Guía del desarrollador de IBM SPSS Modeler 16 CLEF

Nota

Antes de utilizar esta información y el producto al que hace referencia, lea la información del apartado "Avisos" en la página 325.

Información del producto

Esta edición se aplica a la versión 16, release 0, modificación 0 de IBM(r) SPSS(r) Modeler y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

Contenido

Prefacio

IBM® SPSS Modeler es el conjunto de programas de minería de datos de IBM Corp. orientado a las empresas. SPSS Modeler ayuda a las organizaciones a mejorar la relación con sus clientes y los ciudadanos a través de la comprensión profunda de los datos. Las organizaciones utilizan la comprensión que les ofrece SPSS Modeler para retener a los clientes más rentables, identificar las oportunidades de venta cruzada, atraer a nuevos clientes, detectar el fraude, reducir el riesgo y mejorar la prestación de servicios del gobierno.

La interfaz visual de SPSS Modeler invita a la pericia empresarial específica de los usuarios, lo que deriva en modelos predictivos más eficaces y la reducción del tiempo necesario para encontrar soluciones. SPSS Modeler ofrece muchas técnicas de modelado tales como predicciones, clasificaciones, segmentación y algoritmos de detección de asociaciones. Una vez que se crean los modelos, IBM SPSS Modeler Solution Publisher permite su distribución en toda la empresa a los encargados de tomar las decisiones o a una base de datos.

Acerca de IBM Business Analytics

El software IBM Business Analytics ofrece información completa, coherente y precisa en la que confían los encargados de la toma de decisiones para mejorar el rendimiento comercial. Un conjunto integral de inteligencia empresarial, análisis predictivo, rendimiento financiero y gestión de estrategias y aplicaciones de análisis que ofrece una perspectiva clara, inmediata e interactiva del rendimiento actual y la capacidad de predecir resultados futuros. En combinación con extensas soluciones sectoriales, prácticas probadas y servicios profesionales, las organizaciones de cualquier tamaño pueden conseguir el máximo de productividad, automatizar las decisiones de forma fiable y alcanzar mejores resultados.

Como parte de esta familia, el software de análisis predictivo de IBM SPSS ayuda a las organizaciones a predecir eventos futuros y actuar proactivamente según esa información para lograr mejores resultados comerciales. Los clientes comerciales, gubernamentales y académicos de todo el mundo confían en la tecnología de IBM SPSS como ventaja ante la competencia para atraer, retener y hacer crecer a los clientes, reduciendo al mismo tiempo el fraude y el riesgo. Al incorporar el software de IBM SPSS en sus operaciones diarias, las organizaciones se convierten en empresas predictivas, capaces de dirigir y automatizar decisiones para alcanzar los objetivos comerciales y lograr una ventaja considerable sobre la competencia. Para obtener más información o contactar con un representante, visite http://www.ibm.com/spss.

Asistencia técnica

Hay asistencia técnica disponible para los clientes de mantenimiento. Los clientes podrán ponerse en contacto con el servicio de asistencia técnica si desean recibir ayuda sobre la utilización de los productos de IBM Corp. o sobre la instalación en los entornos de hardware admitidos. Para ponerse en contacto con el servicio de asistencia técnica, consulte el sitio web de IBM Corp. en http://www.ibm.com/support. Tenga a mano su acuerdo de asistencia y esté preparado para identificarse a sí mismo y a su organización al solicitar ayuda.

Capítulo 1. Resumen

Introducción a CLEF

Component-Level Extension Framework (CLEF) es una función que permite añadir extensiones definidas por el usuario a las funciones estándar de IBM SPSS Modeler. Normalmente, una extensión suele incluir una biblioteca compartida, por ejemplo, una rutina de procesamiento de datos o un algoritmo de modelado, que se añade a IBM SPSS Modeler y que está disponible como una nueva entrada del menú o como un nuevo nodo en la paleta de nodos.

Para ello, IBM SPSS Modeler requiere detalles del programa personalizado, como su nombre, los parámetros del comando que se debe pasar, cómo IBM SPSS Modeler debe presentar las opciones al programa y los resultados al usuario, etcétera. Para proporcionar esta información, proporciona un archivo en formato XML, denominado **archivo de especificación**. IBM SPSS Modeler traduce la información de este archivo a una nueva entrada del menú o definición del nodo.

Algunas de las ventajas de utilizar CLEF son:

- Proporcionar un entorno fácil de usar, muy flexible y robusto que permite a los ingenieros, asesores y usuarios finales integrar nuevas características en IBM SPSS Modeler.
- Garantizar que los módulos de extensión pueden tener la apariencia y comportarse como los módulos nativos de IBM SPSS Modeler.
- Permitir que los nodos de extensión se ejecuten con una velocidad y eficacia lo más parecidas a las de los nodos nativos de IBM SPSS Modeler.

Arquitectura del sistema

Como el propio IBM SPSS Modeler, CLEF utiliza una arquitectura de dos niveles cliente/servidor, en la que los niveles pueden estar en el mismo ordenador o en ordenadores diferentes.

Componentes de cliente

A continuación se muestran los componentes del nivel de cliente.

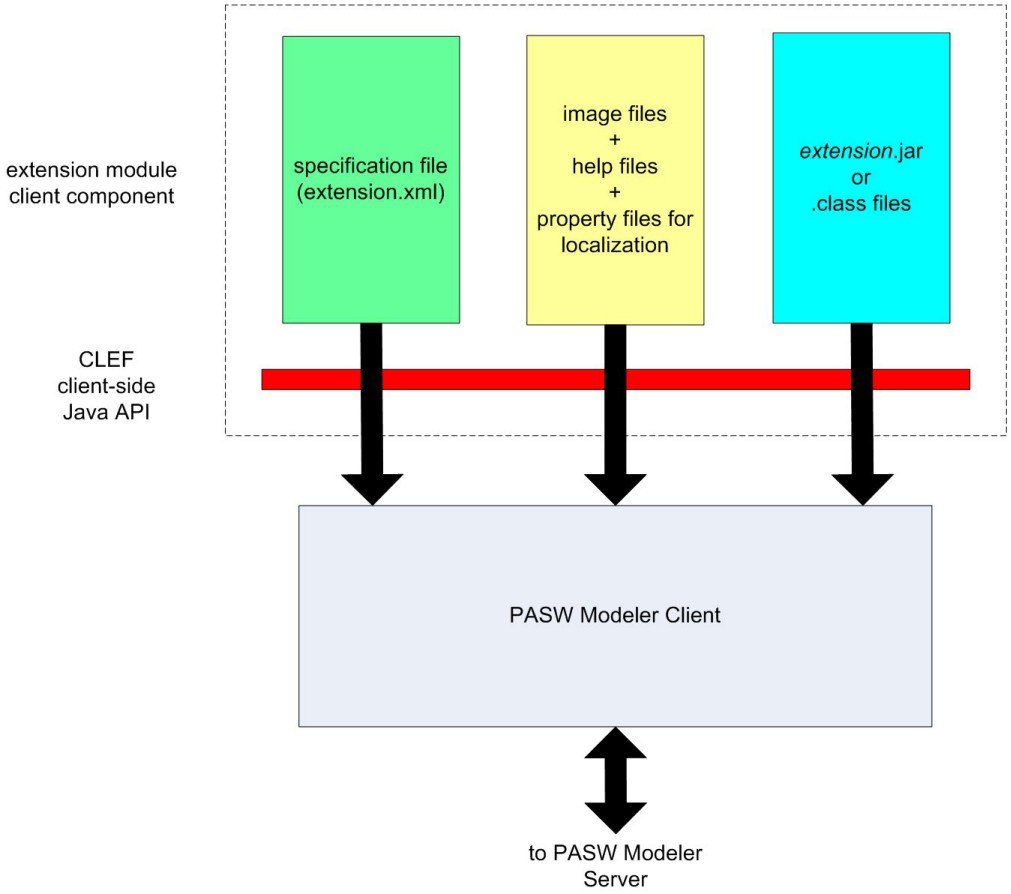

Figura 1. Componentes de cliente

- **Archivo de especificación.** Enumera las propiedades, formatos, cambios del modelo de datos, controles y otras características que se definen por la extensión.
- **Archivos de imagen.** Contienen las imágenes utilizadas para identificar un nodo en la extensión.
- **Archivos de ayuda.** Se utilizan para mostrar información de ayuda acerca de la extensión.
- **Archivos de propiedades.** Contienen cadenas de texto con nombres, etiquetas y mensajes representadas por la extensión en la pantalla.
- **Archivos de Java .jar o .class.** Contienen los recursos de Java que utiliza la extensión.
- **Interfaz de programación de la aplicación de Java (API).** Se puede utilizar por las extensiones que requieren controles adicionales, componentes de interfaz de usuario o interactividad que no proporciona directamente el archivo de especificación.

Componentes de servidor

A continuación se muestran los componentes del nivel de servidor.

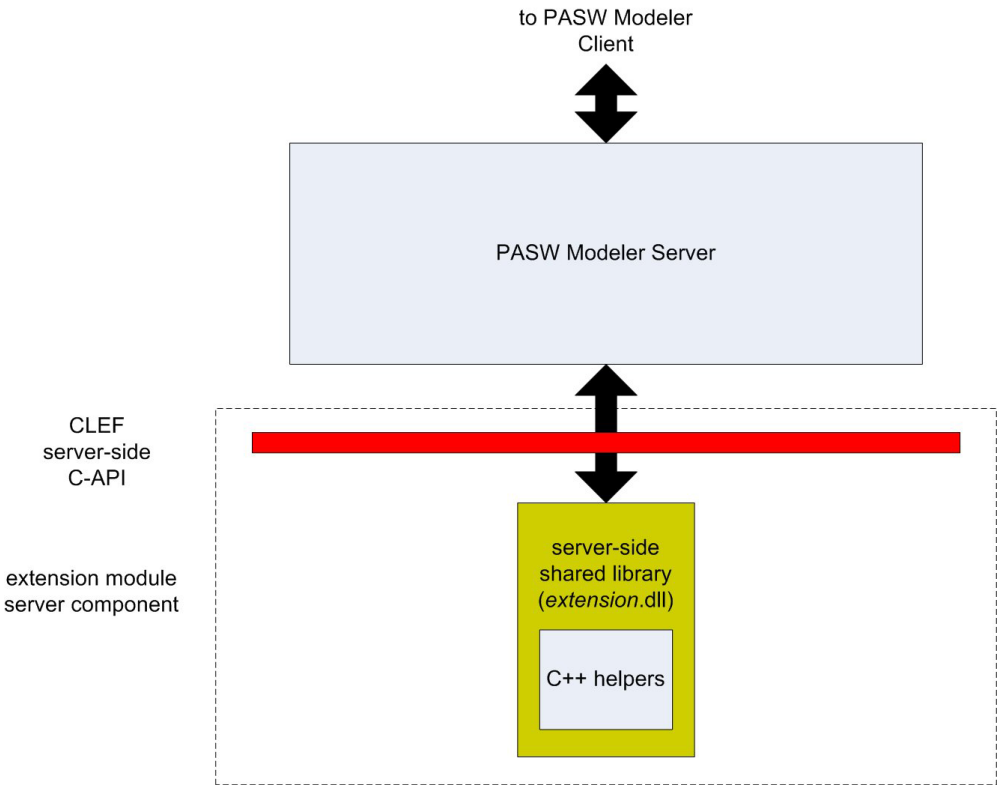

Figura 2. Componentes de servidor

- **API basada en C para bibliotecas compartidas.** Trata aspectos como la configuración y ejecución de los parámetros, la persistencia de los parámetros, comentarios sobre la ejecución, control de trabajos (por ejemplo, interrupción de la ejecución), generación de SQL y objetos devueltos.
- **Biblioteca compartida del servidor.** Una biblioteca de enlaces dinámica (DLL) que admite la ejecución del nodo. Los Aplicaciones de ayuda C++ son encapsuladores de algunas de las API basadas en C que se proporcionan como código fuente y que se pueden compilar fácilmente en un módulo C++ de CLEF.

Características de CLEF

Las secciones siguientes presentan las características clave de CLEF:

- Archivo de especificación
- Nodos
- Modelo de datos
- Archivos de entrada y resultado
- Interfaces de programación de la aplicación (API)

Archivo de especificación

El archivo de especificación de CLEF es un archivo XML que contiene especificaciones estructuradas que describen el comportamiento de la nueva extensión. Un archivo de especificación describe:

- Recursos compartidos que necesita la extensión (por ejemplo, recursos de texto localizado y bibliotecas compartidas del servidor).
- Definiciones comunes, como tipos de archivo o tipos de propiedad.
- Objetos nuevos que el usuario final puede utilizar, como nodos y modelos de resultados.

Cuando IBM SPSS Modeler se inicia, los archivos de especificación se cargan de la ubicación en la que se encuentran, de forma que las características que se definen en los archivos están disponibles de forma inmediata.

Si desea obtener más información, consulte Capítulo 4, “Archivo de especificación”, en la página 31.

Nodos

Si añade una extensión a IBM SPSS Modeler que implementa un nuevo nodo, en primer lugar, debe decidir el tipo de nodo que va a crear (por ejemplo, si el nodo genera un modelo o simplemente transforma los datos). Para obtener más información, consulte el tema “Conceptos básicos sobre nodos” en la página 9.

Después de crear el archivo de especificación y las clases de Java y bibliotecas compartidas necesarias, copie los archivos en ubicaciones concretas desde donde IBM SPSS Modeler pueda leerlos. La próxima vez que inicie IBM SPSS Modeler, el nuevo nodo se añadirá a la paleta correspondiente y estará listo para su uso.

Data Model

El **modelo de datos** representa la estructura de los datos que fluyen por la ruta de IBM SPSS Modeler. Al describir los datos en ese punto de la ruta, el modelo se corresponde con la información del nodo Tipo representado. Incluye los nombres de los campos existentes en un punto concreto de la ruta y describe su tipo.

Cuando añada algún nodo a IBM SPSS Modeler, tenga en cuenta la forma en que el modelo de datos que se ha pasado al nodo afecta al comportamiento de éste. Por ejemplo, un nodo Derivar toma un modelo de datos de entrada, le añade un campo nuevo y genera un modelo de datos de salida que se pasa al siguiente nodo de la ruta de IBM SPSS Modeler. En contraste, un nodo Gráfico toma un modelo de datos de entrada y no genera un modelo de datos de salida porque los datos no se pasan a ningún nodo posterior. IBM SPSS Modeler debe saber lo que sucederá con el modelo de datos de forma que los nodos posteriores puedan presentar la información correcta sobre los campos que están disponibles. La información del modelo de datos que se encuentra en el archivo de especificación proporciona a IBM SPSS Modeler la información necesaria para mantener la coherencia del modelo de datos en toda la ruta.

Dependiendo de si los datos fluyen a, desde o a través del nodo, el archivo de especificación debe describir el modelo de datos de entrada, salida o ambos. Un nodo de CLEF puede afectar al modelo de datos añadiendo campos nuevos a los que se pasan al nodo, o sustituyendo los campos correspondientes al nodo por nuevos campos generados por el propio programa. El elemento `OutputDataModel` del archivo de especificación describe los efectos del nodo de CLEF en el modelo de datos. Para obtener más información, consulte el tema “Modelo de datos de salida” en la página 58.

Archivos de entrada y resultado

Es posible especificar que se genere uno o más archivos temporales antes de que se ejecute un nodo de CLEF. Se denominan **archivos de entrada**, ya que son entradas en la ejecución del nodo en el servidor. Por ejemplo, un nodo generador de modelos puede tener un contenedor de modelos desde donde se transfiere el contenido al archivo de entrada especificado en la ejecución del nodo. Para obtener más información, consulte el tema “Archivos de entrada” en la página 55.

Se generan otros archivos temporales durante la ejecución del nodo en el servidor; por ejemplo, los resultados de ejecutar un nodo generador de modelos o generador de documentos. Se denominan **archivos de resultados** y se vuelven a transferir al cliente después de ejecutar el nodo. Para obtener más información, consulte el tema “Archivos de resultados” en la página 56.

Interfaces de programación de la aplicación (API)

Dependiendo de la acción que desee que ejecute la extensión, es posible que necesite utilizar una interfaz de programación de la aplicación (API). Para una simple transformación de datos, debe poder definir el procesamiento necesario totalmente en el archivo de especificación. Sin embargo, para requisitos más avanzados, deberá interactuar con una o más de las API disponibles:

- API de cliente de CLEF
- API de servidor de CLEF
- API de servidor Predictive (PSAPI)

La **API de cliente** de CLEF es una API de Java que se puede utilizar por las extensiones que requieren controles adicionales, componentes de interfaz de usuario o interactividad que no proporciona directamente el archivo de especificación.

La **API de servidor** de CLEF es una API basada en C que trata aspectos como la configuración y ejecución de los parámetros, la persistencia de los parámetros, comentarios sobre la ejecución, control de trabajos (por ejemplo, interrupción de la ejecución), generación de SQL y objetos devueltos.

La **API de servidor Predictive** es una API de Java que contiene las funciones de IBM SPSS Modeler para el uso de aplicaciones que requieren minería de datos y capacidades de análisis predictivos.

Si desea obtener más información, consulte Capítulo 9, “Programación”, en la página 175.

Estructura de archivos

Una extensión de CLEF se compone de dos grupos de componentes:

- Componentes de cliente
- Componentes de servidor

Los **componentes de cliente** contienen el archivo de especificación de extensiones, las clases de Java y los archivos .jar, paquetes de propiedades que contienen recursos localizables y archivos de imagen y ayuda.

Los **componentes de servidor** son las bibliotecas compartidas y DLL necesarias cuando se ejecuta un nodo de extensión.

Componentes de cliente

Los componentes del cliente se instalan en la carpeta `\ext\lib` del directorio de instalación de IBM SPSS Modeler. Los componentes de cliente son:

- Archivo de especificación
- Clases de Java y archivos `.jar`
- Archivos de propiedades
- Archivos de imagen
- Archivos de ayuda

Carpeta de extensión

Cada extensión se encuentra en su propia **carpeta de extensión** en `\ext\lib`.

La convención de nomenclatura sugerida para la carpeta de extensión es:
`providerTag.id`

donde *providerTag* es el identificador del proveedor del elemento `ExtensionDetails` del archivo de especificación e *id* es el identificador de la extensión del mismo elemento.

Además, por ejemplo, si el elemento ExtensionDetails comienza por:

```
<ExtensionDetails providerTag="myco" id="clasificador" ... />
```

se utilizará el nombre de la carpeta de extensión mico.clasificador.

Archivo de especificación

El archivo de especificación debe tener el nombre extensión.xml y debe residir en el nivel superior de la subcarpeta de extensión. Además, en el ejemplo que se acaba de exponer, la ruta del archivo de especificación sería la que se expone a continuación en el directorio de instalación de IBM SPSS Modeler:

```
\ext\lib\mico.clasificador\extensión.xml
```

Clases de Java y archivos .jar

Las extensiones que utilizan la API de Java de cliente incluyen código compilado de Java. Este código se puede dejar como un conjunto de archivos de .class, o se puede compilar como un archivo .jar.

Los archivos .class de Java se encuentran cerca de la carpeta de extensión del nivel superior. Por ejemplo, una clase que implementa la interfaz ActionHandler puede tener la ruta:

```
com.mi_ejemplo.mi_extensión.MiActionHandler
```

En este caso, el archivo .class debe estar en la siguiente ubicación en el directorio de instalación de IBM SPSS Modeler:

```
\carpeta_extensión\com\mi_ejemplo\mi_extensión\MiActionHandler.class
```

Un archivo .jar se puede encontrar en cualquier ubicación de la carpeta de extensión. Puede especificar la ubicación actual de un archivo .jar mediante el elemento JarFile en el archivo de especificación. Por ejemplo, si una extensión utiliza un archivo .jar con la siguiente ruta:

```
\carpeta_extensión\lib\common-utilities.jar
```

el archivo de especificación debe incluir la siguiente entrada en el elemento Recursos:

```
<Resources>
    <JarFile id="util" path="lib\common-utilities.jar"/>
     ...
</Resources>
```

Para obtener más información, consulte el tema “Archivos Jar” en la página 35.

Archivos de propiedades

Los recursos localizados (por ejemplo, texto en pantalla y mensajes de error con sus traducciones) se pueden guardar en los archivos con la extensión .properties, que se pueden encontrar en cualquier ubicación de la carpeta de extensión. Para obtener más información, consulte el tema “Archivos de propiedades” en la página 168.

Archivos de imagen y ayuda

Los archivos que contienen las imágenes gráficas para la representación del icono y los que contienen sistemas de ayuda se pueden encontrar en la carpeta de extensión. Puede encontrar de gran utilidad separar los archivos de imagen y de ayuda en sus propias subcarpetas.

Puede declarar la ubicación de un archivo de imagen mediante el atributo imagePath de un elemento Icon en el archivo de especificación. Para obtener más información, consulte el tema “Icons” en la página 108.

De la misma forma, puede establecer la ubicación de un sistema de ayuda utilizando el atributo `path` de un elemento `HelpInfo` en el archivo de especificación. Para obtener más información, consulte el tema "Definición de la ubicación y el tipo del sistema de ayuda" en la página 163.

Ejemplo

La estructura del archivo de cliente basada en estos componentes puede tener la siguiente apariencia:

```
\ext\lib\mico.clasificador\
\ext\lib\mico.clasificador\extensión.xml
\ext\lib\mico.clasificador\sorter_en.properties
\ext\lib\mico.clasificador\sorter_fr.properties
\ext\lib\mico.clasificador\sorter_it.properties
\ext\lib\mico.clasificador\com\mi_ejemplo\mi_extensión\MyActionHandler.class
\ext\lib\mico.clasificador\help\sorter.chm
\ext\lib\mico.clasificador\images\lg_sorter.gif
\ext\lib\mico.clasificador\images\sm_sorter.gif
\ext\lib\mico.clasificador\lib\common-utilities.jar
```

Componentes de servidor

Las bibliotecas compartidas necesarias para la ejecución se deben encontrar en una carpeta en la carpeta `\ext\bin` del directorio de instalación de IBM SPSS Modeler, por ejemplo:

directorio_instalación`\ext\bin\mico.clasificador\mi_bib.dll`

Tenga en cuenta que las bibliotecas compartidas no se deben ubicar directamente en la carpeta `\ext\bin`.

Para las bibliotecas compartidas que IBM SPSS Modeler llama directamente durante la ejecución, establece la ubicación en un elemento `SharedLibrary` del archivo de especificación. Para obtener más información, consulte el tema "Bibliotecas compartidas" en la página 36.

La biblioteca compartida principal puede requerir el uso de otras bibliotecas. También debe colocar cualquier biblioteca compartida dependiente en la misma ubicación que la biblioteca compartida principal, para permitir que la biblioteca principal encuentre las bibliotecas dependientes.

Ejemplo

Un ejemplo de una estructura de archivos de servidor puede ser:

```
\ext\bin\mico.clasificador\mi_bib.dll
\ext\bin\mico.clasificador\mi_bib2.dll
```

Capítulo 2. Nodos

Conceptos básicos sobre nodos

Al crear una extensión que implementa un nuevo nodo, debe familiarizarse con las características de los nodos de IBM SPSS Modeler. Lo que le ayudará a definirlos correctamente en el archivo de especificación.

Los nodos de IBM SPSS Modeler se clasifican en nodos de origen, proceso, resultado y modelado, dependiendo de su función. En CLEF, los nodos se clasifican de forma ligeramente diferente. La correlación entre los dos sistemas se muestra en la tabla siguiente.

Tabla 1. Tipos de nodo de CLEF.

Clasificación de IBM SPSS Modeler	**Paleta**	**Tipo de nodos de CLEF**
Nodos de origen	Orígenes	Lector de datos
Nodos de proceso	Oper. con registros	Transformador de datos
	Oper. con campos	
Nodos de resultados	Gráficos	Generador de documentos
	Resultado (nodo Informe)	
	Exportar	Escritor de datos
Nodos de modelado	Modelado	Generador de modelos

Cuando crea un nuevo nodo de CLEF, lo define como uno de los tipos de nodos de CLEF. El tipo de nodo que ha seleccionado depende de la función principal del nodo.

Tabla 2. Tipos y funciones de nodos.

Tipo de nodos de CLEF	**Descripción**	**Paleta de nodo correspondiente**	**Forma de icono**
Lector de datos	Importa datos a IBM SPSS Modeler a partir de un formato distinto.	Orígenes	 *Figura 3. Forma de nodo Origen (círculo)*
Transformador de datos	Obtiene los datos de IBM SPSS Modeler, los modifica y los devuelve modificados a la ruta de IBM SPSS Modeler.	Operaciones con registros, Operaciones con campos	 *Figura 4. Forma de nodo Operaciones (hexágono)*
Generador de modelos	Genera modelos a partir de datos de IBM SPSS Modeler.	Modelado	 *Figura 5. Forma de nodo generador de modelos*

Tabla 2. Tipos y funciones de nodos (continuación).

Tipo de nodos de CLEF	Descripción	Paleta de nodo correspondiente	Forma de icono
Generador de documentos	Genera un gráfico o informe a partir de datos de IBM SPSS Modeler.	Gráficos	 *Figura 6. Forma de nodo Gráficos (triángulo)*
		Resultado (nodo Informe)	 *Figura 7. Forma de nodo Resultado (rectángulo)*
Aplicador del modelo (también conocido como "nugget de modelo")	Define un contenedor para un modelo generado que se ha devuelto al lienzo de IBM SPSS Modeler.	–	 *Figura 8. Forma de nodo aplicador de modelos (diamante de oro)*
Escritor de datos	Exporta los datos del formato de IBM SPSS Modeler a un formato adecuado para el uso con otra aplicación.	Exportar	 *Figura 9. Forma de nodo Exportar (rectángulo)*

Define el tipo de nodo, junto con otros atributos, en un elemento `Nodo` en el archivo de especificación; por ejemplo:

```
<Node name="proceso_ordenación" type="transformadordatos"
      palette="Operacionesconregistro" ...>
     -- elementos de nodo --
</Node>
```

El atributo `palette` define la paleta en la ventana principal de IBM SPSS Modeler desde la que los usuarios podrán acceder al nodo, en este caso, la paleta Operaciones con registros. Si omite este atributo, el nodo aparece en la paleta Operaciones con campos.

Se suministran varios nodos de ejemplo con IBM SPSS Modeler. Para obtener más información, consulte el tema "Acerca de los ejemplos" en la página 25.

Nodos de lector de datos

Un nodo de lector de datos permite que los datos de un origen externo se lean en una ruta de IBM SPSS Modeler. Los nodos de la paleta Orígenes de IBM SPSS Modeler son el equivalente a los nodos de lector de datos y se identifican por una forma de icono circular.

En la especificación de un nodo de lector de datos, se incluye la siguiente información:

- El origen de los datos (como un archivo o base de datos)

- Cualquier procesamiento previo de registros (como el tratamiento de espacios iniciales y finales o el carácter para utilizarlo como delimitador de registro)
- Si filtra o no cualquier campo de registros
- El tipo de datos (por ejemplo, rango, conjunto, marca) y tipo de almacenamiento (cadena, entero, real) para asociar a cada campo
- Si el modelo de datos de entrada se cambia o no

El nodo de lector de datos puede incluir la lógica para leer los registros de datos de origen. De forma alternativa, se puede realizar posteriormente mediante un nodo Tipo en IBM SPSS Modeler.

Se suministra un nodo de lector de datos de ejemplo con IBM SPSS Modeler. Para obtener más información, consulte el tema "Acerca de los ejemplos" en la página 25.

Nodos de transformador de datos

Un nodo de transformador de datos obtiene los datos de una ruta de IBM SPSS Modeler, los modifica y los devuelve modificados a la ruta. Los nodos de las paletas Operaciones con registros y Operaciones con campos de IBM SPSS Modeler son nodos de transformador de datos identificados por una forma de icono hexagonal.

En la especificación de un nodo de transformador de datos, se incluye la siguiente información:

- Los registros o campos que se están transformando
- Cómo se van a modificar los datos

Se suministra un nodo de transformador de datos de ejemplo con IBM SPSS Modeler. Para obtener más información, consulte el tema "Acerca de los ejemplos" en la página 25.

Nodos generadores de modelos

Para ver los conceptos básicos de la generación de modelos en IBM SPSS Modeler, consulte "Introducción al modelado" en el *Manual de aplicaciones de IBM SPSS Modeler 16*.

Los nodos generadores de modelos generan objetos que aparecen en la pestaña Modelos o Resultados del panel del gestor en la ventana principal de IBM SPSS Modeler.

Los nodos de la paleta Modelado de IBM SPSS Modeler son ejemplos de nodos generadores de modelos y se identifican por una forma de icono pentagonal.

Cuando se ejecuta, un nodo generador de modelos genera un **objeto de resultado de modelo** (también conocido como "nugget de modelo") en la pestaña Modelos.

Cuando un modelo generado se añade al lienzo, adquiere la forma de un nodo aplicador de modelos.

En la especificación de un nodo generador de modelos, incluye:

- Los detalles de generación de modelos, como el algoritmo utilizado para generar el modelo y los campos de entrada y salida que se utilizan para puntuar los datos con el modelo
- Propiedades que utiliza el modelo
- Los contenedores utilizados para reservar objetos de resultados
- La interfaz de usuario para el cuadro de diálogo del nodo
- Las propiedades y los archivos utilizados cuando el nodo se ejecuta
- Cómo afecta la ejecución del nodo al modelo de datos de entrada
- El identificador del objeto de resultado de modelo y cualquier otro objeto producido mediante la ejecución del nodo

- El identificador del nodo aplicador de modelos (consulte "Nodos aplicadores de modelos")

Note: Cuando defina un nodo generador de modelos, incluya la definición del objeto de resultado de modelo real y el nodo aplicador de modelos en cualquier lugar del mismo archivo de especificación.

Se suministra un nodo generador de modelos de ejemplo con IBM SPSS Modeler. Para obtener más información, consulte el tema "Acerca de los ejemplos" en la página 25.

Nodos generadores de documentos

Los nodos generadores de documentos generan objetos que aparecen en la pestaña Resultados del panel del gestor en la ventana principal de IBM SPSS Modeler. Los nodos de la paleta Gráficos son ejemplos de nodos generadores de documentos y se identifican por una forma de icono triangular.

Cuando se ejecuta, un nodo generador de documentos genera un **objeto de resultado de documento** en la pestaña Resultados del panel del gestor.

En contraposición a un objeto de resultado de modelo, un objeto de resultado de documento no se puede añadir al lienzo de IBM SPSS Modeler.

En la especificación de un nodo generador de documentos, incluye:

- Los detalles de generación de documentos, como la pestaña del cuadro de diálogo de nodo que debe contener los controles de generación de documentos
- Propiedades que utiliza el documento
- Los contenedores utilizados para reservar objetos de resultados
- La interfaz de usuario para el cuadro de diálogo del nodo
- Las propiedades y los archivos utilizados cuando el nodo se ejecuta
- El identificador del objeto de resultado de documento y cualquier otro objeto producido mediante la ejecución del nodo

Note: Cuando defina un nodo generador de documentos, incluya la definición del objeto de resultado de documento real en cualquier lugar del mismo archivo de especificación.

Nodos aplicadores de modelos

Un nodo aplicador de modelos define un contenedor para un modelo generado que se utiliza cuando el modelo se añade al lienzo de IBM SPSS Modeler desde la pestaña Modelos del panel del gestor.

En la especificación de un nodo aplicador de modelos, se incluye la siguiente información:

- El contenedor para el modelo (o contenedores, si el resultado del modelo se puede producir en más de un formato, por ejemplo, texto y HTML)
- Los detalles de la interfaz de usuario del cuadro de diálogo que se muestra cuando el usuario examina el nodo del aplicador en la pestaña Modelos o lo abre en el lienzo
- El modelo de datos de salida
- El procesamiento que se debe realizar cuando se ejecuta la ruta que contiene el nodo
- Los constructores que tratan los objetos producidos cuando se ejecuta la ruta que contiene el nodo

Nodos de escritor de datos

Un nodo de escritor de datos exporta los datos del formato de IBM SPSS Modeler a un formato adecuado para el uso con otra aplicación. Los nodos de la paleta Exportar de IBM SPSS Modeler son nodos de escritor de datos identificados por una forma de icono rectangular.

En la especificación de un nodo de escritor de datos, se incluye:

- Los detalles del archivo o base de datos en la que se escribirán los datos de la ruta
- De forma opcional, si la ruta completa se va a publicar o no, de manera que se pueda incrustar en una aplicación externa

Menús, barras de herramientas y paletas

Los usuarios pueden acceder a una extensión desde un menú de IBM SPSS Modeler, la barra de herramientas o una paleta. Una extensión puede implementar un nodo o realizar una acción especificada.

Una extensión (nodo o acción) a la que se puede acceder desde un menú especificado explícitamente también se podrá acceder desde la barra de herramientas y viceversa.

Un nodo al que se puede acceder desde una paleta es totalmente accesible desde un elemento correspondiente del menú Insertar.

Menús y submenús

Los usuarios pueden acceder a los nodos estándar de IBM SPSS Modeler desde el menú Insertar. Cada elemento del último grupo de este menú (aparte del menú Modelos) tiene un submenú que ofrece acceso a un conjunto de nodos relacionados.

Estos elementos corresponden directamente a las entradas de las paletas del nodo. Al añadir un nodo a una paleta, se añade automáticamente al grupo correspondiente en el menú Insertar.

Si su extensión define una acción a la que no se puede acceder a través de un nodo, puede habilitar la extensión añadiendo uno o más de los siguientes elementos:

- Un nuevo elemento a un menú o submenú del sistema
- Un nuevo menú a IBM SPSS Modeler
- Un nuevo elemento a la barra de herramientas (consulte “Barras de herramientas”)

Un nuevo menú o elemento de menú puede mostrar opcionalmente el icono asociado con la extensión, como, por ejemplo, en algunos elementos del menú Insertar.

Si desea obtener más información, consulte “Menús” en la página 109 y “Elementos de menú” en la página 111.

Barras de herramientas

Si su extensión define una acción a la que no se puede acceder a través de un nodo, puede habilitar la extensión añadiéndola a la barra de herramientas principal de IBM SPSS Modeler.

En este caso, es aconsejable ocultar la etiqueta de la acción.

También puede añadir un elemento a la barra de herramientas de un cuadro de diálogo de nodo o una ventana de resultados. Puede seleccionar si desea mostrar u ocultar la etiqueta del elemento.

Para obtener más información, consulte el tema “Elementos de la barra de herramientas” en la página 112.

Paletas y subpaletas

Si su extensión define un nuevo nodo, puede colocar el nodo en cualquier posición de una de las paletas o subpaletas estándar de IBM SPSS Modeler.

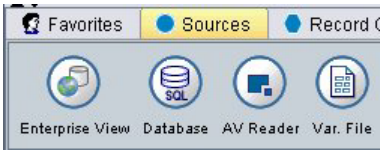

Figura 10. Nuevo nodo en una paleta estándar

Puede añadir una entrada a una subpaleta estándar y posibilitar el acceso al nodo desde ahí.

Figura 11. Nuevo nodo en la adición personalizada a una subpaleta estándar

Puede definir una paleta personalizada y colocar el nuevo nodo ahí.

Figura 12. Nuevo nodo en una paleta personalizada

Una paleta personalizada puede tener subpaletas personalizadas.

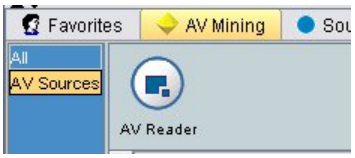

Figura 13. Nuevo nodo en una subpaleta personalizada de una paleta personalizada

Si desea obtener más información, consulte “Nodo” en la página 48 y “Sección de interfaz de usuario (Paletas)” en la página 43.

Diseño de iconos de nodos

Para cada nuevo nodo que cree en CLEF, puede proporcionar una imagen central para el icono que identifique el nodo en la pantalla.

Note: No tiene que proporcionar una imagen: IBM SPSS Modeler proporciona una de forma predeterminada, que se muestra si no especifica ninguna (puede ser útil cuando empieza a desarrollar un nodo).

Figura 14. Imagen predeterminada para los iconos de CLEF

Los iconos estándar de IBM SPSS Modeler se componen de tres capas:

- Borde
- Fondo
- Imagen central

Para un nuevo nodo, sólo tiene que proporcionar la imagen central (conocida como **glifo**); IBM SPSS Modeler realiza el procesamiento del borde y el fondo. La imagen de glifo debe tener un fondo transparente de manera que no oculte la capa de fondo del icono. En esta sección, las representaciones del glifo tienen un fondo coloreado para indicar su transparencia.

Así es cómo se forma un icono de modelado típico de IBM SPSS Modeler.

Tabla 3. Composición de los iconos de nodos y modelos generados

	Icono de nodo	**Icono del modelo generado**
Borde	 *Figura 15. Borde del icono*	Ninguno
Fondo	*Figura 16. Fondo del icono*	 *Figura 17. Fondo del modelo generado*
Glifo	 *Figura 18. Glifo del icono*	 *Figura 19. Glifo del modelo generado*
Imagen mostrada como	 *Figura 20. Icono mostrado*	 *Figura 21. Icono del modelo generado mostrado*

Bordes

La función del nodo se indica por la forma del borde del icono. Para obtener más información, consulte el tema “Conceptos básicos sobre nodos” en la página 9.

Si un nodo tiene el almacenamiento en caché activado, la forma del borde tiene un símbolo de documento en miniatura añadido. Un icono blanco de documento en un nodo indica que la caché está vacía. Cuando la caché está llena, el icono de documento aparece en color verde oscuro.

Tabla 4. Bordes del nodo y estado de almacenamiento en caché

Estado de almacenamiento en caché	Ejemplo
Sin almacenamiento en caché	 *Figura 22. Nodo sin almacenamiento en caché*
Almacenamiento en caché activado	 *Figura 23. Nodo sin almacenamiento en caché activado*
Caché completa	 *Figura 24. Nodo con caché completa*

Los diferentes símbolos de borde los proporciona el sistema y IBM SPSS Modeler se ocupa del procesamiento necesario para mostrar el símbolo correcto en cada momento.

Fondos

Para los iconos de nodo que sean distintos de los de modelos generados y nodos aplicadores de modelos, el fondo cambia de color para indicar el estado.

Tabla 5. Fondos del nodo

Estado	Color	Ejemplo
Sin seleccionar	Gris	*Figura 25. Fondo del icono (gris)*
Seleccionado	Azul	 *Figura 26. Fondo del icono (azul)*
Error	Rojo	 *Figura 27. Fondo del icono (rojo)*

Tabla 5. Fondos del nodo (continuación)

Estado	Color	Ejemplo
Una base de datos está realizando una acción	Púrpura	 *Figura 28. Fondo del icono (púrpura)*

Igual que antes, las imágenes de fondo las proporciona el sistema y IBM SPSS Modeler se ocupa del procesamiento necesario para mostrar el fondo correcto en cada momento.

Requisitos gráficos

Para cada nodo nuevo de CLEF, cree las versiones siguientes de la imagen de la capa de glifo:

- Tamaño grande (49 x 49 píxeles) de los nodos en el lienzo de rutas
- Tamaño pequeño (38 x38 píxeles) para los nodos del gestor de paletas en la parte inferior de la pantalla

Si desea mostrar el icono en un menú, en una barra de herramientas, en la barra de título de un navegador o en una ventana de resultados, también tendrá que crear:

- Tamaño en miniatura (16 x 16 píxeles)

Si el nodo genera un modelo, también tendrá que crear:

- Tamaño pequeño (38 x 38 píxeles) con el diseño desplazado a la esquina inferior izquierda, para superponerlo en el icono del modelo generado (el nugget dorado)

Note: Las imágenes con una tamaño mayor se recortarán cuando se visualicen en IBM SPSS Modeler.

Para obtener más información, consulte el tema “Icons” en la página 108.

Creación de imágenes personalizadas

La imagen que cree para un nodo debe incluir la función principal del nodo. Para un público internacional, asegúrese de utilizar imágenes que no sean específicas de un país y que no puedan provocar malentendidos en usuarios de otros países.

Para crear una imagen personalizada para utilizarla con CLEF:

1. Con un paquete gráfico que admita la transparencia, establezca el tamaño apropiado del lienzo de dibujo y cree la versión de la imagen.
2. Guarde cada versión (grande, pequeña, etc.) como un archivo *.gif* independiente con las siguientes características:
   - Fondo transparente
   - Configuración de 16 colores (4 bits) o superior.

   La forma de convertir el fondo de la imagen en transparente depende del paquete gráfico que utilice. Por ejemplo, puede establecer el color de fondo como transparente directamente o puede proponer un color de transparencia y, a continuación, "pintar" el fondo de la imagen con este color.

   Para los archivos de imágenes, recomendamos que siga las convenciones de nomenclatura de archivos que utiliza IBM SPSS Modeler internamente, que se muestra en la tabla siguiente.

Tabla 6. Convenciones de nomenclatura de archivo de imagen

Tipo de imagen	Nombre de archivo
Grande	lg_*nodo*.gif

Tabla 6. Convenciones de nomenclatura de archivo de imagen (continuación)

Tipo de imagen	Nombre de archivo
Pequeño	sm_*nodo*.gif
En miniatura	*nodo*16.gif
Modelo generado	sm_gm_*nodo*.gif

3. Compruebe el aspecto de la imagen haciendo referencia a los archivos de la imagen del archivo de especificación (consulte "Adición de los archivos de imagen a la especificación del nodo") y añadiendo el nuevo nodo a IBM SPSS Modeler (consulte"Prueba de una extensión de CLEF" en la página 199).

Adición de los archivos de imagen a la especificación del nodo

Cuando haya creado los archivos de imagen, cópielos en una carpeta del equipo desde el que ejecute IBM SPSS Modeler. En el archivo de especificación, tendrá que especificar una ruta de imagen respecto a una carpeta \ext\lib*proveedor.nombrenodo* en el directorio de instalación de IBM SPSS Modeler, de manera que despliegue los archivos en una carpeta de acceso fácil. Para obtener más información, consulte el tema "Icons" en la página 108.

En el archivo de especificación, asocie los archivos gráficos de icono grande y pequeño con un nodo personalizado por medio del elemento `Icons` en la sección `UserInterface` de la especificación `Nodo`; por ejemplo:

```
<Icons>
     <Icon type="Nodoestándar" imagePath="images/lg_minodo.gif" />
     <Icon type="smallNode" imagePath="images/sm_mynode.gif" />
</Icons>
```

Para los nodos generadores de modelos o de documentos, haga referencia también a la versión en miniatura (16 x 16 píxeles) en la sección `UserInterface` de la especificación `ModelOutput` (para un nodo generador de modelos) o la especificación `DocumentOutput` (para un nodo generador de documentos); por ejemplo:

```
<Icons>
     <Icon type="Ventanaestándar" imagePath="images/minodo16.gif" />
</Icons>
```

Para los nodos aplicadores de modelos, haga referencia también a la versión del modelo generado en la sección `UserInterface` de la especificación `Nodo`; por ejemplo:

```
<Icons>
     <Icon type="Nodoestándar" imagePath="images/lg_gm_minodo.gif" />
     <Icon type="Nodopequeño" imagePath="images/sm_gm_minodo.gif" />
</Icons>
```

Cuadros de diálogo de diseño

Esta sección describe las características de los cuadros de diálogo del nodo estándar de IBM SPSS Modeler para ayudarle a diseñar los cuadros de diálogo en CLEF.

Acerca de los cuadros de diálogo del nodo

Un cuadro de diálogo de nodo ofrece una interfaz que permite al usuario final modificar la configuración de ejecución. El aspecto del cuadro de diálogo es muy importante; es donde se altera y modifica el comportamiento del nodo. La interfaz debe contener, además, toda la información necesaria y debe ser fácil de usar.

El comportamiento de nodo se cambia a través del uso de varios **controles** basados en cuadros de diálogo, que son elementos de la interfaz de usuario con los que un usuario puede interactuar. Un cuadro de diálogo puede incluir un número de controles, como los botones de radio, casillas de verificación, cuadros de texto y menús. CLEF proporciona una amplia variedad de controles que le permiten diseñar sus cuadros de diálogo. Para obtener más información, consulte el tema "Especificaciones de control de propiedad" en la página 123.

El tipo de parámetro modificado por un control determina qué control aparece en el cuadro de diálogo; algunos tipos ofrecen controles alternativos. Puede agrupar opciones en una nueva pestaña mediante elementos `Tab` en el archivo de especificación. Para obtener más información, consulte el tema "Área de pestañas" en la página 22.

Note: Puede comprobar el aspecto de la interfaz de usuario para una extensión incluso si no ha especificado el procesamiento que la extensión debe realizar. Para obtener más información, consulte el tema "Comprobación de las extensiones de CLEF" en la página 199.

Directrices del diseño de cuadros de diálogo

Cuando defina los controles para un cuadro de diálogo, tenga en cuenta las siguientes directrices:

- Elabore detenidamente el texto que vaya a utilizar en la etiqueta de visualización del control. El texto debe ser razonablemente conciso, además de mostrar la información correcta. Si está diseñando un mercado internacional, tenga en cuenta que la longitud del texto traducido puede variar de forma significativa respecto al original.
- Utilice el control correcto para un parámetro. Por ejemplo, una casilla de verificación no es siempre la mejor opción para un parámetro que tiene sólo dos valores. El cuadro de diálogo de nodo de IBM SPSS Modeler C5.0 utiliza los botones de radio para permitir a los usuarios seleccionar el tipo de resultados, que tiene uno de estos dos valores:**Árbol de decisión** o **Conjunto de reglas**.

