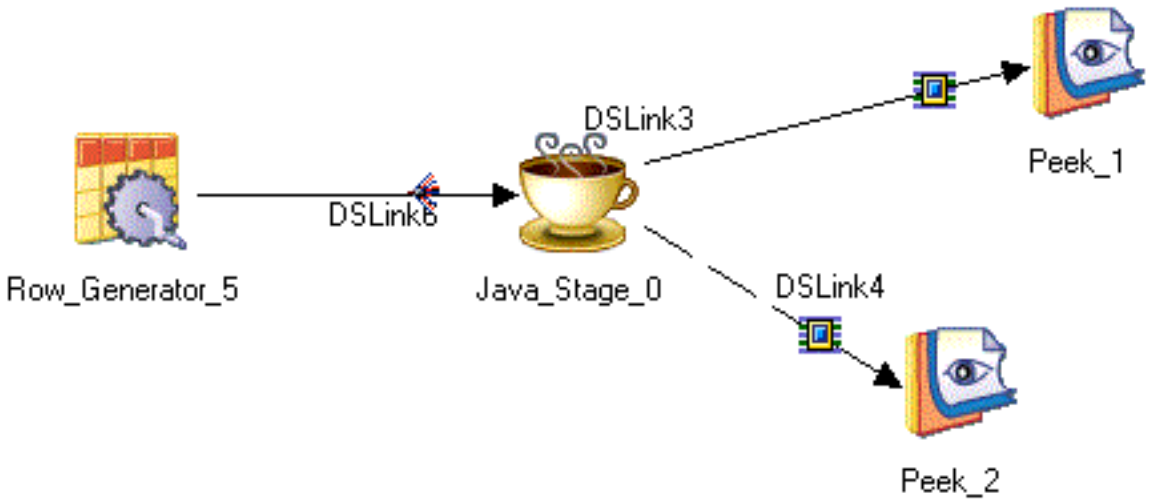


Figura 3. Ejemplo de transformación de datos

Configurar la etapa Java Integration como transformador

Puede configurar la etapa Java Integration como transformador para transformar datos externos. La etapa Java Integration como transformador puede tener 1 o más enlaces de entrada, y 1 o más enlaces de salida o de rechazo.

Procedimiento

1. En el lienzo de diseño de trabajos, efectúe una doble pulsación en el icono **Etapa Java Integration**.
2. Seleccione el separador **Entrada** y seleccione el enlace de entrada que desee editar desde la lista desplegable **Nombre de entrada (etapa en sentido ascendente)** o seleccione el separador **Salida** y seleccione el enlace de salida que desee editar desde la lista desplegable **Nombre de salida (etapa en sentido descendente)**. Al editar el enlace de entrada y de salida, está configurando la etapa Java Integration como transformador.
3. Especifique una descripción opcional del enlace de entrada o de salida en el separador **General**.
4. Opcional: Pulse **Configurar** para configurar las propiedades adicionales. Según el código de usuario, se visualiza el **Editor de propiedad personalizado**, **Editor de correlaciones de columnas** o el **Importador de metadatos de columna**.
   a. Se necesita especificar valores para las propiedades. Si el código de usuario expone propiedades definidas por el usuario utilizando el método

`Processor.getUserPropertyDefinitions()`, aparece la ventana **Editor de propiedad personalizada**, en la que puede especificar el valor de cada propiedad.

b. Si el código de usuario utiliza JavaBeans en su interfaz (mediante la implementación de los métodos `Processor.getBeanForInput()` y `Processor.getBeanForOutput()`), aparece el **Editor de correlaciones de columnas** de la ventana. Complete el esquema de columna de DataStage basándose en las propiedades de los JavaBeans y luego correlacione las propiedades JavaBeans con las columnas de DataStage. Si crea un trabajo donde los enlaces no contienen columnas, entonces el **Editor de correlaciones de columnas** inicial contiene tablas vacías. Pulse **Examinar objetos** para lanzar la ventana **Seleccionar bean Propiedades** y seleccione las propiedades del bean o los argumentos de función definida por el usuario (UDF) que se vayan a importar en el **Editor de correlaciones de columnas** y, a continuación, pulse **Aceptar**. Pulse **Finalizar**.También puede correlacionar las propiedades JavaBean con las columnas de DataStage existentes, en lugar de crear nuevas columnas.

c. Si su código de usuario implementa los métodos `Processor.getColumnMetadataForInput()` y `Processor.getColumnMetadataForOutput()` en lugar de los métodos `Processor.getBeanForInput()` y `Processor.getBeanForOutput()`, aparece la ventana **Importador de metadatos de columna**. En la ventana **Importador de metadatos de columna**, seleccione los metadatos de la columna para completar el esquema de la columna DataStage a partir de una lista de instancias de `ColumnMetadata` que son devueltas por los métodos `Processor.getColumnMetadataForInput()` y `Processor.getColumnMetadataForOutput()` para cada enlace. Pulse **Examinar objetos** y seleccione los metadatos de la columna que vayan a completarse en el trabajo. Pulse **Finalizar**.

5. Especifique los detalles necesarios en el separador **Propiedades** y el separador **Avanzado**.
6. Pulse **Aceptar** para guardar la configuración que haya especificado.