  Esta configuración se puede representar como una casilla de verificación etiquetada con **Árbol de decisión**. Cuando se selecciona, el tipo de resultados es un árbol de decisión; cuando se anula su selección, los resultados son un conjunto de reglas. Aunque el resultado es realmente el mismo, con los botones de radio los usuarios suelen comprender mejor las opciones en este caso.
- Los controles de los nombres de archivo se suelen ubicar en la parte superior del cuadro de diálogo.
- Los controles que forman el foco del nodo se sitúan en la parte alta del cuadro de diálogo. Por ejemplo, los nodos de gráficos muestran campos procedentes de los datos. Seleccionar estos campos es la función principal del cuadro de diálogo, por lo que los parámetros de campos se sitúan en la parte superior de éste.
- Las casillas de verificación o botones de radio suelen permitir al usuario seleccionar una opción que necesita más información. Por ejemplo, la selección de **Utilizar aumento** en el cuadro de diálogo de C5.0 requiere que el análisis incluya un número que indique el **Número de ensayos**.

  La información adicional siempre se encuentra después de la selección de opción, a su derecha o justo debajo de ella.

Los cuadros de diálogo de CLEF utilizan la edición de confirmación de IBM SPSS Modeler de la misma manera que los cuadros de diálogo de IBM SPSS Modeler estándar: los valores mostrados en los cuadros de diálogo no se copian en el nodo hasta que el usuario pulsa **Aceptar**, **Aplicar** o, en el caso de los nodos terminales, **Ejecutar**. De igual modo, la información mostrada por el cuadro de diálogo no se actualiza (por ejemplo, cuando los campos de entrada del nodo han cambiado como resultado de las operaciones anteriores de la ruta del nodo actual) hasta que el usuario cancela y vuelve a mostrar el cuadro de diálogo o pulsa en el botón **Actualizar**.

Componentes de los cuadros de diálogo

Los cuadros de diálogo tienen los siguientes componentes:

- Barra de título
- Área de iconos

- Área de barra de herramientas y menú que incluye:
  – Archivo, Generar, Ver, Presentación preliminar, Actualizar y otros botones (dependiendo del nodo)
  – Botón Maximizar/tamaño normal
  – Botón Ayuda
- Área de estado
- Área de panel
- Área de pestañas
- Área de botones

Cada nodo personalizado requiere un cuadro de diálogo que se muestra cuando el usuario abre el nodo. Siempre que su archivo de especificación incluya un elemento `Nodo` que contenga una sección `UserInterface` con un elemento `Tabs`, verá todos los componentes del cuadro de diálogo enumerados arriba cuando abra el nodo. En función del tipo de nodo, el contenido mínimo del área de pestañas y el área de botones es el que se muestra en la tabla siguiente:

Tabla 7. Contenido mínimo del área de pestañas y área de botones para tipos de nodo diferentes

Tipo de nodo	**Pestañas**	**Botones**
Lector de datos	Anotaciones (con botón Actualizar en el área de la barra de herramientas)	Aceptar, Cancelar, Aplicar, Restablecer
Transformador de datos	Anotaciones	Aceptar, Cancelar, Aplicar, Restablecer
Escritor de datos	Publicar, Anotaciones	Aceptar, Cancelar, Ejecutar, Aplicar, Restablecer
Generador de modelos	Anotaciones	Aceptar, Cancelar, Ejecutar, Aplicar, Restablecer
Generador de documentos	Anotaciones	Aceptar, Cancelar, Ejecutar, Aplicar, Restablecer
Aplicador de modelos	Resumen, Anotaciones	Aceptar, Cancelar, Aplicar, Restablecer

Los cuadros de diálogo de nodo se colocan inicialmente de manera que cuando el usuario abra el nodo, se superponga el icono de nodo en el nodo que representa. El usuario puede mover el cuadro de diálogo, pero la nueva posición no se guardará para la siguiente vez que se abra el nodo. Si el usuario ha movido el cuadro de diálogo y se ha ocultado parcial o completamente por otro cuadro de diálogo, al pulsar dos veces en el nodo original en el lienzo volverá a mostrar el primer cuadro de diálogo delante. El cuadro de diálogo no tiene modo (es decir, una misma entrada de usuario produce la misma acción) y se puede volver a cambiar de tamaño.

Todos los campos editables del cuadro de diálogo admiten los atajos de teclado que se muestran en la tabla siguiente.

Tabla 8. Atajos de teclado para campos editables en cuadros de diálogo

Atajo	**Efecto**
Ctrl-C	Copiar
Ctrl-V	Pegar
Ctrl-X	Cortar

Barra de título

La barra de título de un cuadro de diálogo de nodo incluye una versión en miniatura del icono de la pepita de IBM SPSS Modeler seguido del nombre del modelo. El texto se extrae de la configuración de los controles del nombre del modelo. También se proporciona de forma predeterminada el botón Cerrar (X) en la esquina superior derecha.

Área de iconos

El icono de nodo se muestra en el área de iconos junto a la parte superior derecha del cuadro de diálogo. Es la versión de tamaño pequeño (38 x 38 píxeles) del icono que también se utiliza en la paleta de nodo de la parte inferior de la ventana principal, no la versión de mayor tamaño que aparece en el lienzo.

Note: El icono de la pepita en miniatura del extremo izquierdo de la barra de título está codificado en todos los cuadros de diálogo de nodo

Área de barra de herramientas y menú

El área superior del cuadro de diálogo se reserva como área de barra de herramientas y menú.

Los cuadros de diálogo de los nodos de lector de datos y transformador de datos tienen el botón Presentación preliminar en este área con el que se visualiza una muestra de los datos de entrada.

Los cuadros de diálogo de nodos de lector de datos también tienen el botón Actualizar, que actualiza la información mostrada por el nodo (por ejemplo, cuando cambian los campos de entrada del nodo).

Los nodos aplicadores de modelos tienen los botones Archivo, Generar y Ver, que permiten a los usuarios realizar varias operaciones como exportar el modelo o generar nuevos modos. Los nodos aplicadores de modelos tienen el botón Presentación preliminar, que en este caso, muestra datos de entrada de muestra junto con las columnas adicionales creadas al aplicar el nodo.

El tamaño correcto de esta área contiene dos botones en todos los cuadros de diálogo de nodo:

- Botón Maximizar/tamaño normal
- botón Ayuda

Botón Maximizar/tamaño normal: Este botón cambia el tamaño del cuadro de diálogo a pantalla completa. Si se vuelve a utilizar disminuye el tamaño del cuadro de diálogo devolviéndolo al tamaño anterior.

Botón Ayuda: Este botón abre la ayuda contextual del nodo. En un cuadro de diálogo o ventana de resultados con pestañas, se mostrará la ayuda de dicha pestaña. También se puede utilizar la tecla F1 para acceder a la ayuda.

Área de estado

El resto del área de la parte superior del cuadro de diálogo se reserva para mostrar información, advertencias o texto de error. Los nodos de origen muestran aquí la ruta completa y el nombre de archivo del archivo de datos de origen. Los nodos individuales pueden mostrar en esta área otra información específica del nodo. Cualquier texto especificado en esta área se debe limitar a dos líneas.

Área de panel

Es el área principal del cuadro de diálogo y contiene todos los controles y áreas de visualización del nodo. Cada pestaña tiene un área de panel diferente. Cada panel puede ser de cualquiera de los siguientes tipos:

- Procesador de texto
- Objeto de extensión
- Propiedades

También puede especificar subpaneles, que son cuadros de diálogo separados que se abren en una nueva ventana y se activan mediante los botones de acción del panel.

Para obtener más información, consulte el tema “Especificaciones de panel” en la página 116.

Área de pestañas

Los cuadros de diálogo de nodo tienen las siguientes pestañas:

- Una o más pestañas específicas del nodo proporcionado por el usuario
- Una pestaña Resumen (sólo objetos de resultado de modelo y nodos aplicadores de modelos)
- Una pestaña Anotaciones

Las pestañas específicas de nodo se definen en la sección `Tabs` del archivo de especificación de CLEF. Para obtener más información, consulte el tema “Pestañas” en la página 113.

Los cuadros de diálogo de objetos de resultado de modelo y nodos aplicadores de modelos disponen de una pestaña Resumen proporcionada por el sistema. Muestra información resumida sobre un modelo generado, entre la que se incluyen los campos, la configuración de creación y el proceso de estimación del modelo utilizado. Los resultados se muestran en una vista de árbol que se puede expandir o contraer pulsando los elementos específicos.

El sistema incluye la pestaña Anotaciones en todos los cuadros de diálogo de nodo y permite al usuario especificar la información sobre el nodo. Incluye el nombre del nodo, el texto de información sobre herramientas y un campo de comentario mayor.

Nombre. El nombre de nodo predeterminado se especifica en el atributo `Label` del elemento Node del archivo de especificación (consulte “Nodo” en la página 48). El usuario puede cambiar el nombre del nodo seleccionando **Personalizado**, introduciendo un nombre en el campo de edición Personalizado y pulsando en **Aplicar** o **Aceptar**. El nuevo nombre se conserva en las demás secciones, aunque el nombre predeterminado se pueda restaurar seleccionando **Automático**. Un nombre personalizado especificado en la pestaña Anotaciones sobrescribe un nombre personalizado especificado por otra pestaña del cuadro de diálogo.

Texto de información sobre herramientas. El texto especificado aquí se muestra como la información sobre herramientas del nodo en el lienzo. Si no se especifica el texto de información sobre herramientas, no se mostrará cuando el usuario pase el cursor sobre el nodo.

Palabras clave. El usuario puede especificar las palabras clave que desea utilizar en los informes del proyecto y cuando realice búsquedas o seguimientos de objetos almacenados en IBM SPSS Collaboration and Deployment Services Repository.

Panel Comentarios. Esta área permite al usuario introducir el texto de comentario.

Creación y almacenamiento de información. Es un área de texto no editable que muestra la información de creación, el nombre y la fecha/hora en la que se guardó el archivo (el formato de fecha y hora depende de cada región). Si no se ha guardado, este campo indicará "Este elemento no se ha guardado".

Área de botones

En la parte inferior de cada cuadro de diálogo, se muestran los botones **Aplicar**, **Restablecer**, **Aceptar** y **Cancelar**. Si el nodo es un nodo terminal (un nodo ejecutable que procesa los datos de ruta), también se mostrará el botón **Ejecutar**.

Aceptar. Aplica todos los ajustes y cierra el cuadro de diálogo. Cuando el cuadro de diálogo se abre por primera vez desde el nodo, este botón se resalta (indicado por un rectángulo azul alrededor del botón) y al pulsar la tecla Intro se realiza la misma función que el botón Aceptar.

Cancelar. Cierra el cuadro de diálogo y deja la configuración tal y como estaba antes de abrir el cuadro de diálogo o tras haber pulsado en Aplicar. Si pulsa la tecla Esc cuando todo el cuadro de diálogo está resaltado, realiza la misma función que Cancelar.

Ejecutar. Aplica todos los ajustes, cierra el cuadro de diálogo y ejecuta el nodo terminal.

Aplicar. Guarda los ajustes del cuadro de diálogo de manera que las operaciones posteriores pueden utilizarlos.

Restablecer. Restablece todo el cuadro de diálogo a los valores que tenía tras abrir el cuadro de diálogo o tras haber pulsado en Aplicar.

Diseño de las ventanas de resultados

Esta sección describe las características de las ventanas de resultados estándar de IBM SPSS Modeler para ayudarle a diseñar ventanas de resultados coherentes en CLEF.

Las ventanas de resultados le permiten mostrar el resultado de:

- Un modelo: por ejemplo, a partir de la puntuación (aplicación de un modelo) de un conjunto de datos
- Un documento: por ejemplo, un gráfico o informe

Para obtener más información, consulte el tema "Acerca de interfaces de usuario" en la página 105.

Las ventanas de resultados son similares a los cuadros de diálogo de nodo pero con las siguientes particularidades:

- La barra de título tiene un icono en miniatura específico del nodo en lugar del icono de nugget dorado genérico
- Se omite el icono de nodo principal
- En el área de barra de herramientas y menú, el botón Maximizar/tamaño normal se omite (se puede volver a sustituir por un botón Cerrar y Eliminar en una ventana de resultados de documento) aunque la ventana se puede cambiar de tamaño también utilizando el ratón.
- El área de estado se omite
- Las pestañas suelen ser:
  - Una pestaña Modelo (de ventanas de resultados de modelo) para mostrar los datos de importancia de predictor, si esta opción está seleccionada en el nodo del modelo.
  - Una pestaña individual para el resultado.
  - Una pestaña Resumen (de ventanas de resultados de modelo) para mostrar detalles de resumen sobre el modelo.
  - Una pestaña Anotaciones (los valores de anotación se extraen del nodo que generan los resultados).
- El área de botones tiene los botones Aceptar, Cancelar, Aplicar y Restablecer

CLEF proporciona ventanas de resultados de modelo y de documentos predeterminadas de forma similar a las ilustradas anteriormente. Normalmente se muestran cuando utiliza un elemento `ModelOutput` o `DocumentOutput` en el archivo de especificación. Para obtener más información, consulte el tema "Identificador de objeto" en la página 47.

De forma alternativa, puede especificar un elemento `ModelOutput` o `DocumentOutput` de manera que sustituya completamente la ventana de resultados predeterminada por una ventana personalizada de su propio diseño. Para obtener más información, consulte el tema "Ventanas de resultados personalizadas" en la página 162.

Capítulo 3. Ejemplos de CLEF

Acerca de los ejemplos

Para ayudarle a familiarizarse con CLEF, una instalación de IBM SPSS Modeler incluye un grupo de nodos de ejemplo, junto con su código fuente completo. Son nodos básicos con una funcionalidad limitada, diseñados para ayudarle a comprender el funcionamiento de CLEF y cómo se puede utilizar. Puede probar estos nodos ahora o cuando lo estime conveniente.

Los ejemplos son:

- Nodos de lector de datos (denominado lector de registro de Apache)
- Nodo de transformador de datos (denominado analizador URL)
- Nodo generador de documentos (denominado informe de estado Web)
- Nodo generador de modelos (denominado Interacción)

Es necesario activar los ejemplos para poder utilizarlos.

Activación de los ejemplos

Los ejemplos se instalan en el directorio *Demos* en un formato comprimido como parte de una instalación de IBM SPSS Modeler. Debe activar los ejemplos extrayendo los archivos en las ubicaciones correctas tal y como se explica.

En el ordenador en el que tiene instalado IBM SPSS Modeler:

1. Salga de IBM SPSS Modeler si lo está ejecutando.
2. Busque el archivo *clef_examples_ext_lib.zip* en la carpeta *Demos* de la instalación de IBM SPSS Modeler.
3. Extraiga el contenido de *clef_examples_ext_lib.zip* en la carpeta *\ext\lib* en el directorio de instalación de IBM SPSS Modeler.

En una instalación de IBM SPSS Modeler únicamente, extraiga el contenido de *clef_examples_ext_bin.zip* en la carpeta *\ext\bin* en el directorio de instalación de IBM SPSS Modeler.

En el ordenador en el que tiene instalado IBM SPSS Modeler o IBM SPSS Modeler Server:

1. Extraiga el contenido de *clef_examples_ext_bin.zip* en la carpeta *\ext\bin* en los directorios de instalación de IBM SPSS Modeler y IBM SPSS Modeler Server.
2. Utilice el makefile suministrado en *clef_examples_ext_bin.zip* para compilar el código fuente de los ejemplos. Para obtener más información, consulte el tema “Evaluación del código fuente” en la página 28.

Por último, en todos los casos, inicie IBM SPSS Modeler y asegúrese de que todos los nodos que se muestran en la tabla siguiente sean visibles en la paleta de nodos.

Tabla 9. Nodos que están visibles en la paleta de nodos.

Pestaña Paleta	**Nodo**
Orígenes	Lector de registro de Apache
Oper. con campos	Analizador URL
Modelado	Interacción
Resultado	Informe de estado Web

Nodos de lector de datos (lector de registro de Apache)

El ejemplo del nodo de lector de datos es un nodo de origen que lee datos del archivo de registro de acceso de un servidor Web HTTP de Apache. El registro de acceso contiene información sobre todas las solicitudes que ha procesado el servidor Web. Los registros están en el formato conocido como formato de registro combinado, por ejemplo:

```
dirección_IP - - [09/Jul/2007:07:57:38 +0000] "GET /lsearch.php?county_id=3 HTTP/1.1" 200 16348
"http://www.google.co.uk/search?q=thunderbirds+cliveden&hl=en&start=10&sa=N"
"Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)"
```

Puede utilizar el nodo de ejemplo para convertir los registros en un formato tabular que resulte más fácil de leer.

Para utilizar el nodo del lector de registro de Apache:

1. Si aún no ha activado los ejemplos de CLEF, hágalo ahora. Para obtener más información, consulte el tema "Activación de los ejemplos" en la página 25.
2. Abra IBM SPSS Modeler.
3. En la pestaña Orígenes de la paleta de nodos, seleccione **Lector de registro de Apache** y añada el nodo al lienzo.
4. Edite el nodo. En el campo Lector de registro de Apache de la pestaña Opciones, introduzca:

   carpeta_demos\formato_registro_combinado.txt

   donde *carpeta_demos* es la ubicación de la carpeta *Demos* en el directorio de instalación de IBM SPSS Modeler (no utilice en formato *$CLEO_DEMOS*).
5. Pulse en **Aceptar**.
6. Añada un nodo Tipo a la ruta.
7. Edite el nodo Tipo. Pulse en **Leer valores** para leer los datos y, a continuación, pulse en **Aceptar**.
8. Conecte un nodo Tabla al nodo Tipo y ejecute la ruta. El contenido del archivo de registro se muestra en formato tabular.
9. Guarde la ruta para su uso en los dos siguientes ejemplos.

Nodo de transformador de datos (analizador URL)

El ejemplo del nodo transformador de datos realiza el procesamiento de los datos obtenidos en el ejemplo anterior. Seleccione un campo ID (que debe contener un valor exclusivo para cada fila) y un campo de entrada con una URL. El nodo genera un resultado que se compone de estos dos campos, junto con los datos URL analizados adicionalmente en campos diferentes generados. Por ejemplo, si un registro de URL contiene una cadena de consulta como:

```
http://www.dummydomain.co.uk/resource.php?res_id=89
```

el registro se analiza tal como se muestra en la tabla siguiente.

Tabla 10. Ejemplo de análisis de registro de URL.

Campo generado	Contenido
*campoURL*_servidor	*http://www.dominiovacio.com/*
*campoURL*_ruta	*/recurso.php*
*campoURL*_campo	*res_id*
*campoURL*_valor	*89*

Para utilizar el nodo de analizador URL:

1. Si se ha cerrado la ruta del ejemplo anterior, vuelva a abrirla. La ruta contiene los nodos del lector de registro de Apache y Tipo.
2. En la pestaña Operaciones con campos de la paleta de nodos, añada un nodo de analizador URL al nodo Tipo.
3. Edite el nodo de analizador URL. En la lista desplegable Campo de ID, seleccione **ReturnedContentSize**. En la lista desplegable Campo de URL, seleccione **ReferralURL**. Pulse en **Aceptar**.
4. Añada un nodo Tabla al nodo de analizador URL y ejecute la ruta. Los campos **ReturnedContentSize** y **ReferralURL** se muestran con **ReferralURL** analizado en cuatro campos diferentes generados: **ReferralURL_servidor**, **ReferralURL_ruta**, **ReferralURL_campo** y **ReferralURL_valor**

Nodo generador de documentos (informe de estado Web)

El ejemplo del nodo generador de documentos lee los datos transmitidos del registro del servidor Web y genera un informe de la forma de un archivo HTML. El informe contiene una tabla que muestra los porcentajes de los registros que devuelven diferentes códigos de estado HTTP (por ejemplo, 200, 302, 404, etc.).

Para utilizar el nodo de informe de estado Web:

1. Si se ha cerrado la ruta del primer ejemplo, vuelva a abrirla. Es la ruta que contiene los nodos del lector de registro de Apache y Tipo. Si su ruta contiene un nodo de analizador URL del segundo ejemplo, ese nodo se ignora en este ejemplo.
2. En la pestaña Resultado de la paleta de nodos, añada un nodo de informe de estado Web al nodo Tipo.
3. Edite el nodo de informe de estado Web. En la lista desplegable Campo de código de estado, seleccione **StatusCode**. Pulse en **Ejecutar**. Se abre una ventana de resultado con el contenido del informe.

Nodo generador de modelos (Interacción)

El ejemplo del nodo de generación de modelos funciona con independencia del resto de los ejemplos y permite crear un modelo simple en modo estándar (no interactivo) o para interactuar con el modelo generado anteriormente. El modelo intenta predecir el abandono de clientes de una empresa de telecomunicaciones.

Para utilizar el nodo Interacción:

1. Si aún no ha activado los ejemplos de CLEF, hágalo ahora. Para obtener más información, consulte el tema “Activación de los ejemplos” en la página 25.
2. Cree una nueva ruta en IBM SPSS Modeler.
3. Añada un nodo de origen Archivo Statistics que importe el archivo *telco.sav* del directorio *Demos*.
4. En la pestaña Tipos, pulse en **Leer valores** y pulse en **Aceptar** en el cuadro del mensaje para confirmar.
5. Defina el rol del campo de **abandono** (el último de la lista) como **Objetivo** y pulse en **Aceptar**.
6. En la pestaña Modelados de la paleta de nodos, añada un nodo Interacción al nodo de origen.

Para comprobar la generación de modelos estándar (no interactivos):

1. Ejecute la ruta para crear el nugget de modelo en la ruta y, en la paleta Modelos en la parte superior de la pantalla.
2. Conecte un nodo Tabla a este nugget de modelo.
3. Ejecute el nodo Tabla. Desplácese a la derecha de la ventana de resultado de la tabla para ver las predicciones de abandono. El campo `$I-churn` contiene los valores predichos, mientras que `$IP-churn` muestra los valores de confianza (de 0,0 a 1,0) de las predicciones.

Para probar la generación de modelos interactiva:

1. En la pestaña Modelo del cuadro de diálogo del generador de modelos de Interacción, seleccione **Iniciar una sesión interactiva**.
2. Pulse en **Ejecutar** para abrir el cuadro de diálogo de prueba de interacción.
3. En el cuadro de diálogo de prueba de interacción, pulse en **Iniciar tarea de generación** para mostrar el progreso de generación de modelos.
4. Cuando haya finalizado la operación de generación de modelos, seleccione la fila que se ha añadido a la tabla de tareas de generación del cuadro de diálogo.
5. En el área de la barra de herramientas de la parte superior del cuadro de diálogo, pulse en el botón con el icono en forma de diamante amarillo. Genera el objeto de resultado de modelo (denominado **model_1**) en la paleta Modelos en la parte derecha de la pantalla.

El modelo generado de forma interactiva es idéntico al primer modelo, salvo que tiene un nombre diferente. Si repite el proceso de **Iniciar tarea de generación** se genera otro modelo idéntico denominado **modelo_2**, etcétera.

Evaluación de los archivos de especificación

Una excelente forma de comprender el funcionamiento de CLEF es evaluar los archivos de especificación de los ejemplos proporcionados. Puede encontrar estos archivos en:

dir_instal\ext\lib*carpeta_extensión*\extension.xml

donde *dir_instal* es el directorio de instalación de IBM SPSS Modeler y *carpeta_extensión* es una de los siguientes:

- *spss.lectorregistroapache*
- *spss.interacción*
- *spss.analizadorurl*
- *spss.informeestadoweb*

Puede ver otras carpetas de extensión en *\ext\lib*; son carpetas relacionadas con los nodos del sistema de IBM SPSS Modeler que se generan mediante CLEF. Estos nodos aparecen en su instalación en función de los módulos de IBM SPSS Modeler de que disponga. Puede encontrar de gran utilidad examinar los archivos de especificación, pero **no modifique estos archivos en ninguna circunstancia**. Si lo hace, es posible que los nodos no funcionen correctamente, en cuyo caso deberá volver a instalar IBM SPSS Modeler. IBM Corp. no admite modificaciones en archivos del sistema.

Evaluación del código fuente

Como referencia, también se incluye el código fuente completo de los nodos de ejemplo. Todos los nodos de ejemplo utilizan bibliotecas de servidor de C++, pero únicamente el nodo Interacción utiliza además clases de Java de cliente.

Los archivos de código fuente se extraen automáticamente cuando activa los ejemplos y se instalan tal como se muestra en la tabla siguiente:

Tabla 11. Instalación del archivo de código fuente

Ubicación	Contenido
...\ext\lib\spss.interaction\src	Código fuente de Java de los archivos `.class` en el archivo `ui.jar` en la carpeta padre
...\ext\bin\spss.apachelogreader\src *...\ext\bin\spss.interaction\src* *...\ext\bin\spss.urlparser\src* *...\ext\bin\spss.webstatusreport\src*	Archivos de origen C++ y de proyecto de las DLL de la carpeta padre

Eliminación de los ejemplos

Si ya no desea ver los nodos de ejemplo en IBM SPSS Modeler, puede eliminarlos de la siguiente manera:

1. Salga de IBM SPSS Modeler.
2. Elimine las carpetas de ejemplo de los directorios *\ext\bin* y *\ext\lib* de la instalación de IBM SPSS Modeler. No elimine ninguna de las carpetas estándar de IBM SPSS Modeler por error. Si lo hace, tendrá que volver a instalar el producto de IBM SPSS Modeler. Las carpetas que debe eliminar son las siguientes:

- *spss.lectorregistroapache*
- *spss.analizadorurl*
- *spss.informeestadoweb*
- *spss.interacción*

Los cambios surtirán efecto la próxima vez que inicie IBM SPSS Modeler.

Capítulo 4. Archivo de especificación

Conceptos básicos de los archivos de especificación

Todas las extensiones CLEF deben incluir un archivo XML en el que se definan todas las características de extensión. Este archivo se conoce como el **archivo de especificación**, y siempre se denomina `extension.xml`. Este archivo de especificación consta de las siguientes secciones:

- **declaración XML**. Declaración opcional de la versión XML y de otra información.
- **Elemento de extensión**. Parte principal del archivo, contiene todas las secciones siguientes.
- **Sección de detalles de extensión**. Especifica información básica sobre la extensión.
- **Sección de recursos**. Especifica recursos externos necesarios para que funcione la extensión, como paquetes de recursos, archivos JAR y bibliotecas compartidas.
- **Sección de objetos comunes**. (Opcional.) Define elementos que pueden utilizarse o a los que hacer referencia mediante otros objetos de la extensión como tipos de propiedades, modelos y documentos.
- **Sección de interfaz de usuario (Paletas)**. (Opcional.) Define la paleta o subpaleta personalizada en la que debe aparecer un nodo.
- **Sección de definición de objetos**. Identifica el objeto u objetos definidos por la extensión, como los nodos, resultados de modelo y resultados de documento.

Cada sección puede contener declaraciones estáticas (como componentes en un elemento), procesos dinámicos sencillos (como el cálculo de un modelo de datos de salida de un nodo) o ambos. El formato general de un archivo de especificación CLEF es el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<Extension ...>
   <ExtensionDetails ... />
   <Recursos
      Sección de recursos
   </Resources>
   <CommonObjects>
      Sección de objetos comunes
   </CommonObjects>
   <UserInterface >
      Sección de interfaz de usuario (Paletas)
   </UserInterface>
   Sección de definición de objetos
     definición de objeto
     definición de objeto
     definición de objeto
    ...
</Extension>
```

Líneas de comentarios

En cualquier punto del archivo de especificación puede incluir una línea de comentario con el siguiente formato:

```
<!- texto del comentario ->
```

¿Obligatorio u opcional?

En las definiciones de elemento de las siguientes secciones (por lo general identificadas mediante la cabecera **Formato**), los atributos de elemento y los elementos hijo son opcionales a no ser que se indiquen como "(obligatorios)". Para conocer la sintaxis completa de los elementos, consulte “Esquema XML de CLEF”, en la página 203.

Archivo de especificación de ejemplo

A continuación se incluye un ejemplo completo de un archivo de especificación CLEF, que en este caso es para un nodo de transformador de datos simple.

```
<?xml version="1.0" encoding="UTF-8"?>
<Extension version="1.0" debug="true">
     <ExtensionDetails id="urlparser" providerTag="spss" label="URL CLEF Module" version="1.0"
provider="IBM Corp." copyright="(c) 2005-2011 IBM Corp." description="Extensión CLEF de transformación URL"/>
     <Resources>
          <SharedLibrary id="urlparser_library" path="spss.urlparser/urlparser" />
     </Resources>
     <Node id="urlparser_node" type="dataTransformer" palette="fieldOp" label="URL Parser">
          <Properties>
               <Property name="id_fieldname" valueType="integer" label="ID field" />
               <Property name="url_fieldname" valueType="string" label="URL field" />
          </Properties>
          <UserInterface >
               <Icons />
               <Pestañas>
                    <Tab label="Types" labelKey="optionsTab.LABEL">
                         <PropertiesPanel>
                              <SingleFieldChooserControl property="id_fieldname"  storage="integer" />
                              <SingleFieldChooserControl property="url_fieldname"  storage="string" />
                         </PropertiesPanel>
                    </Tab>
               </Tabs>
               <Controls />
          </UserInterface>
          <Execution>
               <Module libraryId="urlparser_library" name="">
                    <StatusCodes>
                         <StatusCode code="0" status="error" message="No se puede inicializar un homólogo" />
                         <StatusCode code="1" status="error" message="Error al leer los datos de entrada" />
                         <StatusCode code="2" status="error" message="Error interno" />
                         <StatusCode code="3" status="error" message="El campo de entrada no existe" />
                    </StatusCodes>
               </Module>
          </Execution>
          <OutputDataModel mode="replace">
               <AddField name="${id_nombre campo}" fieldRef="${id_nombre campo}"/>
               <AddField name="${url_nombre campo}" fieldRef="${url_nombre campo}"/>
               <AddField name="${url_nombre campo}_server" storage="string" />
               <AddField name="${url_nombre campo}_ruta" storage="string" />
               <AddField name="${url_nombre campo}_campo" storage="string" />
               <AddField name="${url_nombre campo}_ruta" storage="string" />
          </OutputDataModel>
     </Node>
</Extension>
```

El elemento `ExtensionDetails` ofrece información básica sobre la extensión que utiliza internamente IBM SPSS Modeler.

El elemento `Resources` especifica la ubicación de una biblioteca en el servidor a la que se hará referencia posteriormente en el archivo. La especificación de ruta indica que la biblioteca se encuentra en el directorio de instalación de IBM SPSS Modeler, bajo `\ext\bin\spss.urlparser\urlparser.dll`.

Este archivo de especificación concreto no incluye el elemento `CommonObjects`.

El elemento `Node` especifica toda la información sobre el propio nodo:

- En `Properties`, se indican inicialmente dos propiedades para un uso posterior en una pestaña del cuadro de diálogo de nodo.
- El elemento `UserInterface` define el aspecto y diseño de la pestaña de cuadro de diálogo de nodo que es específica de esta extensión (IBM SPSS Modeler ofrece otras pestañas).
- El elemento `Execution` define elementos que se utilizan cuando se ejecuta el nodo. En este caso, estos elementos se encuentran en la biblioteca del servidor que se indicó anteriormente en el archivo, y hay un conjunto de mensajes que se mostrarán si la ejecución muestra un código de estado concreto.
- El elemento `OutputDataModel` define la transformación de datos que realiza este nodo. Especifica que el modelo de datos de entrada (el conjunto de campos introducidos en este nodo) debe sustituirse por el conjunto de nodos definidos aquí, que constituyen el modelo de datos de salida (el conjunto de campos pasados a todos los nodos a partir de aquí, a no ser que el modelo se modifique posteriormente). En este ejemplo en particular, el nodo pasa los dos campos originales (`id_fieldname` y `url_fieldname`) sin modificarlos, pero añade cuatro campos más cuyos nombres derivan de `url_fieldname`.

Este archivo de especificación concreto se toma de uno de los nodos de ejemplo suministrados como parte de la instalación de IBM SPSS Modeler. Para obtener más información, consulte el tema "Nodo de transformador de datos (analizador URL)" en la página 26.

Declaración XML

La declaración XML es opcional, y especifica la versión de XML utilizada, junto con información del formato de codificación de caracteres.

Ejemplo

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

Elemento Extension

El elemento Extension constituye la parte principal del archivo y contiene todas las demás secciones. El formato es:

```
<Extension version="número_versión" debug="true_false">
     Sección de detalles de extensión
     Sección de recursos
     Sección de objetos comunes
     Sección de interfaz de usuario (Paletas)
     Sección de definición de objetos
</Extension>
```

donde:

`version` es el número de versión de la extensión.

La opción `debug` (depurar) es opcional; si se define como `true`, se añadirá la pestaña **Depurar** a cualquier cuadro de diálogo o marco asociado con un nodo o resultado de CLEF, y ofrecerá acceso a las propiedades y contenedores definidos para ese objeto. El valor predeterminado es `false`. Para obtener más información, consulte el tema "Uso de la pestaña Depurar" en la página 200.

Sección de detalles de extensión

La sección de detalles de extensión ofrece información básica sobre la extensión.

Formato

```
<ExtensionDetails providerTag="etiqueta_proveedor_extensión"
     id="identificador_exclusivo_extensión"
     label="nombre_visualización" version="número_versión_extensión"
     provider="proveedor_extensión" copyright="aviso_copyright"
     description="descripción_extensión"/>
```

donde:

`providerTag` (obligatorio) es un nombre que identifica de forma exclusiva al proveedor de esta extensión. Tenga en cuenta que el valor no debe incluir la cadena `spss`, que queda reservada a un uso interno.

`id` (obligatorio) es un nombre que identifica de forma exclusiva a esta extensión y que se utiliza en mensajes de sistema para referirse a ella. Por convención, el archivo de extensión se ubica en una carpeta denominada \ext\lib*providerTag.id* dentro del directorio de instalación de IBM SPSS Modeler.

`etiqueta` (obligatorio) es la etiqueta de visualización de la extensión. Este texto se muestra en el campo Nombre del Gestor de paletas cuando se añade el nodo. Para obtener más información, consulte el tema "Prueba de una extensión de CLEF" en la página 199.

`version` es el número de versión de esta extensión.

`provider` es la cadena que identifica al proveedor de esta extensión. Este texto se muestra en el campo Proveedor del Gestor de paletas cuando se añade el nodo. El valor predeterminado es la cadena `(desconocido)`.

`copyright` es el aviso de copyright de esta extensión. Este texto se muestra en el campo Copyright del Gestor de paletas cuando se añade el nodo.

`description` es la descripción breve que indica el objetivo de la extensión. Este texto se muestra en el campo Descripción del Gestor de paletas cuando se añade el nodo.

Ejemplo

```
<ExtensionDetails providerTag="miempresa" id="clasificador" name="Clasificar datos" version="1.2"
     provider="Mi Empresa S.A." copyright="(c) 2005-2006 Mi Empresa S.A."
     description="Extensión de ejemplo que clasifica los datos mediante comandos integrados del SO."/>
```

Sección de recursos

Esta sección define qué recursos externos son necesarios para que funcione esta extensión.

Formato

```
<Resources>
     <Bundle .../>
      ...
     <JarFile .../> ...
     <SharedLibrary .../> ...
     <HelpInfo .../>
</Resources>
```

donde:

`Bundle` identifica un conjunto de recursos localizados en el cliente. Para obtener más información, consulte el tema "Paquetes" en la página 35.

`JarFile` identifica un archivo jar de Java en el cliente. Para obtener más información, consulte el tema "Archivos Jar" en la página 35.

SharedLibrary identifica una biblioteca o DLL en el servidor. Para obtener más información, consulte el tema “Bibliotecas compartidas” en la página 36.

HelpInfo especifica el tipo de información de ayuda de la extensión, en caso de tener alguna. Para obtener más información, consulte el tema “Implementación de un sistema de ayuda” en la página 163.

Ejemplo

```
<Resources>
     <SharedLibrary id="discriminantnode" path="spss.xd/Discriminant"/>
     <Bundle id="translations.discrim" type="properties" path="messages"/>
     <JarFile id="java" path="discriminant.jar"/>
     <HelpInfo id="help" type="native"/>
</Resources>
```

Paquetes

El elemento Bundle especifica un paquete de recursos en el cliente (como un conjunto de mensajes de texto de localización) que pueden implementarse como archivo .properties como un archivo .class de Java. Para obtener más información, consulte el tema “Localización” en la página 167.

Formato

<Bundle id="*identificador*" path="*ruta*"/>

donde:

id (obligatorio) es un identificador exclusivo para este paquete.

path (obligatorio) especifica la ubicación del archivo de paquete relativo a la carpeta padre de este archivo de especificación. Cuando el paquete se refiere a un archivo .properties, la ruta no debe incluir extensiones de lenguaje ni el sufijo .properties.

Ejemplo

```
<Bundle id="traducciones.discrim" path="mensajes"/>
```

Esto indica que hay un paquete de recursos en un archivo denominado mensajes.properties en la misma carpeta que el archivo de especificación.

Archivos Jar

El elemento JarFile especifica un archivo de Java en el cliente (.jar) que ofrece clases de Java y otros recursos de cliente para esta extensión.

Formato

<JarFile id="*identificador*" path="*ruta*"/>

donde:

id (obligatorio) es un identificador exclusivo para este archivo .jar.

path (obligatorio) especifica la ubicación del archivo .jar relativo a la carpeta padre de este archivo de especificación.

Ejemplo

```
<JarFile id="java" path="coxreg_model_terms.jar"/>
```

Esto indica que el archivo .jar de esta extensión se encuentra en la misma carpeta que el archivo de especificación.

Bibliotecas compartidas

El elemento SharedLibrary especifica una biblioteca compartida o DLL en el lado del servidor. Por lo general, esto sólo es necesario para permitir la ejecución del nodo. Cuando una biblioteca implementa múltiples módulos, un elemento Module de la sección de ejecución de la especificación del nodo identifica un módulo específico de la biblioteca.

Formato

```
<SharedLibrary id="identificador" path="ruta"/>
```

donde:

id (obligatorio) es un identificador exclusivo para esta biblioteca compartida.

path (obligatorio) especifica la ubicación de la biblioteca compartida relativa a la carpeta \ext\bin del directorio de instalación del servidor. Tenga en cuenta que la ruta no debe incluir la extensión de archivo de la biblioteca compartida (p. ej. .dll).

Ejemplo

La siguiente declaración de biblioteca compartida:

```
<SharedLibrary id="agrupación" path="spss.binning/Binning" />
```

especifica que la biblioteca compartida debe cargarse desde:

```
dir_instal\ext\bin\spss.binning\Binning.dll
```

donde *dir_instal* es el directorio en el que están instalador los componentes de CLEF del servidor. Como esta biblioteca implementa más de un módulo, el módulo necesario (IntervaloSupervisado) se identifica mediante un elemento Module en la especificación del nodo de generación, haciendo referencia al identificador de la biblioteca de la siguiente forma:

```
<Execution>
     <Module libraryId="agrupación" name="supervisedBinning" .../>
 ...
</Execution>
```

Información de ayuda

El elemento opcional HelpInfo indica cuál de los posibles tipos de ayuda se ofrecen para esta extensión. Para obtener más información, consulte el tema “Implementación de un sistema de ayuda” en la página 163.

Sección de objetos comunes

La sección opcional de objetos comunes define los objetos que pueden compartirse entre elementos definidos en otro lugar del archivo de especificación. Algunos tipos de objetos de esta sección (como las enumeraciones de propiedades) también pueden definirse localmente donde son necesarias, mientras que otros (como los modelos y documentos) sólo pueden definirse aquí.

Formato

```
<CommonObjects>
     <PropertyTypes .../>
     <PropertySets .../>
```

```
     <ContainerTypes .../>
     <Actions .../>
     <Catalogs .../>
</CommonObjects>
```

donde:

`PropertyTypes` (tipos de propiedad) admite que se compartan definiciones de propiedades comunes entre objetos. Para obtener más información, consulte el tema "Tipos de propiedad".

`PropertySets` (conjuntos de propiedades) suelen utilizarse cuando los nodos generadores de modelos, objetos de resultado de modelo y nodos aplicadores de modelos incluyen el mismo conjunto de propiedades. Para obtener más información, consulte el tema "Conjuntos de propiedades" en la página 38.

`ContainerTypes` (tipos de contenedor) define los tipos de contenedores, que son objetos que pueden rodear estructuras de datos complejos. Para obtener más información, consulte el tema "Tipos de contenedor" en la página 39.

`Actions` (acciones) define información básica de las interacciones de los usuarios mediante menús o barras de herramientas, por ejemplo. Para obtener más información, consulte el tema "Acciones" en la página 40.

`Catalogs` (Catálogos) implementan un control que permite elegir una o más opciones de una lista de valores que genera el servidor dinámicamente. Para obtener más información, consulte el tema "Catalogs" en la página 41.

Ejemplo

```
<CommonObjects>
     <ContainerTypes>
          <ModelType id="discriminant_model" format="utf8" />
          <DocumentType id="resultado_html" />
          <DocumentType id="zip_outputType" format="binary"/>
     </ContainerTypes>
</CommonObjects>
```

Tipos de propiedad

La sección opcional de tipos de propiedad permite que se compartan definiciones de propiedades comunes entre objetos. Esto se debe en parte a una mayor facilidad de mantenimiento. Por ejemplo, una propiedad puede aparecer en un único lugar en vez de duplicada en varias veces. Las definiciones también se comparten para asegurar la compatibilidad entre las propiedades de diferentes objetos cuyos valores se copian cuando se crea una nueva instancia de un objeto.

Los tipos de propiedad sólo pueden definirse en la sección de objetos comunes.

Formato

```
<PropertyTypes>
     <PropertyType id="identificador" isKeyed="true_false" isList="true_false" max="valor_máx"
       min="valor_mín" valueType="tipo_valor">
          <Enumeration ... />
          <Structure ... />
          <DefaultValue ... />
     </PropertyType>
     <PropertyType ... />
      ...
</PropertyTypes>
```

Los atributos PropertyType son los siguientes.

id (obligatorio) es un identificador exclusivo para el tipo de propiedad.

si isKeyed se define como true, el tipo de propiedad cuenta con clave. Una propiedad con clave asocia un conjunto predefinido de operaciones con un campo mediante un control de propiedades definidas por el usuario (consulte "Control de propiedad" en la página 140). Si isKeyed se define como true, el atributo valueType se debe definir como structure. Si desea obtener más información sobre las propiedades estructuradas, consulte "Propiedades estructuradas" en la página 62.

isList especifica si la propiedad es una lista de valores del tipo de valor especificado (true) o un único valor (false).

max y min denotan los valores máximo y mínimo de un rango.

valueType puede ser cualquiera de los siguientes:

- string (cadena)
- encryptedString (cadena cifrada)
- fieldName (nombre de campo)
- integer (entero)
- double (doble)
- boolean (booleano)
- date (fecha)
- enum (enumeración) (consulte "Propiedades enumeradas" en la página 61)
- structure (estructura) (consulte "Propiedades estructuradas" en la página 62)
- databaseConnection (conexión con base de datos)

Los elementos hijo Enumeration y Structure se excluyen entre sí. Los elementos hijo Enumeration, Structure y DefaultValue se utilizan en casos específicos; consulte "Propiedades enumeradas" en la página 61, "Propiedades estructuradas" en la página 62 y "Valores predeterminados" en la página 63.

Conjuntos de propiedades

Los conjuntos de propiedades suelen utilizarse cuando los nodos generadores de modelos, objetos de resultado de modelo y nodos aplicadores de modelos incluyen el mismo conjunto de propiedades. Por ejemplo, un nodo generador de modelos puede definir un conjunto de propiedades que pueden definirse en el generador, pero que no se utilizan realmente hasta la aplicación del modelo. Para poder transferirlo automáticamente, también tiene que incluirse en el resultado del modelo.

Formato

```
<PropertySets>
     <PropertySet id="identificador">
          <Property ... />
          <Property ... />
           ...
     </PropertySet>
       ...
</PropertySets>
```

donde id es un identificador exclusivo de este conjunto de propiedades.

Para obtener una descripción del elemento Property, consulte "Propiedades" en la página 51.

Ejemplo

Este ejemplo muestra la definición de un conjunto de dos propiedades: el número de predicciones a producir y si se deben incluir probabilidades. En la sección de objetos comunes se puede definir:

```
<PropertySets>
     <PropertySet id="propiedades_modelo_común">
          <Property name="prediction_count" valueType="integer" min="1" max="10"/>
          <Property name="include_probabilities" valueType="boolean" defaultValue="false"/>
     </PropertySet>
       ...
</PropertySets>
```

A continuación, en cada una de las definiciones de nodo generador de modelos, objeto de resultado de modelo y nodo aplicador de modelos, contará con un atributo `includePropertySets` como los siguientes (que ilustra la definición del nodo generador de modelos únicamente):

```
<Node id="mi_generador" type="modelBuilder" ...>
     <Properties includePropertySets="[propiedades_modelo_común]">
      ...
     </Properties>
      ...
</Node>
```

Tipos de contenedor

Los contenedores son objetos que actúan como marcadores de posición de estructuras de datos complejos como modelos y documentos. Un contenedor se define como de un tipo particular, y los tipos de contenedor se definen aquí. Es posible definir los siguientes tipos de contenedor:

- tipos de modelo
- tipos de documento

Los tipos de contenedores se pueden transferir entre cliente y servidor, clonarse y guardarse en un archivo o repositorio de contenidos. Un modelo se clona cuando un nodo aplicador de modelos se genera desde un objeto de resultado de modelo.

Cada tipo de contenedor tiene su propio conjunto de propiedades predefinido, aunque es posible añadir propiedades personalizadas. Los tipos de contenedor sólo pueden definirse en la sección de objetos comunes.

Formato

El formato de la sección de tipos de contenedor es:

```
<ContainerTypes>
     <ModelType ... />
      ...
     <DocumentType ... />
      ...
</ContainerTypes>
```

donde:

`ModelType` especifica el formato de un tipo concreto de modelo. Para obtener más información, consulte el tema “Tipos de modelos” en la página 40.

`DocumentType` especifica el formato de un tipo concreto de documento. Para obtener más información, consulte el tema “Tipos de documento” en la página 40.

Ejemplo