Configurar definiciones de columna (Java Integration stage)

Puede configurar definiciones de columna para un enlace y también personalizar la cuadrícula de columnas, guardar definiciones de columna para utilizarlas posteriormente y cargar definiciones de columna predefinidas del repositorio.

Procedimiento

1. En el lienzo de diseño de trabajos, efectúe una doble pulsación en el icono **Etapa Java Integration**.
2. Seleccione el separador **Entrada** y seleccione el enlace de entrada que desee editar desde la lista desplegable **Nombre de entrada (etapa en sentido ascendente)** o seleccione el separador **Salida** y seleccione el enlace de salida que desee editar desde la lista desplegable **Nombre de salida (etapa en sentido descendente)**.
3. En el separador **Columnas**, modifique la cuadrícula de columnas para especificar los metadatos que desee definir.
   a. Pulse con el botón derecho del ratón en la cuadrícula y seleccione **Propiedades** en el menú.
   b. En la ventana Propiedades de cuadrícula, seleccione las propiedades que desee visualizar y el orden en que desee que se visualicen. A continuación, pulse **Aceptar**.

4. Especifique definiciones de columna para la tabla utilizando uno de los métodos siguientes:

Opción	Descripción
Método 1	1. En la columna **Nombre de la columna**, efectúe una doble pulsación dentro de la celda apropiada y escriba un nombre de columna. 2. Para cada celda de la fila, efectúe una doble pulsación dentro de la celda y seleccione las opciones que desee. 3. En la columna **Descripción**, efectúe una doble pulsación dentro de la celda apropiada y escriba una descripción.
Método 2	1. Pulse con el botón derecho del ratón en la cuadrícula y seleccione **Editar fila** en el menú. 2. En la ventana Editar metadatos de columna, especifique los metadatos de la columna.

5. Para compartir metadatos entre varias columnas, seleccione las columnas cuyos metadatos desee compartir.
   a. Pulse con el botón derecho del ratón y seleccione **Propagar valores**.
   b. En la ventana Propagar valores de columnas, seleccione las propiedades que desee que las columnas seleccionadas compartan.
6. Para guardar las definiciones de columna como una definición de tabla en el repositorio, pulse **Guardar**.
   a. Especifique la información apropiada en la ventana Guardar definición de tabla y luego pulse **Aceptar**.
   b. En la ventana Guardar definición de tabla como, seleccione la carpeta en la que desee guardar la definición de tabla y luego pulse **Guardar**.
7. Para cargar definiciones de columna del repositorio, pulse **Cargar**.
   a. En la ventana Definiciones de tabla, seleccione la definición de tabla que desee cargar y luego pulse **Aceptar**.
   b. En la ventana Seleccionar columnas, utilice los botones de flecha para mover columnas de la lista **Columnas disponibles** a la lista **Columnas seleccionadas**. Pulse **Aceptar**.

Asociar índices de enlaces con enlaces (etapa Java Integration)

Puede asociar índices de enlace con enlaces cuando haya varios enlaces de entrada o salida.

Procedimiento

1. En el lienzo de diseño de trabajos, efectúe una doble pulsación en el icono de enlace de entrada o de salida (que se conecta con el icono de la **etapa Java de integración**) para el que desee cambiar el orden.
2. Seleccione el separador **Orden de los enlaces**. Este separador le permite especificar cómo los enlaces de entrada o salida correspondan a etiquetas de enlace numérico. La etiqueta de enlace numérico corresponde al índice del enlace al que se ha accedido a través de la API de la etapa Java Integration. Para reorganizar los enlaces, seleccione el enlace de entrada o de salida necesarios y pulse el botón de flecha arriba o botón de flecha abajo.

3. Pulse **Aceptar** para guardar los detalles de correlación.

Buscar datos utilizando enlaces de referencia

Puede buscar datos utilizando un enlace de referencia para enlazar la etapa Java Integration con una etapa Lookup.

Acerca de esta tarea

Puede seleccionar el tipo de búsqueda como Normal o Dispersa en el editor de etapas. En el caso de una búsqueda normal, el código Java no necesita ninguna implementación, ya que el enlace de referencia de salida se trata del mismo modo que el enlace de secuencia de salida. Sin embargo, la búsqueda dispersa cambia internamente la topología de enlace de manera que el conector contenga un enlace de entrada, un enlace de salida y un enlace de rechazo opcional. Cuando seleccione la búsqueda Dispersa, asegúrese de que sólo hay un enlace de referencia de salida conectado a la etapa Java Integration.

Procedimiento

1. En el lienzo de diseño de trabajos, arrastre un icono de etapa **Java Integration** y un icono de **etapa Lookup** al lienzo de diseño de trabajos. (La **Etapa Lookup** se encuentra en la categoría **Proceso** del menú **Paleta**.)
2. Una las etapas arrastrando un enlace de la etapa Java Integration a la etapa Lookup.
3. Pulse el botón derecho del ratón en el enlace y seleccione la opción **Convertir a referencia** del menú. La línea cambia a una línea con guiones para indicar que el enlace es un enlace de referencia.
4. Pulse **Aceptar**.
5. Efectúe una doble pulsación en el icono de etapa Java Integration.
6. En el separador **Propiedades** del separador **Etapa**, seleccione **Normal** o **Dispersa**.
7. Pulse **Aceptar**.