```
<ContainerTypes>
     <ModelType id="modelo_discriminante" format="utf8" />
     <DocumentType id="resultado_html" />
     <DocumentType id="zip_outputType" format="binary"/>
</ContainerTypes>
```

Tipos de modelos

Un modelo debe ofrecer la información de nombre de algoritmo, tipo de modelo y modelos de entrada y resultados de datos. Una definición de tipo de modelo especifica el formato de un tipo concreto de modelo.

La información de tipo de modelo debe especificarse de forma estática en el archivo de especificaciones o dinámica cuando el modelo se construye mediante el nodo generador de modelos.

Formato

```
<ModelType id="identificador" format="formato_tipo_modelo" />
```

donde:

- `id` (obligatorio) es un identificador exclusivo para el tipo de modelo.
- `format` (obligatorio) es el formato del tipo de modelo, y puede ser `utf8` (texto) o `binary` (binario). El formato de modelo debe especificarse como parte de la información estática.

Ejemplo

```
<ModelType id="mi_modelo" format="utf8" />
```

Tipos de documento

Un **documento** es un objeto de resultados como un gráfico o un informe. Una definición de tipo de documento especifica el formato de un tipo concreto de modelo.

Formato

```
<DocumentType id="identificador" format="formato_tipo_documento" />
```

donde:

- `id` (obligatorio) es un identificador exclusivo para el tipo de documento.
- `format` (obligatorio) es el formato del tipo de documento y puede ser `utf8` (texto) o `binary` (binario).

Ejemplos

```
<DocumentType id="html_output" format="utf8" />
<DocumentType id="zip_outputType" format="binary"/>
```

Acciones

Actions (acciones) define información básica de las interacciones de los usuarios mediante menús o barras de herramientas, por ejemplo. Cada acción define cómo debe representarse en la interfaz de usuario, si como etiqueta, información sobre herramientas o icono. Una colección de acciones se gestiona mediante una clase de Java en el lado del cliente que se define para cada grupo de acciones. Las acciones también pueden definirse dentro de objetos específicos.

Formato

```
<Acciones>
     <Action id="identificador" label="etiqueta_visualización" labelKey="clave_etiqueta"
     description="descripción_acción" descriptionKey="clave_descripción" imagePath="ruta_imagen"
```

```
     imagePathKey="clave_ruta_imagen" mnemonic="carácter_mnemónico" mnemonicKey="tecla_mnemónica"
     shortcut="serie_atajo" shortcutKey="tecla_atajo" />
      ...
</Actions>
```

donde:

`id` (obligatorio) es un identificador exclusivo para la acción.

`label` (obligatorio) es el nombre de visualización de la acción tal y como aparece en la interfaz de usuario.

`labelKey` identifica la etiqueta con fines de localización.

`description` es una descripción de la acción--por ejemplo, para un elemento de menú personalizado o un botón de acción de icono o barra de herramientas, debe tratarse del texto de la información sobre herramientas para ese botón o elemento de menú.

`descriptionKey` identifica la descripción con fines de localización.

`imagePath` es la ubicación de un archivo gráfico, como una imagen de icono. La ubicación proporcionada es relativa al directorio en el que se instala el archivo de especificación.

`imagePathKey` identifica la ruta de imagen con fines de localización.

`mnemonic` es el carácter alfabético utilizado junto con la tecla Alt que activa el control (por ejemplo, si proporciona el valor S, el usuarios puede activar este control mediante Alt-S).

`mnemonicKey` identifica el atributo mnemonic con fines de localización. Si no utiliza `mnemonic` ni `mnemonicKey`, no habrá ningún atributo mnemonic disponible para este control. Para obtener más información, consulte el tema “Claves de acceso y atajos de teclado” en la página 114.

`acceso directo` es una cadena que indica un atajo de teclado (por ejemplo, `CTRL+MAYÚS+A`) que se puede utilizar para iniciar esta acción.

`shortcutKey` identifica el atajo con fines de localización. Si no utiliza `shortcut` ni `shortcutKey`, no habrá ningún acceso directo disponible para esta acción. Para obtener más información, consulte el tema “Claves de acceso y atajos de teclado” en la página 114.

Ejemplo

```
<Actions>
     <Action id="generateSelect" label="Nodo Seleccionar..." labelKey="generate.selectNode.LABEL"
      imagePath="images/generate.gif" description="Genera un nodo Seleccionar"
      descriptionKey="generate.selectNode.TOOLTIP"/>
     <Action id="generateDerive" label="Nodo Derivar..." labelKey="generate.deriveNode.LABEL"
      imagePath="images/generate.gif" description="Genera un nodo de derivación"
      descriptionKey="generate.deriveNode.TOOLTIP"/>
</Actions>
```

Catalogs

Los catálogos permiten asociar una propiedad con un control que permite al usuario seleccionar una o más opciones de una lista de valores que genere el servidor de forma dinámica.

Los valores se muestran en el control como una lista emergente cuando el usuario pulsa en la entrada **<Select>**.

Si el usuario selecciona una fila de la lista, el valor de la fila de una columna especificada en el elemento `Catalog` se coloca en el control.

Formato

```
<CommonObjects>
     <Catalogs>
          <Catalog id="identificador" valueColumn="entero">
               <Attribute label="nombre_visualización" />
                ...
          </Catalog>
           ...
     </Catalogs>
</CommonObjects>
```

donde:

`id` (obligatorio) es un identificador exclusivo para el catálogo.

`valueColumn` (necesario) es el número de la columna cuyos valores se colocarán en el control cuando el usuario selecciona una fila. La numeración de las columnas comienza en 1.

Utilice un elemento `Attribute` por columna, en el orden de las columnas. Consulte el ejemplo siguiente.

Si el usuario activa un control asociado con un catálogo, el catálogo con la lista de valores se recupera del servidor mediante la activación de la función `getCatalogInformation`. Esta función devuelve un documento XML con la lista de los valores. Para obtener más información, consulte el tema "Funciones de homólogo" en la página 178.

Ejemplo

Este ejemplo muestra parte del código utilizado para definir controles de catálogo. Tres catálogos se definen y asocian a tres controles diferentes en una pestaña de un cuadro de diálogo.

En primer lugar, los catálogos se definen en la sección de objetos comunes:

```
<CommonObjects>
     <Catalogs>
          <Catalog id="cat1" valueColumn="1">
               <Attribute label="col1" />
               <Attribute label="col2" />
          </Catalog>
          <Catalog id="cat2" valueColumn="2">
               <Attribute label="col1" />
               <Attribute label="col2" />
               <Attribute label="col3" />
          </Catalog>
          <Catalog id="cat3" valueColumn="1">
               <Attribute label="col1" />
          </Catalog>
     </Catalogs>
</CommonObjects>
```

A continuación, las propiedades que se asocian con los controles se definen en la sección de propiedades de la definición del nodo:

```
<Node id="catalognode" type="dataReader" palette="import" label="Catalog">
     <Properties>
          <Property name="sometext" valueType="string" label="Some Text" />
```

```
        <Property name="selection1" valueType="string" label="Selection 1" />
        <Property name="selection2" valueType="string" isList="true" label="Selection 2" />
        <Property name="selection3" valueType="string" label="Selection 3" />
    </Properties>
```

En la sección de interfaz de usuario de la definición del nodo, los controles se definen y asocian con las definiciones del catálogo en las referencias a las propiedades:

```
<UserInterface >
    <Pestañas>
        <Tab label="Controles de catálogo" labelKey="Catalog.LABEL">
            <PropertiesPanel>
                <TextBoxControl property="sometext" />
                <SingleItemChooserControl property="selection1" catalog="cat1" />
                <MultiItemChooserControl property="selection2" catalog="cat2" />
                <SingleItemChooserControl property="selection3" catalog="cat3" />
            </PropertiesPanel>
        </Tab>
```

Sección de interfaz de usuario (Paletas)

Esta sección es opcional y sólo se incluye si desea que esta extensión defina la paleta o subpaleta personalizada en la que debe aparecer un nodo.

Si una extensión define una paleta o subpaleta personalizada, las extensiones que cargue posteriormente y definan nodos para incluirlos en la misma paleta o subpaleta podrán omitir esta sección de interfaz de usuario (paletas). Todo lo que necesitan es que el elemento `Node` haga referencia a la paleta en el atributo `customPalette`. Las extensiones se cargan por orden alfabético según el valor de *providerTag.id*, donde estos son los valores de los atributos `providerTag` e `id` del elemento `ExtensionDetails` para esta extensión (consulte "Sección de detalles de extensión" en la página 33). Así, por ejemplo, la extensión `mico.abc` se carga antes de la extensión `mico.def`.

Nota: La sección interfaz de usuario (Paletas) es diferente de la sección de interfaz de usuario principal, que aparece como parte de una definición de objeto individual y que se describe en Capítulo 6, "Generación de interfaces de usuario", en la página 105.

Formato

El formato de la sección de interfaz de usuario (paletas) es:

```
<UserInterface >
    <Paletas>
        <Palette id="nombre" systemPalette="nombre_paleta" customPalette="nombre_paleta"
           relativePosition="posición" relativeTo="paleta" label="etiqueta_visualización"
           labelKey="clave_etiqueta" description="descripción" descriptionKey="clave_descripción"
           imagePath="ruta_imagen" />
        <Palette ... />
     ...
    </Palettes>
</UserInterface>
```

Tabla 12. Atributos de Paleta.

Atributo	Descripción
`id`	(Obligatorio.) Identificador exclusivo para la paleta o subpaleta que está definiendo.

Tabla 12. Atributos de Paleta (continuación).

Atributo	Descripción
systemPalette	Sólo se usa cuando se añade una subpaleta a una paleta de sistema, e identifica la paleta de sistema en la que aparecerá esta subpaleta: import - Orígenes recordOp - Operaciones con registros fieldOp - Operaciones con campos graph - Gráficos modeling - Modelado (consulte a continuación) dbModeling - Modelado de bases de datos output - Resultados export - Exportar
customPalette	Sólo se usa cuando se añade una subpaleta a una paleta personalizada, e identifica la paleta personalizada en la que aparecerá esta subpaleta. Se trata del valor del atributo id del elemento Palette que define la paleta personalizada.
relativePosition	Sólo se utiliza cuando se define una paleta personalizada y especifica su posición en la franja de paleta en la parte inferior de la pantalla. Los valores posibles son: primera last before after Si el valor es before (antes) o after (después), el atributo relativeTo (con respecto a) también es obligatorio (véase abajo). Si se omite relativePosition, la paleta se sitúa al final de la franja.
relativeTo	Si el valor de relativePosition (posición relativa) es before (antes) o after (después), entonces relativeTo (con respecto a) se utilizará para especificar el identificador de la paleta a la que precede o sigue esta paleta personalizada. Los identificadores de paleta se enumeran como valores del atributo de paleta del elemento Nodo (consulte "Nodo" en la página 48).
label	(Obligatorio.) Nombre de visualización de la acción tal y como aparece en la interfaz de usuario.
labelKey	Identifica la etiqueta con fines de localización.
description	El texto de la información sobre herramientas aparece cuando se pasa el cursor sobre la pestaña de paleta (no se utiliza en subpaletas). Este valor también actúa como la descripción larga accesible del control. Para obtener más información, consulte el tema "Accesibilidad" en la página 173.
descriptionKey	Identifica la descripción con fines de localización.
imagePath	Identifica la ubicación de la imagen empleada en la pestaña de paleta (no se utiliza en subpaletas). La ubicación proporcionada es relativa al directorio en el que se instala el archivo de especificación. Si se omite este atributo no se utilizará ninguna imagen.

Ejemplo: Adición de un nodo a una paleta de sistema

Supongamos que su organización ha desarrollado un nuevo algoritmo para la minería de datos de audio y vídeo y que desea integrar ese algoritmo en IBM SPSS Modeler. Puede empezar definiendo un nodo de lectura de datos personalizada que leerá la entrada de archivos de vídeo y audio.

En principio ha decidido añadir su nuevo nodo de lector de datos a la paleta de sistema Orígenes. Todo lo que tiene que hacer es identificar la paleta Orígenes mediante el atributo `palette` del elemento `Node`. Para obtener más información, consulte el tema "Nodo" en la página 48.

Así, para añadir el nodo después del nodo Base de datos de la paleta Orígenes, utilizaría:

```
<Node id="lectorAV" type="dataReader" palette="import" relativePosition="after"
relativeTo="basedatos" label="Lector AV">
```

Ejemplo: Adición de una paleta personalizada

El uso de una paleta IBM SPSS Modeler estándar no está mal, pero desea dar a su nuevo nodo más relevancia. El usuario decide si define una paleta personalizada para ello, que colocará después de la paleta Favoritos pero antes de Orígenes. Primero deberá añadir una sección de interfaz de usuario (paletas) para definir la paleta personalizada de la siguiente forma:

```
<UserInterface >
     <Paletas>
          <Palette id="minería_AV" label="Minería AV" relativePosition="before"
      relativeTo="import" description="Minería de audio y vídeo"  />
     </Palettes>
</UserInterface>
```

El atributo `relativeTo` debe utilizar el identificador interno de la paleta Orígenes, que es `import`.

Puede alterar la definición de `Nodo` de la siguiente manera:

```
<Node id="lectorAV" type="dataReader" customPalette="minería_AV" label="Lector AV">
```

De esta forma la paleta **Minería AV** se ubica entre las paletas de sistema Favoritos y Orígenes.

Ejemplo: Adición de una subpaleta personalizada a una paleta personalizada

Según el ejemplo anterior, decide que prefiere que el nodo de lector de datos vaya en su propia subpaleta **Orígenes AV** de la paleta **Minería AV**. Para ello, deberá especificar la subpaleta añadiendo un segundo elemento `Palette` a la sección de interfaz de usuario (paletas):

```
<UserInterface >
     <Paletas>
          <Palette id="Minería_AV" label="Minería AV" description="Minería de audio y vídeo" />
          <Palette id="Minería_AV.sources" customPalette="minería_AV" label="Orígenes AV" />
     </Palettes>
<UserInterface >
```

Después podrá alterar el elemento `Nodo` para que se refiera al identificador de la subpaleta:

```
<Node id="lectorAV" type="dataReader" customPalette="minería_AV.sources" label="Lector AV">
```

Ahora, cuando el usuario pulse en la pestaña **Minería AV** verá dos subpaletas, una etiquetada como **Todos** y otra etiquetada **Orígenes AV**. El nodo de lector AV aparece en ambos.

Si añade otro nodo nuevo a otra subpaleta nueva **Minería AV**, el nuevo nodo aparece tanto en **Todos** como en la nueva subpaleta, pero no en la subpaleta **Orígenes AV**.

Ejemplo: Adición de un nodo a una subpaleta de sistema

Para procesar los datos de origen de audio y vídeo, ahora podrá definir un nodo generador de modelos. Puede decidir añadirlo a la paleta estándar Modelado, que tiene una serie de subpaletas estándar. Puede optar por añadirlo a la subpaleta Clasificación, ubicándolo justo antes del nodo Red neural, por lo cual especifique:

```
<Node id="modeladorAV" type="modelBuilder" palette="modeling.classification"
relativePosition="before" relativeTo="redneuronal" label="Modelador AV">
```

Tenga en cuenta que el nodo también se añade en la misma posición relativa en la subpaleta Todos de la paleta Modelado.

Ejemplo: Adición de una subpaleta personalizada a una paleta de sistema

Al volver a mirar el número de nodos generadores de modelos de la subpaleta Clasificación, se da cuenta de que los usuarios no pueden ver su nuevo nodo fácilmente. Una forma de hacer más visible su nodo es añadir su propia subpaleta a la paleta Modelado y colocar allí el nodo.

Primero deberá definir una subpaleta personalizada añadiendo una sección de interfaz de usuario (paletas) al archivo:

```
<UserInterface >
     <Paletas>
          <Palette id="modelado.av_modelado" systemPalette="modeling" label="Modelador AV"
             labelKey="modelado_av.LABEL" description="Contiene nodos de modelador relacionados
             con la minería de audiovisuales" descriptionKey="modelado_av.TOOLTIP"/>
     </Palettes>
</UserInterface>
```

Tenga en cuenta que debe especificar explícitamente `systemPalette` para identificar la paleta de sistema que está extendiendo.

A continuación, en la sección de interfaz de usuario principal del nodo, especifique lo que debe aparecer en esta subpaleta:

```
<Node id="mi.modeladorav" type="modelBuilder" customPalette="modelado.av_modelado"
   label="Modelador AV">
```

Las subpaletas personalizadas siempre se ubican tras las subpaletas de sistema.

Nota: Si desea añadir más nodos a la subpaleta Modelado AV, sus archivos de especificación **no** necesitarán una sección de interfaz de usuario (paletas) siempre y cuando la extensión Modelador AV se cargara antes.

Ocultación o eliminación de una subpaleta o paleta personalizada

Si ya no desea que se muestre una subpaleta o paleta personalizada, puede ocultarla o eliminarla mediante el gestor de paletas IBM SPSS Modeler.

Tenga en cuenta que la operación de ocultación se mantiene en las distintas sesiones de IBM SPSS Modeler, pero es reversible ya que se controla mediante una casilla de verificación. La operación de eliminación es irreversible en la misma sesión, pero al reiniciar IBM SPSS Modeler el elemento reaparecerá a no ser que lo elimine del archivo de especificación o elimine toda la extensión. Para obtener más información, consulte el tema “Desinstalación de las extensiones de CLEF” en la página 202.

Para ocultar o eliminar una paleta:

1. En el menú principal de IBM SPSS Modeler, seleccione:
   Herramientas > **Administrar paletas**
2. Seleccione una paleta en el campo Nombre de paleta, después:
   - para ocultar la paleta, cancele la casilla de selección ¿Mostrar? casilla de verificación
   - para eliminar la paleta, pulse en el botón de selección Eliminar.
3. Pulse en Aceptar.

Para ocultar o eliminar una subpaleta:

1. En el menú principal de IBM SPSS Modeler, seleccione:
   Herramientas > **Administrar paletas**
2. Seleccione una paleta en el campo Nombre de paleta.
3. Pulse en el botón Subpaletas.
4. Seleccione una subpaleta en el campo Nombre de subpaleta, después:
   - para ocultar la subpaleta, cancele la casilla de selección ¿Mostrar? casilla de verificación
   - para eliminar la subpaleta, pulse en el botón de selección Eliminar.
5. Pulse en Aceptar.

Sección de definición de objetos

Los elementos son las partes más visibles de una extensión. La sección definición de objeto constituye el resto del archivo de especificación de CLEF, y se utiliza para definir varios objetos en la extensión. Es posible definir los siguientes tipos de objeto:

- nodos
- objetos de resultado de modelo
- objetos de resultado de documento
- objetos de resultados interactivos

Nodos son los objetos que aparecen en una ruta. **Los objetos de resultados de modelo** se generan mediante nodos generadores de modelos y aparecen en la pestaña Modelos del panel del gestor en la ventana principal. De la misma forma, los **objetos de resultado de documentos** se generan mediante nodos generadores de documentos, y aparecen en la pestaña Resultados del mismo panel. **Los objetos de resultados interactivos** se generan mediante nodos generadores de modelos interactivos y aparecen en la pestaña Resultados del panel del gestor.

La sección de definición de objeto consiste en una o más de estas definiciones de objetos.

Los elementos que pueden definirse para los diferentes tipos de objeto se describen en las siguientes secciones. Algunos de estos elementos son comunes a todos los tipos de objetos, mientras que los otros son específicos de la definición de resultados de modelo o nodo. Los elementos que no son específicos del objeto se indican como tales en el texto.

- identificador de objeto
- generador de modelos
- generador de documentos
- propiedades
- contenedores
- interfaz de usuario
- ejecución
- modelo de datos de salida
- constructores

Identificador de objeto

El identificador de objeto indica el tipo de objeto, y será uno de los siguientes:

```
<Node .../>

<ModelOutput .../>
```

```
<DocumentOutput .../>

<InteractiveModelBuilder .../>
```

El identificador de objeto también ofrece información sobre la forma en que se expondrá el objeto mediante scripts. El atributo `scriptName` representa un nombre exclusivo para el objeto. Los scripts pueden utilizar este atributo para especificar un objeto concreto (por ejemplo, un nodo en una ruta o un resultado de la pestaña Resultados).

Nodo

Una definición de nodo describe un objeto que puede aparecer en una ruta.

Formato

```
<Node id="identificador" type="tipo_nodo" palette="paleta" customPalette="paleta_personalizada"
     relativePosition="posición" relativeTo="nodo" label="etiqueta_visualización"
     labelKey="clave_etiqueta"
     scriptName="nombre_script" helpLink="id_tema" description="descripción"
     descriptionKey="clave_descripción">
     <ModelBuilder ...>
      ...
     </ModelBuilder>
     <DocumentBuilder ...>
      ...
     </DocumentBuilder>
     <ModelProvider ... />
     <Properties>
      ...
     </Properties>
     <Containers>
      ...
     </Containers>
     <UserInterface >
      ...
     </UserInterface>
     <Execution>
      ...
     </Execution>
     <OutputDataModel ...>
      ...
     </OutputDataModel>
     <Constructors>
      ...
     </Constructors>
</Node>
```

Los elementos admitidos en la definición de nodo se describen en las secciones que comienzan en "Propiedades" en la página 51.

Tabla 13. Atributos de nodo.

Atributo	Descripción
`id`	(Obligatorio.) Identificador para este nodo en formato de cadena de texto.

Tabla 13. Atributos de nodo (continuación).

Atributo	Descripción
`type`	(Obligatorio.) Tipo de nodo: `dataReader`: nodo que lee datos, por ejemplo nodos de la paleta Orígenes. `dataWriter`: nodo que escribe datos, por ejemplo nodos de la paleta Exportar. `dataTransformer`: nodo que transforma datos, por ejemplo los nodos Operaciones con campos/registros `modelBuilder`: nodo generador de modelos, por ejemplo nodos de la paleta Modelado. `documentBuilder`: nodo que crea un gráfico o informe. `modelApplier`: nodo que contiene un modelo generado. El tipo de nodo determina la forma del icono de nodo en la paleta y el lienzo. Para obtener más información, consulte el tema "Conceptos básicos sobre nodos" en la página 9. Si el tipo de nodo es `modelBuilder`, la definición de nodo debe incluir un elemento `ModelBuilder`; consulte "Generador de modelos" en la página 51. Si el tipo de nodo es `documentBuilder`, la definición de nodo debe incluir un elemento `DocumentBuilder`; consulte "Generador de documentos" en la página 51.
`palette`	El identificador de una de las paletas o subpaletas estándar de IBM SPSS Modeler en la que aparecerá el nodo, principalmente: `import` - Orígenes `recordOp` - Operaciones con registros `fieldOp` - Operaciones con campos `graph` - Gráficos `modeling` - Modelado (consulte a continuación) `dbModeling` - Modelado de bases de datos `output` - Resultados `export` - Exportar La paleta Modelado tiene una serie de subpaletas estándar: `modeling.classification` - Clasificación `modeling.association` - Asociación `modeling.segmentation` - Segmentación `modeling.auto` - Automatizado Si omite el atributo `palette`, el nodo aparece en la paleta Operaciones con campos. Nota: el atributo `palette` sólo se utiliza para nodos generadores de modelos.
`customPalette`	El identificador de una paleta o subpaleta personalizada en la que debe aparecer un nodo. Es el valor del atributo `id` de un elemento `Palette`, que se especifica en la sección de interfaz de usuario (paletas) del archivo. Para obtener más información, consulte el tema "Sección de interfaz de usuario (Paletas)" en la página 43.

`relativePosition`	Especifica la posición del nodo en la paleta. Los valores posibles son:
	
	`primera`
	`last`
	`before`
	`after`
	
	Si el valor es `before` (antes) o `after` (después), el atributo `relativeTo` (con respecto a) también es obligatorio (véase abajo).
	
	Si se omite `relativePosition`, el nodo se ubicará el último en la paleta.

Tabla 13. Atributos de nodo (continuación).

Atributo	Descripción
`relativeTo`	Si el valor de `relativePosition` (posición relativa) es `before` (antes) o `after` (después), entonces `relativeTo` (con respecto a) se utilizará para especificar el nodo de la paleta al que precede o sigue este nodo. El valor de `relativeTo` es el nombre de script del nodo. En el caso de un nodo de IBM SPSS Modeler estándar, el nombre de script puede encontrarse en la sección "Referencia de propiedades" de la *IBM SPSS Modeler Guía de scripts y automatización*, aunque sin el sufijo `...node` (por ejemplo, para el nodo de base de datos utilizaría `database`, no `databasenode`). En el caso de un nodo CLEF, es el valor del atributo `scriptName` de ese nodo.
`label`	(Obligatorio.) Nombre de visualización del nodo tal y como aparece en la paleta, lienzos y cuadros de diálogo.
`labelKey`	Identifica la etiqueta con fines de localización.
`scriptName`	Se utiliza para identificar de forma exclusiva el nodo al que se hace referencia en un script. Para obtener más información, consulte el tema “Uso de nodos CLEF en scripts” en la página 76.
`helpLink`	Es posible mostrar un identificador opcional de un tema de ayuda cuando el usuario invoca el sistema de ayuda, en caso de que haya. El formato del identificador depende del tipo de sistema de ayuda (consulte Capítulo 7, “Adición del sistema de ayuda”, en la página 163) : HTML Help - URL del tema de ayuda JavaHelp - ID del tema
`description`	Una descripción del texto del nodo.
`descriptionKey`	Identifica la descripción con fines de localización.

Los elementos que pueden encontrarse en la definición de nodo se describen en las secciones que comienzan en “Generador de modelos” en la página 51.

Ejemplo

Para obtener un ejemplo de una definición de nodo, consulte “Archivo de especificación de ejemplo” en la página 32.

Resultado de modelo

Una definición de resultados de modelo describe un modelo generado; un objeto que aparecerá en la pestaña Modelos del panel de gestor tras la ejecución de una ruta.

Si desea obtener más información sobre la codificación de esta parte del archivo, consulte “Resultado de modelo” en la página 87.

Resultado de documento

Una definición de resultados de documento describe un objeto, como una tabla o un gráfico generado, que aparecerá en la pestaña Modelos del panel de gestor tras la ejecución de una ruta.

Si desea obtener más información sobre la codificación de esta parte del archivo, consulte “Resultado de documento” en la página 99.

Generador de modelos interactivos

Si desea obtener más información sobre la codificación de esta parte del archivo, consulte “Generación de modelos interactivos” en la página 88.

Generador de modelos

Este elemento se utiliza únicamente en las definiciones del elemento Node.

Si desea obtener más información sobre la codificación de esta parte del archivo, consulte Capítulo 5, “Generación de modelos y documentos”, en la página 79.

Generador de documentos

Este elemento se utiliza únicamente en las definiciones del elemento Node.

Si desea obtener más información sobre la codificación de esta parte del archivo, consulte Capítulo 5, “Generación de modelos y documentos”, en la página 79.

Proveedor de modelos

Este elemento se utiliza únicamente en las definiciones del elemento Node.

Cuando defina un objeto de resultado de modelo y un nodo aplicador de modelos puede utilizar el elemento `ModelProvider` para especificar el contenedor que va a contener el modelo. También puede especificar si el modelo se almacena en formato PMML. Los modelos PMML pueden visualizarse mediante un visor personalizado o mediante un visor de resultados de modelos de IBM SPSS Modeler estándar, que ofrece el elemento `ModelViewerPanel`. Para obtener más información, consulte el tema “Panel de visor de modelos” en la página 121.

Formato

```
<ModelProvider container="nombre_contenedor" isPMML="true_false" />
```

donde:

`container` es el nombre del contenedor en el que se encuentra el modelo.

`isPMML` indica si el modelo se almacena en formato PMML.

Ejemplo

```
<ModelProvider container="modelo" isPMML="true" />
```

Si desea consultar un ejemplo del uso de `ModelProvider` en el contexto de un nodo aplicador de modelos, consulte el ejemplo en “Panel de visor de modelos” en la página 121.

Propiedades

Una definición de propiedad consiste en un conjunto de pares de nombre-y-valor. Las definiciones de propiedades individuales, de las que puede haber varias, se almacenan en una única sección de propiedades.

Nota: Si se define una propiedad en la sección de propiedades, no habrá necesidad de definirla para un control de propiedad individual, ya que las definiciones de la sección de propiedades tienen preferencia. Por ello, se recomienda definir las propiedades en la sección Propiedades.

La única excepción a esta regla se refiere al atributo `label`. Si el atributo `label` se define para un control de propiedad, entonces *cualquier* definición de propiedad que se encuentre en esa declaración de control de propiedad (no sólo la definición de etiqueta `label`) tendrá prioridad sobre su definición correspondiente de la sección de propiedades. Tenga en cuenta que esta excepción sólo se aplica a los controles de propiedad, y no a otros tipos de control como los menús, elementos de menú y elementos de barra de herramientas. Estos deben definir una etiqueta de forma explícita, ya sea directamente (menús) o indirectamente a través de un elemento `Action` (elementos de menú y elementos de barra de menú).

Formato

```
<Properties>
     <Property name="nombre" scriptName="nombre_script" valueType="tipo_valor" isList="true_false"
          defaultValue="valor_predeterminado" label="etiqueta_visualización" labelKey="clave_etiqueta"
          description="descripción" descriptionKey="clave_descripción" />
          <Enumeration ... />
          <Structure ... />
          <DefaultValue ... />
     ...
</Properties>
```

Los elementos `Enumeration`, `Structure` y `DefaultValue` se utilizan en casos específicos. Para obtener más información, consulte el tema "Tipos de valor" en la página 60.

Los atributos del elemento `Property` se muestran en la siguiente tabla.

Tabla 14. Atributos de propiedad.

Atributo	Descripción
`name`	(obligatorio) Nombre exclusivo para la propiedad.
`scriptName`	Nombre por el que se hace referencia a la propiedad en un script. Para obtener más información, consulte el tema "Uso de nodos CLEF en scripts" en la página 76.

`valueType`	Especifica el tipo de valor que puede tomar esta propiedad, que puede ser:
	
	`string (cadena)`
	`encryptedString (cadena cifrada)`
	`fieldName (nombre de campo)`
	`integer (entero)`
	`double (doble)`
	`boolean (booleano)`
	`date (fecha)`
	`enum (enumeración)`
	`estructura`
	`databaseConnection (conexión con base de datos)`
	
	Para obtener más información, consulte el tema "Tipos de valor" en la página 60.

`isList`	Especifica si la propiedad es una lista de valores del tipo de valor especificado (`true`) o un único valor (`false`).
`defaultValue`	El valor predeterminado de esta propiedad. Se puede expresar mediante un atributo de valor sencillo o un elemento complejo, y debe ser coherente con los valores válidos especificados.
`label`	Nombre de visualización del valor de propiedad tal y como aparece en la interfaz de usuario.
`labelKey`	Identifica la etiqueta con fines de localización.
`description`	Una descripción de la propiedad.
`descriptionKey`	Identifica la descripción con fines de localización.

Las propiedades pueden declarar cómo se pueden determinar los valores válidos:

- En el caso de los valores numéricos, será por el valor mínimo o máximo.
- En el caso de las cadenas, suele ser una selección de campos (por ejemplo, todos los campos, todos los campos numéricos, todos los campos discretos, etc.) pero también puede ser una selección de archivos.
- En el caso de las enumeraciones, será el conjunto de valores válidos.

Las propiedades con clave también deben indicar cómo se determinan las claves válidas. Tenga en cuenta que el tipo de clave de una propiedad con clave debe ser una cadena o una enumeración. Para obtener más información, consulte el tema “Tipos de propiedad” en la página 37.

El valor predeterminado opcional asociado a la propiedad se evalúa cuando se crea el objeto asociado. Por ejemplo, los valores predeterminados de la propiedad de nodo se evalúan cada vez que se crea una nueva instancia del nodo, las propiedades de ejecución se evalúan cada vez que se ejecuta el nodo. La evaluación se realiza en el mismo orden en el que se declararon las propiedades.

Tenga en cuenta que una definición de propiedad puede hacer referencia a un tipo de propiedad declarado en la sección de objetos comunes.

Containers

Un contenedor es un marcador de un objeto de resultados cuya generación viene definida en la sección de constructores.

Formato

```
<Containers>
     <Container name="nombre_contenedor"/>
      ...
</Containers>
```

donde:

`name` corresponde al valor del atributo de destino de un elemento `CreateModel` o `CreateDocument` (consulte en “Uso de constructores” en la página 100) y asocia indirectamente el contenedor con uno de los tipos de contenedor declarados en la sección de objetos comunes.

Ejemplo

En primer lugar, los tipos de contenedor se declaran en la sección de objetos comunes. Hay un tipo de contenedor para los modelos, en formato de texto, y dos tipos de contenedor para los objetos de resultados de documentos, uno en el formato predeterminado (texto) para los resultados HTML y otro en formato binario para los resultados comprimidos.

```
<CommonObjects>
     <ContainerTypes>
          <ModelType id="mi_modelo" format="utf8" />
          <DocumentType id="resultado_html" />
          <DocumentType id="tipo resultado_zip" format="binary" />
     </ContainerTypes>
</CommonObjects>
```

En la sección de ejecución de la definición de nodo, los archivos de resultados se definen como archivos contenedores con tipos de contenedor que corresponden a los identificadores especificados en la sección de objetos comunes:

```
<Node id="minodo" ...>
 ...
     <Execution>
      ...
          <OutputFiles>
               <ContainerFile id="pmml" path="${tempfile}.pmml" containerType="mi_modelo" />
               <ContainerFile id="resultadoshtml" path="${tempfile}.html" containerType="resultados_
               html" />
               <ContainerFile id="resultadoszip" path="${tempfile}.zip" containerType="zip_
               outputType" />
          </OutputFiles>
```

Así, la sección de constructores define los objetos de resultados que se generarán cuando se ejecute el nodo. Aquí, los elementos CreateModel y CreateDocument tienen un atributo sourceFile correspondiente al archivo contenedor, como se especifica en la sección anterior Archivos de resultado:

```
          <Constructors>
               <CreateModelOutput type="miresultado">
                    <CreateModel target="modelo" sourceFile="pmml" />
                    <CreateDocument target="resultados_avanzados" sourceFile="resultadohtml" />
                    <CreateDocument target="resultado_zip" sourceFile="resultadozip" />
               </CreateModelOutput>
          </Constructors>
     </Execution>
</Node>
```

Por último, la sección de resultados de modelo asocia un contenedor con un objeto de resultados de documento o de modelo. En el elemento Container, el atributo name corresponde con el atributo target en los elementos CreateModel y CreateDocument que se acaban de especificar:

```
<ModelOutput id="miresultado" label="Mi modelo">
     <Containers>
          <Container name="modelo" />
          <Container name="resultados_avanzados" />
          <Container name="resultados_zip" />
     </Containers>
      ...
</ModelOutput>
```

Interfaz de usuario

El archivo de especificación admite una serie de componentes de interfaz de usuario que permiten que se muestren los objetos y se modifiquen los controles y propiedades. Se ofrecen recursos para especificar el diseño y comportamiento de cambio de tamaño del componente, y si éste se activará o hará visible si se modifican otros controles.

La sección de interfaz de usuario especifica el aspecto visible de un objeto. La especificación puede utilizarse para personalizar un componente de interfaz de usuario básica, como un cuadro de diálogo de propiedades de nodo o una ventana de resultados.

La sección de interfaz de usuario es una parte obligatoria de la especificación del elemento Node.

Si desea obtener más información sobre la codificación de esta parte del archivo, consulte Capítulo 6, “Generación de interfaces de usuario”, en la página 105.

Ejecución

Este elemento se utiliza únicamente en las definiciones del elemento Node.

El elemento Execution define las propiedades y archivos que se utilizan cuando se ejecuta un nodo.

Formato

```
<Execution>
     <Properties>
      ...
     </Properties>
     <InputFiles>
          <ContainerFile ... />
           ...
     </InputFiles>
     <OutputFiles>
          <ContainerFile ... />
```

```
        ...
    </OutputFiles>
    <Module ...>
        <StatusCodes ... />
    </Module>
    <Constructors ... />
</Execution>
```

La sección de ejecución incluye la definición de un conjunto de propiedades que se vuelven a crear cada vez que se ejecuta el nodo y que sólo están disponibles mientras se ejecuta el nodo.

La información de ejecución también puede definir el conjunto de archivos de entrada que se generarán antes de la ejecución del nodo y los archivos de resultados que se generarán durante la ejecución.

Es posible especificar cualquier número de archivos de entrada y resultados. Cada archivo de entrada está asociado con un contenedor definido por el nodo. Cada archivo de resultados suele utilizarse para crear contenedores para objetos generados. El formato de un archivo de entrada o resultados viene determinado por la declaración del contenedor de la sección de objetos comunes.

Ejemplo

Si desea conocer un ejemplo de una sección de ejecución, consulte "Archivo de especificación de ejemplo" en la página 32.

Propiedades (Runtime)

Esta sección define el conjunto de propiedades de tiempo de ejecución que sólo están disponibles mientras se ejecuta el nodo.

Formato

El formato es similar al de la sección `Properties` de la parte principal de la definición de elemento. Para obtener más información, consulte el tema "Propiedades" en la página 51.

Durante la ejecución de un nodo generador de modelos o de documentos, se crea un **archivo temporal de servidor** para almacenar el objeto de resultados de documento o de modelo. El servidor accede a este archivo y coloca el objeto en el cliente, donde se introduce en un contenedor. Deberá especificar este archivo aquí.

Ejemplo

Este ejemplo muestra cómo especificar el archivo temporal del servidor.

```
<Properties>
    <Property name="tempfile" valueType="string">
        <DefaultValue>
            <ServerTempFile basename="datatmp"/>
        </DefaultValue>
    </Property>
</Properties>
```

Archivos de entrada

Esta sección define el conjunto de archivos de entrada que se generarán antes de ejecutar el nodo. Los archivos de entrada de este contexto son archivos que se introducen en la ejecución del nodo en el servidor. Por ejemplo, un nodo aplicador de modelos tiene un contenedor de modelos que se transfiere al archivo de entrada especificado en la ejecución del nodo.

Formato

```
<InputFiles>
     <ContainerFile id="identificador" path="ruta" container="contenedor">
      ...
</InputFiles>
```

En el elemento `ContainerFile` de un archivo de entrada, los atributos son tal y como se muestran en la siguiente tabla.

Tabla 15. Atributos de archivo contenedor - archivos de entrada.

Atributo	Descripción
`id`	Identificador exclusivo para el archivo contenedor.
`path`	La ubicación en el servidor en el que desea que se genere el archivo de entrada (por ejemplo, la ubicación de un archivo temporal de servidor, consulte "Propiedades (Runtime)" en la página 55).
`contenedor`	Identificador del contenedor que contiene el objeto que se está enviando al servidor como entrada.

Ejemplo

```
<InputFiles>
     <ContainerFile id="pmml" path="${archivo temporal}.pmml" container="modelo"/>
</InputFiles>
```

Archivos de resultados

Esta sección especifica los archivos de resultados que se generan durante la ejecución del nodo en el servidor. Los archivos de resultados (por ejemplo, los resultados de la ejecución de un nodo generador de modelos o de documentos) se vuelven a transferir al cliente tras la ejecución.

Formato

```
<OutputFiles>
     <ContainerFile id="identificador" path="ruta" containerType="contenedor">
      ...
</OutputFiles>
```

En el elemento `ContainerFile`, los atributos son tal y como se muestran en la siguiente tabla.

Tabla 16. Atributos de archivo contenedor - archivos de resultados.

Atributo	Descripción
`id`	Identificador exclusivo para el archivo contenedor.
`path`	La ubicación en el servidor del objeto que va a transferirse al cliente (por ejemplo, la ubicación de un archivo temporal de servidor, consulte "Propiedades (Runtime)" en la página 55).
`containerType`	Identificador del tipo de contenedor del objeto (es decir, el ID del tipo de modelo o documento) que permite que el objeto se transfiera en el formato correcto. Para obtener más información, consulte el tema "Tipos de contenedor" en la página 39.

Ejemplo

```
<OutputFiles>
     <ContainerFile id="pmml" path="${archivo temporal}.pmml" containerType="minodo_modelo" />
     <ContainerFile id="htmloutput" path="${archivo temporal}.html" containerType="resultado_html" />
     <ContainerFile id="zipoutput" path="${archivo temporal}.zip" containerType="tipo resultado_zip" />
</OutputFiles>
```

Módulos

Esta sección especifica una biblioteca compartida en el lado del servidor que se utilizará durante la ejecución del nodo (por ejemplo, una DLL que se cargue en memoria).

Formato

```
<Module libraryId="identificador_biblioteca_compartida" name="nombre_nodo">
     <StatusCodes ... />
</Module>
```

donde:

`libraryId` es el identificador de una biblioteca declarada en un elemento Shared Library en la sección Resources. Para obtener más información, consulte el tema "Bibliotecas compartidas" en la página 36.

`name` se utiliza si la biblioteca la comparte más de un nodo, e identifica el nodo concreto que se ejecuta. Si sólo un nodo está utilizando la biblioteca, el nombre puede quedarse en blanco.

Ejemplo

```
<Module libraryId="minodo1" name="minodo">
     <StatusCodes>
          <StatusCode code="0" status="error" message="Se ha producido una excepción" />
          <StatusCode code="1" status="error" message="Error al leer los datos de entrada" />
           ...
     </StatusCodes>
</Module>
```

Códigos de estado

La mayoría de los programas realizan algún tipo de comprobación de errores y muestran los mensajes adecuados al usuario. Suelen devolver enteros para indicar una finalización correcta u otro estado. La API del servidor puede devolver un código de estado tras la ejecución de una ruta que contenga el nodo. Para obtener más información, consulte el tema "Documento de detalles de estado" en la página 193.

La sección de códigos de estado le permite asociar un mensaje con un código de estado concreto para mostrárselo al usuario.

Formato

```
<StatusCodes>
     <StatusCode code="númerocódigo" status="estado" message="mensaje_texto"
       messageKey="clave_mensaje" />
      ...
</StatusCodes>
```

Los atributos de código de estado se muestran en la siguiente tabla.

Tabla 17. Atributos de código de estado.

Atributo	Descripción
`código`	El código de estado (valor entero) al que se asociará el mensaje.
`status`	Clasificación de estado: `success`: el nodo se ha ejecutado correctamente. `warning`: el nodo de ha ejecutado con advertencias `error`: la ejecución del nodo se ha realizado sin éxito.
`message`	El mensaje se mostrará cuando se devuelva este código de estado.
`messageKey`	Identifica el mensaje con fines de localización.

Ejemplo

En este ejemplo, el texto del mensaje de error se incluye en el elemento `StatusCode`:

```
<StatusCodes>
     <StatusCode code="0" status="error" message="No se puede inicializar un homólogo" />
     <StatusCode code="1" status="error" message="Error al leer los datos de entrada" />
     <StatusCode code="2" status="error" message="Error interno" />
     <StatusCode code="3" status="error" message="El campo de entrada no existe" />
</StatusCodes>
```

Durante la ejecución, si la API del servidor devuelve el código de estado 3, se mostrará el mensaje siguiente al usuario.

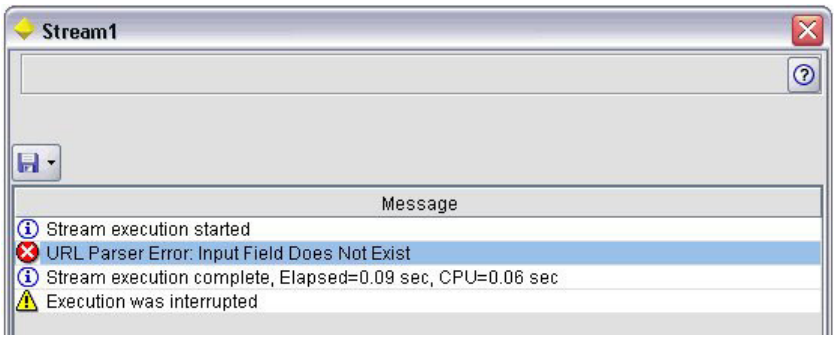

Figura 29. Visualización del mensaje de error

En el siguiente ejemplo, se hace referencia a los textos de mensaje de error mediante un atributo `messageKey`:

```
<StatusCodes>
     <StatusCode code="0" status="error" messageKey="initErrMsg.LABEL"/>
     <StatusCode code="1" status="error" messageKey="inputErrMsg.LABEL"/>
     <StatusCode code="2" status="error" messageKey="internalErrMsg.LABEL"/>
     <StatusCode code="3" status="error" messageKey="invalidMetadataErrMsg.LABEL"/>
      ...
</StatusCodes>
```

Un archivo de propiedades (como `messages.properties`), situado en la misma carpeta que el archivo de especificación, es el que contiene los mensajes de texto, así como otro texto que se visualizará:

```
 ...
initErrMsg.LABEL=Error de inicialización.
inputErrMsg.LABEL=Error al leer los datos de entrada.
internalErrMsg.LABEL=Error interno.
invalidMetadataErrMsg.LABEL=Metadatos (en campos de entrada/salida) no válidos.
 ...
```

Este método resulta útil cuando el texto que se muestra debe localizarse para mercados de diferentes idiomas, ya que el texto de localización se encuentra en un archivo. Para obtener más información, consulte el tema “Localización” en la página 167.

Modelo de datos de salida

Este elemento se utiliza únicamente en las definiciones del elemento Node.

La sección de modelo de datos de salida especifica la forma en que ciertas propiedades afectan al modelo de datos.

El modelo de datos de salida se puede determinar de una de las tres formas siguientes:

- Mediante las características de definición de conjunto de campos en el archivo de especificación. Para obtener más información, consulte el tema “Conjuntos de campos” en la página 69.
- Mediante una clase de Java en el cliente que implemente una interfaz de proveedor de modelo de datos que reciba un conjunto de propiedades y el modelo de datos de entrada y devuelva una instancia de modelo de datos.
- Mediante un componente de biblioteca compartida en el lado del servidor que reciba un conjunto de propiedades y el modelo de datos de entrada y devuelva un documento de metadatos.

La sección de modelo de datos de salida define la forma en que las propiedades de un nodo afectan a los campos que pasan por el nodo. El modelo de datos de salida puede:

- dejar el modelo de datos de entrada sin cambios
- modificar el modelo de datos de entrada
- reemplazar el modelo de datos de entrada por un modelo de datos distinto

Por ejemplo, un nodo Ordenar no afecta a las propiedades en sí aunque las reclasifica, un nodo Derivar modifica el modelo de datos añadiendo un nuevo campo y un nodo Agregar sustituye completamente el modelo de datos.

En el caso de que se modifique el modelo de datos de entrada, la definición puede añadir nuevos campos o modificar o eliminar los existentes. Cuando se reemplaza el modelo de datos, sólo es posible añadir nuevos campos. El archivo de especificación admite estas operaciones básicas (incluyendo la posibilidad de crear nuevos campos cuyo tipo se basa en un campo de entrada), así como la capacidad de iterar el conjunto de campos de entrada o una clave o propiedad de lista que represente un grupo de campos en el conjunto de campos de entrada.

Formato

El formato general de la sección de modelo de datos de salida es el que sigue, aunque puede consultar las secciones “Control de selector de campos múltiples” en la página 137 y “Control de selector de campo único” en la página 144 para conocer los formatos específicos de estos casos.