Capítulo 4. Migrar los trabajos de Java Pack de herencia a la etapa Java Integration

Utilice la **Herramienta de migración de conectores** para ver y migrar los trabajos de Java Pack de herencia a la etapa Java Integration. La etapa Java Integration es compatible con el Java Pack actual de manera que las clases Java existentes funcionan sin modificar con la etapa Java Integration. La etapa Java Integration también da soporte a la API existente que se expone en el Java Pack existente. La API de Java Pack no se extenderá más allá de su capacidad actual. Las vías de migración soportadas son de Java Transformer (en paralelo) a la etapa Java Integration (en paralelo) y del cliente Java (en paralelo) a la etapa Java Integration (en paralelo).

Procedimiento

1. Seleccione **Inicio** > **Programas** > **IBM InfoSphere Information Server** > **Herramienta de migración de conectores**.
2. Inicie sesión especificando el nombre de host donde se ejecuta el cliente del Diseñador y la identificación del proyecto donde están definidos los trabajos de Java Pack.
3. Seleccione **Ver** > **Ver todos los trabajos migrables** para visualizar todos los trabajos que están en el proyecto y que se pueden migrar para utilizar la etapa Java Integration. Los trabajos que no incluyan ninguna etapa que se pueda migrar quedan excluidos de la lista de trabajos. Un icono indica el estado de cada trabajo. Un icono atenuado en gris indica que no se puede migrar el trabajo. Un icono de color gris con un signo de interrogación indica que es posible que el trabajo se pueda migrar correctamente.
4. Lleve a cabo los pasos siguientes para analizar trabajos:
   a. Marque el trabajo en la lista de trabajos.
   b. Expanda el trabajo en la lista de trabajos para ver las etapas del trabajo.
   c. Seleccione uno o varios trabajos y pulse **Analizar**.

   Depués del análisis, el color del icono de trabajo, etapa o propiedad indica si se puede migrar o no.

   Icono verde
   Un icono de color verde indica que el trabajo, la etapa o la propiedad se puede migrar.

   Icono rojo
   Un icono rojo indica que no se puede migrar el trabajo o la etapa.

   Icono naranja
   Un icono naranja indica que un trabajo o etapa se puede migrar parcialmente y que una propiedad de una etapa no tiene un equivalente en la etapa Java Integration.

   Icono gris
   Un icono de color gris indica que el trabajo o la etapa no es apto para la migración.

   La herramienta de migración de conectores muestra los nombres de propiedad internos, en lugar de los nombres que las etapas muestran. Para ver una tabla que contenga el nombre interno y el nombre de visualización correspondiente

para cada propiedad, en el cliente de IBM InfoSphere DataStage and QualityStage Designer, abra la carpeta Tipos de etapas en el árbol de repositorio. Efectúe una doble pulsación en el icono de etapa y pulse el separador **Propiedades** para ver las propiedades de la etapa.

5. Pulse **Preferencias** y seleccione cómo desea migrar el trabajo:
   - Pulse **Clonar y migrar trabajo clonado** para hacer una copia del trabajo y, a continuación, migrar la copia. El trabajo original queda intacto.
   - Pulse **Hacer copia de seguridad del trabajo y migrar el trabajo original** para realizar una copia del trabajo y luego migrar el trabajo original.
   - Pulse **Migrar trabajo original** para migrar el trabajo sin realizar una copia de seguridad.
6. Seleccione los trabajos y etapas para migrar, y pulse **Migrar**.

   Los trabajos seleccionados se migran a la etapa Java Integration y se colocan en la misma carpeta que el trabajo original. Si el registro está habilitado, se crea un archivo de registro que incluye un informe de la tarea de migración. Una vez migrado correctamente, aparece una marca de selección de color verde al lado del nombre del trabajo en la lista Trabajos para indicar que el trabajo se ha migrado.

Capítulo 5. Compilar y ejecutar trabajos de la etapa Java Integration

Puede compilar los trabajos de la etapa Java Integration en scripts ejecutables que puede planificar y ejecutar.

Procedimiento

1. En el cliente InfoSphere DataStage and QualityStage Designer, abra el trabajo que desee compilar.
2. Pulse el botón **Compilar**.
3. Si el área Estado de la compilación muestra errores, edite el trabajo para resolver los errores. Una vez resueltos los errores, pulse el botón **Recompilar**.
4. Cuando el trabajo se compile satisfactoriamente, pulse el botón **Ejecutar** y especifique las opciones de ejecución del trabajo:
   a. Especifique los parámetros de trabajo necesarios.
   b. Pulse el botón **Validar** para verificar que el trabajo se ejecuta satisfactoriamente sin, de hecho, extraer, convertir o grabar ningún dato.
   c. Pulse el botón **Ejecutar** para extraer, convertir o grabar datos.
5. Para ver los resultados de la validación o ejecución de un trabajo.
   a. En el Cliente del Diseñador, seleccione **Herramientas** > **Ejecutar Director** para abrir el Cliente del Director.
   b. En la columna Estado, verifique que el trabajo haya sido validado o completado satisfactoriamente.
   c. Si el trabajo o la validación finalizan anómalamente, seleccione **Ver** > **Registro** para identificar cualquier problema de ejecución.
6. Si el trabajo tiene problemas de ejecución, corrija los problemas, recompile, valide (opcionalmente) y ejecute el trabajo hasta que se complete satisfactoriamente.

Capítulo 6. Resolución de problemas de la etapa Java Integration

Hay varios errores comunes que son específicos de la etapa Java Integration.

Errores en el editor de etapas

Al utilizar la etapa Java Integration, es posible que encuentre algunos errores de la interfaz gráfica de usuario (GUI).

Síntomas

- Al pulsar **Configurar** en el **Editor de propiedades de la etapa Java Integration**, puede recibir el siguiente mensaje de aviso:

  ```
  No se ha podido instanciar la clase de derivador recursos
  ```

- Al pulsar **Seleccionar** en la propiedad de etapa **Uso>Clase de usuario**, puede recibir el siguiente mensaje de aviso:

  ```
  No se ha podido enviar la solicitud al manejador: el agente en HOST123:31531
  no está disponible.
  ```

Resolución del problema

Verifique que el agente ASB está funcionando y reinicie el agente ASB si es necesario.

El archivo de registro del agente ASB puede ayudarle a entender los errores que se producen en el editor de etapas. El archivo se llama `asb-agent-xx.out` y se encuentra en la carpeta `ASBNode/logs` si `com.ibm.iis.cas.level=ALL` se ha añadido al archivo `ASBNode/conf/asbagent-logging.properties`.

Errores de tiempo de ejecución

Después de configurar y compilar el trabajo de la etapa Java Integration, es posible que encuentre algunos errores de tiempo de ejecución.

Problemas relacionados con la configuración de la etapa

Al ejecutar un trabajo que incluye una etapa Java Integration, se recibe un error que está relacionado con la configuración de la etapa.

Síntomas

Cuando no se puede encontrar una clase de usuario `userClass` en la vía de acceso de clases especificada en el **Editor de propiedades de la etapa Java Integration**, se puede producir el siguiente error:

```
com.ascential.e2.common.CC_Exception: java.lang.ClassNotFoundException:
userClass
at java.net.URLClassLoader.findClass(URLClassLoader.java:588)
 at java.lang.ClassLoader.loadClassHelper(ClassLoader.java:743)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:711)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:690)
 at com.ibm.is.cc.javastage.connector.CC_JavaConnection.connect
(CC_JavaConnection.java:155)
 at com.ibm.is.cc.javastage.connector.CC_JavaConnection.connect
(CC_JavaConnection.java: 250)
```

Si una clase de usuario no pasa la prueba de conexión durante la inicialización, se puede producir el siguiente error para una clase que no se hereda de la clase `com.ibm.is.cc.javastage.api.Processor` o de la clase `com.ascentialsoftware.jds.Stage`.

```
La prueba de conexión ha fallado (PXBridgeOp::negotiate, file pxbridge.c,
 line 1,684)
```

Resolución del problema

Añada el archivo de clase de usuario que se hereda de la clase Proccesor o Stage, o un archivo `.jar` que contenga la clase, a un directorio al que pueda acceder el motor paralelo. En la etapa Java Integration, establezca la propiedad **Uso - Java - Vía de acceso de clases** en el nombre de vía de acceso que contiene el archivo.

Para confirmar que se ha especificado la vía de acceso de clases correcta, pulse **Seleccionar** en la propiedad **Uso - Clase de usuario** y seleccione la clase de usuario.

Problemas relacionados con la JVM

Puede encontrar un error relacionado con la creación de una JVM al ejecutar una etapa Java Integration.

Síntomas

El conductor se ejecuta como un proceso único y crea una JVM individual. Las opciones para la JVM se especifican en una de las etapas Java Integration del trabajo creado. Cuando ejecute un trabajo que tiene varias etapas Java Integration, puede encontrarse con el siguiente error:

```
JVMDUMP039I Procesando suceso de volcado
"systhrow", detalle "java/lang/OutOfMemoryError"
el 2013/08/15 11:46:11
com.ibm.tools.attach.enable espere, por favor.
JVMDUMP032I La JVM ha solicitado un volcado del sistema utilizando
'C:\IBM\InformationServer\Server\Projects\dstage1\core.20130815.
114611.3932.0001.dmp'
en respuesta a un suceso
```

Este error puede producirse si el tamaño máximo de almacenamiento dinámico es suficiente para una etapa Java Integration pero no para todas las etapas Java Integration.

Resolución del problema

1. Compruebe qué opciones de JVM son necesarias.
2. Establezca una variable de entorno **CC_MSG_LEVEL** en 2.
3. Compile y ejecute el trabajo.

   Puede ver las opciones de JVM en la siguiente sección del registro de trabajo:

   ```
   CC_JNICommon.cpp:(634)
   La JVM se inicia con estas opciones =
   -Dcom.ibm.tools.attach.enable=no
   -Dcom.ibm.is.cc.options=noisfjars
   -Xmx256m -Djava.class.path=
   C:\IBM\InformationServer\Server\DSEngine/../DSComponents/bin/ccapi.jar;.;
   C:\IBM\SQLLIB\java\db2java.zip;C:\IBM\SQLLIB\java\db2jcc.jar;
   C:\IBM\SQLLIB\java\sqlj.zip;C:\IBM\SQLLIB\java\db2jcc_license_cu.jar;
   C:\IBM\SQLLIB\bin;C:\IBM\SQLLIB\java\common.jar
   ```