```
<OutputDataModel mode="modo" libraryId="nombre_contenedor">
     -- operaciones del modelo de datos --
</OutputDataModel>
```

Los atributos del modelo de datos de salida se muestran en la siguiente tabla.

Tabla 18. Atributos del modelo de datos de salida.

Atributo	Descripción
`mode`	Efecto en el modelo de datos: `extend`: añade un nuevo campo al modelo existente. `fixed`: no se modifica. `modify`: cambia los campos existentes (por ejemplo eliminar o cambiar el nombre) `replace`: sustituye el modelo existente
`libraryId`	Nombre del contenedor del servidor desde el que se obtiene el modelo de datos.

Las operaciones de modelo de datos son aquellas que añaden nuevos campos o modifican o eliminan campos existentes. Para obtener más información, consulte el tema “Operaciones del modelo de datos” en la página 64.

Ejemplo

Se incluye un elemento `OutputDataModel` en el ejemplo de un archivo de especificación. Para obtener más información, consulte el tema "Archivo de especificación de ejemplo" en la página 32.

Constructores

Los constructores definen los objetos que se producen como resultado de ejecutar un nodo en una ruta o de generar un objeto en la ruta.

Si desea obtener más información sobre la codificación de esta parte del archivo, consulte las secciones que comienzan en "Uso de constructores" en la página 100.

Características comunes

Algunas características pueden utilizarse en más de una sección del archivo de especificación, principalmente:

- tipos de valor
- cadenas evaluadas
- operaciones
- campos y metadatos de campos
- conjuntos de campos
- roles
- operadores lógicos
- condiciones

Tipos de valor

Las declaraciones de tipo de valor especifican el tipo de valor que puede tomar una especificación de tipo de propiedad, propiedad o columna.

Cadenas y cadenas cifradas

El formato `valueType="string"` especifica que el valor es una cadena de texto. Se utiliza una declaración `valueType="encryptedString"` para una propiedad relacionada con un campo cuyo contenido deba ocultarse a medida que los escribe el usuario, como un campo de contraseña.

Nombres de campos

Si un valor toma la forma de un nombre de campo, utilice el formato `valueType="fieldName"`.

Expresiones matemáticas, lógicas y de fecha

Si el valor es una expresión matemática (número entero o de doble precisión), lógica (verdadera o falsa) o de fecha, defina `valueType` como `integer` (entero), `double` (doble), `boolean` (booleano) o `date` (fecha) respectivamente.

Propiedades enumeradas

Las propiedades enumeradas se incluyen en una sección de enumeración que siga inmediatamente a una declaración `valueType="enum"`. Para obtener más información, consulte el tema "Propiedades enumeradas" en la página 61.

Declaraciones de estructura

Una declaración `valueType="structure"` especifica un valor compuesto que contiene otros atributos nombrados. Los atributos son similares a las propiedades, pero no pueden estructurarse ni incluir claves. Para obtener más información, consulte el tema "Propiedades estructuradas" en la página 62.

- **Indicador con clave.** Especifica si la propiedad es un valor único o una tabla hash donde cada valor de la tabla es del tipo de valor especificado.
- **Conjunto de valores.** Especifica cómo se determina el conjunto de valores disponibles.
- **Conjunto de claves.** En las propiedades con clave, se utiliza para especificar cómo se determina el conjunto de claves disponibles. Esta información también se utiliza para suministrar sugerencias a la interfaz de usuario sobre el tipo de controlador que se debe utilizar.

Conexiones a la base de datos

Se trata de cadenas de conexión que permiten a los usuarios iniciar sesión en una base de datos, por ejemplo `usuario1@bdprueba`. La información de inicio de sesión ya debe haberse definido para la base de datos. Para obtener más información, consulte el tema "Control de selector de conexión de base de datos" en la página 135.

Propiedades enumeradas

Una propiedad enumerada es una que puede tomar un valor de una lista predefinida de valores.

Formato

El formato de las propiedades enumeradas utiliza una sección `Enumeration` en la que la lista de valores se define como sigue:

```
<PropertyTypes>
     <PropertyType id="identificador" valueType="enum">
          <Enumeration>
               <Enum value="valor" label="etiqueta_visualización" labelKey="clave_etiqueta"
                    description="descripción" descriptionKey="clave_descripción" />
           ...
          </Enumeration>
     </PropertyType>
<PropertyTypes>
```

donde los atributos de `PropertyType`:

- `id` es un identificador exclusivo para el tipo de propiedad.
- `valueType` indica que el tipo de propiedad es enumerado.

y los atributos de `Enum` son:

- `value` (obligatorio) es el valor de propiedad que debe aparecer en la lista de valores.
- `label` (obligatorio) es el nombre de visualización del valor de propiedad tal y como aparece en la interfaz de usuario.
- `labelKey` identifica la etiqueta con fines de localización.
- `description` es la descripción que indica el valor enumerado.
- `descriptionKey` identifica la descripción con fines de localización.

Ejemplo

```
<PropertyTypes>
     <PropertyType id="enum_compartida1" valueType="enum">
          <Enumeration>
               <Enum value="value1" label="Value 5.1" labelKey="enum5.value1.LABEL" />
               <Enum value="value2" label="Value 5.2" labelKey="enum5.value2.LABEL" />
```

```
            <Enum value="value3" label="Value 5.3" labelKey="enum5.value3.LABEL" />
        </Enumeration>
    </PropertyType>
<PropertyTypes>
```

Propiedades estructuradas

Una propiedad estructurada es aquella que se utiliza en una estructura de rejilla como un control de tabla en un cuadro de diálogo.

Formato

El formato de las propiedades estructuradas utiliza una sección `Structure` en la que se define la estructura y consiste en una serie de elementos `Attribute` como sigue:

```
<PropertyTypes>
    <PropertyType id="identificador" valueType="structure" isList="true_false">
        <Structure>
            <Attribute name="ID_columna" valueType="tipo_valor" isList="true_false"
                label="etiqueta_columna" labelKey="clave_etiqueta" defaultValue="valor"
                description="descripción" descriptionKey="clave_descripción" />
         ...
        </Structure>
    </PropertyType>
</PropertyTypes>
```

donde los atributos del elemento `PropertyType` son:

- `id` es un identificador exclusivo para el tipo de propiedad.
- `valueType` indica que el tipo de propiedad es estructurado.
- `isList` especifica si la propiedad es una lista de valores del tipo de valor especificado (`true`) o un único valor (`false`).

y los atributos de elemento `Attribute` son:

- `name` (obligatorio) es el identificador de la columna.
- `valueType` especifica el tipo de valor que puede tomar el contenido de esta columna, y puede ser:

  `string (cadena)`

  `encryptedString (cadena cifrada)`

  `integer (entero)`

  `double (doble)`

  `boolean (booleano)`

  `date (fecha)`

  `enum (enumeración)`
- `isList` especifica si el atributo es una lista de valores del tipo de valor especificado (`true`) o un único valor (`false`). De esta forma, una propiedad con clave se puede asociar con un conjunto fijo de atributos conocido (por ejemplo, atributos booleanos que representan las diferentes operaciones de agregación que se realizan en un campo concreto) o una lista de valores (por ejemplo, asociando una lista de nombres de campo con otras nombres de campo).
- `label` (obligatorio) es el nombre de visualización de la columna tal y como aparece en la interfaz de usuario.
- `labelKey` identifica la etiqueta con fines de localización.
- `defaultValue` es un valor que debe aparecer en la columna cuando se muestra.
- `description` es la descripción de la columna.
- `descriptionKey` identifica la descripción con fines de localización.

Ejemplo: control de tabla

Si desea consultar un ejemplo de la forma en que se utilizan las propiedades estructuradas en una tabla de control, consulte “Control de tabla” en la página 147.

Ejemplos: tipos de propiedades con clave

El primero de estos ejemplos ilustra el uso de una propiedad con clave en el que cada valor asociado es una estructura que representa la operación de agregado que se aplicará a un campo a partir de un conjunto fijo de operaciones:

```
<PropertyType id="aggregateOps" isKeyed="true" valueType="structure">
  <Structure>
    <Attribute name="MIN" valueType="boolean" label="Min" />
    <Attribute name="MAX" valueType="boolean" label="Max" defaultValue="true"/>
    <Attribute name="SUM" valueType="boolean" label="Sum" defaultValue="false"/>
    <Attribute name="MEAN" valueType="boolean" label="Mean" defaultValue="false"/>
    <Attribute name="SDEV" valueType="boolean" label="SDev" defaultValue="false"/>
  </Structure>
</PropertyType>
```

Además, una propiedad que utilice el tipo de propiedad `aggregateOps` puede ser:

```
<Property name="aggregationSettings" scriptName="aggregation_settings" type="aggregateOps"/>
```

Aquí la propiedad está compuesta por varios valores, cada uno de los cuales con una clave diferente. Por ejemplo, la clave `name` es el nombre de un campo (MIN, MAX, etc.).

En el siguiente ejemplo de propiedad con clave, cada valor asociado es una estructura que contiene un atributo único. En este caso, el atributo es una lista de expresiones de precisión doble que representan multiplicadores que se aplicarán al campo:

```
<PropertyType id="multiplierOps" isKeyed="true" valueType="structure">
  <Structure>
    <Attribute name="multipliers" valueType="double" isList="true"/>
  </Structure>
</PropertyType>
```

Además, una propiedad que utilice el tipo de propiedad `multiplierOps` puede ser:

```
<Property name="multiplierSettings" scriptName="multiplier_settings" type="multiplierOps"/>
```

Valores predeterminados

El elemento `DefaultValue` se utiliza para especificar un archivo, un directorio temporal del servidor o ambos. Se crean para almacenar un objeto de resultado de modelo o de resultados de documento.

Formato

```
<DefaultValue>
     <ServerTempDir basename="nombre"/>
     <ServerTempFile basename="nombre"/>
</DefaultValue>
```

donde `basename` (obligatorio) es el nombre del archivo o directorio temporal.

Ejemplo

```
<DefaultValue>
     <ServerTempFile basename="datatmp"/>
</DefaultValue>
```

Cadenas evaluadas

Algunas de las cadenas declaradas en el archivo de especificación pueden incluir referencias a los nombres de propiedad. Estas cadenas se denominan cadenas evaluadas.

La sintaxis de una referencia de propiedad es:

`"${nombre_propiedad}"`

Cuando se accede a una cadena evaluada, cualquier referencia de propiedad se sustituirá con el valor de la propiedad a la que se hace referencia. Si la propiedad no existe, se producirá un error. Por ejemplo, cuando se añade un nuevo campo, es posible que tenga una propiedad denominada `mi_nuevo_campo` en la definición de nodo y un control en la sección de interfaz de usuario que permitirá al usuario editar el valor de esta propiedad.

Ejemplo

```
<AddField name="${mi_nuevo_campo}" ...>
```

Operaciones

Algunas secciones del archivo de especificación admiten operaciones como la adición de campos, la creación de componentes y la inicialización de propiedades. Las secciones que admiten operaciones son:

- modelo de datos de salida (nodos de proceso y origen)
- modelo de datos de entrada y salida (componentes)
- creación de objeto de resultados (nodo generador de documentos y nodo generador de modelos)
- creación de aplicador de modelos (salidas de modelo)

Las operaciones se dividen en los siguientes tipos:

- operaciones del modelo de datos: `AddField`, `ChangeField`, `RemoveField`
- iteración: `ForEach`

Operaciones del modelo de datos

Las operaciones que puede realizar en el modelo de datos son:

- añadir un nuevo campo al modelo de datos existente
- modificar un campo existente a un modelo de datos
- eliminar un campo de un modelo de datos

Adición de un campo: El elemento `AddField` le permite añadir un nuevo campo a un modelo de datos existente.

Formato

```
<AddField prefix="prefijo" name="nombre" direction="rol_campo" directionRef="ref_rol_campo"
     fieldRef="referencia_campo" group="id_grupo" label="etiqueta" missingValuesRef="ref_valor_perdido"
     storage="tipo_almacenamiento" storageRef="ref_almacenamiento" targetField="target_field"
     type="tipo_datos" typeRef="ref_tipo" role="rol" tag="tipo_propensión" value="valor" >
     <Range min="valor_mín" max="valor_máx" />
</AddField>
```

Los atributos de `AddField` son los siguientes.

Tabla 19. Atributos de AddField.

Atributo	Descripción
`prefix`	Prefijo que se añade a un nombre de campo, por ejemplo para denotar un campo de salida de modelo.

Tabla 19. Atributos de AddField (continuación).

Atributo	Descripción
`name`	(obligatorio) El nombre del campo que se va a añadir. Puede ser una cadena codificada (por ejemplo, `field8`) o una cadena evaluada (por ejemplo, `${target}`) que hace referencia a un campo. Para obtener más información, consulte el tema "Cadenas evaluadas" en la página 64.
`dirección`	El rol del campo, es decir, si el campo es una entrada o un destino. Puede ser `in` (entrada), `out` (salida), `both` (ambos), `partition` (partición) o bien `none` (ninguno).
`directionRef`	Especifica que la dirección del campo se deriva de la dirección del campo que identifica una cadena evaluada (por ejemplo, `${field1}`) que hace referencia a un campo. Para obtener más información, consulte el tema "Cadenas evaluadas" en la página 64.
`fieldRef`	Especifica que todos los valores referenciados (`directionRef`, `missingValuesRef`, `storageRef` y `typeRef`) se derivan de los valores correspondientes del campo identificado por una cadena evaluada (por ejemplo `${field1}`) que hace referencia a un campo. Para obtener más información, consulte el tema "Cadenas evaluadas" en la página 64.
`group`	Especifica que el campo es un miembro de un grupo de campos. Para obtener más información, consulte el tema "Campos de modelo" en la página 85.
`label`	Una etiqueta del campo que se añadirá.
`missingValuesRef`	Especifica que la forma en que se gestionan los valores que faltan se deriva de la especificación de los valores perdidos del campo que identifica una cadena evaluada (por ejemplo, `${field1}`) que hace referencia a un campo. Para obtener más información, consulte el tema "Cadenas evaluadas" en la página 64.
`storage`	El tipo de almacenamiento de datos para el valor de campo, que puede ser `integer` (entero), `real` (real), `string` (cadena), `date` (fecha), `time` (hora), `timestamp` (marca de tiempo) o `unknown` (desconocido).
`storageRef`	Especifica que el tipo de almacenamiento se deriva del tipo de almacenamiento que identifica una cadena evaluada (por ejemplo, `${field1}`) que hace referencia a un campo. Para obtener más información, consulte el tema "Cadenas evaluadas" en la página 64.
`targetField`	En un campo de resultado de un modelo, especifica el campo objetivo del que derivan los datos de este nuevo campo. Puede ser una cadena codificada (por ejemplo, `field8`) o una cadena evaluada (por ejemplo, `${target}`) que hace referencia a un campo. Para obtener más información, consulte el tema "Cadenas evaluadas" en la página 64.
`type`	Tipo de datos del campo, que puede ser `auto` (automático), `range` (rango), `discrete` (discreto), `set` (conjunto), `orderedSet` (conjunto ordenado), `flag` (marca) o `typeless` (sin tipo).

Tabla 19. Atributos de AddField (continuación).

Atributo	Descripción
typeRef	Especifica que el tipo de datos se deriva del tipo de datos que identifica una cadena evaluada (por ejemplo, ${field1}) que hace referencia a un campo. Para obtener más información, consulte el tema "Cadenas evaluadas" en la página 64.
rol	El tipo de datos incluido en un campo de salida de modelos, que puede ser unknown (desconocido), predictedValue (valor predicho), predictedDisplayValue (valor de visualización predicho), probability (probabilidad), residual (residual), standardError (error estándar), entityId (id de entidad), entityAffinity (afinidad de entidad), upperConfidenceLimit (límite superior confianza), lowerConfidenceLimit (límite inferior confianza), propensity (propensión), value (valor) o supplementary (adicional). Para obtener más información, consulte el tema "Roles" en la página 71.
tag	Sólo se utiliza si role tiene el valor propensity; indica el tipo de propensión y puede ser o bien RAW (bruto) o ADJUSTED (ajustado).
value	Especifica que los valores que el nuevo campo contendrá derivan del campo que identifica una cadena evaluada (por ejemplo, ${field1}) que hace referencia a un campo. Para obtener más información, consulte el tema "Cadenas evaluadas" en la página 64.

Los atributos de Rango son los siguientes.

Tabla 20. Atributos de rango

Atributo	Descripción
min	Valor mínimo que puede aceptar el campo.
máx	Valor máximo que puede aceptar el campo.

Ejemplos

Lo siguiente añade un campo de cadena denominado campo8:

```
<AddField name="campo8" storage="string" />
```

El siguiente ejemplo le muestra cómo utilizar una referencia a un nombre de propiedad cuando se añade un campo. Aquí el campo se añade con un nombre que coincide con el valor de la propiedad definida previamente prop1:

```
<AddField name="${prop1}" ... />
```

En el siguiente ejemplo, si el campo de destino se denomina campo1, el modelo crea un campo de salida denominado $S-campo1 que contendrá el valor predicho para campo1:

```
<AddField prefix="$S" name="${target}" role="predictedValue" targetField="${target}"/>
```

El siguiente ejemplo añade un campo de salida de modelos que contiene una puntuación de probabilidad de entre 0,0 y 1,0:

```
<AddField prefix="$SC" name="${target}" storage="real" role="probability" targetField="${target}">
     <Range min="0.0" max="1.0"/>
</AddField>
```

En el ejemplo final, para cada campo de resultado de modelo, se añade un campo de salida que contiene una puntuación de probabilidad entre 0.0 y 1.0 y que deriva su valor del de la variable `fieldValue`:

```
<ForEach var="fieldValue" inFieldValues="${field}">
     <AddField prefix="$SP" name="${fieldValue}" storage="real" role="probability" targetField=
     "${field}" value="${fieldValue}">
          <Range min="0.0" max="1.0"/>
     </AddField>
</ForEach>
```

Para obtener más información, consulte el tema “Cadenas evaluadas” en la página 64.

Cambio de campo: El elemento `ChangeField` le permite modificar un campo existente en un modelo de datos.

Formato

```
<ChangeField name="nombre" fieldRef="referencia_campo" direction="rol_campo" storage="tipo_
almacenamiento" type="tipo_datos" >
     <Range min="valor_mín" max="valor_máx" />
</ChangeField>
```

Los atributos de `ChangeField` son los siguientes.

Tabla 21. Atributos de ChangeField.

Atributo	Descripción
`name`	(obligatorio) El nombre del campo que se va a modificar.
`fieldRef`	Un valor de referencia para el campo.
`dirección`	El rol del campo, es decir, si el campo es una entrada o un destino. Puede ser `in` (entrada), `out` (salida), `both` (ambos), `partition` (partición) o bien `none` (ninguno).
`storage`	El tipo de almacenamiento de datos para el valor de campo, que puede ser `integer` (entero), `real` (real), `string` (cadena), `date` (fecha), `time` (hora), `timestamp` (marca de tiempo) o `unknown` (desconocido).
`type`	Tipo de datos del campo, que puede ser `auto` (automático), `range` (rango), `discrete` (discreto), `set` (conjunto), `orderedSet` (conjunto ordenado), `flag` (marca) o `typeless` (sin tipo).

Los atributos de `Rango` son los siguientes.

Tabla 22. Atributos de rango

Atributo	Descripción
`min`	Valor mínimo que puede aceptar el campo.
`máx`	Valor máximo que puede aceptar el campo.

Eliminación de campo: El elemento RemoveField le permite eliminar un campo de un modelo de datos.

Formato

```
<RemoveField fieldRef="referencia_campo" />
```

donde `fieldRef` es un valor de referencia para el campo.

Iteración con el elemento ForEach

En algunos lugares resulta útil poder realizar la misma operación repetidamente para procesar cada uno de un conjunto de valores. El archivo de especificación admite un iterador `ForEach` sencillo que une una propiedad temporal a cada valor del conjunto suministrado. El bucle `ForEach` puede definirse para que realice la iteración de una de las formas siguientes:

- entre dos valores enteros con un tamaño de incremento opcional
- sobre los valores de una propiedad de lista
- sobre los valores de una propiedad con clave
- sobre los valores de un grupo de campos

Formato

```
<ForEach var="nombre_campo" from="exp_entero" to="exp_entero" step="exp_entero"
inFields="campos" inFieldValues="nombre_campo" inProperty="nombre_propiedad" >
   -- operación del modelo de datos --
</ForEach>
```

donde:

`var` (obligatorio) especifica el campo que contiene los valores a los que se aplicará la iteración.

`from` y `to` especifican los números enteros (o expresiones que evalúan enteros) que denotan los límites inferior y superior de la iteración, con el atributo opcional `step` que indica un tamaño de paso de entero.

`inFields`, `inFieldValues` e `inProperty` son alternativas al formato `from/to/step`:

- `inFields` especifica un conjunto de campos sobre el que realizar la iteración y es uno de los siguientes:
  `inputs`: los campos de entrada del nodo.
  `outputs`: campos de entrada desde el nodo
  `modelInput`: los campos de entrada especificados en la firma del modelo.
  `modelOutput`: los campos de salida especificados en la firma del modelo.
- `inFieldValues` especifica un nombre de campo (o una propiedad que representa un nombre de campo) e itera los valores de los metadatos de ese campo.
- `inProperty` especifica el nombre de una propiedad en la que realizar la iteración.

La operación del modelo de datos que se puede especificar en un elemento `ForEach` es cualquiera de los elementos `AddField`, `ChangeField` o `RemoveField`. Para obtener más información, consulte el tema "Operaciones del modelo de datos" en la página 64. Los alimentos `ForEach` también pueden anidarse.

Ejemplos

Lo siguiente realiza una operación diez veces:

```
<ForEach var="val" from="1" to="10"> ...
</ForEach>
```

Lo siguiente realiza una operación el número de veces especificado por una propiedad de entero:

```
<ForEach var="val" from="1" to="${history_count}"> ...
</ForEach>
```

En el siguiente ejemplo, el procesamiento se itera a través de los valores de los campos de salida del nodo:

```
<ForEach var="field" inFields="outputs">
 ...
</ForEach>
```

En el siguiente ejemplo se itera a través de los valores de los metadatos del campo identificado mediante `${field}`:

```
<ForEach var="fieldValue" inFieldValues="${field}"> ...
</ForEach>
```

El siguiente ejemplo itera los valores de una propiedad de lista:

```
<ForEach var="val" inProperty="mi_propiedad_lista"> ...
</ForEach>
```

El siguiente ejemplo itera los valores clave de una propiedad con clave:

```
<ForEach var="key" inProperty="mi_propiedad_clave"> ...
</ForEach>
```

Campos y metadatos de campos

Los nodos, modelos y orígenes de datos actúan como **proveedores de modelos de datos**, es decir, pueden definir los metados de campos a los que se puede acceder desde otros objetos.

Los proveedores de modelos de datos tienen un modelo de datos de entrada y un modelo de datos de salida. Un modelo de datos de salida se puede definir según los términos del modelo de datos de entrada, por ejemplo cuando se extiende un modelo de entrada mediante la adición de un campo o cuando se cambia un modelo existente.

Cada uno de estos objetos tiene requisitos ligeramente distintos.

Nodos. Se puede hacer referencia al modelo de datos de entrada, pero no es posible modificarlo. El modelo de datos de salida puede basarse en el modelo de datos de entrada o bien reemplazarlo. El modelo de datos de salida vuelve a calcularse siempre que las propiedades de nodo o el modelo de datos de entrada cambien. El modelo de datos de salida de un nodo aplicador de modelos también puede hacer referencia al modelo de datos de salida del componente de modelo.

Modelos. De forma predeterminada, los modelos de datos de entrada y salida (la firma de modelo) se basan en los ajustes de campo de entrada y salida que se emplearon al crear el modelo. En principio, el proceso de generación de modelo devolverá un archivo de metadatos que define los campos de entrada obligatorios y los campos de salida generados. Una vez definida, la firma de modelo no puede modificarse. Sin embargo, las propiedades de un nodo aplicador de modelos pueden modificar la salida de modelo de datos desde el nodo aplicador. Por ejemplo, estas propiedades pueden definir si se devuelve un ID clúster como una cadena o entero, o cuántos ID de secuencia generar. Además, la firma de modelo suele especificar salidas que tienen rol de campo (`dirección`) establecido como salida `"out"` mientras que el nodo podrá generarlos en el rol de campo como entrada `"in"`.

Orígenes de datos. Los orígenes de datos que se emplean en nodos de lector de datos pueden especificar un modelo de datos de salida. El modelo de datos de entrada siempre está vacío.

Conjuntos de campos

Es posible utilizar un conjunto de campos en numerosas ubicaciones para seleccionar un subconjunto de campos de un proveedor de modelos de datos. El proveedor del modelo de datos puede ser el objeto que lo encierra o un contenedor del objeto que lo encierra. El estado inicial de un filtro de campo puede ser incluir todos los campos disponibles y después excluir los tipos de campos específicos o empezar con un conjunto de campos vacío e incluir los campos obligatorios o añadir nuevos.

El siguiente ejemplo muestra la forma en que un nodo de extensión puede especificar el modelo de datos de salida. Los campos clave se especifican en una propiedad de lista denominada `keys`, que sigue con un campo opcional de recuento de registros que puede generarse y cuyo nombre también se especifica mediante una propiedad.

```
<OutputDataModel mode="replace">
     <ForEach var="field" inProperty="keys">
          <AddField name="${campo}" fieldRef="${campo}"/>
     </ForEach>
     <AddField name="${nombre_recuento_registros}" storage="integer">
          <Condition property="incluir_recuento_registros" op="equals" value="true"/>
     </AddField>
</OutputDataModel>
```

Generación de modelos y conjuntos de datos

El siguiente ejemplo muestra cómo un aplicador de modelos puede utilizar la información de un componente de modelo creado previamente para generar sus campos de salida:

```
<OutputDataModel mode="modify">
     <AddField provider="model" dataModel="output">
</OutputDataModel>
```

Tanto `AddField` como `ForEach` especifican un proveedor del modelo de datos, así como cuál de los modelos de entrada o salida de datos debe utilizarse. Ofrecen un mecanismo para especificar un conjunto (o subconjunto) de campos del proveedor de modelos de datos. El proveedor predeterminado es `this`, que representa el elemento que lo encierra (en vez del objeto que se crea), con el conjunto de campos de entrada que se utiliza de forma predeterminada. Si no se especifica ningún conjunto de campos, se utilizarán todos los campos disponibles.

Los conjuntos de campos se pueden basar en almacenamiento, tipo, rol del campo o nombres. Cuando se basan en nombres, se requiere una referencia a una propiedad de lista. El conjunto de campos puede estar lleno (valor predeterminado) o vacío; el primero permite que se excluyan los campos, mientras que el último permite que se incluyan los campos. Es posible especificar múltiples valores para cada uno de los filtros individuales, y estos valores actuarán como operadores "intersection" o "and", por ejemplo:

```
<FieldSet include="none">
     <Include direction="in" storage="string"/>
</FieldSet>
```

Aquí se comienza con un conjunto de campos vacío (especificado por `include="none"`) e incluirá campos que tengan el rol de campo (`dirección` como entrada `"in"` y el almacenamiento de cadena (string).

Otro ejemplo sería:

```
<FieldSet include="all">
     <Exclude type="typeless"/>
</FieldSet>
```

Aquí se incluyen todos los campos disponibles (especificados por el atributo `include="all"`, que es el comportamiento predeterminado) y excluye cualquiera que tenga como tipo `typeless` (sin tipo). De esta forma se incluyen campos con la `dirección` establecida como entrada `"in"` o ambas `"both"`.

Es posible especificar múltiples filtros que actuarán como operadores "union" u "or", por ejemplo:

```
<FieldSet include="all">
     <Exclude type="discrete" storage="real"/>
     <Exclude type="discrete" storage="integer"/>
</FieldSet>
```

Esto excluye campos que son discretos o almacenamiento real o discreto con almacenamiento entero.

Tenga en cuenta que cuando se incluyen campos en un conjunto de campos inicialmente vacío, el orden de las instrucciones `include` no suele afectar el orden en que se incluyen los campos. Es decir, los campos del proveedor de conjunto de campos se evalúan en su orden natural frente a cada condición para determinar si deben incluirse en el conjunto de campos.

Roles

Los roles describen el tipo de datos que se tienen en un campo de salida de modelo de datos. Cada rol puede especificarse mediante un elemento `AddField` y se comprueba mediante un elemento `Condition`.

Los roles posibles son los siguientes.

Tabla 23. Roles de salida de modelo

Rol	Significado
`unknown`	No se ha especificado ningún rol (no se encuentra).
`predictedValue`	Este campo contiene el valor predicho del campo de destino.
`probabilidad`	La probabilidad o confianza de la predicción.
`residual`	El valor residual.
`standardError`	Error estándar de la predicción.
`entityId`	El ID de entidad, normalmente representa el ID de clúster en un modelo de clúster.
`entityAffinity`	La afinidad de entidad, normalmente representa la distancia desde el centro del clúster en un modelo.
`upperConfidenceLimit`	El límite superior de confianza de la predicción.
`lowerConfidenceLimit`	El límite inferior de confianza de la predicción.
`propensity`	La puntuación de propensión. Un atributo adicional `tag` especifica si se refiere a una propensión en bruto o ajustada.
`value`	Se utiliza para representar un valor para modelos que está asociado a otra salida (véase a continuación).
`supplementary`	Información generada por el modelo que no está cubierta por otros roles.

Como ejemplo de `value`, el modelo Detección de anomalías genera grupos de campos en los que cada grupo se compone de dos campos, uno de los cuales representa un nombre de campo y el otro especifica una medida de lo anómalo que es el campo. En este caso, `value` puede ser el nombre de campo.

Operadores lógicos

Una serie de elementos puede utilizar los operadores lógicos `And`, `Or` y `Not` para especificar varios tipos de procesamiento, por ejemplo al establecer condiciones compuestas (consulte "Condiciones compuestas" en la página 76).

Formato

El formato del elemento `And` es el siguiente. El formato de los elementos `Or` y `Not` es casi idéntico, siendo las únicas diferencias las etiquetas de inicio y cierre `<Or>...</Or>` y `<Not>...</Not>`, respectivamente.

```
<And>
     <Condition .../>
     <And ... />
     <Or ... />
     <Not ... />
</And>
```

El elemento `Condition` especifica una condición que va a comprobarse. Para obtener más información, consulte el tema "Condiciones" en la página 72.

Tenga en cuenta que es posible anidar los elementos hijo `And`, `Or` y `Not`

Condiciones

El comportamiento de ciertos objetos puede modificarse mediante condiciones, que se especifican mediante elementos `Condition` (el equivalente a las instrucciones IF). Por ejemplo, un nodo que se ejecute mediante un comando puede añadir una condición a la información de ejecución, de forma que sólo se incluya una opción particular si una propiedad tiene un valor específico. De la misma forma, es posible activar o hacer visible un control de propiedad de la interfaz de usuario si otro control tiene un valor específico.

Las condiciones pueden ser simples o compuestas. Una **condición simple** consiste en lo siguiente:

- un origen de valor (ya sea una propiedad o un control)
- una prueba
- un valor de prueba opcional

Una **condición compuesta** permite que otras condiciones se combinen para formar condiciones lógicas complejas. Las condiciones compuestas implican el uso de:

- `And`
- `Or`
- `Not`

Formato

```
<Condition container="nombre_contenedor" control="nombre_prop" property="nombre" op="operador"
     value="valor" />
```

donde:

`container` especifica el nombre del contenedor particular cuyo valor va a comprobar la condición.

`control` especifica el control de propiedad cuyo valor va a comprobar la condición. *nombre_prop* es el valor del atributo `property` del elemento en el que se define el control (por ejemplo, en un panel de propiedades de una pestaña de cuadro de diálogo).

`property` especifica la propiedad cuyo valor va a comprobar la condición. *name* es el valor del atributo `name` del elemento `Property` en el que se define la propiedad.

`op` es el operador de condición. Para obtener más información, consulte el tema “Operadores de condición” en la página 73.

`value` es el valor específico que la condición va a probar.

Ejemplos

Si desea ver ejemplos de definición de condiciones, consulte “Condiciones simples” en la página 75 y “Condiciones compuestas” en la página 76.

Operadores de condición

Hay disponible un conjunto de operadores dirigidos a la mayoría de las condiciones.

Tabla 24. Pruebas admitidas en cualquier valor

Operador	Valor	Descripción
`equals`	*valor*	Es verdadero si la propiedad iguala al valor suministrado (distingue mayúsculas de minúsculas en valores de cadena).
`notEquals`	*valor*	Es verdadero si la propiedad no iguala al valor suministrado (distingue mayúsculas de minúsculas en valores de cadena).
`in`	*lista de valores*	Es verdadero si la propiedad se encuentra en la lista de valores suministrada.

Tabla 25. Pruebas admitidas en valores numéricos

Operador	Valor	Descripción
`lessThan`	*número*	Es verdadero si la propiedad es inferior al número suministrado.
`lessOrEquals`	*número*	Es verdadero si la propiedad es inferior o igual al número suministrado.
`greaterThan`	*número*	Es verdadero si la propiedad es superior al número suministrado.
`greaterOrEquals`	*número*	Es verdadero si la propiedad es superior o igual al número suministrado.

Tabla 26. Pruebas admitidas en valores de cadena

Operador	Valor	Descripción
`isEmpty`	-	Es verdadero si la propiedad tiene una cadena de longitud cero.
`isNotEmpty`	-	Es verdadero si la propiedad tiene una cadena de una longitud distinta de cero.
`startsWith`	*cadena*	Es verdadero si la propiedad comienza con la cadena suministrada (distingue mayúsculas de minúsculas).
`startsWithIgnoreCase`	*cadena*	Es verdadero si la propiedad comienza con la cadena suministrada, e ignora si se trata de mayúsculas de minúsculas.
`endsWith`	*cadena*	Es verdadero si la propiedad termina con la cadena suministrada (no distingue mayúsculas de minúsculas).
`endsWithIgnoreCase`	*cadena*	Es verdadero si la propiedad termina con la cadena suministrada, e ignora si se trata de mayúsculas de minúsculas.
`equalsIgnoreCase`	*cadena*	Es verdadero si la propiedad es igual a la cadena suministrada, e ignora si se trata de mayúsculas de minúsculas.
`hasSubstring`	*cadena*	Es verdadero si la propiedad contiene la cadena suministrada (distingue mayúsculas de minúsculas).
`hasSubstringIgnoreCase`	*cadena*	Es verdadero si la propiedad contiene la cadena suministrada, e ignora si se trata de mayúsculas de minúsculas.

Tabla 26. Pruebas admitidas en valores de cadena (continuación)

Operador	Valor	Descripción
isSubstring	*cadena*	Es verdadero si la propiedad es una subcadena de la cadena suministrada (distingue mayúsculas de minúsculas).
isSubstringIgnoreCase	*cadena*	Es verdadero si la propiedad es una subcadena de la cadena suministrada, e ignora si se trata de mayúsculas de minúsculas.

Tabla 27. Pruebas admitidas en propiedades de lista

Operador	Valor	Descripción
isEmpty	-	Es verdadero si el número de elementos de la lista es 0.
isNotEmpty	-	Es verdadero si el número de elementos de la lista no es 0.
countEquals	*número*	Es verdadero si el número de elementos de la lista iguala al valor suministrado.
countLessThan	*número*	Es verdadero si el número de elementos de la lista es inferior al valor suministrado.
countLessOrEquals	*número*	Es verdadero si el número de elementos de la lista es inferior o igual al valor suministrado.
countGreaterThan	*número*	Es verdadero si el número de elementos de la lista es superior al número suministrado.
countGreaterOrEquals	*número*	Es verdadero si el número de elementos de la lista es superior o igual al número suministrado.
contains	*valor*	Es verdadero si el elemento suministrado se encuentra en la lista.

Tabla 28. Pruebas admitidas en propiedades de campo

Operador	Descripción
storageEquals	Es verdadero si el almacenamiento iguala el valor suministrado.
typeEquals	Es verdadero si el tipo iguala el valor suministrado.
directionEquals	Es verdadero si el rol de campo (dirección) iguala el valor suministrado.
isMeasureDiscrete	Es verdadero si el tipo de datos de campo es discrete (discreto), es decir, sólo puede ser set (conjunto), flag (marca) o orderedSet (conjunto ordenado).
isMeasureContinuous	Es verdadero si el tipo de datos de campo es range (rango).
isMeasureTypeless	Es verdadero si el tipo de datos de campo es typeless (sin tipo).
isMeasureUnknown	Es verdadero si el tipo de datos de campo es unknown (desconocido).
isStorageString	Es verdadero si el tipo de almacenamiento de campo es string (cadena).
isStorageNumeric	Es verdadero si el tipo de almacenamiento de campo es numeric (numérico).
isStorageDatetime	Es verdadero si el tipo de almacenamiento de campo es datetime (fecha y hora).

Tabla 28. Pruebas admitidas en propiedades de campo (continuación)

Operador	Descripción
isStorageUnknown	Es verdadero si el tipo de almacenamiento de campo es unknown (desconocido).
isModelOutput	Es verdadero si el campo es un campo de salida de modelo.
modelOutputRoleEquals	Es verdadero si el rol del campo es uno de los roles válidos que aparecen en la siguiente tabla.
modelOutputTargetFieldEquals	Es verdadero si el campo de destino iguala el valor especificado (cadena).
modelOutputHasValue	Es verdadero si el campo es un campo de salida de modelo con un valor asociado.
modelOutputTagEquals	Es verdadero si la etiqueta iguala el valor especificado (cadena).

Los operadores de condición compatibles con propiedades con clave son:

- isEmpty
- isNotEmpty
- countEquals
- countLessThan
- countLessOrEquals
- countGreaterThan
- countGreaterOrEquals
- contains

Condiciones simples

Una condición simple consiste en el origen del valor inicial que va a comprobarse (ya sea un nombre de controlador o propiedad o bien una expresión evaluada), la prueba que va a realizarse y opcionalmente un valor con el que realizar la prueba.

Ejemplos

Lo siguiente resulta verdadero si la propiedad booleana denominada valores_agrupados es verdadera:

```
<Condition property="valores_agrupados" op="equals" value="true"/>
```

El siguiente ejemplo es verdadero si se ha seleccionado un control denominado valores_agrupados que muestra valores booleanos:

```
<Condition control="valores_agrupados" op="equals" value="true"/>
```

El siguiente ejemplo es verdadero si una propiedad de lista denominada campos_gráfico cuenta con al menos un valor.