4. Debe verificar qué cantidad de la JVM es necesaria y establecer la variable de entorno **CC_JVM_OVERRIDE_OPTIONS** con las opciones de JVM después de cambiar el valor del tamaño máximo de almacenamiento dinámico con una opción de JVM -Xmx.

```
$CC_JVM_OVERRIDE_OPTIONS =
-Dcom.ibm.tools.attach.enable=no
-Dcom.ibm.is.cc.options=noisfjars
-Xmx512m -Djava.class.path=
C:\IBM\InformationServer\Server\DSEngine/../DSComponents/bin/ccapi.jar;.;
C:\IBM\SQLLIB\java\db2java.zip;C:\IBM\SQLLIB\java\db2jcc.jar;
C:\IBM\SQLLIB\java\sqlj.zip;C:\IBM\SQLLIB\java\db2jcc_license_cu.jar;
C:\IBM\SQLLIB\bin;C:\IBM\SQLLIB\java\common.jar
```

Problemas relacionados con Java Pack

Puede encontrar un error muy grave al ejecutar una etapa Java Integration migrada desde un trabajo de Java Pack existente.

Síntomas

1. Es posible que encuentre una excepción java.lang.ClassNotFoundException al ejecutar un trabajo de la etapa Java Integration migrado con la Herramienta de migración de conectores aunque el trabajo de Java Pack original no generase ningún error.

```
Java_Client: java.lang.ClassNotFoundException:
myClass.MyTester at
java.net.URLClassLoader.findClass(URLClassLoader.java:588)
at java.lang.ClassLoader.loadClass(ClassLoader.java:743)
at java.lang.ClassLoader.loadClass(ClassLoader.java:711)
at java.lang.ClassLoader.loadClass(ClassLoader.java:690)
at com.ibm.is.cc.javastage.connector.CC_JavaConnection.connect
(CC_JavaConnection.java:155)
at (com.ibm.is.cc.javastage.connector.CC_JavaConnection::connect
(CC_JavaConnection.java: 250)
```

2. Puede encontrar el siguiente error muy grave al ejecutar un trabajo de la etapa Java Integration utilizando la API de Java Pack:

```
JavaPackTransformer: java.lang.IllegalAccessError:
Class com/ibm/is/cc/javastage/connector/CC_JavaPackProcessor
illegally accessing "package private" member of class
com/ascentialsoftware/jds/Stage
at com.ibm.is.cc.javastage.connector.CC_JavaPackProcessor.<init>
(CC_JavaPackProcessor.java: 121)
at com.ibm.is.cc.javastage.connector.CC_JavaConnection.connect
(CC_JavaConnection.java: 167)
```

Resolución del problema

El primer error puede producirse porque, en Java Pack, el directorio $DSHOME se considera el directorio raíz de la vía de acceso relativa especificada en la propiedad de etapa **Vía de acceso de clases del usuario** del **Editor de propiedades de Java Pack**. En la etapa Java Integration, el directorio $DSHOME puede no ser el directorio raíz. La excepción en cuestión se produce si los archivos de clase de usuario sólo se han colocado en ese directorio relativo a $DSHOME.

Para corregir el primer error, debe especificar la vía de acceso absoluta en la propiedad de etapa **Uso - Java - Vía de acceso de clases** del **Editor de propiedades de la etapa Java Integration** o copiar los archivos de clase en el directorio relativo al directorio de proyecto $DSHOME/../Projects/<nombre_proyecto>.

El segundo error puede producirse porque el tiempo de ejecución de la etapa Java Integration podría invocar erróneamente un método Java Pack que no es para la etapa Java Integration. Puede haber especificado un archivo JAR llamado `tr4j.jar` para Java Pack en la vía de acceso de clases.

Para corregir el segundo error, debe eliminar `tr4j.jar`, que es un archivo JAR para Java Pack, de lo siguiente:

- El entorno CLASSPATH especificado en el sistema operativo, las propiedades del proyecto de DataStage y las propiedades del trabajo.
- **Uso - Java - Vías de acceso de clases** en el editor de propiedades de la etapa Java Integration.
- La opción **-classpath** o **-cp** especificada en **Uso - Java - Opción VM** en el **Editor de propiedades de la etapa Java Integration**.

Problemas relacionados con la configuración de la etapa

Al ejecutar un trabajo que incluye una etapa Java Integration, se recibe un error que está relacionado con la configuración de la etapa.

Síntomas

Cuando no se puede encontrar una clase de usuario `userClass` en la vía de acceso de clases especificada en el **Editor de propiedades de la etapa Java Integration**, se puede producir el siguiente error:

```
com.ascential.e2.common.CC_Exception: java.lang.ClassNotFoundException:
userClass
at java.net.URLClassLoader.findClass(URLClassLoader.java:588)
 at java.lang.ClassLoader.loadClassHelper(ClassLoader.java:743)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:711)
 at java.lang.ClassLoader.loadClass(ClassLoader.java:690)
 at com.ibm.is.cc.javastage.connector.CC_JavaConnection.connect
(CC_JavaConnection.java:155)
 at com.ibm.is.cc.javastage.connector.CC_JavaConnection.connect
(CC_JavaConnection.java: 250)
```

Si una clase de usuario no pasa la prueba de conexión durante la inicialización, se puede producir el siguiente error para una clase que no se hereda de la clase `com.ibm.is.cc.javastage.api.Processor` o de la clase `com.ascentialsoftware.jds.Stage`.

```
La prueba de conexión ha fallado (PXBridgeOp::negotiate, file pxbridge.c,
 line 1,684)
```

Resolución del problema

Añada el archivo de clase de usuario que se hereda de la clase Proccesor o Stage, o un archivo `.jar` que contenga la clase, a un directorio al que pueda acceder el motor paralelo. En la etapa Java Integration, establezca la propiedad **Uso - Java - Vía de acceso de clases** en el nombre de vía de acceso que contiene el archivo.

Para confirmar que se ha especificado la vía de acceso de clases correcta, pulse **Seleccionar** en la propiedad **Uso - Clase de usuario** y seleccione la clase de usuario.

Problemas relacionados con la correlación de columnas

Puede encontrar un error muy grave y un aviso al ejecutar un trabajo de la etapa Java Integration que utilice JavaBeans.

Síntomas

1. Puede encontrar el siguiente error muy grave al ejecutar un trabajo de la etapa Java Integration que utilice JavaBeans:

```
Com.ascential.e2.common.CC_Exception: El bean Java no se ha podido establecer.
Nombre de clase: userClass
La correlación de columnas a propiedades de bean no se ha definido.
Inicie el editor de correlaciones de columnas para definir las correlaciones
 de columnas para el bean.
Nombre de clase de bean: userInputBean
at com.ibm.is.cc.javastage.connector.CC_JavaLinkImpl.setBeanInfo
(CC_JavaLinkImpl.java: 146)
at com.ibm.is.cc.javastage.connector.CC_JavaInputLinkImpl.setBeanClass
(CC_JavaInputLinkImpl.java: 128)
at com.ibm.is.cc.javastage.connector.CC_JavaAdapter.updateLinkWithBeanInfo
(CC_JavaAdapter.java: 208)
at com.ibm.is.cc.javastage.connector.CC_JavaAdapter.preRunNode
(CC_JavaAdapter.java: 421)
```

2. Puede encontrar el siguiente aviso al ejecutar un trabajo de la etapa Java Integration que utilice JavaBeans:

```
com.ascential.e2.common.CC_Exception: El bean Java no se ha podido establecer.
Nombre de clase: userOutputBean
Se ha producido una no coincidencia de tipo. Nombre de columna: aaa,
Clase de tipo de columna: java.sql.Date, Nombre de propiedad de bean: bbb,
Clase de tipo de propiedad de bean: java.lang.String,
Nombre de clase de bean: userOutputBean
at com.ibm.is.cc.javastage.connector.CC_JavaLinkImpl.setBeanInfo
(CC_JavaLinkImpl.java: 146)
at com.ibm.is.cc.javastage.connector.CC_JavaOutputLinkImpl.setBeanClass
(CC_JavaOutputLinkImpl.java: 278)
at com.ibm.is.cc.javastage.connector.CC_JavaAdapter.updateLinkWithBeanInfo
(CC_JavaAdapter.java: 218)
at com.ibm.is.cc.javastage.connector.CC_JavaAdapter.preRunNode
(CC_JavaAdapter.java: 421)
```

Resolución del problema

Este error muy grave puede ser debido a que ha especificado `userInputBean` para un enlace de entrada codificando `getBeanForInput()` en su clase de usuario *userClass*, pero no ha establecido la correlación de columnas para ese enlace.

El aviso se puede producir porque puede haber especificado `userOutputBean` para un enlace de salida y también haber configurado ese enlace, pero puede que haya seleccionado columna DataStage `aaa` incorrecta, cuyo tipo de columna es `java.sql.Date`, y la haya correlacionado con la propiedad de bean Java `bbb`, cuyo tipo es `java.lang.String`.

Para solucionar los problemas, realice las siguientes acciones:

1. Verifique la propiedad de enlace **Uso - Correlación de columnas**.
2. Pulse **Configurar** para actualizar la propiedad de enlace.
3. Puede ver el valor actual en el **Editor de correlaciones de columnas**. Modifique la correlación seleccionando el valor necesario en la lista desplegable de cada columna de DataStage.
4. Si desea ver todas las propiedades de JavaBean, pulse **Examinar objetos**. Puede ver y seleccionar las propiedades pulsando los recuadros de selección en el diálogo **Seleccionar propiedades de bean**.
5. Si desea guardar las nuevas selecciones, pulse **Aceptar** en el diálogo **Seleccionar propiedades de bean** y pulse **Finalizar** en el **Editor de correlaciones de columnas**.

Problemas relacionados con enlaces

Puede encontrar un error muy grave al ejecutar un trabajo de la etapa Java Integration con varios enlaces de entrada o enlaces de rechazo.