```
<Condition property="campos_gráfico" op="countGreaterThan" value="0"/>
```

El siguiente ejemplo es verdadero si una propiedad de lista denominada campos_entrada sólo contiene los valores que son instanciados:

```
<Condition property="campos_entrada" op="instantiated" listMode="all"/>
```

El siguiente ejemplo es verdadero si una propiedad de lista denominada campos_entrada tiene al menos un valor que representa un campo sin instancia:

```
<Condition property="campos_entrada" op="uninstantiated" listMode="any "/>
```

Condiciones compuestas

Grupos de condiciones simples que pueden combinarse mediante operadores lógicos.

Ejemplos

Lo siguiente resulta verdadero si la propiedad booleana `valores_agrupados` es verdadera y `campos_grupo` contiene al menos un valor:

```
<And>
     <Condition property="valores_agrupados" op="equals" value="true"/>
     <Condition property="campos_grupo" op="countGreaterThan" value="0"/>
</And>
```

Lo siguiente resulta verdadero si la propiedad booleana `valores_agrupados` es verdadera o si `campos_grupo` contiene al menos un valor:

```
<Or>
     <Condition property="valores_agrupados" op="equals" value="true"/>
     <Condition property="campos_grupo" op="countGreaterThan" value="0"/>
</Or>
```

Lo siguiente es verdadero si `campos_grupo` contiene al menos un valor:

```
<Not>
     <Condition property="campos_grupo" op="equals" value="0"/>
</Not>
```

Las condiciones compuestas pueden anidarse para ofrecer cualquier combinación de condiciones.

Uso de nodos CLEF en scripts

Puede hacer referencia a un nodo de CLEF en un script utilizando el atributo `scriptName` del elemento `Node`. De la misma manera, puede hacer referencia a una propiedad del nodo en un script mediante el atributo `scriptName` del elemento `Property`.

El atributo `scriptName` es opcional en ambos casos, aunque es recomendable utilizar el atributo para evitar conflictos de nombre entre extensiones o propiedades.

Si omite el nombre del script en una definición de nodo, un script puede hacer referencia al nodo mediante el valor del atributo `id` delante del nombre de la extensión. Por ejemplo, con una extensión denominada `myext` y que define un nodo de lector de datos con la ID `import`, un script podrá hacer referencia al tipo de nodo como `myextimport`.

Si omite el nombre del script en una definición de propiedad, un script puede hacer referencia a la propiedad mediante el valor de su atributo `name`.

Si desea obtener más información, consulte *IBM SPSS ModelerAutomatización y scripts.*

Ejemplo: Edición y ejecución de un nodo

El siguiente ejemplo muestra cómo utilizar un script para automatizar las tareas de editar y ejecutar el nodo de ejemplo de lector de datos que se muestra en “Nodos de lector de datos (lector de registro de Apache)” en la página 26.

En el archivo de especificación del nodo del lector de registro de Apache, la especificación del nodo comienza de la siguiente manera:

```
<Node id="apachelogreader" type="dataReader" palette="import" labelKey="apacheLogReader.LABEL">
     <Properties>
          <Property name="nombre archivo_registro" valueType="string" labelKey="logfileName.LABEL" />
     </Properties>
```

En el script, haría referencia al nodo y a la propiedad de la siguiente manera:

```
create apachelogreader
set :apachelogreader.nombre archivo_registro='directorio_instalación\Demos\combined_log_format.txt'
create tablenode at 200 100
connect :apachelogreader to :tablenode
execute :tablenode
```

siendo *directorio_instalación* el directorio en el que se instala IBM SPSS Modeler.

Ejecución del script:

- crea el nodo del lector de datos
- especifica *combined_log_format.txt* como el archivo de registro de Apache que se leerá
- crea un nodo de tabla
- conecta el nodo del lector de datos al nodo de tabla
- ejecuta el nodo de tabla

Ejemplo: propiedades con clave

Las propiedades con clave admiten sintaxis de scripts estándar. Por ejemplo, la estructura del primer ejemplo de los tipos de propiedades con clave de "Propiedades estructuradas" en la página 62 se podría definir en un script como:

```
set :mynode.aggregation_settings.Age = {true true false false false}
```

Un atributo simple se puede modificar de la siguiente manera:

```
set :mynode.aggregation_settings.Age.MIN = true
```

Conservación de la compatibilidad con versiones anteriores

Cuando planifique actualizaciones con una extensión existente, tenga cuidado de conservar la compatibilidad con una versión distribuida previamente de esa extensión. Algunos cambios no tendrán efectos negativos, algunos incluyen importantes riesgos y otros pueden impedir la compatibilidad y deberá evitarse.

Cambios sin riesgos

Los siguientes cambios no afectarán a la compatibilidad con versiones anteriores:

- adición de nuevos elementos `Node`, `ModelOutput`, `DocumentOutput` o `InteractiveModelBuilder`
- adición de nuevas definiciones de `Property` y sus controles nuevos asociados a estos elementos
- adición de nuevos contenedores a estos elementos*
- adición de nuevos valores a una propiedad de enumeración existente

* Tenga en cuenta que cualquier código que utilice estos nuevos contenedores debe contemplar que estos contenedores estarán vacíos para objetos creados con la versión anterior de la extensión.

Cambios con riesgos significativos

Los cambios que aplique a instrucciones existentes conllevan importantes riesgos de impedir la compatibilidad. Deben comprobarse adecuadamente antes de su distribución.

Cambios que se deben evitar

Se ha comprobado que los siguientes cambios impiden la compatibilidad y deben evitarse:

- cambio del valor de los atributos `id` o `providerTag` en el elemento `ExtensionDetail`.
- cambio del valor del atributo `id` en los elementos `Node`, `ModelOutput`, `DocumentOutput` o `InteractiveModelBuilder`.
- eliminación del elemento `Node`, `ModelOutput`, `DocumentOutput` o `InteractiveModelBuilder` de la extensión.
- cambio del valor del atributo `valueType` de un elemento `Property` o `PropertyType`

Capítulo 5. Generación de modelos y documentos

Introducción a generación de modelos y documentos

Los módulos estándar de IBM SPSS Modeler incluyen nodos que permiten a los usuarios generar (o "construir") una amplia variedad de modelos y gráficos. CLEF permite definir más nodos para generar otros modelos y documentos (gráficos e informes) que no se proporcionan como opciones estándar.

Cuando defina nodos generadores de modelos o documentos, también debe definir los objetos que se producen cuando se ejecutan estos nodos. Se realiza mediante la acción de elementos denominados "Constructors".

Las secciones siguientes describen detalladamente este proceso.

Modelos

Un **modelo** es un conjunto de reglas, una fórmula o una ecuación que se puede utilizar para predecir un resultado basándose en un conjunto de campos de entrada. La capacidad para predecir un resultado es el objetivo central de los análisis predictivos. En IBM SPSS Modeler se consigue:

- Generando un modelo a partir de los datos existentes
- Aplicando el modelo generado a los datos para realizar predicciones

El proceso de generar un modelo también se conoce como "crear" un modelo y en IBM SPSS Modeler se realiza mediante un nodo de modelado. En CLEF, los nodos de modelado se denominan **nodos generadores de modelos**, un nombre derivado de la sintaxis de la declaración XML utilizada para definirlos. Para obtener más información, consulte el tema “Nodos generadores de modelos” en la página 11.

El proceso de aplicar un modelo a los datos se denomina "puntuar los datos". La puntuación de datos le permite utilizar la información obtenida a partir de la generación de modelos para crear predicciones para nuevos registros. En IBM SPSS Modeler, se realiza añadiendo el icono de un modelo generado al lienzo de rutas. El icono adopta la forma de una pepita de oro, por lo que un modelo generado se denomina en IBM SPSS Modeler "nugget de modelo". En CLEF, un nugget de modelo en la pestaña Modelos del panel de gestor se denomina **objeto de resultado de modelo** y cuando se añade al lienzo, se denomina **nodo aplicador de modelos**. Para obtener más información, consulte el tema “Nodos aplicadores de modelos” en la página 12.

Documentos

En algunos casos, deseará generar un objeto que no sea un modelo, como un gráfico o un informe de resultados. En IBM SPSS Modeler, estos objetos se conocen como **documentos** y se generan mediante un **nodo generador de documentos**. Para obtener más información, consulte el tema “Nodos generadores de documentos” en la página 12.

Constructores

Los constructores definen los objetos que se producen como resultado de ejecutar un nodo en una ruta o de generar un objeto en la ruta.

Los constructores se pueden definir mediante uno de los siguientes elementos:

- nodo generador de modelos
- Nodo generador de documentos
- Nodos aplicadores de modelos

- Objeto de resultado de modelo

En el caso de un nodo de **generador de modelos** o **generador de documentos**, un constructor permite a los nodos definir cómo se generará el objeto de resultado cuando se ejecute el nodo. Una definición de objeto de resultado puede incluir múltiples propiedades y componentes y la sección Constructores define cómo se inicializan o se crean a partir del objeto generado por la ejecución.

En un nodo de **aplicador de modelos**, un constructor define el tipo de objeto que el nodo puede generar en la ruta o en la pestaña Modelos.

En el caso de que se definan constructores para un **objeto de resultado de modelo**, pueden:
- Especificar el nodo aplicador de modelos que se creará si el objeto de resultado de modelo se suelta en el lienzo de rutas
- Generar un nodo generador de modelos con los parámetros utilizados para crear el objeto de resultado de modelo

Para obtener más información, consulte el tema “Uso de constructores” en la página 100.

Generación de modelos

Si especifica un nodo desde el que se puede generar un modelo (es decir, un nodo generador de modelos), debe definir cómo se comunicará el nodo con el componente de generador de modelos de IBM SPSS Modeler, que es el proceso que crea el modelo. Se realiza en la definición de un elemento `Node` en el archivo de especificaciones.

Salvo que se especifique lo contrario, la generación del modelo comienza en cuanto el usuario pulsa en el botón **Ejecutar** en el cuadro de diálogo del nodo generador de modelos. Sin embargo, también es posible definir un **modelo interactivo**, en el que el usuario final puede refinar o modificar los valores de datos después de pulsar en **Ejecutar** pero antes de generar el modelo. La generación de modelos interactivos también requiere la inclusión de elementos específicos mediante los cuales se define la interactividad. Para obtener más información, consulte el tema “Generación de modelos interactivos” en la página 88.

Cuando defina un nodo generador de modelos, el elemento `Node` debe incluir:
- Un atributo `type="modelBuilder"`
- Un elemento hijo `ModelBuilder`
- Un elemento hijo `Constructors` que contiene un elemento `CreateModelOutput` (consulte “Uso de constructores” en la página 100)

Para conocer el formato de especificación de un elemento `Node`, consulte “Nodo” en la página 48.

Note: En las definiciones de elemento de las siguientes secciones (por lo general identificadas mediante la cabecera **Formato**), los atributos de elemento y los elementos hijo son opcionales a no ser que se indiquen como "(obligatorios)". Para conocer la sintaxis completa de los elementos, consulte “Esquema XML de CLEF”, en la página 203.

Para la creación de modelos, la extensión también necesita un elemento `ModelOutput` para describir el modelo generado (consulte “Resultado de modelo” en la página 87). El elemento `ModelOutput` debe incluir un elemento hijo `Constructors` que contiene una definición `CreateModelApplier`. Para obtener más información, consulte el tema “Creación de aplicador de modelos” en la página 102.

Generador de modelos

El elemento `ModelBuilder` define el comportamiento de un nodo generador de modelos. Se realiza mediante los atributos de elementos y uno o más elementos hijo.

Formato

```
<ModelBuilder allowNoInputs="true_false" allowNoOutputs="true_false" nullifyBlanks="true_false"
     miningFunctions="[función1 función2 ... ]" >
     <Algorithm ... />
     <ModelingFields ... />
     <ModelGeneration ... />
     <ModelFields ... />
     <AutoModeling ... />
</ModelBuilder>
```

donde:

- `allowNoInputs` y `allowNoOutputs` se deben utilizar de forma explícita en caso de que desee crear un modelo que no tenga campos de entrada ni de resultado, respectivamente.
- `nullifyBlanks`, si se define como `false`, desactiva la característica mediante la que los valores vacíos se sustituyen por valores nulos (representados por `$null$`) en los datos que se pasan al componente de generación de modelos de IBM SPSS Modeler. La opción predeterminada es sustituir los espacios vacíos por valores nulos, pero es posible que desee desactivar esta característica, por ejemplo, si su algoritmo necesita considerar los elementos vacíos de forma diferente a los valores nulos.
- `miningFunctions` (obligatorio) identifica la función o funciones de minería de datos o las funciones que ejecuta el modelo.

Tabla 29. Funciones de minería de datos.

Función	Descripción
`classification`	Predice un valor discreto (es decir, tipos de datos `conjuntos`, `marca` o `orderedSet`) de un atributo de destino desconocido de los registros con valores objetivos conocidos.
`approximation`	Predice un valor continuo (es decir, tipos de datos `rango`) de un atributo de destino desconocido de los registros con valores objetivos conocidos.
`clustering`	Identifica grupos de registros similares y los etiqueta correctamente.
`association`	Identifica los eventos o atributos relativos de los datos.
`sequence`	Busca patrones secuenciales en datos con estructura temporal.
`reduction`	Reduce la complejidad de los datos, por ejemplo, mediante campos derivados que resumen el contenido de los campos de datos originales.
`conceptExtraction`	Se utiliza en minería de texto.
`categorize`	Se utiliza en minería de texto.
`timeSeries`	Prevé valores futuros a partir de patrones en datos pasados.
`anomalyDetection`	Busca casos atípicos basados en desviaciones de las normas de sus grupos de clústeres.
`attributeImportance`	Identifica los atributos con la mayor influencia en un atributo de destino.
`supervisedMultiTarget`	Estima la posibilidad de un resultado (*sí* o *no*) entre un conjunto de posibilidades.

Si el modelo ejecuta más de una función, los nombres de la función están separados por espacios en los paréntesis, como en el ejemplo siguiente:

```
<ModelBuilder miningFunctions="[aproximación de clasificación]"> ...
</ModelBuilder>
```

Elementos hijo

Los elementos hijo del elemento `ModelBuilder` se muestran en la siguiente tabla.

Tabla 30. Elementos hijo de una declaración de generador de modelos.

Elemento hijo	Descripción	Consulte...
`Algoritmo`	(obligatorio) Especifica el algoritmo que se utiliza para generar el modelo.	"Algoritmo"
`ModelingFields`	Especifica el identificador que se utilizará posteriormente en la sección de interfaz de usuario para definir la ubicación de los controles del campo de entrada y resultado del modelo. Los controles se definen en los elementos hijo `InputFields` y `OutputFields` de `ModelingFields`.	"Campos de modelado"
`ModelGeneration`	Especifica el identificador que se utilizará posteriormente en la sección de interfaz de usuario para definir la ubicación de los controles del nombre del modelo generado.	"Generación de modelos" en la página 85
`ModelFields`	Especifica el conjunto campos de entrada y resultado que se utilizan para puntuar los datos con este modelo.	"Campos de modelo" en la página 85
`AutoModeling`	Activa este modelo para que se utilice con un nodo de modelado de conjunto, como Clasificador automático, Agrupación en clústeres automática o Autonumérico.	"Modelado automático" en la página 92

Algoritmo

El elemento `Algorithm` define los detalles del algoritmo utilizado para generar el modelo.

```
<Algorithm value="id_resultado_modelo" label="etiqueta_visualización" labelKey="clave_etiqueta"/>
```

donde:

- `value` (obligatorio) es el nombre interno del algoritmo. Se incluye en otras ubicaciones en el archivo de especificación. Para obtener más información, consulte el tema "Ejemplo de generador de modelos" en la página 86.
- `label` (obligatorio) es una descripción del algoritmo.
- `labelKey` identifica la etiqueta con fines de localización.

Campos de modelado

Los campos de entrada y resultado del modelo se especifican de manera estándar mediante el nodo Tipo. Los usuarios definen el rol del campo como **Entrada** o **Resultado**, según se desee. Como opción, en un nodo generador de modelos, puede ofrecer a los usuarios la oportunidad de sustituir los parámetros en un nodo Tipo situado en un punto anterior y utilizar la configuración personalizada.

Se realiza mediante el elemento `ModelingFields`. Este elemento especifica un identificador que se utiliza posteriormente en la sección de interfaz de usuario de la declaración del nodo generador de modelos para definir la ubicación de los controles de los campos de entrada y resultado del modelo. Los controles se definen en los elementos hijo `InputFields` y `OutputFields`.

Formato

```
<ModelingFields controlsId="identificador_control" ignoreBOTH="true_false" >
      <InputFields ... />
      <OutputFields ... />
</ModelingFields>
```

donde:

- controlsId (obligatorio) es el identificador que se utilizará posteriormente en un elemento SystemControls en la sección de interfaz de usuario de la declaración del nodo generador de modelos. De esta forma identifica la pestaña del cuadro de diálogo del nodo que contiene los controles del campo de entrada y resultado del modelo.
- ignoreBOTH, si se define como true (obligatorio), especifica que el modelo ignora los campos con un rol de campo definido como **Ambas**.

Los elementos InputFields y OutputFields se describen en las secciones que comienzan en "Campos de entrada".

Ejemplo

Este ejemplo ilustra el uso de un conjunto de controles de los campos de modelado en una pestaña Campos de un cuadro de diálogo del nodo generador de modelos. En primer lugar, se especifica un identificador para el conjunto de controles:

```
<ModelBuilder miningFunctions="[clasificación]">
 ...
     <ModelingFields controlsId="modelingFields">
          <InputFields property="entradas" onlyNumeric="true" multiple="true" label="Entradas"
            labelKey="inputFields.LABEL"/>
          <OutputFields property="objetivo" multiple="false" types="[conjunto marca]" label="Objetivo"
            labelKey="targetField.LABEL"/>
     </ModelingFields>
 ...
</ModelBuilder>
```

El identificador modelingFields se incluye posteriormente en la sección de interfaz de usuario del cuadro de diálogo del nodo, en el punto donde se define la pestaña Campos:

```
<UserInterface ...>
     <Tabs defaultTab="1">
          <Tab label="Campos" labelKey="Fields.LABEL" helpLink="pestañacampos_modelado.htm">
               <PropertiesPanel>
                    <SystemControls controlsId="modelingFields">
                    </SystemControls>
               </PropertiesPanel>
          </Tab>
 ...
</UserInterface>
```

Campos de entrada: El elemento InputFields define el conjunto de campos desde el que los usuarios podrán seleccionar uno o más campos de entrada (es decir, predictores) del modelo.

El conjunto está compuesto por todos los campos visibles en este nodo. Si se han filtrado campos en un punto anterior de la ruta de este nodo, sólo serán visibles los campos que hayan pasado por el filtro. La lista también puede restringirse aun más especificando que sólo estarán disponibles para su selección los campos que tengan unos tipos específicos de almacenamiento y datos.

```
<InputFields storage="tipos_almacenamiento" onlyNumeric="true_false" onlySymbolic="true_false"
     onlyDatetime="true_false" types="tipos_datos" onlyRanges="true_false"
     onlyDiscrete="true_false" property="nombre_propiedad" multiple="true_false" label="etiqueta"
     labelKey="clave_etiqueta"/>
```

Puede restringir la lista de campos que se utilizarán como campos de entrada especificando dos atributos, uno de los cuales debe proceder de la lista siguiente:

- storage es una propiedad de lista que especifica el tipo de almacenamiento de los campos que se van a admitir en la lista; por ejemplo, storage="[integer real]" significa que sólo se enumerarán los

campos con tipos de almacenamiento de número entero real. Para conocer el conjunto de posibles tipos de almacenamiento, consulte la tabla que se encuentra debajo de "Tipos de almacenamiento y datos" en la página 183.

- `onlyNumeric`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento numérico.
- `onlySymbolic`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento simbólico (es decir, cadena).
- `onlyDatetime`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento de fecha y hora.

El segundo atributo especificado debe pertenecer a esta lista:

- `types` es una propiedad de lista que especifica el tipo de datos de los campos que se van a admitir en la lista; por ejemplo, `types="[range flag]"` significa que sólo se enumerarán los campos con tipos de almacenamiento de marca de rango. El conjunto de tipos de datos posibles es:

  `rango`

  `distintivo`

  `set`

  `orderedSet`

  `numérico`

  `discrete`

  `typeless`
- `onlyRanges`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de datos de rango.
- `onlyDiscrete`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de datos discreto (es decir, marca, conjunto o sin tipo).

Por eso, por ejemplo, un control que especifica `storage="[integer]"` y `types="[flag]"` garantiza que en la lista sólo aparecerán campos de números enteros que sean marcas.

Los atributos restantes son los siguientes:

- `property` es el identificador de la propiedad que va a utilizarse para almacenar los valores de campo.
- `multiple` especifica si los valores de campo son una lista enumerada (`true`) o no (`false`).
- `label` es el nombre de visualización del control.
- `labelKey` identifica la etiqueta con fines de localización.

Campos de resultado: El elemento `OutputFields` define el conjunto de campos desde el que los usuarios podrán seleccionar uno o más campos de resultado (es decir, objetivos) del modelo.

El conjunto está compuesto por todos los campos visibles en este nodo. Si se han filtrado campos en un punto anterior de la ruta de este nodo, sólo serán visibles los campos que hayan pasado por el filtro. La lista también puede restringirse aun más especificando que sólo estarán disponibles para su selección los campos que tengan unos tipos específicos de almacenamiento y datos.

```
<OutputFields storage="tipos_almacenamiento" onlyNumeric="true_false" onlySymbolic="true_false"
     onlyDatetime="true_false" types="tipos_datos" onlyRanges="true_false"
     onlyDiscrete="true_false" property="nombre_propiedad" multiple="true_false" label="etiqueta"
     labelKey="clave_etiqueta"/>
```

Puede restringir la lista de campos que se utilizarán como campos de resultado especificando dos atributos, uno de los cuales debe proceder de la lista siguiente:

- `storage` es una propiedad de lista que especifica el tipo de almacenamiento de los campos que se van a admitir en la lista; por ejemplo, `storage="[integer real]"` significa que sólo se enumerarán los

campos con tipos de almacenamiento de número entero real. Para conocer el conjunto de posibles tipos de almacenamiento, consulte la tabla que se encuentra debajo de "Tipos de almacenamiento y datos" en la página 183.

- `onlyNumeric`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento numérico.
- `onlySymbolic`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento simbólico (es decir, cadena).
- `onlyDatetime`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento de fecha y hora.

El segundo atributo especificado debe pertenecer a esta lista:

- `types` es una propiedad de lista que especifica el tipo de datos de los campos que se van a admitir en la lista; por ejemplo, `types="[range flag]"` significa que sólo se enumerarán los campos con tipos de almacenamiento de marca de rango. El conjunto de tipos de datos posibles es:

  ```
  rango
  distintivo
  set
  orderedSet
  numérico
  discrete
  typeless
  ```

- `onlyRanges`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de datos de rango.
- `onlyDiscrete`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de datos discreto (es decir, marca, conjunto o sin tipo).

Por eso, por ejemplo, un control que especifica `storage="[integer]"` y `types="[flag]"` garantiza que en la lista sólo aparecerán campos de números enteros que sean marcas.

Los atributos restantes son los siguientes:

- `property` es el identificador de la propiedad que va a utilizarse para almacenar los valores de campo.
- `multiple` especifica si los valores de campo son una lista enumerada (`true`) o no (`false`).
- `label` es el nombre de visualización del control.
- `labelKey` identifica la etiqueta con fines de localización.

Generación de modelos

El elemento `ModelGeneration` especifica el identificador que se utilizará en otra parte del archivo para definir la pestaña del cuadro de diálogo del nodo generador de modelos que contendrá los controles del nombre del modelo generado.

El formato es:

```
<ModelGeneration controlsId="identificador_control" />
```

El atributo `controlsId` especifica el identificador que se utilizará posteriormente en un elemento `SystemControls` en la sección de interfaz de usuario de la especificación del nodo generador de modelos. La pestaña cuya especificación incluye este elemento `SystemControls` es la que contendrá los controles del nombre del modelo.

Campos de modelo

El elemento `ModelFields` se utiliza para generar la **firma de modelo**: el conjunto de campos de entrada y resultado utilizados para puntuar los datos con este modelo.

```
<ModelFields inputDirections="[entrada]" outputDirections="[resultado]">
     <AddField prefix="prefijo_campo" ... />
      ...
     <ForEach ...>
          <AddField prefix="prefijo_campo" ... />
     </ForEach>
      ...
</ModelFields>
```

donde inputDirections y outputDirections especifican cómo se genera la firma del modelo. Los valores pueden ser de entrada, resultado o ambos.

Los campos se especifican por uno o más elementos AddField. El atributo prefix especifica el prefijo que se añadirá a un nombre de campo para denotar un campo generado por el modelo. Por ejemplo, si el nombre del campo es campo1 y el valor del prefijo es $S, el campo generado se denomina $S-campo1. Para obtener más información, consulte el tema "Adición de un campo" en la página 64.

La iteración es posible mediante los elementos ForEach. Para obtener más información, consulte el tema "Iteración con el elemento ForEach" en la página 68.

Grupos de campos le permite agrupar dos o más campos de salida del modelo con el fin de realizar iteraciones. A los nombres de campo de salida se les añade un sufijo que indica la iteración, por ejemplo, $S-campo1-1, $S-campo1-2, etc. Uno de los usos de los grupos de campos es lograr que el mismo conjunto de campos aparezca varias veces en el resultado de modelo. Para obtener más información, consulte el tema "Ejemplo de grupo de campos".

Automodeling

El elemento AutoModeling activa el modelo que se utilizará por un nodo de modelado de conjunto, como Clasificador automático, Agrupación en clústeres automática o Autonumérico. Para obtener más información, consulte el tema "Modelado automático" en la página 92.

Ejemplo de generador de modelos

A continuación se muestra la sección completa del generador de modelos en el archivo de especificación del ejemplo del nodo Interacción (consulte "Nodo generador de modelos (Interacción)" en la página 27):

```
<Node id="generador.interacciones" type="modelBuilder" palette="modelado" label="Interacción">
     <ModelBuilder miningFunctions="[clasificación]">
          <Algorithm value="robd" label="Algoritmo de Roberto" />
          <ModelingFields controlsId="modellingFields">
               <InputFields property="entradas" multiple="true" label="Entradas" onlyDiscrete="true" />
               <OutputFields property="objetivo" multiple="false" label="Objetivo" onlyDiscrete="true" />
          </ModelingFields>
          <ModelFields inputDirections="[entrada]" outputDirections="[resultado]">
               <ForEach var="field" inFields="outputs">
                    <AddField prefix="$I" name="${campo}" fieldRef="${campo}" role=
                     "Valorpredicho" targetField="${campo}" />
                    <AddField prefix="$IP" name="${campo}" storage="real" role="probabilidad"
                     targetField="${campo}">
                          <Range min="0.0" max="1.0"/>
                    </AddField>
               </ForEach>
          </ModelFields>
     </ModelBuilder>
 ...
</Node>
```

Ejemplo de grupo de campos

Este ejemplo se ha tomado del nodo SLRM y añade un grupo con dos campos nuevos a la firma de modelo para incluir los datos generados cuando puntúe el modelo. En el caso de cada registro de

entrada, los datos se puntúan para cada uno de los campos nuevos las veces que especifique el usuario, determinado por el valor de la propiedad max_predictions.

Los dos campos nuevos son:

- $S-*destino*: contiene el valor predicho del campo de destino.
- $SC-*destino*: contiene el valor de probabilidad de esta predicción.

Para agrupar estos dos campos, se les asigna el mismo identificador de grupo cuando se declaran en la sección ModelFields. Los identificadores de grupo se asignan mediante el atributo group del elemento AddField.

De este modo, la declaración del nodo generador de modelos contiene:

```
<Node ... type="modelBuilder" ...>
     <ModelBuilder ...>
     ...
          <ModelFields inputDirections="[entrada]" outputDirections="[resultado]">
               <AddField prefix="$S" name="${objetivo}" fieldRef="${objetivo}" role=
                "Valorpredicho" targetField="${objetivo}" group="[1]"/>
               <AddField prefix="$SC" name="${objetivo}" storage="real" role="probabilidad"
                  targetField="${objetivo}" group="[1]">
                    <Range min="0.0" max="1.0"/>
               </AddField>
          </ModelFields>
     </ModelBuilder>
</Node>
```

La declaración del nodo aplicador de modelos contiene:

```
<Node ... type="modelApplier" ...>
...
     <OutputDataModel mode="extend">
          <ForEach var="group" from="1" to="${max_predictions}">
               <ForEach var="field" inFields="modelOutputs" container="model">
                    <AddField name="${field}" group="[${group}]" fieldRef="${field}" />
               </ForEach>
          </ForEach>
     </OutputDataModel>
</Node>
```

El campo objetivo se denomina **campaña** y las entradas de usuario 2 en el campo correspondiente a la propiedad max_predictions. La ejecución del nodo generador de modelos hace que los siguientes campos se añadan al modelo:

- $S-campaña-1
- $SC-campaña-1
- $S-campaña-2
- $SC-campaña-2

Resultado de modelo

El elemento ModelOutput describe un objeto de resultado de modelo; un objeto que aparecerá en la pestaña Modelos del panel de gestor tras la ejecución de una ruta.

Formato

```
<ModelOutput id="identificador" label="etiqueta_visualización" labelKey="clave_etiqueta" >
     <ModelProvider ... />
     <Properties>
          <Property ... />
```

```
        ...
    </Properties>
    <Containers ... />
    <UserInterface ... />
    <Constructors ... />
</ModelOutput>
```

donde:

- `id` (obligatorio) es un identificador exclusivo para el modelo generado.
- `label` (obligatorio) es el nombre de representación del valor de propiedad tal y como aparece en la pestaña Modelos.
- `labelKey` identifica la etiqueta con fines de localización.

Los elementos hijo que se pueden incluir en el elemento `ModelOutput` se muestran en la tabla siguiente.

Tabla 31. Elementos hijo de una declaración del resultado del modelo.

Elemento hijo	Define	Consulte...
`ModelProvider`	El contenedor que contiene el resultado del modelo y si el resultado tiene el formato PMML.	"Proveedor de modelos" en la página 51
`Properties`	Propiedades que utilizará el modelo generado.	"Propiedades" en la página 51
`Containers`	Los contenedores en los que se ubicará el resultado del modelo generado.	"Containers" en la página 53
`UserInterface`	La interfaz de usuario mediante la que se puede visualizar el resultado del modelo generado, por ejemplo, las pestañas Modelo y Configuración en un objeto de resultado de modelo.	"Interfaz de usuario" en la página 54
`Constructores`	Objetos producidos por el modelo generado.	"Uso de constructores" en la página 100

Ejemplo

```
<ModelOutput id="modelo.interacción" label="Modelo interacción">
    <Properties>
    </Properties>
    <Containers>
        <Container name="modelo" />
    </Containers>
    <UserInterface >
        <Pestañas>
            <Tab label="Modelo">
                <TextBrowserPanel container="modelo" textFormat="Textoplano" />
            </Tab>
        </Tabs>
    </UserInterface>
    <Constructors>
        <CreateModelApplier type="aplicador.interacción">
            <SetContainer target="modelo" source="modelo" />
        </CreateModelApplier>
    </Constructors>
</ModelOutput>
```

Generación de modelos interactivos

El modelado interactivo es una característica que permite crear un objeto de resultado con el que el usuario final puede interactuar antes de generar un modelo. Este objeto de resultados interactivos se coloca en la pestaña Resultados del panel de gestor y contiene un conjunto de datos provisionales. El

conjunto de datos provisionales se puede utilizar para refinar o simplificar el modelo antes de que se genere. El modelado interactivo se realiza añadiendo elementos adicionales a la especificación de un nodo generador de modelos normal:

- La sección `Constructors` de la definición del `Nodo` incluye un elemento `CreateInteractiveModelBuilder`.
- La extensión incluye un elemento `InteractiveModelBuilder` exclusivo.

La interacción del usuario con el conjunto de datos provisional se produce mediante una ventana denominada **ventana de interacción**, que aparece en cuanto se crea el objeto de resultado.

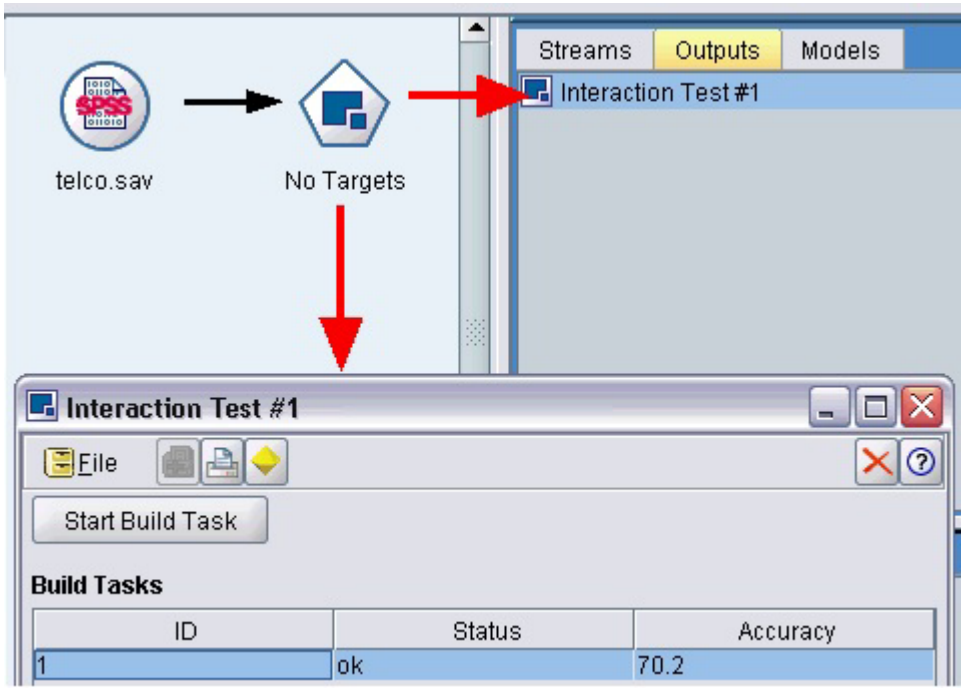

Figura 30. Objeto de resultados interactivos y ventana de interacción

La interacción es específica del algoritmo utilizado y se aplica según la extensión. La ventana de interacción se define en la sección de interfaz de usuario del elemento `InteractiveModelBuilder`. Puede definir una ventana de interacción especificando uno de los siguientes elementos:

- Una clase de marco (consulte "Sección de interfaz de usuario" en la página 106), que define la ventana en su totalidad
- Una clase de panel, especificado como un atributo de un panel de objeto de extensión (consulte "Panel de objeto de extensión" en la página 118), para cada pestaña de la ventana

Una vez cerrada, la ventana de interacción se puede volver a abrir si pulsa dos veces en el nombre del objeto de la pestaña Resultados.

La especificación de la ventana de interacción necesita incluir un código para generar el modelo una vez el usuario ha completado la interacción. En el ejemplo ilustrado, se realiza mediante el botón de la barra de herramientas con el icono de pepita de oro, que está asociado con una acción para generar el modelo. El código se muestra en la sección `InteractiveModelBuilder` en "Ejemplo de modelado interactivo" en la página 91.

Creación de un generador de modelos interactivos

El elemento `CreateInteractiveModelBuilder` describe el objeto de resultado con el que interactuará el usuario. Se trata de una versión interactiva del elemento `CreateModelOutput`.

Formato

Este elemento se utiliza en la sección de ejecución de una definición de nodo generador de modelos:

```
<Node ... type="modelBuilder" ...>
 ...
     <Execution>
      ...
          <Constructors>
           ...
               <CreateInteractiveModelBuilder ...>
                 ...
               </CreateInteractiveModelBuilder>
          </Constructors>
     </Execution>
 ...
</Node>
```

El formato del elemento es:

```
<CreateInteractiveModelBuilder type="id_objeto_resultado">
     <Condition ... ./>
     <And>
     <Or>
     <Not>
     <CreateModel type="id_modelo" target="id_contenedor" sourceFile="id_archivo_contenedor" />
     <CreateDocument type="id_modelo" target="id_contenedor" sourceFile="id_archivo_contenedor" />
</CreateInteractiveModelBuilder>
```

donde `tipo` (obligatorio) es el identificador del objeto de resultado que crea el elemento `InteractiveModelBuilder`.

La sección `Condition` le permite especificar una o más condiciones. Para obtener más información, consulte el tema "Condiciones" en la página 72.

También puede especificar condiciones complejas que incluyen los operadores `And`, `Or` y `Not`. Para obtener más información, consulte el tema "Operadores lógicos" en la página 71.

En los elementos `CreateModel` y `CreateDocument`:

- `type` es el identificador del modelo o documento que se está definiendo.
- `target` (obligatorio) es el identificador del contenedor del modelo; este contenedor se define en una sección de resultado de modelo. Para obtener más información, consulte el tema "Resultado de modelo" en la página 87.
- `sourceFile` (obligatorio) es el identificador de un archivo de salida generado durante la ejecución del nodo; este archivo se define en una sección de archivos de resultados. Para obtener más información, consulte el tema "Archivos de resultados" en la página 56.

Ejemplo

```
<CreateInteractiveModelBuilder type="mi.interacción">
     <Condition property="interactiva" op="igual" value="true" />
</CreateInteractiveModelBuilder>
```

El ejemplo especifica que se creará un objeto de resultado con el identificador `mi.interacción` al ejecutar el nodo generador de modelos para la especificación que contiene el elemento. El objeto de resultado se define en otra ubicación del archivo de especificación, mediante un elemento `InteractiveModelBuilder` que hace referencia a este identificador, por ejemplo:

```
<InteractiveModelBuilder id="mi.interacción" label=...>
 ...
</InteractiveModelBuilder>
```

Generador de modelos interactivos

Este elemento define un objeto de resultados interactivos, que permite a un usuario final refinar o simplificar un modelo antes de generarlo.

El elemento `InteractiveModelBuilder` sigue la definición de un nodo generador de modelos que contiene el elemento `CreateInteractiveModelBuilder` correspondiente. Para obtener más información, consulte el tema "Creación de un generador de modelos interactivos" en la página 89.

Formato

El formato del elemento `InteractiveModelBuilder` es:

```
<Node ... type="modelBuilder" ...>
 ...
     -- Sección de creación de un generador de modelos interactivos --
 ...
</Node>
 ...
<InteractiveModelBuilder id="identificador" label="etiqueta_visualización" labelKey="clave_etiqueta" >
     <Properties>
          <Property name=... />
           ...
     </Properties>
     <Containers>
          <Container name="nombre_contenedor"/>
     </Containers>
     <UserInterface ... />
     <Constructors ... />
</InteractiveModelBuilder>
```

donde:

- `id` (obligatorio) es un identificador exclusivo para el modelo generado.
- `label` (obligatorio) es el nombre de representación del valor de propiedad tal y como aparece en la pestaña Modelos.
- `labelKey` identifica la etiqueta con fines de localización.

Para obtener más información acerca de los elementos `Properties`, `Containers`, `UserInterface` y `Constructors`, consulte "Propiedades" en la página 51, "Containers" en la página 53, "Sección de interfaz de usuario" en la página 106 y "Uso de constructores" en la página 100.

Ejemplo de modelado interactivo

Este ejemplo ilustra cómo puede definir un nodo generador de modelos de forma que los usuarios puedan seleccionar mediante una simple casilla de verificación si desean interactuar antes de generar el modelo.

Para ver esta función en la práctica, utilice el nodo Interacción de ejemplo que se incluye con esta versión. Para obtener más información, consulte el tema "Nodo generador de modelos (Interacción)" en la página 27.

En primer lugar, el nodo generador de modelos especifica una propiedad booleana:

```
<Node id="generador.interacciones" type="modelBuilder" ...>
 ...
     <Properties>
      <Property name="interactive" valueType="boolean" />
    </Properties>
```

En la sección de interfaz de usuario de la especificación del nodo, la sección que define la pestaña Modelo incluye una referencia a esta propiedad:

```
<Tab label="Modelo">
     <PropertiesPanel>
          <CheckBoxControl property="interactive" label="Iniciar una sesión interactiva" />
     </PropertiesPanel>
</Tab>
```

En la sección `CreateInteractiveModelBuilder` del mismo nodo, se comprueban los parámetros de la propiedad y si es `true`, se crea un objeto de resultados interactivos:

```
<CreateInteractiveModelBuilder type="mi.interacción">
     <Condition property="interactive" op="equals" value="true" />
</CreateInteractiveModelBuilder>
```

El objeto de resultado al que se refiere se define en la sección `InteractiveModelBuilder` de la extensión:

```
<InteractiveModelBuilder id="mi.interacción" label="Prueba de interacción">
     <Properties>
     </Properties>
     <Containers>
     </Containers>
     <UserInterface actionHandler="ui.InteractionHandler">
          <controles>
               <ToolbarItem action="generateModel" showLabel="false" />
          </Controls>
          <Pestañas>
               <Tab label="Modelo">
                    <ExtensionObjectPanel id="panel.modelo" panelClass="ui.SampleInteractionPanel" />
               </Tab>
               <Tab label="Genérico">
                    <ExtensionObjectPanel id="panel.genérico" panelClass="ui.GenericInteractionPanel" />
               </Tab>
          </Tabs>
     </UserInterface>
</InteractiveModelBuilder>
```

La acción para generar el modelo está controlada por el botón de la barra de herramientas que define el elemento `ToolbarItem`.

Tenga en cuenta que si utiliza el atributo `panelClass` del elemento `ExtensionObjectPanel` para especificar una clase de Java para controlar la interfaz del usuario para cada pestaña de la ventana de interacción.

Modelado automático

IBM SPSS Modeler proporciona un grupo de nodos de modelado de conjunto como opción estándar, como los nodos Clasificador automático, Agrupación en clústeres automática y Autonumérico. Estos nodos automatizan la creación de un número diferente de modelos de forma simultánea, permitiendo al usuario comparar los resultados y seleccionar el mejor modelo para sus datos. CLEF incluye el elemento `AutoModeling` para activar un modelo especificado por el elemento `ModelBuilder` que pueden utilizar cualquiera de estos nodos de conjunto.

El formato del elemento `AutoModeling` es:

```
<AutoModeling enabledByDefault="true_false">
     <SimpleSettings ... />
     <ExpertSettings ... />
     <Constraint ... />
     <Constraint ... />
      ...
</AutoModeling>
```

donde `enabledByDefault` especifica si el modelo está activado para su uso de forma predeterminada en el nodo de modelado de conjunto (o sea, la columna **Use?** está marcada de forma predeterminada para ese modelo en particular). Si se omite este atributo, se asume el valor `true`.

En un cuadro de diálogo del nodo de modelado de conjunto, la pestaña Experto muestra los modelos que los usuarios pueden seleccionar.

Si pulsa en **Especificar** en el campo **Parámetros del modelo** de un modelo particular, se abrirá el cuadro de diálogo Configuración del algoritmo, donde el usuario puede seleccionar las opciones del tipo de modelo.

El cuadro de diálogo Configuración del algoritmo contiene las pestañas Simple y Experto, correspondiente a los modos de ejecución Simple y Experto del nodo de modelado. El contenido de las pestañas Simple y Experto está controlado por los elementos `SimpleSettings` y `ExpertSettings`, que se describen en las secciones siguientes.

Además, los elementos `Constraint` permiten especificar las condiciones que permiten al usuario final editar o restringir de alguna forma los parámetros del cuadro de diálogo Configuración del algoritmo. Para obtener más información, consulte el tema “Restricciones” en la página 96.

Algunos parámetros del cuadro de diálogo Configuración del algoritmo pueden tener valores múltiples. Si se especifican múltiples valores, el nodo de conjunto intentará crear modelos para todas las combinaciones posibles de valores de parámetros. Por ejemplo, con un modelo lineal generalizado, si el usuario especifica dos distribuciones (normal y gamma) y tres funciones de enlace (identidad, registro y potencia), el nodo Autonumérico intenta generar seis Modelos lineales generalizados, uno para cada combinación posible de estos parámetros.

Configuración simple

El elemento `SimpleSettings` determina los parámetros que aparecen en la pestaña Simple del cuadro de diálogo Configuración del algoritmo para este modelo en un nodo de modelado de conjunto. Para obtener más información, consulte el tema “Modelado automático” en la página 92.

Formato

```
<SimpleSettings>
     <PropertyGroup label="nombre_grupo" labelKey="clave_recurso" properties="[nombre_prop1
      nombre_prop2 ...]"/>
     <PropertyGroup ... />
</SimpleSettings>
```

En un elemento `PropertyGroup` (al menos uno es obligatorio):

`label` es una etiqueta de visualización para el grupo de propiedades, insertada como cabecera secundaria en el cuadro de diálogo que se encuentra por delante del primer parámetro del grupo.

`labelKey` identifica la etiqueta con fines de localización. Si no se utiliza `label` ni `labelKey`, no se inserta ninguna cabecera secundaria.

properties (obligatorio) es una lista de una o más propiedades que deben aparecer en la pestaña. El valor de *nombre_prop1*, *nombre_prop2*, etc., es el valor del atributo name del elemento Property en el que se define esta propiedad. Para obtener más información, consulte el tema "Propiedades" en la página 51.

Ejemplo

```
<SimpleSettings>
     <PropertyGroup properties="[método]"/>
</SimpleSettings>
```

Este ejemplo del nodo Discriminante especifica que sólo el parámetro **Método** aparecerá en la pestaña Simple del cuadro de diálogo Configuración del algoritmo para ese modelo en el nodo de modelado de conjunto relevante (en este caso, el nodo Clasificador automático). Como no se ha especificado ningún atributo label o labelKey, no se muestra ninguna cabecera secundaria para el parámetro en el cuadro de diálogo.

Configuración de experto

El elemento ExpertSettings determina los parámetros que aparecen en la pestaña Experto del cuadro de diálogo Configuración del algoritmo para este modelo en un nodo de modelado de conjunto. Para obtener más información, consulte el tema "Modelado automático" en la página 92.

Formato

```
<ExpertSettings>
     <Condition ... />
     <PropertyGroup label="nombre_grupo" labelKey="clave_recurso"
      properties="[propiedad1 propiedad2 ...]"/>
     <PropertyGroup ... />
      ...
</ExpertSettings>
```

El elemento Condition especifica una condición que, si es true, activa los parámetros identificados por el elemento o elementos PropertyGroup posteriores. Para obtener más información, consulte el tema "Condiciones" en la página 72.

En un elemento PropertyGroup (al menos uno es obligatorio):

label es una etiqueta de visualización para el grupo de propiedades, insertada como cabecera secundaria en el cuadro de diálogo que se encuentra por delante del primer parámetro del grupo.

labelKey identifica la etiqueta con fines de localización. Si no se utiliza label ni labelKey, no se inserta ninguna cabecera secundaria.

properties (obligatorio) es una lista de una o más propiedades que deben aparecer en la pestaña. El valor de *nombre_prop1*, *nombre_prop2*, etc., es el valor del atributo name del elemento Property en el que se define esta propiedad. Para obtener más información, consulte el tema "Propiedades" en la página 51.

Ejemplos

En el ejemplo siguiente, la pestaña Experto del cuadro de diálogo Configuración del algoritmo tiene el parámetro **Modo** definido inicialmente como **Simple**.

Figura 31. Configuración de experto desactivada

La siguiente cadena especifica que el resto de parámetros de la pestaña Experto sólo se activan si el usuario cambia el ajuste del parámetro **Modo** a **Experto**:

```
<ExpertSettings>
     <Condition property="modo" op="igual" value="Experto"/>
     <PropertyGroup properties="[matriz_covarianza_probabilidades_previa_modo]"/>
      ...
</ExpertSettings>
```

Si cambia el parámetro **Modo** a **Experto** activará los dos parámetros en el grupo de propiedades.

Figura 32. Configuración de experto activada

El siguiente ejemplo ilustra el uso de etiquetas del grupo de propiedades:

```
<ExpertSettings>
      ...
     <PropertyGroup labelKey="automodelado.opciones_criterios_pasos"
      properties="[método V_pasos_introducir_criterios]"/>
      ...
</ExpertSettings>
```

En este caso, el elemento `PropertyGroup` controla los parámetros resaltados en la ilustración siguiente.

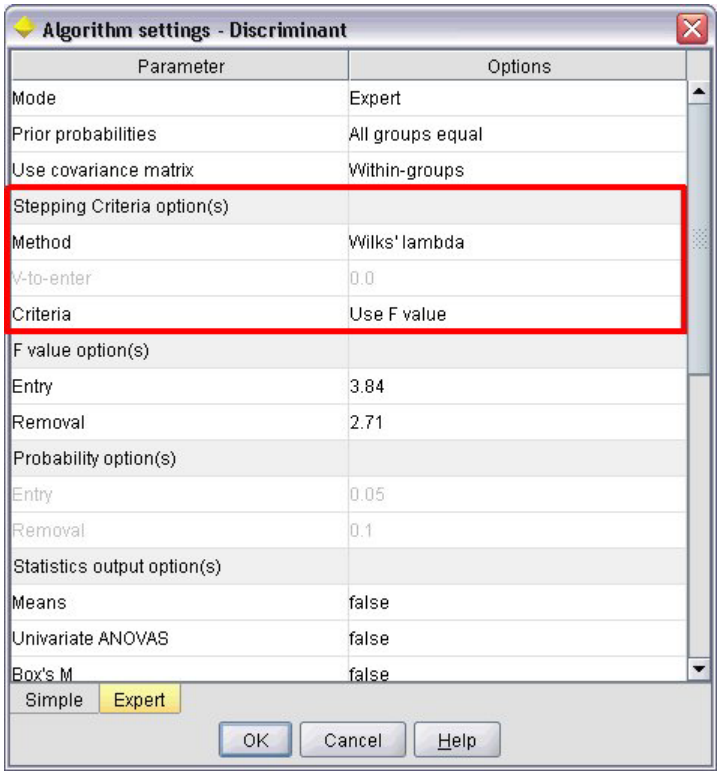

Figura 33. Configuración de experto desactivada

El atributo `labelKey` permite que CLEF recupere el texto de representación de la cabecera secundaria del grupo de propiedades de la entrada correspondiente en el archivo de propiedades de la extensión:

```
automodelado.opciones_criterios_pasos=Opciones de criterios del método por pasos
```

Para obtener más información, consulte el tema "Archivos de propiedades" en la página 168.

Restricciones

El elemento `Constraint` especifica las condiciones que permiten editar o limitar de alguna manera los parámetros que aparecen en el cuadro de diálogo Configuración del algoritmo de un modelo en un nodo de modelado de conjunto. Por ejemplo, algunos parámetros se pueden desactivar si el usuario final no puede modificarlos.

Formato