Síntomas

1. Puede encontrar el siguiente error muy grave al ejecutar un trabajo de la etapa Java de integración con varios enlaces de entrada:

   ```
   [Enlace de entrada 0] com.ascential.e2.common.CC_Exception:
   El código de usuario ha devuelto la clase de bean "YourInput" para este enlace,
   pero no coincide con "MyInput", especificado en el diseño.
   Inicie el editor de correlaciones de columnas para volver a configurar las
   correlaciones de columnas.
   at com.ibm.is.cc.javastage.connector.CC_JavaInputLinkImpl.setBeanClass
   (CC_JavaInputLinkImpl.java: 123)
   at com.ibm.is.cc.javastage.connector.CC_JavaAdapter.updateLinkWithBeanInfo
   (CC_JavaAdapter.java: 208)
   at com.ibm.is.cc.javastage.connector.CC_JavaAdapter.preRunNode
   (CC_JavaAdapter.java: 421)
   ```

2. Puede encontrar el siguiente error muy grave al ejecutar un trabajo de la etapa Java de integración con enlaces de rechazo:

   ```
   com.ascential.e2.common.CC_Exception: El enlace de rechazo no está configurado.
   Configure el enlace de rechazo y guárdelo.
   at com.ibm.is.cc.javastage.connector.
   CC_JavaRecordDataSetConsumer.setRejectManager
   (CC_JavaRecordDataSetConsumer.java: 182)
   at com.ibm.is.cc.javastage.connector.CC_JavaAdapter.getRejectDataSetProducer
   (CC_JavaAdapter.java: 349)
   ```

Resolución del problema

El primer error se puede producir debido a que podría haber cambiado el orden de los enlaces después de configurar la correlación de columnas.

Para corregir el error, pulse el botón **Configurar** en el **Editor de propiedades de la etapa Java Integration** y configure de nuevo la correlación de columnas. Siempre debe verificar el orden de los enlaces en el separador **Orden de los enlaces** del **Editor de propiedades de la etapa Java Integration** si el trabajo tiene varios enlaces de entrada y de salida para evitar cualquier incoherencia entre el diseño de trabajo y el código Java.

Para corregir el segundo error, abra el separador **Rechazar** y confirme que se ha seleccionado la fila de entrada correcta en la propiedad de etapa **Enlace de entrada de rechazo** en el **Editor de propiedades de la etapa Java Integration**.

Capítulo 7. Variables de entorno: etapa Java Integration

La etapa Java Integration utiliza las siguientes variables de entorno.

CC_IGNORE_TIME_LENGTH_AND_SCALE

Establezca esta variable de entorno para cambiar el comportamiento del conector en el lienzo paralelo.

Cuando esta variable de entorno se establece en 1, el conector que se ejecuta con el motor paralelo hace caso omiso de la longitud y escala especificadas para la columna de indicación de fecha y hora. Por ejemplo, cuando el valor de esta variable de entorno no está establecido y si la longitud de la columna de indicación de fecha y hora es 26 y la escala es 6, el conector en el lienzo paralelo considera que la indicación de fecha y hora tiene una resolución de microsegundos. Cuando el valor de esta variable de entorno se establece en 1, el conector en el lienzo paralelo no considera que la indicación de fecha y hora tiene una resolución de microsegundos a menos que la propiedad ampliada de microsegundos esté establecida, aunque la longitud de la columna de indicación de fecha y hora sea 26 y la escala sea 6.

CC_JNI_EXT_DIRS

Establezca esta variable de entorno para añadir un prefijo a la vía de acceso de clases de la propiedad del sistema `java.ext.dirs`.

Cuando se establece el valor de esta variable de entorno, se añade un prefijo a la vía de acceso de clases de la propiedad del sistema `java.ext.dirs`.

CC_JVM_OPTIONS

Establezca esta variable de entorno para especificar los argumentos de JVM que se utilizan cuando se ejecuta un trabajo.

Cuando se especifica esta variable, tiene prioridad sobre el valor de los argumentos de JVM predeterminados para los conectores basados en Java. Por ejemplo, si establece **`CC_JVM_OPTIONS`** como `-Xmx512M`, el tamaño máximo de almacenamiento dinámico se establece en 512 MB cuando se crean instancias de JVM para el conector.

CC_JVM_OVERRIDE_OPTIONS

Establezca esta variable de entorno para alterar temporalmente las opciones de JVM del nodo conductor para que pueda evitar o solucionar un posible conflicto.

En el nodo conductor en un trabajo paralelo, los conectores Java se inicializan para la reconciliación de esquema. Por lo tanto, todos los conectores Java en un trabajo se inicializan en la misma JVM. En un trabajo individual, varias etapas podrían estar desarrolladas en Java. Cada una de estas etapas puede definir opciones de JVM, tales como vía de acceso de clases, propiedad del sistema, tamaño de almacenamiento dinámico, etcétera. Si dos etapas se ejecutan en la misma JVM física, las opciones de JVM pueden entrar en conflicto entre sí.

CC_MSG_LEVEL

Establezca esta variable de entorno para especificar la gravedad mínima de los mensajes que el conector notifica en el archivo de registro.

Con el valor predeterminado de 3, los mensajes informativos y los mensajes de una gravedad superior se notifican en el archivo de registro.

La siguiente lista contiene los valores válidos:

- 1 - Rastreo
- 2 - Depuración
- 3 - Informativo
- 4 - Aviso
- 5 - Error
- 6 - Muy grave

JAVASTAGE_API_DEBUG

Establezca esta variable de entorno para controlar si los mensajes de depuración que se especifican mediante la API `com.ibm.is.cc.javastage.api.Logger.debug()` se notifican en el archivo de registro del cliente del Director.

Cuando el valor de esta variable es 1, los mensajes de depuración se notifican en el archivo de registro. Si el valor de esta variable de entorno no es 1, todos los mensajes de depuración se ignoran. Para obtener más detalles, consulte la información de Javadoc para la API de la etapa Java Integration.

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

Esta documentación utiliza caracteres especiales para definir la sintaxis de la línea de mandatos.