```
<Constraint property="nombre_propiedad" singleSelection="true_false">
     <Condition property="nombre_propiedad" op="operador" value="valor"/>
     <And ... />
     <Or ... />
     <Not ... />
</Constraint>
```

donde:

- `property` (obligatorio) identifica un parámetro que se debe editar o limitar. *nombre_prop* es el valor del atributo `name` del elemento `Property` en donde se define la propiedad correspondiente a este parámetro. Para obtener más información, consulte el tema "Propiedades" en la página 51.
- `singleSelection` controla si el usuario final puede seleccionar más de uno de los valores disponibles de un parámetro. Si se define como `true`, sólo se puede seleccionar un valor, incluso si existe más de un valor disponible en la lista del campo Opciones para ese parámetro en el cuadro de diálogo

Configuración del algoritmo. Si se define como `false` (valor predeterminado), el usuario puede seleccionar uno o más de los valores disponibles, tal y como se muestra en el ejemplo siguiente.

El elemento `Condition` especifica la limitación real. Para obtener más información, consulte el tema "Condiciones" en la página 72.

Los elementos `And`, `Or` y `Not` se pueden utilizar para especificar condiciones compuestas. Para obtener más información, consulte el tema "Operadores lógicos" en la página 71.

Ejemplo

Este ejemplo se toma del archivo de especificación del nodo de modelos lineales generalizados. Incluso en modo Experto, algunos parámetros no está activados de forma predeterminada.

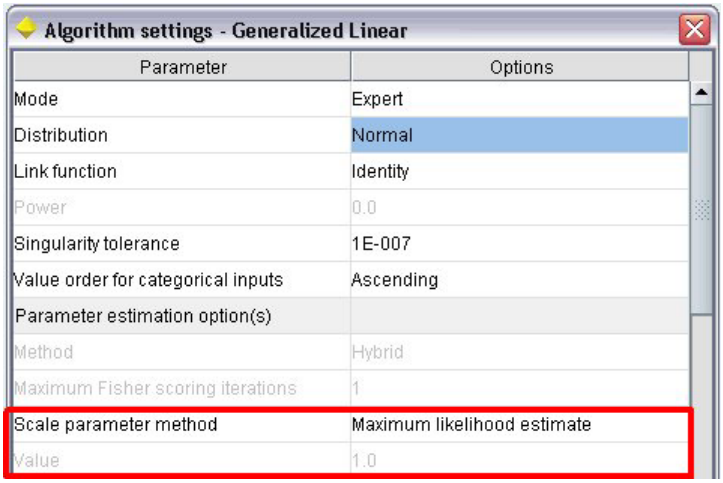

Figura 34. Efecto de un parámetro de limitación desactivado

La limitación especifica las condiciones en las que se activa el parámetro Valor:

```
<Constraint property="valor_escala">
     <And>
          <Condition property="método_escala" op="igual" value="Valor fijo"/>
          <Condition property="distribución" op="entrada" value="[IGAUSS GAMMA NORMAL]"/>
     </And>
</Constraint>
```

El parámetro **Método de parámetro de escala** (identificado por la propiedad `método_escala`) se debe definir como **Valor fijo** y **Distribución** debe ser **Normal**, **de Gauss inversa** o **Gamma** para poder activar el parámetro **Valor**.

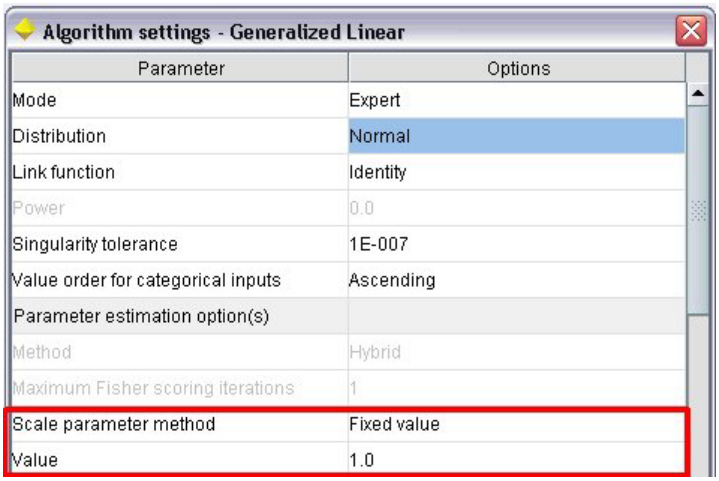

Figura 35. Efecto de un parámetro de limitación activado

Aplicación de modelos

Aplicar un modelo significa utilizar un modelo generado para puntuar los datos, es decir, para utilizar la información obtenida de la generación del modelo para crear predicciones para nuevos registros. En IBM SPSS Modeler, se realiza mediante un nodo aplicador de modelos. Para obtener más información, consulte el tema "Nodos aplicadores de modelos" en la página 12.

La definición de un nodo aplicador de modelos en el archivo de especificación crea el marco para aplicar un modelo generado. En IBM SPSS Modeler, crea una instancia de un nodo aplicador de modelos arrastrando el icono que representa el objeto de resultado de modelo de la pestaña Modelos del panel de gestor en el lienzo de rutas. Sin una definición del nodo aplicador de modelos, si ejecuta el nodo generador de modelos, generaría un modelo sin refinar, que no se pueden añadir al lienzo de rutas.

Cuando defina un nodo aplicador de modelos, el elemento `Node` debe incluir:

- Un atributo `type="modelApplier"`
- Un elemento hijo `Constructors` que contiene un elemento `CreateModelOutput` (consulte "Uso de constructores" en la página 100)

Para conocer el formato de especificación de un elemento `Node`, consulte "Nodo" en la página 48.

Generación de documentos

Cuando defina un nodo generador de documentos, el elemento `Node` debe incluir:

- Un atributo `type="documentBuilder"`
- Un elemento hijo `DocumentBuilder`

Para generar documentos, la extensión también necesita un elemento `DocumentOutput` para describir el documento generado. Para obtener más información, consulte el tema "Resultado de modelo" en la página 87.

Para conocer el formato de especificación de un elemento `Node`, consulte "Nodo" en la página 48.

Generador de documentos

El elemento `DocumentBuilder` define el comportamiento de un nodo generador de documentos. La definición debe incluir un elemento hijo `DocumentGeneration` para especificar la pestaña del cuadro de diálogo de nodo generador de documentos que contendrá los controles de generación del documento. Los controles se definen en la sección de interfaz de usuario (consulte Capítulo 6, "Generación de interfaces de usuario", en la página 105) .

Formato

```
<DocumentBuilder>
     <DocumentGeneration controlsId="identificador_control />
</DocumentBuilder>
```

donde `controlsId` (obligatorio) es el identificador utilizado por los controles del sistema para especificar dónde deben aparecer los controles de generación de documentos.

Ejemplo

```
<DocumentBuilder>
     <DocumentGeneration controlsId="1"/>
</DocumentBuilder>
```

Resultado de documento

El elemento `DocumentOutput` describe un objeto de resultado de documento; un objeto que aparecerá en la pestaña Resultados del panel de gestor tras la ejecución de una ruta.

Formato

```
<DocumentOutput id="identificador" label="etiqueta_visualización" labelKey="clave_etiqueta" >
     <Properties>
          <Property ... />
           ...
     </Properties>
     <Containers>
          <Container ... />
           ...
     </Containers>
     <UserInterface ... />
     <Constructors ... />
</DocumentOutput>
```

donde:

- `id` (obligatorio) es un identificador exclusivo para el documento generado.
- `label` (obligatorio) es el nombre de representación del documento generado tal y como aparece en la pestaña Resultados.
- `labelKey` identifica la etiqueta con fines de localización.

Los elementos hijo que se pueden incluir en el elemento `DocumentOutput` se muestran en la tabla siguiente.

Tabla 32. Elementos hijo de una declaración del resultado del documento.

Elemento hijo	Define	Consulte...
`Properties`	Propiedades que utilizará el documento generado.	"Propiedades" en la página 51
`Containers`	Los contenedores en los que se ubicará el resultado del documento generado.	"Containers" en la página 53

Tabla 32. Elementos hijo de una declaración del resultado del documento (continuación).

Elemento hijo	Define	Consulte...
`UserInterface`	Interfaz de usuario mediante la cual se puede visualizar el resultado del documento generado.	"Interfaz de usuario" en la página 54
`Constructores`	Objetos producidos por el documento generado.	"Uso de constructores"

Ejemplo

```
<DocumentOutput id="informeestadoweb">
     <Containers>
          <Container name="datosinformeestadoweb" />
     </Containers>
     <UserInterface >
          <Pestañas>
               <Tab label="Advanced" labelKey="advancedTab.LABEL" >
                    <TextBrowserPanel container="datosinformeestadoweb" textFormat="html" />
               </Tab>
          </Tabs>
     </UserInterface>
</DocumentOutput>
```

Uso de constructores

Un elemento `Constructors` puede aparecer en uno de los dos lugares del archivo de especificación:

- En la sección Ejecución de una definición de nodo (para objetos de resultado de modelo o documentos)
- En una definición de resultados de modelo (para un nodo aplicador de modelos)

Un nodo sólo puede generar un objeto de resultado. Esta limitación es obligatoria para mantener la coherencia entre los nodos existentes y los scripts y las interfaces API del cliente en este tipo de nodos.

Formato

El formato del elemento `Constructors` cuando se utiliza en una sección Execution es:

```
<Constructors>
     <CreateModelOutput ... />
      ...
     <CreateDocumentOutput ... />
      ...
     <CreateInteractiveModelBuilder ... />
      ...
</Constructors>
```

En una definición de resultados de modelo, el formato es:

```
<Constructors>
     <CreateModelApplier ... />
</Constructors>
```

Creación de resultado de modelo

Esta sección define cómo se crea un objeto de resultado de modelo en la pestaña Modelos o un objeto de resultado de documento en la pestaña Resultados.

Formato

```
<CreateModelOutput type="id_objeto_resultado">
     <Condition ... ./>
     <And>
     <Or>
     <Not>
     <CreateModel type="id_modelo" target="id_contenedor" sourceFile="id_archivo_contenedor" />
      ...
     <CreateDocument type="id_documento" target="id_contenedor" sourceFile="id_archivo_contenedor" />
      ...
</CreateModelOutput>
```

En el elemento `CreateModelOutput`:

- `tipo` (obligatorio) es el identificador de un objeto de resultado de modelo que se define en una sección de resultado de modelo. Para obtener más información, consulte el tema "Resultado de modelo" en la página 50.

La sección `Condition` le permite especificar una o más condiciones. Para obtener más información, consulte el tema "Condiciones" en la página 72.

También puede especificar condiciones complejas que incluyen los operadores `And`, `Or` y `Not`. Para obtener más información, consulte el tema "Operadores lógicos" en la página 71.

En los elementos `CreateModel` y `CreateDocument`:

- `type` es el identificador del modelo o documento que se está definiendo.
- `target` (obligatorio) es el identificador del contenedor del modelo; este contenedor se define en una sección de resultado de modelo. Para obtener más información, consulte el tema "Resultado de modelo" en la página 87.
- `sourceFile` (obligatorio) es el identificador de un archivo de salida generado durante la ejecución del nodo; este archivo se define en una sección de archivos de resultados. Para obtener más información, consulte el tema "Archivos de resultados" en la página 56.

Ejemplo

```
<Constructors>
     <CreateModelOutput type="naivebayes">
          <CreateModel type="modelo_naivebayes" target="modelo" sourceFile="pmml"/>
          <CreateDocument type="resultado_html" target="resultado_avanzado" sourceFile="resultadohtml" />
          <CreateDocument type="Tiporesultado_zip" target="resultado_zip" sourceFile="resultadozip" />
     </CreateModelOutput>
</Constructors>
```

Creación de resultado de documento

Este elemento se utiliza en la sección de ejecución de una definición de nodo generador de documentos e identifica el objeto de resultado de documento que se está creando.

Formato

```
<CreateDocumentOutput type="id_objeto_resultado">
     <Condition ... ./>
     <And>
     <Or>
     <Not>
     <CreateModel type="id_modelo" target="id_contenedor" sourceFile="id_archivo_contenedor" />
      ...
     <CreateDocument type="id_documento" target="id_contenedor" sourceFile="id_archivo_contenedor" />
      ...
</CreateModelOutput>
```

donde tipo (obligatorio) es el identificador de un objeto de resultado de documento que se define en una sección de resultado de documento. Para obtener más información, consulte el tema “Resultado de documento” en la página 50.

La sección Condition le permite especificar una o más condiciones. Para obtener más información, consulte el tema “Condiciones” en la página 72.

También puede especificar condiciones complejas que incluyen los operadores And, Or y Not. Para obtener más información, consulte el tema “Operadores lógicos” en la página 71.

En los elementos CreateModel y CreateDocument:

- type es el identificador del modelo o documento que se está definiendo.
- target (obligatorio) es el identificador del contenedor del modelo; este contenedor se define en una sección de resultado de modelo. Para obtener más información, consulte el tema “Resultado de modelo” en la página 87.
- sourceFile (obligatorio) es el identificador de un archivo de salida generado durante la ejecución del nodo; este archivo se define en una sección de archivos de resultados. Para obtener más información, consulte el tema “Archivos de resultados” en la página 56.

Ejemplo

```
<Constructors>
     <CreateDocumentOutput type="informeestadoweb">
          <CreateDocument type="informeestadoweb" target="datosinformeestadoweb"
               sourceFile="archivo_resultado_informeestadoweb" />
     </CreateDocumentOutput>
</Constructors>
```

Creación de un generador de modelos interactivos

Este elemento se utiliza en la sección de ejecución de una definición de nodo generador de modelos interactivos e identifica el objeto de resultado con el que el usuario interactuará. Para obtener más información, consulte el tema “Creación de un generador de modelos interactivos” en la página 89.

Creación de aplicador de modelos

Este elemento se utiliza en un elemento Constructors en la sección de resultados del modelo de una definición de nodo generador de modelos (consulte “Resultado de modelo” en la página 87). El elemento CreateModelApplier define cómo se crea un nodo aplicador de modelos cuando el nodo generador de modelos genera un objeto de resultado de modelo y se suelta en el lienzo.

Formato

```
<CreateModelApplier type="aplicar_identificador_nodo">
     <Condition ... ./>
     <And>
     <Or>
     <Not>
     <CreateModel type="id_modelo" target="id_contenedor" sourceFile="id_archivo_contenedor" />
      ...
     <CreateDocument type="id_documento" target="id_contenedor" sourceFile="id_archivo_contenedor" />
      ...
</CreateModelApplier>
```

En el elemento CreateModelApplier:

- tipo (obligatorio) es el identificador del nodo aplicador de modelos que se va a crear; este nodo se define en un elemento Node ... type="modelApplier" posteriormente en el archivo.

La sección `Condition` le permite especificar una o más condiciones. Para obtener más información, consulte el tema “Condiciones” en la página 72.

También puede especificar condiciones complejas que incluyen los operadores `And`, `Or` y `Not`. Para obtener más información, consulte el tema “Operadores lógicos” en la página 71.

En los elementos `CreateModel` y `CreateDocument`:

- `type` es el identificador del modelo o documento que se está definiendo.
- `target` (obligatorio) es el identificador del contenedor del modelo; este contenedor se define en una sección de resultado de modelo. Para obtener más información, consulte el tema “Resultado de modelo” en la página 87.
- `sourceFile` (obligatorio) es el identificador de un archivo de salida generado durante la ejecución del nodo; este archivo se define en una sección de archivos de resultados. Para obtener más información, consulte el tema “Archivos de resultados” en la página 56.

Ejemplo

En el ejemplo siguiente, el elemento `CreateModelApplier` contiene una referencia al nodo aplicador de modelos denominado `miaplicador`. Este nodo se define en el elemento `Node` siguiente.

```
<ModelOutput>
     <Constructors>
          <CreateModelApplier type="miaplicador"></CreateModelApplier>
     </Constructors>
</ModelOutput>
<Node id="miaplicador" type="modelApplier">
 ...
</Node>
```

Capítulo 6. Generación de interfaces de usuario

Acerca de interfaces de usuario

Cuando añada un nuevo nodo de CLEF, debe definir la forma en que el usuario final interactuará con el nodo y cualquier salida de modelo, salida de documento o nodos de aplicador que especifique la extensión. La interfaz de usuario de cada objeto se define en una sección `UserInterface` del archivo de especificación para la extensión. Este capítulo ofrece una descripción detallada de la sección `UserInterface`.

Note: Puede haber más de una sección `UserInterface` en un único archivo de especificación, dependiendo de los tipos de objeto que se definen en el archivo.

La interfaz de usuario a un objeto de CLEF consiste en uno o más de los siguientes elementos:

- Entrada de menú
- Entrada de paleta
- Icons
- Una o varias ventanas de cuadro de diálogo
- Una o varias ventanas de resultados

Las **entradas de menú** y las **entradas de paleta** ofrecen acceso al objeto desde el sistema de menú y las paletas de nodo de IBM SPSS Modeler respectivamente. Los **iconos** identifican objetos de los menús, en el lienzo de dibujo y en las distintas paletas de nodo. Las **ventanas de cuadro de diálogo** contienen pestañas y controles que permiten a los usuarios especificar varias opciones antes de ejecutar una ruta, además de especificar el resultado que se generará cuando se complete la ejecución. Las **ventanas de resultados** se utilizan para mostrar los resultados de un modelo (como los resultados de aplicar un modelo a un conjunto de datos) o de un documento (como un gráfico).

CLEF le permite añadir nuevos tipos de objetos a los que IBM SPSS Modeler ofrece de forma estándar, y permite definir de forma coherente la interfaz de usuario de acuerdo con estos nuevos objetos. “Diseño de iconos de nodos” en la página 14 y “Cuadros de diálogo de diseño” en la página 18 ofrecen directrices de creación de iconos y cuadros de diálogo en CLEF.

Los **iconos** toman la forma de gráficos que identifican un nodo en particular y aparecen dentro de las formas geométricas que identifican el tipo de nodo.

Las **ventanas de resultados** y **cuadros de diálogo** tienen las siguientes características:

- Barra de título que contiene un icono en miniatura y el título del objeto
- Barra de menú, que contiene:
  - Menús
  - Botones de acción específicos de los objetos
  - Botones de acciones comunes (por ejemplo, Maximizar o Ayuda)
- Área del contenido principal
- Múltiples pestañas para organizar los componentes de la interfaz de usuario en grupos lógicos
- Cambiar capacidades

Una pestaña puede cambiar el área de contenido principal de una ventana de muchas formas distintas. En una ventana de cuadro de diálogo, las diferentes pestañas muestran distintos conjuntos de controles para las propiedades de objeto. Los controles pueden modificarse, y los resultados se aplicarán al objeto subyacente cuando el usuario pulse en los botones **Aplicar** o **Aceptar**.

En el caso de las ventanas de resultados, las pestañas permiten al usuario realizar diferentes acciones relacionadas con el resultado, como mostrar un resumen de los resultados o añadirles anotaciones.

Sección de interfaz de usuario

La interfaz de usuario de cada objeto se define en el elemento `UserInterface` de la definición del objeto, dentro del archivo de especificación. Puede haber más de un elemento `UserInterface` en el mismo archivo de especificación, dependiendo de cuántos objetos se definan en el archivo (por ejemplo, nodo generador de modelos, objeto de resultado de modelo o nodo aplicador de modelos).

En cada sección de interfaz de usuario puede definir:

- Iconos para mostrarlos en el lienzo o las paletas
- Controles (menú personalizado y elementos de la barra de herramientas) para que aparezcan en las ventanas de resultados o cuadros de diálogo
- Pestañas que definan conjuntos de controles de propiedades (para ventanas de resultados o cuadros de diálogo)

Note: En las definiciones de elemento de las siguientes secciones (por lo general identificadas mediante la cabecera **Formato**), los atributos de elemento y los elementos hijo son opcionales a no ser que se indiquen como "(obligatorios)". Si desea conocer la sintaxis completa de todos los elementos, consulte “Esquema XML de CLEF”, en la página 203.

En caso de cada interfaz de usuario, puede especificar el procesamiento que se llevará a cabo. Puede realizarlo ya sea mediante un controlador de acción o un atributo de clase de marco, siendo ambos opcionales. Si no se especifica ningún controlador de acción ni ningún atributo de clase de marco, el procesamiento que se realizará se especificará en otro lugar del archivo.

Formato

El formato básico del elemento `UserInterface` es:

```
<UserInterface >
     <Icons>
          <Icon ... />
           ...
     </Icons>
     <controles>
          <Menu ... />
          <MenuItem ... />
           ...
          <ToolbarItem ... />
           ...
     </Controls>
     <Pestañas>
          <Tab ... />
           ...
     </Tabs>
</UserInterface>
```

Los **controladores de acción** se utilizan para añadir acciones personalizadas a ventanas estándar de IBM SPSS Modeler. El controlador de acción especifica la clase de Java que se activa cuando un usuario selecciona una opción de menú personalizada o un botón de la barra de herramientas en un cuadro de diálogo de nodo, una ventana de resultados de modelo o una ventana de resultados de documento. Se trata de una implementación de una clase `ExtensionObjectFrame` o `ActionHandler`. En cualquier caso, automáticamente se incluyen los componentes estándar de la ventana, como son los botones de barra de herramienta, menús y pestañas estándar. Para obtener más información, consulte el tema “Clases de API de cliente” en la página 175.

El formato de un controlador de acción es el mismo que el formato básico excepto en la primera línea:

```
<UserInterface actionHandler="clase_Java" >
      ...
</UserInterface>
```

donde `actionHandler` es el nombre de la clase de Java de controlador de acción.

Las **clases de marco** se utilizan para los objetos de resultados de documento o de modelo, donde la expresión ofrece su propia ventana en vez de personalizar una ventana de IBM SPSS Modeler estándar. Una clase de marco es una clase de Java que especifica totalmente toda la ventana y su procesamiento. Ningún componente de ventana estándar se incluye automáticamente, por lo que la clase debe especificarlos de forma individual. Las clases de marca sólo pueden utilizarse para objetos de resultados de modelo o documento en vez de para nodos, que siempre utilizan cuadros de diálogo de IBM SPSS Modeler. Para obtener más información, consulte el tema "Ventanas de resultados personalizadas" en la página 162.

El formato de una clase de marco es sencillo:

```
<UserInterface frameClass="clase_Java" />
```

donde `frameClass` es el nombre de clase de marco de un objeto de resultado de modelo o documento. Cualquier icono, control y pestaña se especifica mediante la propia clase de marco, por lo que los elementos no se emplean en este formato.

Los elementos hijo de un elemento `UserInterface` se describen en las siguientes secciones.

Ejemplos

El primer ejemplo muestra la interfaz de usuario de un nodo generador de modelos:

```
<UserInterface actionHandler="com.spss.myextension.MyActionHandler">
     <Icons>
          <Icon type="standardNode" imagePath="images/lg_discriminant.gif" />
          <Icon type="smallNode" imagePath="images/sm_discriminant.gif" />
     </Icons>
     <Tabs defaultTab="1">
      ...
     </Tabs>
</UserInterface>
```

La sección correspondiente de un objeto de resultado de modelo es:

```
<UserInterface >
     <Icons>
          <Icon type="standardWindow" imagePath="images/browser_discriminant.gif" />
     </Icons>
     <Pestañas>
          <Tab label="Advanced" labelKey="advancedTab.LABEL"
           helpLink="discriminant_output_advancedtab.htm">
               <ExtensionObjectPanel id="DiscriminantPanel"
                panelClass="com.spss.clef.discriminant.DiscriminantPanel"/>
          </Tab>
     </Tabs>
</UserInterface>
```

La sección de interfaz de usuario de un nodo aplicador de modelos tendría el siguiente aspecto:

```
<UserInterface >
     <Icons>
          <Icon type="standardNode" imagePath="images/lg_gm_discriminant.gif" />
```

```
          <Icon type="smallNode" imagePath="images/sm_gm_discriminant.gif" />
     </Icons>
     <Pestañas>
          <Tab label="Advanced" labelKey="advancedTab.LABEL"
           helpLink="discriminant_output_advancedtab.htm">
               <ExtensionObjectPanel id="DiscriminantPanel"
                panelClass="com.spss.clef.discriminant.DiscriminantPanel"/>
          </Tab>
     </Tabs>
</UserInterface>
```

Icons

Esta sección define los iconos asociados al objeto.

En el caso de los nodos, esta sección debe definir dos elementos `Icon`:

- Una versión mayor para mostrarla en el lienzo
- Una versión más reducida para mostrarla en la paleta

En el caso de los resultados de documentos y los resultados de modelos, define el icono de miniatura que aparece en la esquina superior izquierda del marco de ventana.

"Diseño de iconos de nodos" en la página 14 ofrece directrices de creación de iconos en CLEF.

Formato

```
<Icons>
     <Icon type="tipo_icono" imagePath="ruta_imagen" />
      ...
</Icons>
```

donde:

`type` (obligatorio) es uno de los tipos de icono que se muestran en la tabla siguiente.

Tabla 33. Tipos de icono.

Tipo	Ejemplo	Descripción
`standardNode`	 *Figura 36. Icono de nodo estándar*	Icono de tamaño grande (49 x 49 píxeles) que se muestra en la forma de los nodos en el lienzo.
`smallNode`	 *Figura 37. Icono de nodo pequeño*	Icono de tamaño más reducido (38 x 38 píxeles) que se muestra en la forma de nodo en la paleta de nodo.

Tabla 33. Tipos de icono (continuación).

Tipo	Ejemplo	Descripción
`window`	 *Figura 38. Icono de ventana*	El icono de miniatura (16 x 16 píxeles) se muestra en una ventana de resultados o navegador.

`imagePath` (obligatorio) identifica la ubicación de la imagen empleada en este icono. La ubicación proporcionada es relativa al directorio en el que se instala el archivo de especificación.

Ejemplos

```
<Icon type="Nodoestándar" imagePath="images/lg_minodo.gif" />
<Icon type="smallNode" imagePath="images/sm_mynode.gif" />
<Icon type="window" imagePath="images/mynode16.gif" />
```

Controles

Esta sección define elementos de menús personalizados y barras de herramientas que se emplean para invocar acciones declaradas en la sección de objetos comunes del archivo de especificación. Para obtener más información, consulte el tema "Acciones" en la página 40.

Formato

```
<controles>
     <Menu ... />
      ...
     <MenuItem ... />
      ...
     <ToolbarItem ... />
      ...
</Controls>
```

Los elementos de `Menu`, `MenuItem` y `ToolbarItem` se describen en las siguientes secciones.

Ejemplo

El siguiente ejemplo añade un elemento al menú Generar y a la barra de herramientas del cuadro de diálogo en el que se incluye la especificación. Ambos elementos implementan una acción definida previamente denominada `generateDerive` que genera un nodo Derivar.

```
<controles>
     <MenuItem action="generateDerive" systemMenu="generate"/>
     <ToolbarItem action="generateDerive" showLabel="false"/>
      ...
</Controls>
```

Menús

Puede definir un menú personalizado para añadirlo a uno de los menús estándar.

Formato

```
<Menu id="nombre" label="etiqueta_visualización" labelKey="clave_etiqueta" systemMenu="nombre_menú"
showLabel="true_false" showIcon="true_false" separatorBefore="true_false" separatorAfter="true_
false" offset="entero" mnemonic="mnemonic_char" mnemonicKey="mnemonic_key"/>
```

donde:

`id` (obligatorio) es un identificador exclusivo para el menú que está añadiendo.

`label` (obligatorio) es el nombre de visualización del menú tal y como aparece en la interfaz de usuario (siempre y cuando `showLabel` se haya definido como `true`).

`labelKey` identifica la etiqueta con fines de localización.

`systemMenu` (obligatorio) identifica el menú estándar al que se añade el menú personalizado. El valor de *nombre_menú* será uno de los siguientes:

- `file`
- `edición`
- `insertar*`
- `ver*`
- `herramientas*`
- `ventana*`
- `generar`
- `ayuda*`
- `file.project`
- `file.outputs`
- `file.models`
- `edit.stream`
- `edit.node`
- `edit.outputs`
- `edit.models`
- `edit.project`
- `tools.repository`
- `tools.options`
- `tools.streamProperties`

* sólo válido si se añade a la ventana principal de IBM SPSS Modeler

`showLabel` especifica si la etiqueta de visualización del elemento debe mostrarse en la interfaz de usuario.

`showIcon` especifica si el icono asociado al elemento debe mostrarse en la interfaz de usuario.

`separatorBefore` especifica si debe añadirse un separador (por ejemplo, una barra horizontal para elementos de menú o un espacio para botones de la barra de herramientas) al menú delante de este nuevo elemento.

`separatorAfter` especifica si debe añadirse un separador al menú detrás de este nuevo elemento.

`offset` es un número entero no negativo que define la posición del nuevo elemento; por ejemplo, un desplazamiento de 0 lo añade como el primer elemento (el valor predeterminado es añadirlo al final).

`mnemonic` es el carácter alfabético utilizado junto con la tecla Alt que activa el control (por ejemplo, si proporciona el valor S, el usuarios puede activar este control mediante Alt-S).

mnemonicKey identifica el atributo mnemonic con fines de localización. Si no utiliza mnemonic ni mnemonicKey, no habrá ningún atributo mnemonic disponible para este control. Para obtener más información, consulte el tema "Claves de acceso y atajos de teclado" en la página 114.

Elementos de menú

Puede definir un menú personalizado para añadirlo a uno de los menús estándar o personalizados.

Formato

```
<MenuItem action="identificador" systemMenu="nombre_menú" customMenu="nombre_menú"
     showLabel="true_false" showIcon="true_false" separatorBefore="true_false"
     separatorAfter="true_false" offset="entero" />
```

donde:

action (obligatorio) es el identificador de una acción que se define en la sección de objetos comunes y especifica la acción que realizará este elemento de menú.

systemMenu especifica que el elemento debe aparece en el menú estándar *nombre_menú*, que es uno de los siguientes:

- file
- edición
- insertar*
- ver*
- herramientas*
- ventana*
- generar
- ayuda*
- file.project
- file.outputs
- file.models
- edit.stream
- edit.node
- edit.outputs
- edit.models
- edit.project
- tools.repository
- tools.options
- tools.streamProperties

* sólo válido si se añade a la ventana principal de IBM SPSS Modeler

customMenu es un identificador de un elemento Menu y especifica que el elemento de menú aparezca en este menú personalizado.

showLabel especifica si la etiqueta de visualización del elemento debe mostrarse en la interfaz de usuario.

showIcon especifica si el icono asociado al elemento debe mostrarse en la interfaz de usuario.

separatorBefore especifica si debe añadirse un separador (por ejemplo, una barra horizontal para elementos de menú o un espacio para botones de la barra de herramientas) al menú delante de este nuevo elemento.

separatorAfter especifica si debe añadirse un separador al menú detrás de este nuevo elemento.

offset es un número entero no negativo que define la posición del nuevo elemento; por ejemplo, un desplazamiento de 0 lo añade como el primer elemento (el valor predeterminado es añadirlo al final).

Ejemplo

```
<MenuItem action="generateSelect" systemMenu="generate" showIcon="true"/>
```

Elementos de la barra de herramientas

Puede definir un elemento de barra de elementos personalizada para una ventana de cuadro de diálogo o resultados.

Formato

```
<ToolbarItem action="acción" showLabel="true_false" showIcon="true_false"
     separatorBefore="true_false" separatorAfter="true_false" offset="entero" />
```

donde:

action (obligatorio) es el identificador de una acción que se define en la sección de objetos comunes y especifica la acción que realizará este elemento de barra de herramientas.

showLabel especifica si la etiqueta de visualización del elemento debe mostrarse en la interfaz de usuario.

showIcon especifica si el icono asociado al elemento debe mostrarse en la interfaz de usuario.

separatorBefore especifica si debe añadirse un separador (por ejemplo, una barra horizontal para elementos de menú o un espacio para botones de la barra de herramientas) al menú delante de este nuevo elemento.

separatorAfter especifica si debe añadirse un separador al menú detrás de este nuevo elemento.

offset es un número entero no negativo que define la posición del nuevo elemento; por ejemplo, un desplazamiento de 0 lo añade como el primer elemento (el valor predeterminado es añadirlo al final).

Ejemplo

```
<ToolbarItem action="generateDerive" showLabel="true"/>
```

Ejemplo: Adición a la ventana principal

Es un ejemplo de un archivo de especificaciones que añade un nuevo elemento al menú Herramientas en la ventana principal. No define ningún objeto estándar (por ejemplo nodo, ventana de resultados de modelo o ventana de resultados de documento) pero cuenta con un elemento UserInterface en el nivel superior Extension, lo que significa un cambio de la ventana principal. Todas las demás secciones de UserInterface deben aparecer en una de las definiciones de objetos estándar y afectarían a los cuadros de diálogo asociados a estos objetos.

```
<?xml version="1.0" encoding="UTF-8"?>
<Extension version="1.0">
     <ExtensionDetails providerTag="example" id="main_window" label="Main Window" version="1.0"
          provider="IBM Corp." copyright="(c) 2005-2006 IBM Corp."
          description="Extensión de ejemplo que se conecta con la ventana principal"/>
     <Resources/>
     <CommonObjects>
          <Acciones>
               <Action id="customTool1" label="Herramienta personalizada..." labelKey="customTool.LABEL"
                    imagePath="images/generate.gif" description="Invoca la herramienta personalizada"
                    descriptionKey="customTool.TOOLTIP"/>
```

```
            <Action id="customTool2" label="Herramienta personalizada..." labelKey="customTool.LABEL"
                imagePath="images/generate.gif" description="Invoca la herramienta personalizada"
                descriptionKey="customTool.TOOLTIP"/>
        </Actions>
    </CommonObjects>
    <UserInterface actionHandler="com.spss.cleftest.MainWindowActionHandler">
        <controles>
            <Menu systemMenu="tools" id ="toolsExtension" separatorBefore="true"
                label="Elementos extensión" offset="3"/>
            <MenuItem action="customTool2" customMenu="toolsExtension" showIcon="true"/>
            <MenuItem action="customTool2" customMenu="toolsExtension" showIcon="true"/>
            <ToolbarItem action="customTool1" offset="0"/>
        </Controls>
    </UserInterface>
</Extension>
```

El efecto de esta acción es añadir un nuevo submenú denominado **Extension Items** al menú Herramientas. Este nuevo submenú tiene una única entrada denominada **Herramienta personalizada**.

Puede probar este ejemplo guardando el código XML en un archivo llamado *extension.xml* y añadiendo la extensión a CLEF. Para obtener más información, consulte el tema "Prueba de una extensión de CLEF" en la página 199.

Pestañas

La sección `Tabs` define las pestañas que pueden aparecer en:

- El cuadro de diálogo que aparece cuando el usuario abre un nodo en el lienzo
- Una ventana de resultados de modelo
- Una ventana de resultados de documento

Cada sección `Tabs` puede contener múltiples elementos `Tab`, cada uno de los cuales declara que se visualice una pestaña personalizada:

```
<Tabs defaultTab="entero">
    <Tab ... />
    <Tab ... />
     ...
</Tabs>
```

donde `defaultTab` es un número entero no negativo que especifica qué pestaña debe mostrarse cuando la ventana o el cuadro de diálogo de nodo se abre por primera vez en una ruta. Si el usuario selecciona una pestaña diferente, cuando se cierre y se vuelva a abrir el cuadro de diálogo o la ventana con la ruta activa, la pestaña visualizada más recientemente se mostrará en lugar de la que aparece de forma predeterminada. La numeración de la pestaña comienza en 0.

Tenga en cuenta que pueden incluirse automáticamente otras pestañas en el cuadro de diálogo o ventana. Por ejemplo, todos los cuadros de diálogo y ventanas de resultados pueden incluir una pestaña **Anotaciones** y todos los cuadros de diálogo de nodo de origen de datos incluyen las pestañas **Filtro** y **Tipos**.

Un elemento `Tab` debe declarar la etiqueta de la pestaña (el texto que aparece en la propia pestaña) y también debe incluir una clave de etiqueta para que se utilice como búsqueda si la etiqueta de texto se va a traducir.

Dentro de cada elemento `Tab` hay una especificación de panel, que define la forma en que se presenta el área de contenido principal de la pestaña. Cada especificación del panel puede ser de cualquiera de los siguientes tres tipos: procesador de texto, objeto de extensión o propiedades. Para obtener más información, consulte el tema "Especificaciones de panel" en la página 116.

El formato de un elemento Tab individual es:

```
<Tab id="identificador" label="etiqueta_visualización" labelKey="clave_etiqueta" helpLink="ID_ayuda"
          mnemonic="mnemonic_char" mnemonicKey="mnemonic_key" >
     <TextBrowserPanel ... />
     <ExtensionObjectPanel ... />
     <PropertiesPanel ... />
     <ModelViewerPanel ... />
</Tab>
```

donde:

id es un identificador exclusivo que puede utilizarse para hacer referencia a la pestaña en código Java.

label (obligatorio) es el nombre de visualización de la pestaña tal y como aparece en la interfaz de usuario.

labelKey identifica la etiqueta con fines de localización.

helpLink es el identificador de un tema de ayuda que aparece cuando el usuario invoca el sistema de ayuda, en caso de que lo haya. El formato del identificador depende del tipo de sistema de ayuda (consulte "Definición de la ubicación y el tipo del sistema de ayuda" en la página 163):

HTML help: URL del tema de ayuda

JavaHelp: ID del tema

mnemonic es el carácter alfabético utilizado junto con la tecla Alt que activa el control (por ejemplo, si proporciona el valor S, el usuarios puede activar este control mediante Alt-S).

mnemonicKey identifica el atributo mnemonic con fines de localización. Si no utiliza mnemonic ni mnemonicKey, no habrá ningún atributo mnemonic disponible para este control. Para obtener más información, consulte el tema "Claves de acceso y atajos de teclado".

Ejemplos

Si desea conocer ejemplos de elementos Tab, consulte las secciones de los diferentes tipos de especificación de panel que pueden contener: "Panel de procesador de texto" en la página 116, "Panel de objeto de extensión" en la página 118, "Panel de propiedades" en la página 119 y "Panel de visor de modelos" en la página 121.

Claves de acceso y atajos de teclado

Como una alternativa al acceso mediante el ratón a características en la interfaz de usuario, puede especificar varias combinaciones de teclas para que los usuarios para acceder a características mediante el teclado.

Claves de acceso

Las claves de acceso son teclas que se pueden utilizar junto con la tecla Alt. En los elementos de menú, pestañas Cuadros de diálogo y diferentes controles del cuadro de diálogo, puede especificar claves de acceso mediante el atributo mnemonic de los siguientes elementos.

Tabla 34. Características en la interfaz de usuario.

Característica	Elemento	Consulte...
Acción de pantalla (por ejemplo, para un elemento de menú)	`acción`	"Acciones" en la página 40
Menú	`menú`	"Menús" en la página 109
Pestaña Cuadro de diálogo	`pestaña`	"Pestañas" en la página 113
Controladores	Varios	"Controladores" en la página 129

Por ejemplo, considere el siguiente menú.

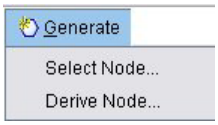

Figura 39. Elementos de menú

Para especificar claves de acceso para este menú debería utilizar el siguiente código:

```
<Acciones>
     <Action id="generateSelect" label="Nodo Seleccionar..." labelKey="generate.selectNode.LABEL"
          imagePath="images/generate.gif" description="Genera un nodo Seleccionar"
          descriptionKey="generate.selectNode.TOOLTIP" mnemonic="S" />
     <Action id="generateDerive" label="Nodo Derivar..." labelKey="generate.deriveNode.LABEL"
          imagePath="images/generate.gif" description="Genera un nodo de derivación"
          descriptionKey="generate.deriveNode.TOOLTIP"  mnemonic="D" />
   ...
</Actions>
```

Proporciona caracteres de guión bajo en los siguientes elementos de menú.

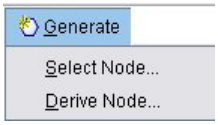

Figura 40. Uso de claves de acceso con los elementos del menú

Los usuarios pueden acceder a los elementos del menú mediante Alt-S y Alt-D, o pulsando con el ratón.

Teclas de acceso directo

Las teclas de acceso directo son combinaciones que permiten a los usuarios finales navegar por la interfaz de usuario. IBM SPSS Modeler proporciona diferentes atajos de teclado como opción estándar. En CLEF, sólo puede añadir los atajos en elementos de menú que haya añadido.

Por ejemplo, para especificar los atajos para los elementos de menú que aparecen en el ejemplo, debe utilizar el siguiente código:

```
<Acciones>
     <Action id="generateSelect" label="Nodo Seleccionar..." labelKey="generate.selectNode.LABEL"
          imagePath="images/generate.gif" description="Genera un nodo Seleccionar"
          descriptionKey="generate.selectNode.TOOLTIP" mnemonic="S" shortcut="CTRL+SHIFT+S" />
     <Action id="generateDerive" label="Nodo Derivar..." labelKey="generate.deriveNode.LABEL"
```

```
        imagePath="images/generate.gif" description="Genera un nodo de derivación"
        descriptionKey="generate.deriveNode.TOOLTIP"  mnemonic="D"  shortcut="CTRL+SHIFT+D" />
    ...
</Actions>
```

Las combinaciones de teclas de acceso directo se añadirán a los elementos de menú.

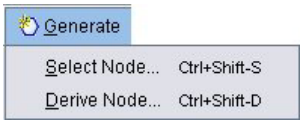

Figura 41. Uso de teclas de acceso directo con los elementos del menú

Los usuarios pueden acceder a los elementos del menú mediante atajos de teclado o pulsando con el ratón. Puede especificar teclas de acceso directo junto con las claves de acceso, como en el ejemplo.

Asegúrese de no utilizar atajos que ya se hayan especificados en el mismo cuadro de diálogo o cualquiera de los atajos estándar de IBM SPSS Modeler.

Especificaciones de panel

Cada elemento `Tab` puede contener la especificación de un único panel, que puede ser de uno de los tipos que se muestran en la tabla siguiente.

Tabla 35. Tipos de especificación de panel

Panel	Muestra...	Consulte...
Panel de procesador de texto	Contenido de texto de un contenedor especificado.	"Panel de procesador de texto"
Panel de objeto de extensión	Contenido definido por una clase de Java especificada.	"Panel de objeto de extensión" en la página 118
Panel de propiedades	Controles de propiedad (por ejemplo, botones, casillas de verificación o campos de entrada).	"Panel de propiedades" en la página 119
Panel de visor de modelos	Resultado de modelo en formato PMML de un contenedor especificado.	"Panel de visor de modelos" en la página 121

Panel de procesador de texto

Un panel de procesador de texto muestra el contenido de texto de un contenedor especificado en la extensión. Los formatos admitidos (codificación UTF-8) son texto plano, HTML y formato de texto enriquecido (RTF).

A continuación se incluye un ejemplo de una ventana de resultados de modelo que contiene un panel de procesador de texto en formato HTML.

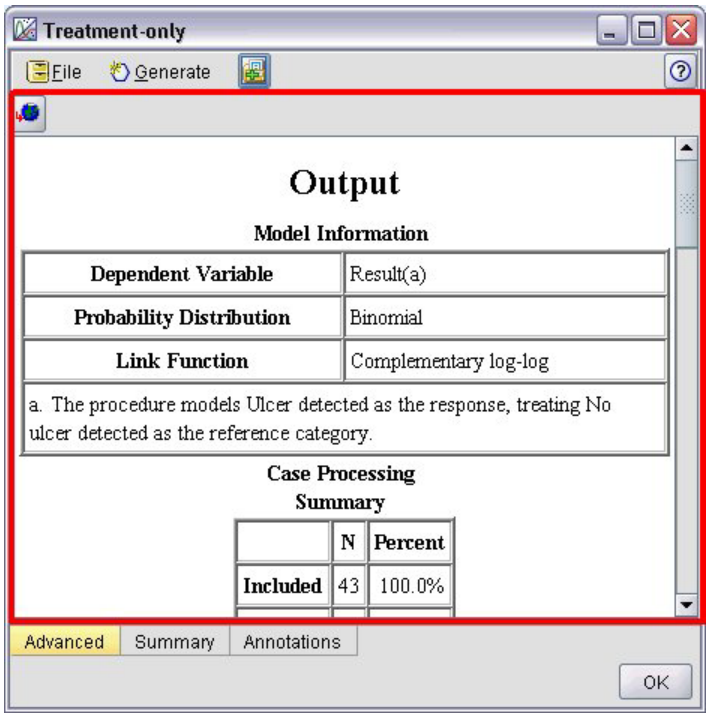

Figura 42. Ventana de resultados de modelo con el panel de procesador de texto resaltado

Formato

```
<TextBrowserPanel container="nombre" textFormat="formato_texto" rows="entero"
     columns="entero" wrapLines="true_false" >
     -- opciones avanzadas de diseño personalizado --
</TextBrowserPanel>
```

donde:

`container` (obligatorio) es el nombre del contenedor desde el que se obtienen los contenidos del panel.

`textFormat` (obligatorio) especifica el formato del texto que aparece en el panel y puede ser:

- `plainText`
- `html`
- `rtf`

`rows` y `columns` especifican el número de filas y columnas de texto que pueden verse cuando se abre la ventana que contiene el panel.

`wrapLines` especifica si utilizar ajuste de línea en líneas de texto largo (`true`) o que sea necesario desplazarse horizontalmente para leer las líneas de texto largo (`false`). El valor predeterminado es `false`.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema “Diseño personalizado avanzado” en la página 153.

Ejemplo

El siguiente ejemplo ilustra la sección de pestaña que define el panel de procesador de texto mostrado anteriormente:

```
<Tab label="Advanced" labelKey="advancedTab.LABEL" helpLink="genlin_output_advancedtab.htm">
     <TextBrowserPanel container="resultados_avanzados" textFormat="html" />
</Tab>
```

Los resultados de modelo se envían a un contenedor que se define en la misma sección `ModelOutput` que la especificación de pestaña:

```
<ModelOutput id="generalizedlinear" label="Generalized Linear">
     <Containers>
      ...
          <Container name="resultados_avanzados"/>
     </Containers>
     <UserInterface >
      ...
          <Pestañas>
               <Tab label="Avanzado" ...>
                    <TextBrowserPanel container="resultados_avanzados" ... />
               </Tab>
          </Tabs>
     </UserInterface>
</ModelOutput>
```

Se hace referencia al contenedor en un elemento `CreateDocument` en la sección de ejecución para el nodo de generación relevante:

```
<Execution>
 ...
     <Constructors>
          <CreateModelOutput type="generalizedlinear">
               <CreateModel type="generalizedlinear_model" target="model" sourceFile="pmml"/>
               <CreateDocument type="resultados_html" target="resultados_avanzados" sourceFile=
                "htmloutput"/>
          </CreateModelOutput>
     </Constructors>
</Execution>
```

Panel de objeto de extensión

Un panel de objeto de extensión funciona de forma similar a un panel de procesador de texto, aunque en vez de mostrar el contenido de texto de un contenedor, crea una instancia de una clase de Java especificada que implementa la interfaz `ExtensionObjectPanel` definida por la API de Java de CLEF.

A continuación se incluye un ejemplo de un cuadro de diálogo de nodo aplicador de modelos que contiene un panel de objeto de extensión.

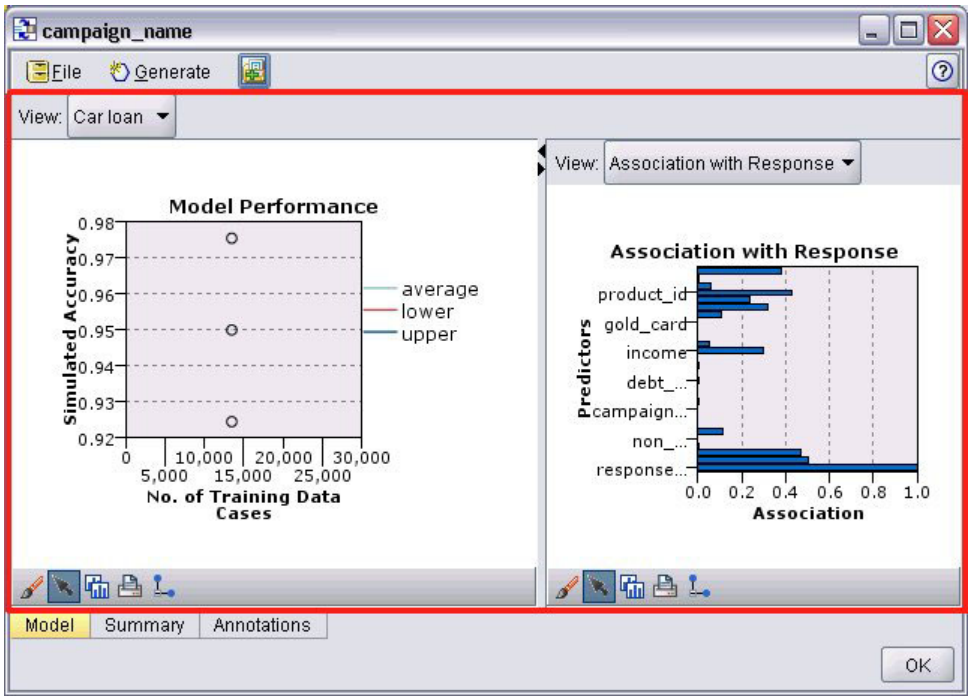

Figura 43. Ventana de resultados de modelo con el panel de objeto de extensión resaltado

Formato

```
<ExtensionObjectPanel id="identificador" panelClass="clase_Java" >
     -- opciones avanzadas de diseño personalizado --
</ExtensionObjectPanel>
```

donde:

`id` es un identificador exclusivo que puede utilizarse para hacer referencia al panel en código Java.

`panelClass` (obligatorio) es el nombre de la clase de Java que implementa el panel.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema “Diseño personalizado avanzado” en la página 153.

Ejemplo

El siguiente ejemplo ilustra la sección de pestaña que define el panel de objeto de extensión mostrado anteriormente:

```
<Tab label="Model" labelKey="Model.LABEL" helpLink="selflearnnode_output.htm">
     <ExtensionObjectPanel id="SelfLearningPanel"
          panelClass="com.spss.clef.selflearning.SelfLearningPanel"/>
</Tab>
```

Panel de propiedades

Un panel de propiedades permite que una pestaña o subpanel de propiedades (consulte “Subpanel de propiedades” en la página 127) muestra **controles de propiedad**, que son componentes de pantalla (como botones, casillas de verificación y campos de entrada) que pueden emplearse para modificar las propiedades de un objeto mostrado en pantalla. El panel de propiedades aplica automáticamente los

cambios realizados con estos controles cada vez que el usuario pulsa en **Aceptar** o **Aplicar**. Cuando el usuario pulsa en **Cancelar** o **Restablecer**, el panel desecha cualquier cambio realizado desde la última operación de aplicación.

A continuación se incluye un ejemplo de un cuadro de diálogo de nodo que contiene un panel de propiedades.

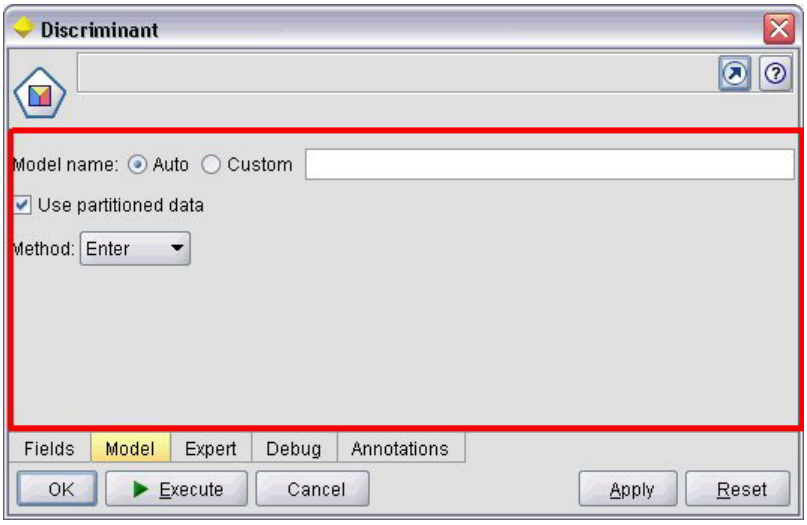

Figura 44. Cuadro de diálogo de nodo con el panel de propiedades resaltado

El siguiente ejemplo ilustra un subpanel de propiedades que contiene tres paneles de propiedades.

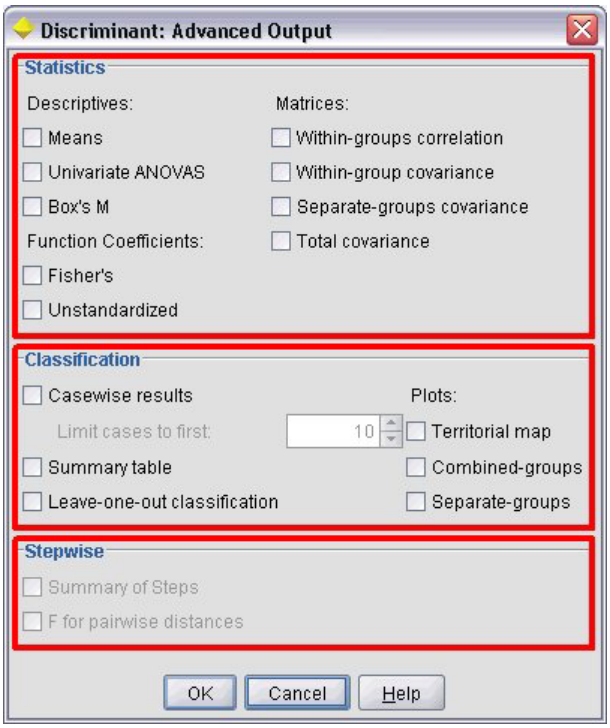

Figura 45. Subpanel de propiedades con los paneles de propiedades resaltados

Formato

```
<PropertiesPanel id="identificador" label="etiqueta_visualización" labelKey="clave_etiqueta">
     -- opciones avanzadas de diseño personalizado --
     -- especificaciones de control de propiedad --
</PropertiesPanel>
```

donde:

`id` es un identificador exclusivo que puede utilizarse para hacer referencia al panel en código Java.

`label` es la cabecera de visualización para un grupo de controles de propiedad (por ejemplo, **Statistics**, **Classification** y **Stepwise** en el último ejemplo).

`labelKey` identifica la etiqueta con fines de localización.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Las especificaciones de control de propiedad individual se describen en "Especificaciones de control de propiedad" en la página 123.

Ejemplo

```
<Tab label="Model" labelKey="Model.LABEL" helpLink="discriminant_node_model.htm">
     <PropertiesPanel>
          <SystemControls controlsId="ModelGeneration" />
          <ComboBoxControl property="method">
               <Layout fill="none" />
          </ComboBoxControl>
     </PropertiesPanel>
</Tab>
```

Panel de visor de modelos

Un panel de visor de modelo muestra cualquier resultado de modelo de formato PMML desde un contenedor especificado en la extensión.

A continuación se incluye un ejemplo de una ventana de nodo aplicador de modelos que contiene un panel de visor de modelo.

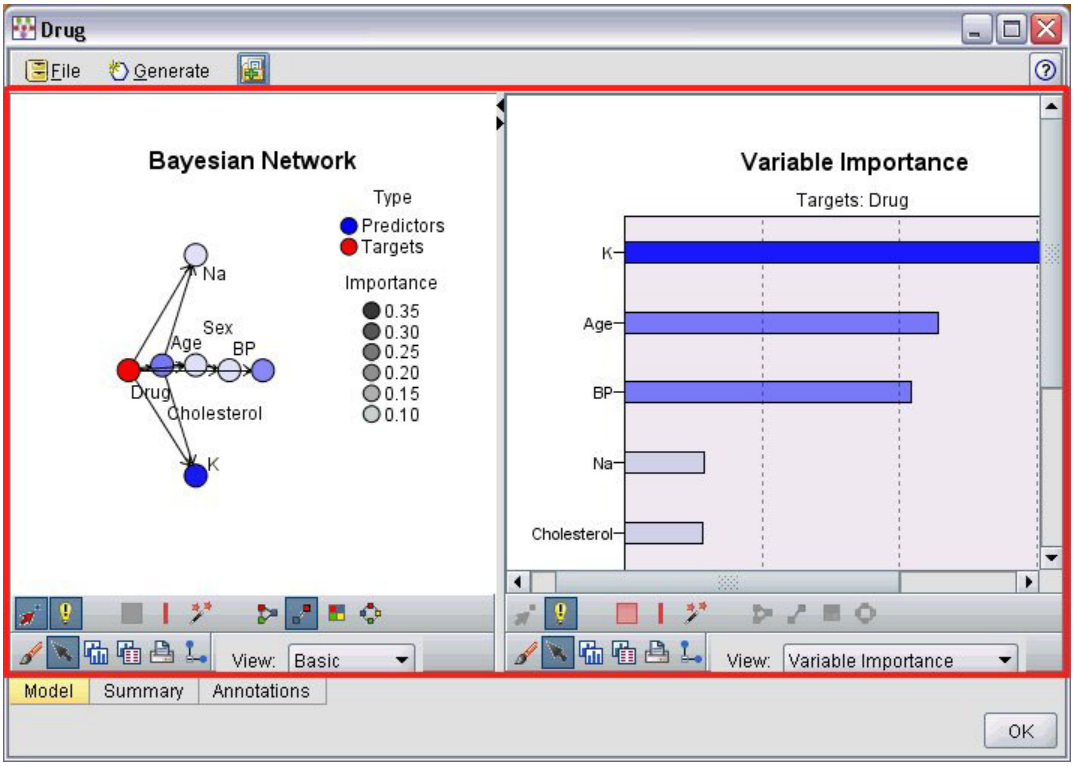

Figura 46. Ventana de resultados con el panel de visor de modelo resaltado

Formato

```
<ModelViewerPanel container="nombre_contenedor">
     -- opciones avanzadas de diseño personalizado --
</ModelViewerPanel>
```

donde `container` (obligatorio) es el nombre del contenedor al que está asignado el resultado de modelo.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

Este ejemplo ilustra el uso de un panel de visor de modelo en una especificación de aplicador de modelos. El resultado de modelo se ha asignado previamente al contenedor denominado `modelo`. Aquí la especificación de aplicador de modelos selecciona este contenedor y lo asocia al panel de visor de modelo:

```
<Node id="applyBN" type="modelApplier">
     <ModelProvider container="model" isPMML="true" />
      ...
     <Containers>
          <Container name="model" />
     </Containers>
     <UserInterface >
      ...
     <Pestañas>
          <Tab label="Model" labelKey="modelTab.LABEL" helpLink="BN_output_modeltab.htm">
               <ModelViewerPanel container="model"/>
          </Tab>
      ...
```

```
      </Tabs>
      </UserInterface>
       ...
</Node>
```

Especificaciones de control de propiedad

Los controles de propiedad son componentes de pantalla como botones, casillas de verificación y campos de entrada que pueden utilizarse para modificar las propiedades de un objeto que aparece en pantalla. El formato de una especificación de control de propiedad depende del tipo de control de propiedad, que puede ser uno de los siguientes:

- Componente de IU
- Panel de propiedades
- Controlador

Los controles **componente de IU** son botones de acción, texto estático de la pantalla y controles de sistema (conjunto de controles que gestionan las propiedades comunes a todos los cuadros de diálogo).

Los controles del **panel de propiedades** son paneles individuales dentro de la especificación del panel de propiedades.

Los **controladores** forman el mayor grupo de controles de propiedad e incluye elementos como casillas de verificación, cuadros combinados y controles de número.

Controles de componentes de IU

Los controles de componentes de IU se muestran en la siguiente tabla.

Tabla 36. Controles de componentes de IU

Control	Descripción
`ActionButton`	Botón de pantalla que ejecuta una acción predefinida cuando se pulsa.
`StaticText`	Cadena de texto no variable que se muestra en la pantalla.
`SystemControls`	Conjuntos estándar de controles que controlan las propiedades comunes a todos los modelos.

Botón de acción

Define un botón de barra de herramientas o cuadro de diálogo que ejecuta una acción especificada en la sección de objetos comunes. La acción (por ejemplo, mostrar una nueva pantalla) se realiza cuando el usuario pulse en este botón.

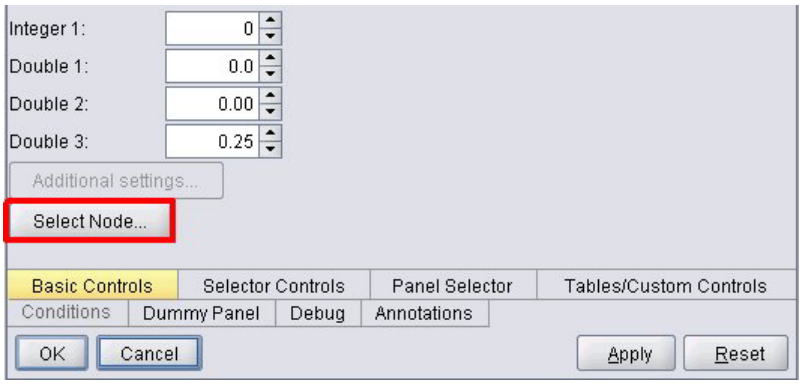

Figura 47. Cuadro de diálogo con el botón de acción resaltado

Formato

```
<ActionButton action="acción" showLabel="true_false" showIcon="true_false" >
     -- opciones avanzadas de diseño personalizado --
</ActionButton>
```

donde:

`action` (obligatorio) es el identificador de la acción que va a realizarse.

`showLabel` especifica si mostrar (`true`) u ocultar (`false`) la etiqueta del botón (por ejemplo, un botón de la barra de herramientas, puede optar por mostrar un icono y no una etiqueta). El valor predeterminado es `true`.

`showIcon` especifica si se debe mostrar (`true`) u ocultar (`false`) un icono asociado con el botón. El valor predeterminado es `false`.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

El código para crear el botón de acción mostrado anteriormente es:

```
<ActionButton action="generateSelect"/>
```

La acción se define en la sección de objetos comunes tal y como sigue (tenga en cuenta que el botón se define aquí también):

```
<CommonObjects extensionListenerClass="com.spss.cleftest.TestExtensionListener"> ...
    <Acciones>
      <Action id="generateSelect" label="Nodo Seleccionar..." labelKey="generate.selectNode.LABEL"
           imagePath="images/generate.gif" description="Genera un nodo Seleccionar"
           descriptionKey="generate.selectNode.TOOLTIP"/>
        ...
    </Actions>
</CommonObjects>
```

Texto estático

Este elemento le permite incluir una cadena de texto no variable en una ventana de resultados o cuadro de diálogo. A continuación se muestra un panel de propiedades con texto estático.

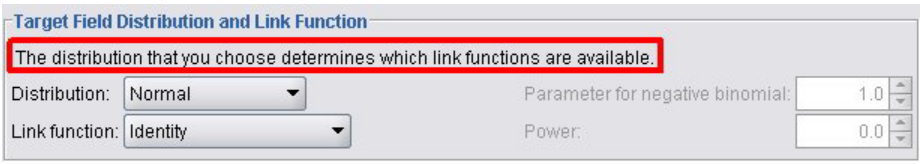

Figura 48. Panel de propiedades con el texto estático resaltado

Formato

```
<StaticText text="texto_estático" textKey="clave_texto" >
     -- opciones avanzadas de diseño personalizado --
</StaticText>
```

donde:

`text` es la cadena de texto que desea utilizar.

`textKey` identifica el texto estático con fines de localización.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

El siguiente ejemplo muestra la declaración empleada para el texto estático mostrado anteriormente:

```
<StaticText text="La distribución que seleccione determinará qué funciones de
enlace están disponibles."
textKey="Genlin_staticText1"/>
```

Controles de sistema

Algunas propiedades son comunes para todos los modelos. En un nodo generador de modelos, los controles estándar son conjuntos estándar de controles que gestionan estas propiedades.

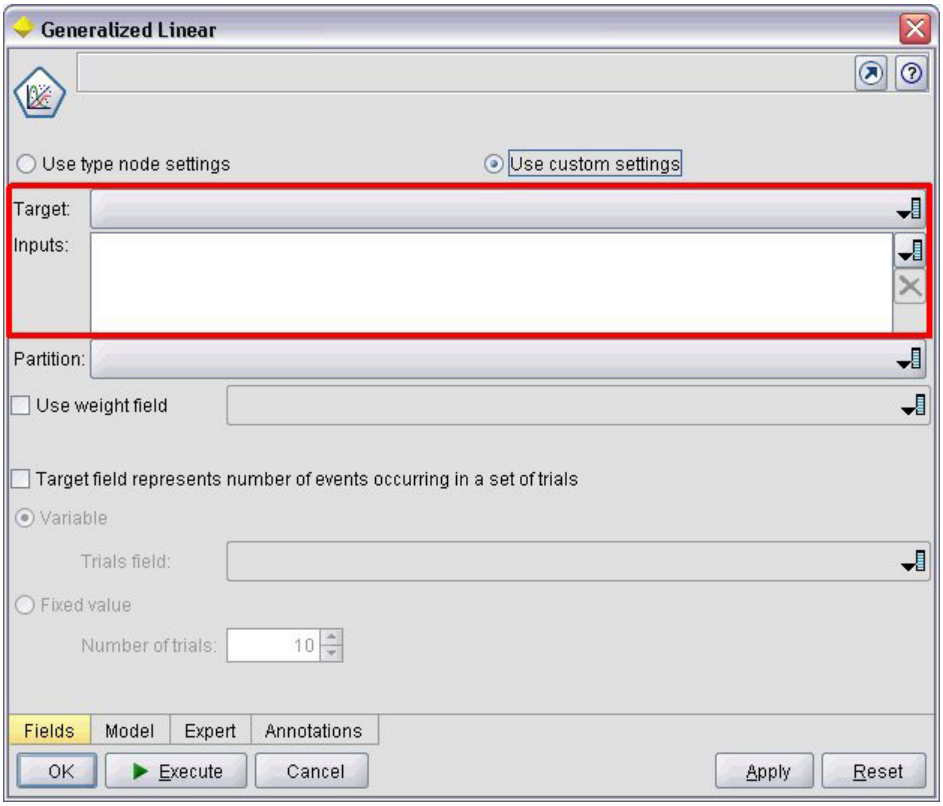

Figura 49. Ejemplo de cuadro de diálogo con los controles de sistema resaltados

Formato

```
<SystemControls controlsID="identificador" >
     -- opciones avanzadas de diseño personalizado --
</SystemControls>
```

donde `controlsID` es el identificador del conjunto de controles. Este identificador debe ser el mismo que uno especificado en el atributo `controlsID` de un elemento `ModelingFields` en una declaración de generador de modelo (consulte "Generador de modelos" en la página 51).

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

Este ejemplo muestra la declaración empleada para los controles de sistema en la ilustración anterior.

En la sección de generador de modelos de la especificación de nodo, lo siguiente define un conjunto de controles de sistema, que en este caso comprende los selectores de campo para los campos de entrada y salida del modelo:

```
<ModelBuilder ...>
     <ModelingFields controlsId="modelingFields">
          <InputFields property="inputs" multiple="true" label="Inputs" types="[range set
orderedSet flag]" labelKey="inputFields.LABEL"/>
          <OutputFields property="target" multiple="false" types="[range flag]"
```

```
label="Target" labelKey="targetField.LABEL"/>
     </ModelingFields>
      ...
</ModelBuilder>
```

Posteriormente en el archivo, se hace referencia a este conjunto de controles en la definición de la pestaña del cuadro de diálogo de nodo generador de modelos en que aparece:

```
<Tab label="Fields" labelKey="Fields.LABEL" helpLink="genlin_node_fieldstab.htm">
     <PropertiesPanel>
          <SystemControls controlsId="modelingFields">
          </SystemControls>
      ...
     </PropertiesPanel>
</Tab>
```

Controles del panel de propiedades

Los controles de panel de propiedades se muestran en la siguiente tabla.

Tabla 37. Controles del panel de propiedades

Control	Descripción
`PropertiesSubPanel`	Cuadro de diálogo diferente que se muestra cuando el usuario pulsa en un botón de un panel de propiedades.
`PropertiesPanel`	Panel de propiedades anidado en una declaración de subpanel de propiedades o anidado en una declaración de panel de propiedades a nivel superior.

Subpanel de propiedades

Define un cuadro de diálogo diferente que se muestra cuando el usuario pulsa en un botón de un panel de propiedades. La declaración del subpanel de propiedades se realiza como parte de la especificación del panel de propiedades principal de una pestaña.

Formato

```
<PropertiesSubPanel buttonLabel="etiqueta_visualización" buttonLabelKey="clave_etiqueta"
     dialogTitle="título_visualización" dialogTitleKey="clave_título" helplink="ID_ayuda"
     mnemonic="mnemonic_char" mnemonicKey="mnemonic_key" >
     -- opciones avanzadas de diseño personalizado --
     -- especificaciones de control de propiedad --
</PropertiesSubPanel>
```

donde:

`buttonLabel`: es la etiqueta del botón que ofrece acceso al subpanel.

`buttonLabelKey` identifica la etiqueta de botón con fines de localización.

`dialogTitle` es el texto que aparece en la barra de título del cuadro de diálogo del subpanel.

`dialogTitleKey` identifica el título del cuadro de diálogo del subpanel con fines de localización.

`helpLink` es el identificador de un tema de ayuda que aparece cuando el usuario invoca el sistema de ayuda, en caso de que lo haya. El formato del identificador depende del tipo de sistema de ayuda (consulte “Definición de la ubicación y el tipo del sistema de ayuda” en la página 163):

HTML help: URL del tema de ayuda

JavaHelp: ID del tema

`mnemonic` es el carácter alfabético utilizado junto con la tecla Alt que activa el control (por ejemplo, si proporciona el valor S, el usuarios puede activar este control mediante Alt-S).

`mnemonicKey` identifica el atributo mnemonic con fines de localización. Si no utiliza `mnemonic` ni `mnemonicKey`, no habrá ningún atributo mnemonic disponible para este control. Para obtener más información, consulte el tema "Claves de acceso y atajos de teclado" en la página 114.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Las especificaciones de control de propiedad individual se describen en "Especificaciones de control de propiedad" en la página 123.

Ejemplo

Lo siguiente muestra un subpanel de propiedades que se muestra cuando el usuario pulsa en el botón **Resultados** en el panel principal de propiedades de una pestaña.

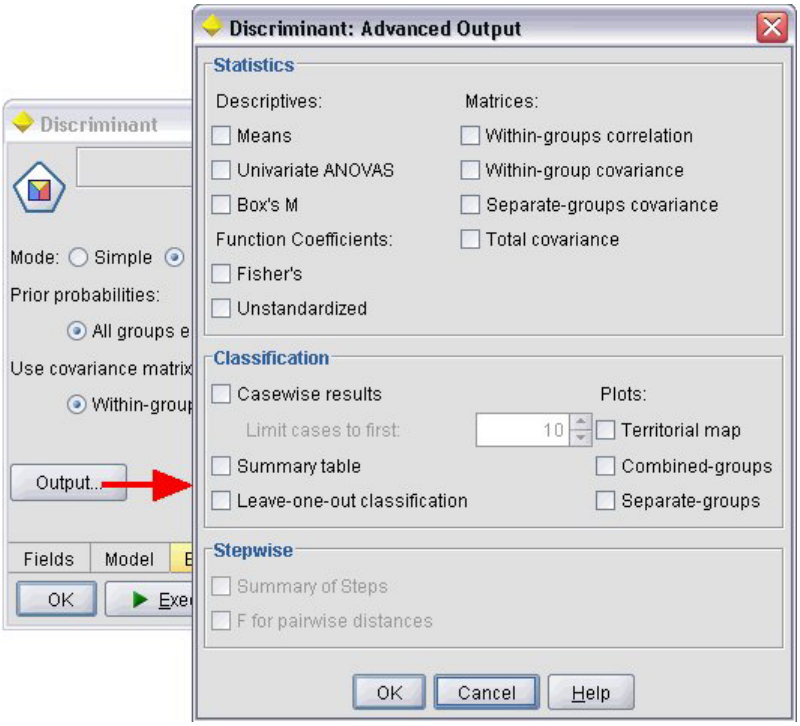

Figura 50. Subpanel de propiedades

El siguiente código muestra las partes más importantes de la declaración utilizada para obtener el subpanel de propiedades ilustrado. Tenga en cuenta cómo, dentro de la declaración de subpanel, cada uno de los grupos de campo (**Statistics**, **Classification** y **Stepwise**) tiene su propia especificación del panel de propiedades:

```
<PropertiesSubPanel buttonLabel="Output..." buttonLabelKey="OutputSubPanel.LABEL"
     dialogTitle="Discriminante: Salida avanzada" dialogTitleKey="AdvancedOutputSubDialog.LABEL"
     helpLink="discriminant_node_outputdlg.htm">
     ...
     <PropertiesPanel>
          <PropertiesPanel label="Statistics" ...>
```

```
          ...
         </PropertiesPanel>
         <PropertiesPanel label="Classification" ...>
          ...
         </PropertiesPanel>
         <PropertiesPanel label="Stepwise" ...>
          ...
         </PropertiesPanel>
     </PropertiesPanel>
</PropertiesSubPanel>
```

Panel de propiedades (anidado)

Puede anidar una especificación de panel de propiedades dentro de una declaración de subpanel de propiedades para definir el contenido del cuadro de diálogo mostrado desde el subpanel. Para obtener más información, consulte el tema "Subpanel de propiedades" en la página 127.

También puede anidar una especificación de panel de propiedades en una declaración de panel de propiedades de nivel superior. Un ejemplo de cuándo puede desear hacerlo sería cuando el contenido de toda una pestaña, que se compone de varios paneles de propiedades, se activan o desactivan en función de si selecciona un botón concreto de la pestaña. En este caso, la especificación de la pestaña tendría un aspecto similar al siguiente:

```
<Tab .... >
     <PropertiesPanel>
      --- especificación de botón ---
         <PropertiesPanel>
              <Activado>
               --- valor del botón que incluye la condición ---
              </Enabled>
          ...
         </PropertiesPanel>
         <PropertiesPanel>
              <Activado>
               --- valor del botón que incluye la condición ---
              </Enabled>
          ...
         </PropertiesPanel>
          ...
     </PropertiesPanel>
</Tab>
```

El formato de una especificación de panel de propiedades anidadas es el mismo que para el elemento de nivel superior. Para obtener más información, consulte el tema "Panel de propiedades" en la página 119.

Controladores

Los controladores forman el mayor grupo de controles de propiedad.

Tabla 38. Controladores

Control	Descripción
`CheckBoxControl`	Casilla de verificación.
`CheckBoxGroupControl`	Conjunto de casillas de verificación, una por cada valor `enum`.
`ClientDirectoryChooserControl`	Campo de texto de una línea y su botón asociado que permite al usuario seleccionar un directorio en el cliente.
`ClientFileChooserControl`	Campo de texto de una línea y su botón asociado que permite al usuario seleccionar un archivo en el cliente.

Tabla 38. Controladores (continuación)

Control	Descripción
`ComboBoxControl`	Lista desplegable de cuadro combinado que contiene los valores `enum`.
`DBConnectionChooserControl`	Permite al usuario seleccionar un origen de datos y conectarse a una base de datos.
`DBTableChooserControl`	Permite al usuario seleccionar una tabla de base de datos tras haberse conectado correctamente a una base de datos.
`MultiFieldChooserControl`	(Sólo nodos) Lista de nombres de campos que permite al usuario seleccionar uno o más campos de la lista.
`MultiItemChooserControl`	Permite al usuario seleccionar uno o más elementos de menú de una lista de valores.
`PasswordBoxControl`	Campo de una línea de texto en el que los caracteres de entrada están ocultos.
`PropertyControl`	Control definible por el usuario para una propiedad.
`RadioButtonGroupControl`	Conjunto de botones de radio en el que sólo es posible seleccionar un botón cada vez. En el caso de las propiedades `enum`, hay un botón de radio por valor `enum`; mientras que en el caso de las propiedades booleanas, se mostrarán dos botones de radio.
`ServerDirectoryChooserControl`	Campo de texto de una línea con un botón asociado que permite al usuario seleccionar un directorio en el servidor.
`ServerFileChooserControl`	Campo de texto de una línea con un botón asociado que permite al usuario seleccionar un archivo en el servidor.
`SingleFieldChooserControl`	(Sólo nodos) Lista de nombres de campos que permite al usuario seleccionar un único campo de la lista.
`SingleItemChooserControl`	Permite al usuario seleccionar un único elemento de una lista de valores.
`SpinnerControl`	Control de número (campo numérico con flechas hacia arriba y hacia abajo que permiten modificar el valor).
`TableControl`	Añade una tabla a un cuadro de diálogo o ventana.
`TextAreaControl`	Campo de texto de múltiples líneas.
`TextBoxControl`	Campo de texto de una única línea.

Atributos de controlador

Las especificaciones del controlador pueden incluir los siguientes atributos.

```
property="valor" showLabel="true_false" label="etiqueta_visualización" labelKey="clave_etiqueta"
labelWidth="ancho_etiqueta" labelAbove="true_false" description="descripción"
descriptionKey="clave_descripción" mnemonic="mnemonic_char" mnemonicKey="mnemonic_key"
```

donde:

`property` (obligatorio) es el identificador exclusivo del control de propiedad.

`showLabel` especifica si se debe mostrar (`true`) u ocultar (`false`) la etiqueta de visualización del control de propiedad. El valor predeterminado es `true`.

`label` es el nombre de visualización del control de propiedad tal y como aparece en la interfaz de usuario. Este valor también actúa como la descripción breve accesible del control de propiedad. Para obtener más información, consulte el tema "Accesibilidad" en la página 173.

`labelKey` identifica la etiqueta con fines de localización.

`labelWidth` es el número de columnas de rejilla de visualización que muestra la etiqueta. El valor por omisión es 1.

`labelAbove` especifica si la etiqueta del control debe aparecer por encima del control (`true`) o a su lado (`false`). El valor predeterminado es `false`.

`description` es el texto de información sobre herramientas que se muestra cuando el puntero del ratón pasa por encima del control. Este valor también actúa como la descripción larga accesible del control de propiedad. Para obtener más información, consulte el tema "Accesibilidad" en la página 173.

`descriptionKey` identifica la descripción con fines de localización.

`mnemonic` es el carácter alfabético utilizado junto con la tecla Alt que activa el control (por ejemplo, si proporciona el valor S, el usuarios puede activar este control mediante Alt-S).

`mnemonicKey` identifica el atributo mnemonic con fines de localización. Si no utiliza `mnemonic` ni `mnemonicKey`, no habrá ningún atributo mnemonic disponible para este control. Para obtener más información, consulte el tema "Claves de acceso y atajos de teclado" en la página 114.

Control de casilla de verificación

Define una casilla de verificación.

Figura 51. Casillas de verificación

Formato

```
<CheckBoxControl atributos_controlador invert="true_false" >
     -- opciones avanzadas de diseño personalizado --
</CheckBoxControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`invert` se utiliza muy raramente pero, si se define como `true`, invierte el efecto de la selección y cancelación de la casilla de verificación. El valor predeterminado es `false`.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

El siguiente ejemplo muestra el código empleado para distribuir las casillas de verificación mostradas anteriormente (las etiquetas de casilla de verificación se definen en otro lugar del archivo de especificación). El elemento `Layout` se describe en "Diseño personalizado avanzado" en la página 153.

```
<CheckBoxControl property="means">
     <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<CheckBoxControl property="within_groups_correlation">
     <Layout gridColumn="2" />
```

```
</CheckBoxControl>
<CheckBoxControl property="univariate_anovas">
     <Layout gridWidth="1" rowIncrement="0" />
</CheckBoxControl>
<CheckBoxControl property="within_group_covariance">
     <Layout gridColumn="2" />
</CheckBoxControl>
<CheckBoxControl property="box_m">
     <Layout gridWidth="1" rowIncrement="0" />
</CheckBoxControl>
<CheckBoxControl property="separate_groups_covariance">
     <Layout gridColumn="2" />
</CheckBoxControl>
```

Control de grupo de casillas de verificación

Define un conjunto de casillas de verificación agrupadas y tratadas como una única unidad. Esto sólo puede utilizarse en conjunto con una propiedad de lista enumerada que define los miembros del grupo.

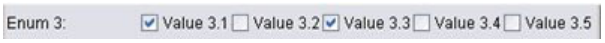

Figura 52. Grupo de casillas de verificación

Formato

```
<CheckBoxGroupControl atributos_controlador rows="entero" layoutByRow="true_false"
     useSubPanel="true_false" >
     -- opciones avanzadas de diseño personalizado --
</CheckBoxGroupControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`rows` es un número entero positivo que especifica el número de filas de la pantalla que ocupará el grupo de casillas de verificación. El valor por omisión es 1.

`layoutByRow` especifica si las casillas de verificación deben disponerse en primer lugar a lo largo de la fila (`true`) o de la columna (`false`). El valor predeterminado es `true`. Si desea conocer un uso similar de `layoutByRow` con un grupo de botón de radio, consulte "Modificación del orden de los controles" en la página 152.

`useSubPanel` especifica si las casillas de verificación van a mostrarse (`true`) o no (`false`) como subpanel. El valor predeterminado es `true`.

Los grupos de casillas de verificación suelen disponerse como un subpanel que contiene todas las casillas del grupo. Sin embargo, se pueden producir problemas de alineación si el grupo de casillas de verificación se asocia con un campo de texto adyacente. Este problema se evita si `useSubPanel` se define como `false`.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

El código para crear el grupo de casillas de verificación mostrado anteriormente es:

```
<CheckBoxGroupControl property="enum3" label="Enum 3" labelKey="enum3.LABEL"/>
```

Las etiquetas y valores asociados con las casillas de verificación individuales se definen en la sección Propiedades del nodo pertinente:

```
<Property name="enum3" valueType="enum" isList="true" defaultValue="[value1 value3]">
     <Enumeration>
          <Enum value="value1" label="Value 3.1" labelKey="enum3.value1.LABEL"/>
          <Enum value="value2" label="Value 3.2" labelKey="enum3.value2.LABEL"/>
          <Enum value="value3" label="Value 3.3" labelKey="enum3.value3.LABEL"/>
          <Enum value="value4" label="Value 3.4" labelKey="enum3.value4.LABEL"/>
          <Enum value="value5" label="Value 3.5" labelKey="enum3.value5.LABEL"/>
     </Enumeration>
</Property>
```

Control de selector de directorio en el cliente

El campo de texto de una línea y el botón asociado para permitir al usuario seleccionar un directorio en el cliente. El directorio debe existir previamente. Los usuarios podrán o bien abrir un archivo desde este directorio o guardar en él un archivo, dependiendo de la configuración de modo.

Figura 53. Control de selector de directorio en el cliente

El usuario puede introducir la ruta y nombre del directorio directamente en el campo de texto o pulsar en el botón adyacente para mostrar un cuadro de diálogo desde el que pueden seleccionar un directorio.

Formato

```
<ClientDirectoryChooserControl atributos_controlador mode="modo_selector" >
     -- opciones avanzadas de diseño personalizado --
</ClientDirectoryChooserControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`mode` determina el botón que se muestra en el cuadro de diálogo desde donde los usuarios seleccionan un directorio, y será uno de los siguientes:

- `open` (default) muestra un botón **Open**.
- `save` muestra un botón **Save**.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

```
<ClientDirectoryChooserControl property="directory2" label="Client Directory"
     labelKey="directory2.LABEL"/>
```

Control de selector de archivo en el cliente

Define un campo de texto de una línea y su botón asociado para permitir al usuario seleccionar un directorio en el cliente. El archivo debe existir previamente. Los usuarios podrán o bien abrir el archivo o guardarlo, dependiendo de la configuración de modo.

Figura 54. Control de selector de archivo en el cliente

El usuario puede introducir la ruta y nombre del archivo directamente en el campo de texto o pulsar en el botón adyacente para mostrar un cuadro de diálogo desde el que pueden seleccionar un archivo.

Formato

```
<ClientFileChooserControl atributos_controlador mode="modo_selector" >
     -- opciones avanzadas de diseño personalizado --
</ClientFileChooserControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`mode` determina el botón que se muestra en el cuadro de diálogo desde donde los usuarios seleccionan un directorio, y será uno de los siguientes:

- `open` (default) muestra un botón **Open**.
- `save` muestra un botón **Save**.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

```
<ClientFileChooserControl property="archivo2" label="Archivo cliente" labelKey="archivo2.LABEL"/>
```

Control de cuadro combinado

Define una lista desplegable de cuadro combinado.

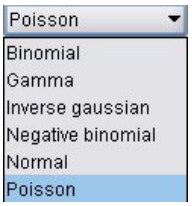

Figura 55. Cuadro combinado

Formato

```
<ComboBoxControl atributos_controlador >
     -- opciones avanzadas de diseño personalizado --
</ComboBoxControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

El siguiente ejemplo muestra el código empleado para disponer la lista desplegable de cuadro combinado en la ilustración anterior:

```
<ComboBoxControl property="distribución" >
     <Layout rowIncrement="0" gridWidth="1" fill="none"/>
</ComboBoxControl>
```

El elemento Layout se describe en "Diseño personalizado avanzado" en la página 153.

Note: Las entradas de lista reales se definen en la sección Propiedades del nodo pertinente; en este caso, como una lista enumerada en la declaración de la propiedad distribution:

```
<Property name="distribution" valueType="enum" label="Distribution" labelKey="distribution.
LABEL" defaultValue="NORMAL">
     <Enumeration>
          <Enum value="BINOMIAL" label="Binomial" labelKey="distribution.BINOMIAL.LABEL"/>
          <Enum value="GAMMA" label="Gamma" labelKey="distribution.GAMMA.LABEL"/>
          <Enum value="IGAUSS" label="Inverse gaussian" labelKey="distribution.IGAUSS.LABEL"/>
          <Enum value="NEGBIN" label="Negative binomial" labelKey="distribution.NEGBIN.LABEL"/>
          <Enum value="NORMAL" label="Normal" labelKey="distribution.NORMAL.LABEL"/>
          <Enum value="POISSON" label="Poisson" labelKey="distribution.POISSON.LABEL"/>
     </Enumeration>
</Property>
```

Control de selector de conexión de base de datos

Define un control que permite al usuario seleccionar un origen de datos y conectarse a una base de datos.

Figura 56. Control de selector de conexión de base de datos

Los usuarios no pueden introducir texto en el campo de texto, en su lugar deben pulsar en el botón para mostrar el cuadro de diálogo de conexiones de base de datos de IBM SPSS Modeler estándar.

Al realizar la conexión correctamente, los detalles de conexión se mostrarán en el campo de texto del control de selector de conexión de base de datos.

Formato

```
<DBConnectionChooserControl atributos_controlador >
     -- opciones avanzadas de diseño personalizado --
</DBConnectionChooserControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

El siguiente ejemplo muestra la forma en que el control requiere la definición de una propiedad de cadena que puede utilizar para la cadena de conexión.

```
<Node ...>
     <Properties>
      ...
          <Property name="dbconnect" valueType="databaseConnection" />
     </Properties>
      ...
```

```
    <UserInterface >
     ...
         <Pestañas>
              <Tab label="Database">
                   <PropertiesPanel>
                        <DBConnectionChooserControl property="dbconnect" label="Database
                         Connection"/>
                    ...
                   </PropertiesPanel>
                    ...
    </UserInterface>
</Node>
```

Control de selector de tabla de base de datos

Define un campo de texto y su botón asociado que permite al usuario seleccionar una tabla de base de datos tras haberse conectado correctamente a una base de datos.

Figura 57. Control de selector de tabla de base de datos

Los usuarios pueden o bien introducir el nombre de tabla directamente en el campo de texto o bien pulsar en el botón y seleccionarlo en una lista.

Formato

```
<DBTableChooserControl connectionProperty="propiedad_conexión_BD" atributos_controlador >
    -- opciones avanzadas de diseño personalizado --
</DBTableChooserControl>
```

donde:

`connectionProperty` es el nombre de la propiedad de conexión de base de datos que ya se ha definido. Se trata del valor del atributo `property` del elemento `DBConnectionChooserControl` que se ha definido previamente para el nodo.

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

Este ejemplo continúa del anterior de `DBConnectionChooserControl` y muestra cómo puede incluir también un elemento `DBTableChooserControl` para seleccionar una tabla de base de datos una vez que se ha establecido una conexión de base de datos con éxito.

```
<Node ...>
    <Properties>
     ...
        <Property name="dbconnect" valueType="databaseConnection" />
        <Property name="dbtable" valueType="string" />
    </Properties>
      ...
    <UserInterface >
     ...
         <Pestañas>
              <Tab label="Database">
```

```
                    <PropertiesPanel>
                         <DBConnectionChooserControl property="dbconnect" label="Database
                          connection"/>
                         <DBTableChooserControl property="dbtable" connectionProperty=
                          "dbconnect" label="Database Table" />
                     ...
                    </PropertiesPanel>
          ...
         </UserInterface>
</Node>
```

Control de selector de campos múltiples

Define un control que permite al usuario seleccionar uno o más nombres de campo de una lista.

Figura 58. Control de selector de campos múltiples

Cuando el usuario pulsa en este control, aparece una lista de campos entre los que el usuario puede elegir uno o más.

El conjunto está compuesto por todos los campos visibles en este nodo. Si se han filtrado campos en un punto anterior de la ruta de este nodo, sólo serán visibles los campos que hayan pasado por el filtro. La lista también puede restringirse aun más especificando que sólo estarán disponibles para su selección los campos que tengan unos tipos específicos de almacenamiento y datos.

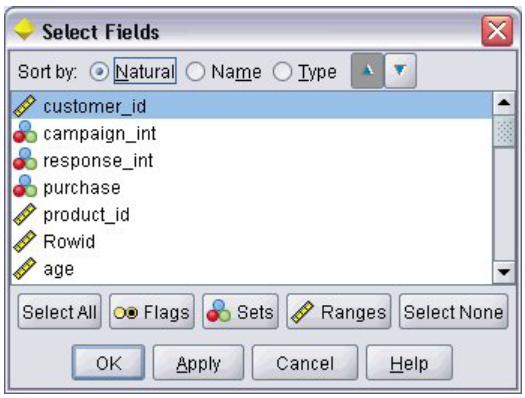

Figura 59. Lista de campos múltiples

Cada control selector de campos múltiples especifica un atributo de propiedad, que se declara en algún otro lugar del archivo y define la forma en que se muestra la lista en el cuadro de diálogo de nodo.

Formato

```
<MultiFieldChooserControl atributos_controlador storage="tipos_almacenamiento" onlyNumeric="true_false"
     onlySymbolic="true_false" onlyDatetime="true_false" types="tipos_datos"
     onlyRanges="true_false" onlyDiscrete="true_false" >
     -- opciones avanzadas de diseño personalizado --
</MultiFieldChooserControl>
```

donde:

atributos_controlador son los descritos en “Atributos de controlador” en la página 130.

Además, puede restringir aún más la lista de campos especificando otros dos atributos, uno de los cuales debe pertenecer a la lista siguiente:

- `storage` es una propiedad de lista que especifica el tipo de almacenamiento de los campos que se van a admitir en la lista; por ejemplo, `storage="[integer real]"` significa que sólo se enumerarán los campos con tipos de almacenamiento de número entero real. Para conocer el conjunto de posibles tipos de almacenamiento, consulte la tabla que se encuentra debajo de "Tipos de almacenamiento y datos" en la página 183.
- `onlyNumeric`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento numérico.
- `onlySymbolic`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento simbólico (es decir, cadena).
- `onlyDatetime`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento de fecha y hora.

El segundo atributo especificado debe pertenecer a esta lista:

- `types` es una propiedad de lista que especifica el tipo de datos de los campos que se van a admitir en la lista; por ejemplo, `types="[range flag]"` significa que sólo se enumerarán los campos con tipos de almacenamiento de marca de rango. El conjunto de tipos de datos posibles es:

  `rango`

  `distintivo`

  `set`

  `orderedSet`

  `numérico`

  `discrete`

  `typeless`
- `onlyRanges`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de datos de rango.
- `onlyDiscrete`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de datos discreto (es decir, marca, conjunto o sin tipo).

Por eso, por ejemplo, un control que especifica `storage="[integer]"` y `types="[flag]"` garantiza que en la lista sólo aparecerán campos de números enteros que sean marcas.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Note: Este control se utiliza únicamente en las definiciones del elemento Node. Para especificar un selector de campos múltiples dentro de una definición de modelo de datos de salida, utilice el siguiente formato:

```
<OutputDataModel mode="modo">
 ...
    <ForEach var="field" inProperty="nombre_prop">
         <AddField name="${nombre_campo}_NEW" fieldRef="${nombre_campo}" />
    </ForEach>
 ...
</OutputDataModel>
```

Para obtener más información, consulte el tema "Modelo de datos de salida" en la página 58. El elemento `ForEach` se describe en "Iteración con el elemento ForEach" en la página 68. `AddField` se describe en "Adición de un campo" en la página 64.

Ejemplo

El siguiente ejemplo muestra el código empleado para especificar el control de selector de campos múltiples en la ilustración anterior:

```
<MultiFieldChooserControl property="inputs" >
     <Activado>
          <Condition control="custom_fields" op="equals" value="true"/>
     </Enabled>
</MultiFieldChooserControl>
```

La sección `Enabled` provoca que el control sólo se active si se selecciona el control `campos_personalizados`.

Note: El contenido de esta lista se controla mediante la declaración de propiedad `inputs` en la sección de propiedades del nodo pertinente:

```
<Property name="inputs" valueType="string" isList="true" label="Inputs" labelKey="inputs.
LABEL"/>
```

Control de selector de elementos múltiples

Define un control que permite al usuario seleccionar uno o más elementos de una lista de valores. Asocia una propiedad con un catálogo que tiene una lista de valores. Para obtener más información, consulte el tema "Catalogs" en la página 41.

Figura 60. Control de selector de elementos múltiples

Formato

```
<MultiItemChooserControl atributos_controlador catalog="nombre_catálogo" >
     -- opciones avanzadas de diseño personalizado --
</MultiItemChooserControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`catalog` (obligatorio) es el nombre del catálogo que se asociará. La biblioteca en la que se obtiene el catálogo es la que se especifica en el elemento `Module` de la sección de ejecución. Para obtener más información, consulte el tema "Módulos" en la página 57.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

```
<MultiItemChooserControl property="selection2" catalog="cat2" />
```

La propiedad a la que hace referencia el atributo `property` (`selection2` en este caso) debe tener un atributo `isList="true"`. Para obtener una explicación y un ejemplo del uso de `MultiItemChooserControl`, consulte "Catalogs" en la página 41.