Los siguientes caracteres especiales definen la sintaxis de la 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 información que se excluye mutuamente. 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 único uso del mandato.

{ } Delimita un conjunto de argumentos que se excluyen mutuamente cuando se necesita uno de los argumentos. Si los argumentos son opcionales, se escriben entre delimitadores ([ ]).

Nota:

- El número máximo de caracteres de un argumento es de 256.
- Escriba los valores de argumentos que tengan espacios incrustados entre comillas simples o dobles.

Por ejemplo:

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

El argumento *origen* es el único argumento necesario para el mandato **wsetsrc**. Los delimitadores de los otros argumentos indican que dichos argumentos son opcionales.

wlsac *[formato -l | -f] [clave...] perfil*

En este ejemplo, los argumentos de formato -l y -f se excluyen mutuamente y son opcionales. El argumento *perfil* es necesario. El argumento *clave* es opcional. La elipsis (...) que sigue al argumento *clave* indica que puede especificar varios nombres de clave.

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

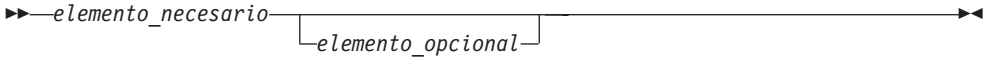
En este ejemplo, los argumentos rule_pack y rule_set se excluyen mutuamente, pero debe especificarse uno de ellos. Además, los puntos suspensivos (...) indican que puede especificar varios paquetes de reglas y conjuntos de reglas.

Apéndice C. Cómo leer diagramas de sintaxis

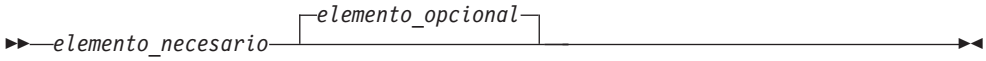
Las reglas siguientes se aplican a los diagramas de sintaxis que se utilizan en esta documentación:

- Lea los diagramas de sintaxis de izquierda a derecha y de arriba abajo, siguiendo el recorrido de la línea. Se utilizan los convenios siguientes:
  - El símbolo >>--- indica el inicio 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 de un diagrama de sintaxis.
- Los elementos necesarios aparecen en la línea horizontal (la línea principal).

- Los elementos opcionales aparecen debajo de la línea principal.

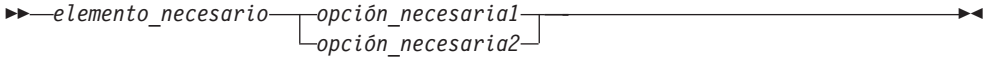


  Si aparece un elemento opcional sobre la línea principal, dicho elemento no tendrá efecto sobre el elemento de sintaxis y sólo se utilizará para facilitar la lectura.

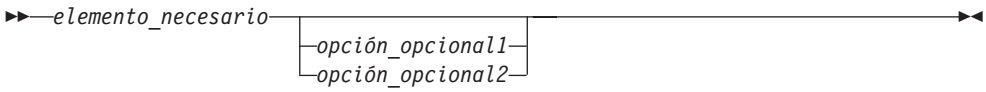


- Si se puede elegir entre dos o más elementos, éstos aparecerán apilados verticalmente.

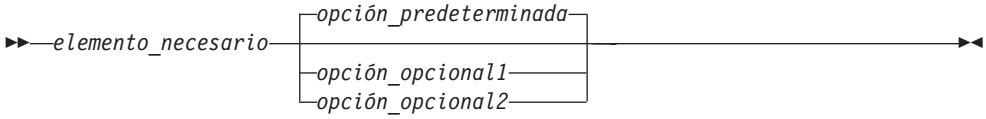
  Si se debe elegir uno de los elementos, un elemento de la pila aparece en la línea principal.



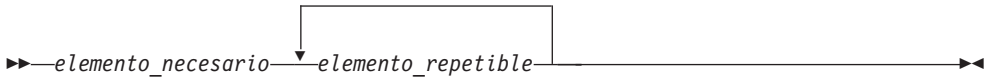
  Si la elección de uno de los elementos es opcional, toda la pila aparecerá por debajo de la línea principal.



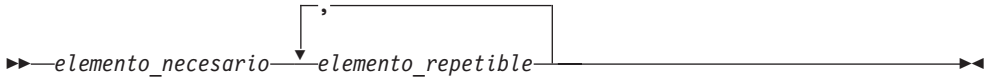
  Si uno de los elementos es el predeterminado, aparecerá por encima de la línea principal y las opciones restantes se mostrarán por debajo.



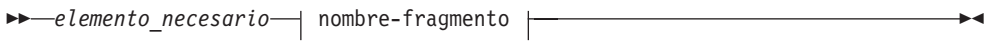
- Una flecha que vuelve hacia la izquierda, sobre la línea principal, indica un elemento que se puede repetir.



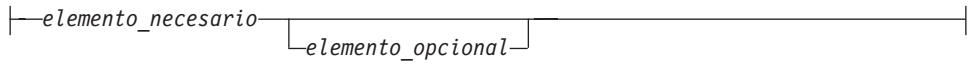
  Si la flecha de repetición contiene una coma, los elementos repetidos se deben separar mediante una coma.



  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.
- Las notas a pie de página se muestran mediante 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 5. 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 6. 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 6. 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

SC43-1238-00