Control de cuadro de contraseña

Define un campo de una línea de texto en el que los caracteres de entrada se ocultan a medida que se escriben.

Figura 61. Control de cuadro de contraseña

Formato

```
<PasswordBoxControl atributos_controlador columns="entero" >
     -- opciones avanzadas de diseño personalizado --
</PasswordBoxControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`columns` es un número entero positivo que especifica el número de columnas de caracteres que debe ocupar el cuadro de contraseña. El valor predeterminado es 20.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

```
<PasswordBoxControl property="cadena_cifrada1" label="Cadena cifrada 1" labelKey=
"cadenaCifrada1.LABEL"/>
```

El campo de texto se cifra asociándolo a una propiedad que se define como una cadena cifrada en la sección Propiedades del nodo pertinente:

```
<Property name="cadena_cifrada1" valueType="encryptedString"/>
```

Control de propiedad

Un control de propiedad es un control que el usuario puede definir completamente y permite a los usuarios introducir propiedades para el nodo. El procesamiento se gestiona mediante una clase de Java escrita por el usuario. La ilustración siguiente es un ejemplo de un control de propiedad.

Figura 62. Sección del cuadro de diálogo con un ejemplo de control de propiedad resaltado

Formato

```
<PropertyControl atributos_controlador controlClass="clase_Java" >
     -- opciones avanzadas de diseño personalizado --
</PropertyControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

controlClass (obligatorio) es la ruta dentro de un archivo *.jar* hacia la clase de Java que implementa el control de propiedad. (*Note*: El archivo *.jar* se declara en un elemento JarFile de la sección Resources. Para obtener más información, consulte el tema "Archivos Jar" en la página 35).

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

```
<PropertyControl property="target_field_values_specify" labelAbove="true"
     controlClass="com.spss.clef.selflearning.propertycontrols.list.CustomListControl" label=""
     labelKey="target_field_values_specify.LABEL">
     <Activado>
          <Condition control="target_field_values" op="equals" value="specify"/>
     </Enabled>
     <Layout rowIncrement="2" />
</PropertyControl>
```

El control de propiedad se asocia con una propiedad que se define en la sección Propiedades del nodo pertinente:

```
<Property name="target_field_values_specify" valueType="string" isList="true" label=""
     labelKey="target_field_values_specify.LABEL"/>
```

Control de grupo de botones de radio

Define un conjunto de botones de radio en el que sólo es posible seleccionar un botón cada vez.

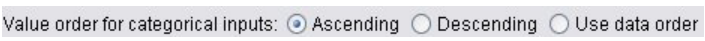

Figura 63. Control de grupo de botones de radio

Cada control de grupo de botones de radio tiene un atributo de propiedad que asocia el grupo con una propiedad particular. Esta propiedad se define en otro lugar del archivo y especifica los botones que forman el grupo.

La propiedad asociada puede ser una lista enumerada o una propiedad booleana. En el caso de las listas enumeradas (donde el atributo de propiedad valueType="enum"), se muestra un botón de radio por cada valor enum. En el caso de las propiedades booleanas (donde valueType="boolean"), siempre se muestran dos botones de radio.

Formato

```
<RadioButtonGroupControl atributos_controlador
     rows="entero" layoutByRow="true_false" useSubPanel="true_false"
     falseLabel="etiqueta_botón" falseLabelKey="clave_etiqueta" trueLabel="?etiqueta_botón"
     trueLabelKey="clave_etiqueta" trueFirst="true_false" >
     -- opciones avanzadas de diseño personalizado --
</RadioButtonGroupControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

rows es un número entero positivo que especifica el número de filas de la pantalla sobre las que se mostrará el grupo. El valor por omisión es 1.

layoutByRow especifica si los botones de radio deben disponerse en primer lugar a lo largo de la fila (true) o a lo largo de la columna (false). El valor predeterminado es true. Si desea conocer un uso similar de layoutByRow con un grupo de botón de radio, consulte “Modificación del orden de los controles” en la página 152.

useSubPanel especifica si los botones de radio van a mostrarse (true) o no (false) como subpanel. El valor predeterminado es true.

Los grupos de casillas de verificación suelen disponerse como un subpanel que contiene todas las casillas del grupo. Sin embargo, se pueden producir problemas de alineación si el grupo de botones de radio se asocia con un campo de texto adyacente. Este problema se evita si useSubPanel se define como false.

falseLabel es la etiqueta del valor "false" de una propiedad booleana (consulte el segundo ejemplo que aparece más abajo). Sólo se utiliza con las propiedades booleanas, en cuyo caso es obligatorio.

falseLabelKey identifica la etiqueta "false" con fines de localización.

trueLabel es la etiqueta del valor "true" de una propiedad booleana (consulte el segundo ejemplo que aparece más abajo). Sólo se utiliza con las propiedades booleanas, en cuyo caso es obligatorio.

trueLabelKey identifica la etiqueta "true" con fines de localización.

Si trueFirst se ha definido como true, el orden de visualización de los botones de una propiedad booleana puede invertirse, por lo que el botón que representa el valor "true" se mostrará en primer lugar. El valor predeterminado es false, lo que significa que el botón que representa el valor "false" se muestra en primer lugar.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema “Diseño personalizado avanzado” en la página 153.

Ejemplos

El primer ejemplo ilustra el código empleado para el grupo de botones de radio mostrado anteriormente.

```
<RadioButtonGroupControl property="value_order" labelWidth="2">
     <Layout gridWidth="4"/>
</RadioButtonGroupControl>
```

El elemento Layout se describe en “Diseño personalizado avanzado” en la página 153.

Note: El número de botones y sus etiquetas se definen en la sección de propiedades del nodo relevante; en este caso, como una lista enumerada en la declaración de la propiedad value_order: Esta declaración también incluye la etiqueta del propio grupo:

```
<Property name="value_order" valueType="enum" label="Value order for categorical
   inputs" labelKey="value_order.LABEL">
     <Enumeration>
          <Enum value="Ascending" label="Ascending" labelKey="value_order.Ascending.LABEL"/>
          <Enum value="Descending" label="Descending" labelKey="value_order.Descending.LABEL"/>
          <Enum value="DataOrder" label="Use data order" labelKey="value_order.UseDataOrder.LABEL"/>
     </Enumeration>
</Property>
```

El segundo ejemplo ilustra el uso de falseLabel y trueLabel para un grupo de botones de radio que controla una propiedad booleana, como uno para controlar si se activa la configuración estándar o personalizada.

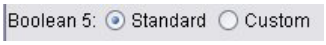

Figura 64. Grupo de botones de radio que controla una propiedad booleana

El código necesario es:

```
<RadioButtonGroupControl property="boolean5" label="Boolean 5" labelKey="boolean5.LABEL"
     falseLabel="Standard" falseLabelKey="boolean5.false.LABEL" trueLabel="Custom"
     trueLabelKey="boolean5.true.LABEL" />
```

En este caso, las etiquetas de botón se definen en el propio elemento `RadioButtonGroupControl`. La propiedad con la que se asocia el grupo se define en la sección de propiedades del nodo:

```
<Property name="boolean5" valueType="boolean" defaultValue="false"/>
```

Control de selector de directorio en el servidor

Define un campo de texto de una línea y el botón asociado para permitir al usuario seleccionar un directorio en el servidor. El directorio debe existir previamente. Los usuarios podrán o bien abrir un archivo desde este directorio o guardar en él un archivo, dependiendo de la configuración de modo.

Figura 65. Control de selector de directorio en el servidor

El usuario puede introducir la ruta y nombre del directorio directamente en el campo de texto o pulsar en el botón adyacente para mostrar un cuadro de diálogo desde el que pueden seleccionar un directorio.

Formato

```
<ServerDirectoryChooserControl atributos_controlador mode="modo_selector" >
     -- opciones avanzadas de diseño personalizado --
</ServerDirectoryChooserControl>
```

donde:

atributos_controlador son los descritos en “Atributos de controlador” en la página 130.

`mode` determina el botón que se muestra en el cuadro de diálogo desde donde los usuarios seleccionan un directorio, y será uno de los siguientes:

- `open` (default) muestra un botón **Open**.
- `save` muestra un botón **Save**.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema “Diseño personalizado avanzado” en la página 153.

Ejemplo

```
<ServerDirectoryChooserControl property="directory1" label="Server Directory"
     labelKey="directory1.LABEL"/>
```

Control de selector de archivo en el servidor

Define un campo de texto de una línea y el botón asociado para permitir al usuario seleccionar un directorio en el servidor. El archivo debe existir previamente. Los usuarios podrán o bien abrir el archivo o guardarlo, dependiendo de la configuración de modo.

Figura 66. Control de selector de archivo en el servidor

El usuario puede introducir la ruta y nombre del archivo directamente en el campo de texto o pulsar en el botón adyacente para mostrar un cuadro de diálogo desde el que pueden seleccionar un archivo.

Formato

```
<ServerFileChooserControl atributos_controlador mode="modo_selector" >
     -- opciones avanzadas de diseño personalizado --
</ServerFileChooserControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`mode` determina el botón que se muestra en el cuadro de diálogo desde donde los usuarios seleccionan un directorio, y será uno de los siguientes:

- `open` (default) muestra un botón **Open**.
- `save` muestra un botón **Save**.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

```
<ServerFileChooserControl property="archivo1" label="Archivo servidor" labelKey="archivo1.LABEL"/>
```

Control de selector de campo único

Define un control que permite al usuario seleccionar un único campo de una lista.

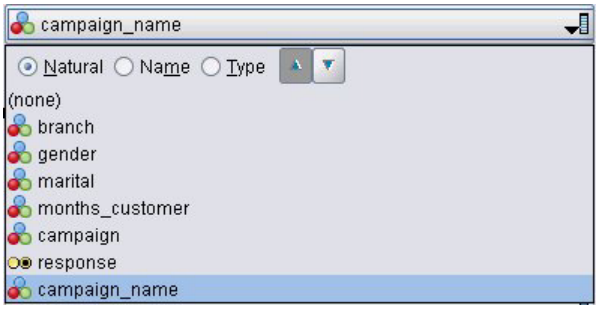

Figura 67. Control de selector de campo único

Cuando el usuario pulsa en este control, aparece una lista de campos entre los que puede elegir sólo uno.

El conjunto está compuesto por todos los campos visibles en este nodo. Si se han filtrado campos en un punto anterior de la ruta de este nodo, sólo serán visibles los campos que hayan pasado por el filtro. La lista también puede restringirse aun más especificando que sólo estarán disponibles para su selección los campos que tengan unos tipos específicos de almacenamiento y datos.

Formato

```
<SingleFieldChooserControl atributos_controlador storage="tipos_almacenamiento" onlyNumeric="true_
     false" onlySymbolic="true_false" onlyDatetime="true_false" types="tipos_datos"
     onlyRanges="true_false" onlyDiscrete="true_false" >
     -- opciones avanzadas de diseño personalizado --
</SingleFieldChooserControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

Además, puede restringir aún más la lista de campos especificando otros dos atributos, uno de los cuales debe pertenecer a la lista siguiente:

- `storage` es una propiedad de lista que especifica el tipo de almacenamiento de los campos que se van a admitir en la lista; por ejemplo, `storage="[integer real]"` significa que sólo se enumerarán los campos con tipos de almacenamiento de número entero real. Para conocer el conjunto de posibles tipos de almacenamiento, consulte la tabla que se encuentra debajo de "Tipos de almacenamiento y datos" en la página 183.
- `onlyNumeric`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento numérico.
- `onlySymbolic`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento simbólico (es decir, cadena).
- `onlyDatetime`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de almacenamiento de fecha y hora.

El segundo atributo especificado debe pertenecer a esta lista:

- `types` es una propiedad de lista que especifica el tipo de datos de los campos que se van a admitir en la lista; por ejemplo, `types="[range flag]"` significa que sólo se enumerarán los campos con tipos de almacenamiento de marca de rango. El conjunto de tipos de datos posibles es:

  `rango`

  `distintivo`

  `set`

  `orderedSet`

  `numérico`

  `discrete`

  `typeless`
- `onlyRanges`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de datos de rango.
- `onlyDiscrete`, si se establece como `true`, especifica que sólo se enumeran los campos con un tipo de datos discreto (es decir, marca, conjunto o sin tipo).

Por eso, por ejemplo, un control que especifica `storage="[integer]"` y `types="[flag]"` garantiza que en la lista sólo aparecerán campos de números enteros que sean marcas.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Note: Este control se utiliza únicamente en las definiciones de nodo. Para especificar un selector de campos múltiples dentro de una definición de modelo de datos de salida, utilice el siguiente formato:

```
<OutputDataModel mode="modo">
 ...
     <ForEach var="field" from="1" to="${entero}">
```

```
        <AddField name="${cadena}_${campo}" fieldRef="${referencia_campo}" />
    </ForEach>
 ...
</OutputDataModel>
```

Para obtener más información, consulte el tema "Modelo de datos de salida" en la página 58. El elemento `ForEach` se describe en "Iteración con el elemento ForEach" en la página 68. `AddField` se describe en "Adición de un campo" en la página 64.

Ejemplo

El siguiente ejemplo muestra el código empleado para especificar el control de selector de un campo en la ilustración anterior:

```
<SingleFieldChooserControl property="target"  storage="string" onlyDiscrete="true"/>
```

Note: El contenido real de la lista se define en la sección de propiedades del nodo pertinente; en este caso, como una lista enumerada en la declaración de la propiedad `target`:

```
<Property name="target" valueType="string" label="Campo objetivo" labelKey="objetivo.LABEL"/>
```

Control de selector de elemento único

Define un control que permite al usuario seleccionar un único elemento de una lista de valores. Asocia una propiedad con un catálogo que tiene una lista de valores. Para obtener más información, consulte el tema "Catalogs" en la página 41.

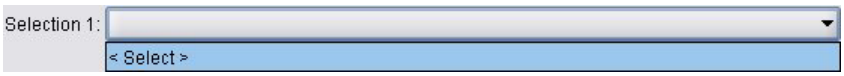

Figura 68. Control de selector de elemento único

Formato

```
<SingleItemChooserControl atributos_controlador catalog="nombre_catálogo" >
     -- opciones avanzadas de diseño personalizado --
</MultiItemChooserControl
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`catalog` (obligatorio) es el nombre del catálogo que se asociará. La biblioteca en la que se obtiene el catálogo es la que se especifica en el elemento `Module` de la sección de ejecución. Para obtener más información, consulte el tema "Módulos" en la página 57.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

```
<SingleItemChooserControl property="selection1" catalog="cat1" />
```

Para obtener una explicación y un ejemplo del uso del control, consulte"Catalogs" en la página 41.

Control de número

Define un número (campo numérico con flechas hacia arriba y hacia abajo que permiten modificar el valor de campo).

Figura 69. Número

Formato

```
<SpinnerControl atributos_controlador columns="entero" stepSize="incremento"
     minDecimalDigits="número" maxDecimalDigits="número" >
     -- opciones avanzadas de diseño personalizado --
</SpinnerControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`columns` es un número entero positivo que especifica el número de columnas de caracteres que abarca el control. El valor predeterminado es 5.

`stepSize` es un número decimal que define la cantidad que cambia el valor de campo cuando el usuario pulsa en una de las flechas. El valor predeterminado es 1,0.

`minDecimalDigits` es el número mínimo de decimales que se mostrarán para el valor de campo. El valor por omisión es 1.

`maxDecimalDigits` es el número máximo de decimales que se mostrarán para el valor de campo.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

El siguiente ejemplo muestra el código empleado para especificar el control de números en la ilustración anterior:

```
<SpinnerControl property="double1" label="Double 1" labelKey="double1.LABEL"/>
```

La precisión y el intervalo válido del contenido del campo numérico se define en la sección de propiedades del nodo pertinente; en este caso, en la declaración de la propiedad `double1`:

```
<Property name="double1" valueType="double" min="0" max="100"/>
```

Control de tabla

Define un elemento de presentación de tabla que se mostrará en un cuadro de diálogo de nodo o ventana de resultados.

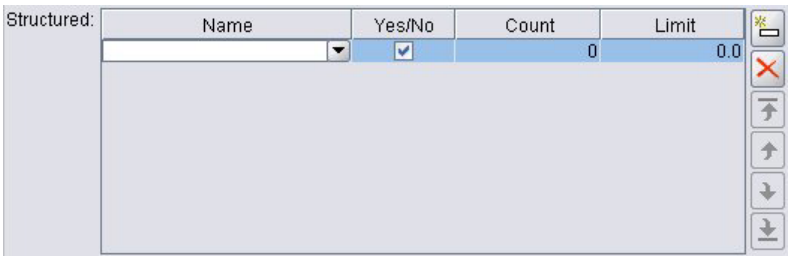

Figura 70. Control de tabla

Formato

```
<TableControl atributos_controlador rows="entero" columns="entero" columnWidths="lista" >
     -- opciones avanzadas de diseño personalizado --
</TableControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

`rows` es un número entero positivo que especifica el número de filas de tabla visibles en la pantalla. El valor predeterminado es 8.

`columns` es un número entero positivo que especifica el número de columnas de caracteres que abarca la tabla. El valor predeterminado es 20.

`columnwidths` es una lista de valores que especifica las anchuras de columna relatives. Además, por ejemplo, un valor de `[30 5 10]` especifica que la columna 1 es tres veces más ancha que la columna 3.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

El atributo *ColumnControl* del ejemplo siguiente es como se describe en "Control de columnas" en la página 149.

Ejemplo

El código para especificar el control de la tabla en la ilustración anterior es:

```
<TableControl property="estructura1" allowReorder="true" label="Structured"
  labelKey="structure1.LABEL" columnWidths="[20 6 10 10]">
     <ColumnControl column="0" editor="fieldValue" fieldControl="field1"/>
</TableControl>
```

La estructura del control de tabla se define como un tipo de propiedad en la sección de objetos comunes del archivo de especificación:

```
<PropertyType id="estructura_compartida1" valueType="structure" isList="true">
     <Structure>
          <Attribute name="id" valueType="string" label="Name" labelKey="structure1.id.LABEL"/>
          <Attribute name="yesno" valueType="boolean" label="Yes/No" labelKey="structure1.
           yesno.LABEL" defaultValue="true"/>
          <Attribute name="count" valueType="integer" label="Count" labelKey="structure1.
           count.LABEL" defaultValue="0"/>
          <Attribute name="limit" valueType="double" label="Limit" labelKey="structure1.
           limit.LABEL" defaultValue="0.0"/>
     </Structure>
</PropertyType>
```

En la especificación del nodo, el identificador de este tipo de propiedad se asocia con el identificador del control de tabla mediante una declaración de propiedad:

```
<Property name="estructura1" type="estructura_compartida1"/>
```

Si hace referencia al nodo en un script, puede definir los valores de la propiedad utilizando corchetes [] para la lista y llaves {} para la estructura. Por ejemplo, puede configurar el gráfico de dos estructuras para la propiedad `estructura1` de la siguiente forma:

```
set :ID_nodo.structure1 = [{"hello" true 4 0.21} {"bye" false 5 0.95}]
```

Tenga en cuenta que el orden de valor debe ser coherente con el orden en que se realizan las definiciones del `Atributo`.

Control de columnas: Define el diseño de las columnas en las tablas

Cada columna del control de tabla comparte el mismo tipo de datos; a partir de esta base, puede especificar un editor para una determinada columna para todas las filas. Por tanto, cada columna solo necesita un editor para editar. Por ejemplo, si la columna X necesita que el usuario introduzca un entero, puede establecer un editor de enteros para la columna X.

El atributo *editor* especifica el tipo de editor de la columna. Hay cuatro tipos de editores: *default*, *field*, *fieldValue* y *enumeration*, y cada tipo de editor es un elemento comboBox editable.

En el tipo de editor *fieldValue*, la lista desplegable contiene todos los valores del campo que se especificó en *fieldControl*. De este modo, el elemento XML siguiente define que al editar la columna 0, el editor es un cuadro combinado y la lista desplegable contiene todos los valores de field1:

```
<ColumnControl column="0" editor="fieldValue" fieldControl="field1" />
```

Puede sustituir *fieldControl* por *fieldDirection*. Por ejemplo: *fieldDirecton="[in out]"* significa que la lista desplegable del cuadro combinado contendrá todos los valores del primero de los campos cuya dirección sea *in* o *out*.

En el editor del tipo *field*, la lista desplegable contiene todos los campos que se adaptan a la condición de filtro del campo. El ejemplo siguiente define que la columna 0 utiliza un cuadro combinado field como editor, y la lista desplegable contiene todos los campos reales y enteros:

```
<ColumnControl column="0" editor="field" storage="[real integer]" />
```

Además, puede utilizar el atributo *types* para especificar los tipos de medida que deberían obedecer los campos de la lista desplegables. Los atributos boolean aplicables al campo son: *onlyRanges*, *onlyDiscrete*, *onlyNumeric*, *onlySymbolic* y *onlyDatetime*.

Control del área de texto

Define un campo de entrada de texto de varias líneas.

Figura 71. Control del área de texto

Formato

```
<TextAreaControl atributos_controlador rows="entero" columns="entero" wrapLines="true_false" >
     -- opciones avanzadas de diseño personalizado --
</TextAreaControl>
```

donde:

atributos_controlador son los descritos en “Atributos de controlador” en la página 130.

`rows` es un número entero positivo que especifica el número de pantalla que ocupa el área de texto. El valor predeterminado es 8.

`columns` es un número entero positivo que especifica el número de columnas de caracteres que abarca el área de texto. El valor predeterminado es 20.

wrapLines especifica si utilizar ajuste de línea en líneas de texto largo (true) o que sea necesario desplazarse horizontalmente para leer las líneas de texto largo (false). El valor predeterminado es true.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

El código para crear el ejemplo anterior es:

```
<TextAreaControl property="string2" label="String 2" labelKey="string2.LABEL"/>
```

En este caso, la etiqueta del área de texto se define en la declaración de control del área de texto, mientras el tipo de datos de entrada se define en la sección de propiedades del nodo pertinente; en la declaración de la propiedad string2:

```
<Property name="string2" valueType="string"/>
```

Control de cuadro de texto

Define un campo de entrada de texto de una línea.

Figura 72. Control de cuadro de texto

Formato

```
<TextBoxControl atributos_controlador columns="entero" >
     -- opciones avanzadas de diseño personalizado --
</TextBoxControl>
```

donde:

atributos_controlador son los descritos en "Atributos de controlador" en la página 130.

columns es un número entero positivo que especifica el número de columnas de caracteres que abarca el cuadro de texto. El valor predeterminado es 20.

Las opciones avanzadas de diseño personalizado ofrecen un grado preciso de control sobre el posicionamiento y la visualización de los componentes de pantalla. Para obtener más información, consulte el tema "Diseño personalizado avanzado" en la página 153.

Ejemplo

El código para crear el cuadro de texto anterior es:

```
<TextBoxControl property="string1" label="String 1" labelKey="string1.LABEL"/>
```

El tipo de datos de entrada del cuadro texto se define en la sección de propiedades del nodo pertinente; en este caso, en la declaración de la propiedad string1:

```
<Property name="string1" valueType="string"/>
```

Diseños de control de propiedad

Esta sección describe los métodos de diseño estándar utilizados en los cuadros de diálogos y ventanas, así como diferentes formas para modificarlas para obtener sus diseños personalizados.

Diseño de controles estándar

Un panel de propiedades se puede considerar un gráfico de casillas de dos dimensiones. Cada fila puede tener una altura diferente y cada columna una anchura diferente. Los componentes de la IU se pueden asignar a múltiples casillas contiguas, aunque normalmente un componente de IU se asigna únicamente a una casilla.

De forma predeterminada, un control de propiedad se asigna a una fila y cada control ocupa dos columnas: una para la etiqueta y una para el componente o componentes de control. La columna que contiene las etiquetas se expande a la anchura de la etiqueta más ancha. Por ejemplo, con los siguientes ejemplos en el archivo de especificación:

```
<TextBoxControl property="string1" label="String 1"/>
<PasswordBoxControl property="encryptedString1" label="Encrypted string 1"/>
<TextAreaControl property="string2" label="String 2"/>
```

el panel resultante se muestra en la siguiente figura.

Figura 73. Panel de propiedades simple

Tenga en cuenta que el carácter ":" al final de la etiqueta se añade automáticamente.

Un control de propiedad que contiene múltiples componentes de interfaz de usuario crea sus propia área rectangular invisible en la que se disponen estos componentes. Los elementos `RadioButtonGroupControl` y `CheckBoxGroupControl` son ejemplos de estos controles.

Tenga en cuenta que la forma del área rectangular en la que se disponen los componentes puede ser diferente, dependiendo del control de la propiedad. Además, es posible que el diseño de los diferentes controles no esté siempre correctamente alineado.

Algunos controles de propiedades incluyen componentes que completan totalmente la columna de componentes y se redimensionan horizontalmente cuando se agranda o reduce la anchura de la ventana. Ejemplos de estos controles son los que especifican los elementos `TextBoxControl`, `PasswordBoxControl` y `TextAreaControl`. Sin embargo, no todos los componentes se comportan de la misma manera. Por ejemplo, las casillas de verificación y los controles de número sólo ocupan una cantidad de espacio horizontal fija, incluso si se reduce la anchura de la ventana:

Diseño de controles personalizados

El diseño estándar de los controles se pueden modificar de diferentes formas, algunas de ellas más simples y otras más complejas.

Diseño personalizado simple

Los tres métodos simples de personalización del diseño de control son:

- Colocar una etiqueta por encima de su componente
- Cambiar el número de filas sobre las que se ubican los controles
- Cambiar el orden en que se disponen los controles

Ubicación de una etiqueta por encima de su componente: Puede colocar una etiqueta en una fila diferente por encima de su componente definiendo el atributo `labelAbove` del control a `true`. Por ejemplo:

```
<TextBoxControl property="string0" label="String 0" labelAbove="true"/>
<TextBoxControl property="string1" label="String 1"/>
<PasswordBoxControl property="encryptedString1" label="Encrypted string 1"/>
```

Además de colocar la etiqueta por encima del componente, el componente o componentes de IU se asignan a la columna de la etiqueta del diseño. Se muestran el siguiente panel, con la etiqueta **String 0** por encima del campo correspondiente.

Figura 74. Panel con etiqueta de campo en una fila diferente

Modificación del número de filas: De forma predeterminada, el botón de opción y las casillas de verificación se disponen en una única fila y la anchura del cuadro de diálogo se ajusta para acomodarlas. Si un botón de opción o un grupo de casillas de verificación tiene múltiples opciones, puede resultar un cuadro de diálogo de grandes dimensiones. Puede evitarlo si cambia el número de filas que se utilizan en el control. Defina el atributo `rows` de la definición de control al valor que desee. Por ejemplo:

```
<RadioButtonGroupControl property="enum1" label="Enum 1" rows="2"/>
```

Da como resultado un panel con el grupo de botones de radio en dos filas.

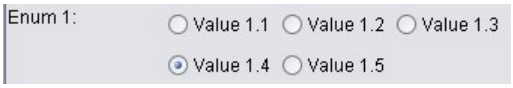

Figura 75. Panel con grupo de botones de radio en dos filas

Modificación del orden de los controles: En los grupos de botones de radio y casillas de verificación, también puede modificar el orden en que los controles de cada valor `enum` se añaden al panel.

De forma predeterminada, los controles se añaden en el orden de fila, como en el ejemplo anterior, donde el primer, segundo y tercer valor se añaden a la primera fila, con el cuarto y quinto valor añadidos a la segunda fila. En su lugar, puede añadir los controles en el orden de las columnas en el número especificado de filas, definiendo `layoutByRow` como `false`. Por ejemplo:

```
<RadioButtonGroupControl property="enum1" label="Enum 1" rows="2" layoutByRow="false"/>
```

Los valores se muestran en dos filas, pero el primer y segundo valor se añaden a la primera columna, el tercer y cuarto valor a la segunda columna y el quinto valor a la tercera columna.

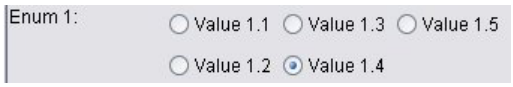

Figura 76. Panel con grupo de botones de radio en el orden de columnas

Con las propiedades booleanas como dos botones de radio, el comportamiento de ordenación predeterminado es mostrar el botón "False" antes del botón "True". Puede invertir el orden definiendo el atributo `trueFirst` a `true`.

También puede evitar que los grupos del botón de radio y de casillas de verificación utilicen un subpanel, definiendo el atributo `useSubPanel` a `false`. Sin embargo, puede provocar un comportamiento no deseado del diseño, salvo que lo utilice junto con el elemento `Diseño` (consulte “Especificación de las posiciones de control precisas con el elemento Layout” en la página 153).

Diseño personalizado avanzado

En cada declaración de control, puede especificar diseños de control complejos, utilizando diferentes elementos. Tiene la posibilidad de:

- Especificar posiciones de control precisas en la pantalla con el elemento `Layout`
- Características de diseño de control con el elemento `Enabled`
- Visibilidad de control de los componentes de pantalla con el elemento `Visible`

Especificación de las posiciones de control precisas con el elemento Layout: Las posiciones de control precisas se pueden lograr especificando un elemento `Layout` específico y asociándolo con el control.

Formato

```
<control_propiedad ... >
     <Layout atributos
      --- especificación de casillas ---
          ...
</control_propiedad>
```

donde:

control_propiedad es uno de los controles de propiedad (consulte "Especificaciones de control de propiedad" en la página 123).

atributos son cualquiera de los atributos que se muestran en la tabla siguiente.

Tabla 39. Atributos de diseño.

Atributo	Valores	Descripción

`anchor`	`north`
	`northeast`
	`east`
	`southeast`
	`south`
	`southwest`
	`west`
	`northwest`
	`centrado`
		Define el punto de anclaje del control.

`columnWeight`	0.0–1.0	Define cómo afecta una modificación del tamaño horizontal de la ventana a la anchura del control. La suma de todos los atributos `columnWeight` en un panel no debe superar 1.0.

`fill`	`ninguno`
	`horizontal`
	`vertical`
	`ambos`
		Define si el control debe rellenar las casillas a las que se ha asignado y cómo debe hacerlo.

`gridColumn`	entero >= 0	Define la primera columna en la que el control debe comenzar la ordenación.
`gridHeight`	entero	Define el número de filas que ocupa el control. Un valor de 0 (el predeterminado) distribuye el control por el resto de filas.
`gridRow`	entero >= 0	Define la primera fila en la que el control debe comenzar la ordenación. De forma predeterminada, el índice de fila de gráfico se incrementa automáticamente.
`gridWidth`	entero	Define el número de columnas que ocupa el control. Un valor de 0 (valor predeterminado) distribuye el control por el resto de columnas.
`leftIndent`	entero	Define el número de píxeles de sangría que se aplica al control desde su posición predeterminada.

Tabla 39. Atributos de diseño (continuación).

Atributo	Valores	Descripción
rowWeight	0.0–1.0	Define cómo afecta una modificación del tamaño vertical de la ventana a la altura del control. La suma de todos los atributos rowWeight en un panel no debe superar 1.0.

Una **especificación de casillas** permite especificar la posición precisa de un control en la pantalla. El formato es el siguiente:

```
<Cell row="entero" column="entero" width="entero" />
```

donde:

`fila` (obligatorio) es un entero no negativo que especifica la posición de fila en la que inicia el control.

`columna` (obligatorio) es un entero no negativo que especifica la posición de columna en la que inicia el control.

`anchura` (obligatorio) es un entero no negativo que especifica el número de columnas de gráfico que ocupa el control.

Además, por ejemplo, asumiendo un gráfico de tres columnas y tres filas, considere un diseño de controles personalizados en el siguiente formato.

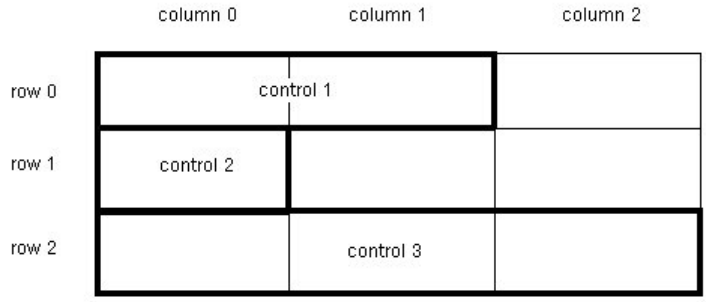

Figura 77. Ejemplo de diseño de control con casillas

Este formato necesita un elemento `Layout` con las siguientes especificaciones de casillas:

```
<Layout ...>
     <Cell row="0" column="0" width="2">
     <Cell row="1" column="0" width="1">
     <Cell row="2" column="0" width="3">
</Layout>
```

A continuación se incluyen algunos ejemplos que proporcionan ilustraciones sobre cómo puede utilizar el elemento `Layout`.

Ejemplo: Casilla de verificación Activación de campo de texto: Este ejemplo muestra cómo utilizar una casilla de verificación para activar un campo de texto en la misma línea del diseño.

Si utiliza una casilla de verificación para activar otro control en la misma línea, se necesita un elemento `Layout` simple para que los controles se visualicen correctamente. (*Note*: El mecanismo para activar y desactivar los controles se describe en "Control de las características de diseño con el elemento Enabled" en la página 160).

Supongamos que queremos incluir el panel siguiente en una representación.

Figura 78. Casilla de verificación Activación de campo de texto

Se incluyen dos controles:

- Una casilla de verificación con una etiqueta que actúa como la etiqueta de un campo de texto
- El campo de texto

El punto inicial es una declaración normal de los dos controles:

```
<CheckBoxControl property="boolean3" label="Check box 3"/>
<TextBoxControl property="string3" label="String 3"/>
```

Obtendremos el siguiente panel.

Figura 79. Casilla de verificación y campo de texto en filas diferentes

En primer lugar, queremos que no aparezca la etiqueta del campo de texto **String 3**. Esto se consigue definiendo el atributo `showLabel` del control de campo de texto a `false`:

```
<CheckBoxControl property="boolean3" label="Check box 3"/>
<TextBoxControl property="string3" label="String 3" showLabel="false"/>
```

El campo de texto se expande rellenando el área que antes ocupaba la etiqueta.

Figura 80. Casilla de verificación y campo de texto sin etiqueta

Ahora queremos que la etiqueta de texto aparezca en la misma línea que la casilla de verificación. Para ello, añadimos un elemento `Layout` al elemento `CheckBoxControl` para definir el incremento de filas a 0 (de forma predeterminada, esta opción se incrementa en 1 para cada control):

```
<CheckBoxControl property="booleano3" label="Casilla de verificación 3">
     <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false"/>
```

No obstante, obtendremos la siguiente representación.

Figura 81. Campo de texto superpuesto en la casilla de verificación

El campo de texto se ha subido una línea, pero sigue ocupando la fila completa, por lo que tapa la casilla de verificación.

Nota: La casilla de verificación se comienza a dibujar después de campo de texto, si la representación se parece a la siguiente.

Figura 82. Casilla de verificación superpuesta en el campo de texto

Los primeros caracteres del campo de texto aparecen sombreados.

Con independencia del objeto que se dibuje primero, no es recomendable asignar varios componentes de IU en la misma casilla, ya que su comportamiento no será el deseado o no se podrá definir, por lo que se debe evitar. Para resolver el problema, es necesario añadir un segundo elemento `Layout`, esta vez al elemento `TextBoxControl`, para forzar que el elemento de texto comience en la segunda columna de la representación:

```
<CheckBoxControl property="booleano3" label="Casilla de verificación 3">
     <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1"/>
</TextBoxControl>
```

No obstante, esto solo es una solución parcial. Ambos controles se colocan correctamente, pero el campo de texto es demasiado corto, tal como se muestra en la representación siguiente.

Figura 83. Colocación correcta, pero campo de texto muy corto

El problema es que una vez que se asocia un diseño personalizado con un control, sobrescribe los valores predeterminados "inteligentes" relacionados con cada tipo de control. En este caso, el comportamiento de relleno predeterminado del elemento `Layout` (es decir, cómo el componente rellena las casillas disponibles) no es rellenar las casillas disponibles y ocupar el menor espacio posible en la pantalla. Para modificar este valor, indique al campo de texto que complete el espacio horizontal:

```
<CheckBoxControl property="booleano3" label="Casilla de verificación 3">
     <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Es necesario añadir un reducido valor `columnWeight` para que Java distribuya el espacio relleno correctamente.

De esta forma obtendremos el diseño que pretendemos.

Figura 84. Casilla de verificación Activación de campo de texto

La apariencia parece correcta, pero queda un problema que resolver. La casilla de verificación intenta ocupar la fila completa, aunque estemos añadiendo otro control en la misma fila. El problema no es visible porque la etiqueta de la casilla de verificación es relativamente corta y el resto de etiquetas del panel (que no aparecen en la ilustración) han movido la segunda columna de visualización para que no se solape. El problema será obvio si se agranda la etiqueta de la casilla de verificación:

```
<CheckBoxControl property="boolean3" label="Casilla de verificación 3 con una etiqueta mucho más grande que
     <Layout rowIncrement="0"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

De esta forma, obtendremos lo siguiente.

Figura 85. Campo de texto superpuesto a la etiqueta larga de casilla de verificación

Lo que tenemos que hacer es limitar la anchura disponible de la casilla de verificación a una sola columna:

```
<CheckBoxControl property="boolean3" label="Casilla de verificación 3 con una etiqueta mucho
más grande que la que teníamos">
     <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Finalmente obtendremos lo que queríamos.

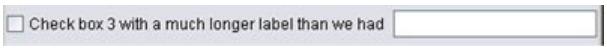

Figura 86. Etiqueta larga de la casilla de verificación visualizada correctamente

Ejemplo: Grupo de botones de radio y campos de texto: Este ejemplo muestra una forma de asociar cada botón en un grupo de botones de radio con su propio campo de texto.

Queremos definir un panel con la siguiente apariencia:

Figura 87. Grupo de botones de radio con campos de texto

Esta vez tenemos cuatro controles:

- Un grupo de botones de radio para una lista enumerada de tres valores
- Tres campos de texto, uno para cada valor

Como en el ejemplo anterior, comenzamos con una simple declaración de los controles:

```
<RadioButtonGroupControl property="enum4" label="Enum 4" >
<TextBoxControl property="string4" label="String 4"/>
<TextBoxControl property="string5" label="String 5"/>
<TextBoxControl property="string6" label="String 6"/>
```

De esta forma, obtendremos lo siguiente.

Figura 88. Grupo de botones de radio con campos de texto y etiquetas

Queremos utilizar las etiquetas del botón de radio para identificar los campos de texto, por lo que nuestra primera tarea es alinear los botones de radio en una única columna de tres filas y ocultar las etiquetas de campo de texto:

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3"/>
<TextBoxControl property="string4" label="String 4" showLabel="false"/>
<TextBoxControl property="string5" label="String 5" showLabel="false"/>
<TextBoxControl property="string6" label="String 6" showLabel="false"/>
```

Obtendremos la siguiente representación.

Figura 89. Botones de radio en una única columna y campos de texto

Ya podemos ver un pequeño problema; la etiqueta del grupo de botones de radio no está alineada con el primer botón de radio. Solucionaremos este problema más tarde. Ahora necesitamos alinear los campos de texto con su botón de radio correspondiente.

El procedimiento es similar al del ejemplo 1. Necesitamos:

- Modificar el incremento de la fila del grupo de botones de radio a 0.
- Limitar la anchura de la rejilla de forma que los campos de texto y los botones de opción no se solapen.
- Colocar cada campo de texto en la misma fila que su botón de radio.

Tenemos que añadir algunos elementos `Layout`, al igual que en el ejemplo anterior. En este caso, cambiamos el archivo de especificación de la siguiente forma:

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3" >
     <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="String 4" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="String 5" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="String 6" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Desgraciadamente, ahora tenemos la siguiente representación.

Figura 90. Campos de texto superpuestos a los botones de radio

Hemos utilizado los mismos elementos `Layout` del ejemplo 1, ¿qué ha ocurrido?

La respuesta es que, al contrario que el control de la casilla de verificación del ejemplo anterior, el grupo de botón de radio (como la mayoría de controles) tiene una etiqueta diferente, al igual que el control. Significa que el grupo de botones de radio requiere una columna extra, por lo que necesitamos que los campos de texto comiencen en una columna posterior, en la columna 2 en lugar de la columna 1. Además, en los elementos `Layout` de los campos de texto, definimos los valores `gridColumn` a 2:

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3" >
     <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="String 4" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="String 5" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string6" label="String 6" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Tenga en cuenta que, aunque hemos incrementado la columna del gráfico del campo de texto a 2, no aumentamos la anchura del gráfico del grupo de botones de radio de 1. Se debe a que en los controles de propiedad, la mayoría de atributos de `Layout` sólo afectan a los componentes de IU que componen la parte editable del control, en lugar de a la etiqueta del control.

Ahora tenemos la siguiente representación.

Figura 91. Los campos de texto ya no se superponen a los botones de radio

Estamos mucho más cerca de lo que pretendemos. Sin embargo, aún existen algunos problemas de alineación entre los botones de radio y los campos de texto.

El problema es que los botones de radio se representan en un subpanel diferente y por eso no existe una relación de diseño real entre un botón de radio y su campo de texto. Todo lo que tenemos que hacer es configurar el grupo de botones de radio para que dejen de utilizar un subpanel:

```
<RadioButtonGroupControl property="enum4" label="Enum 4" rows="3" useSubPanel="false">
     <Layout rowIncrement="0" gridWidth="1"/>
</RadioButtonGroupControl>
<TextBoxControl property="string4" label="String 4" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
<TextBoxControl property="string5" label="String 5" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
```

```
</TextBoxControl>
<TextBoxControl property="string6" label="String 6" showLabel="false">
     <Layout gridColumn="2" fill="horizontal" columnWeight="0.001"/>
</TextBoxControl>
```

Finalmente, tenemos el diseño que queríamos.

Figura 92. Grupo de botones de radio con campos de texto

Control de las características de diseño con el elemento Enabled: Puede utilizar el elemento `Enabled` para activar o desactivar un elemento, normalmente en función de si se cumple una condición concreta.

Los paneles y controles de propiedad pueden tener condiciones asociadas para determinar diferentes características de representación. Por ejemplo, una casilla de verificación se puede utilizar para activar un campo de texto asociado o un botón de radio puede causar que un grupo de campos ocultos sean visibles.

Las condiciones de la interfaz de usuario se suelen basar en el valor de un control diferente al de una propiedad. Las condiciones basadas en las propiedades surten efecto sólo cuando se hayan vuelto a aplicar los cambios al objeto subyacente (por ejemplo, nodo, resultado de modelo o resultado de documento). En la interfaz de usuario, los controles se deben activar en cuanto se modifique el control relacionado.

Formato

```
<Activado>
     <Condition .../>
     <And ... />
     <Or ... />
     <Not ... />
</Enabled>
```

El elemento `Condition` especifica que una condición se debe comprobar para determinar si el control se ha activado.

Los elementos `And`, `Or` y `Not` permiten especificar condiciones compuestas.

Para obtener más información, consulte el tema "Condiciones" en la página 72.

Ejemplo: Activación de controles con una condición simple: En "Ejemplo: Casilla de verificación Activación de campo de texto" en la página 154, hemos incluido una casilla de verificación diseñada para activar un campo de texto cuando se selecciona.

Queremos activar el campo de texto en cuanto seleccione la casilla de verificación y no cuando se modifique la propiedad del objeto subyacente. Para ello, necesitamos añadir una condición `Enabled`:

```
<CheckBoxControl property="boolean3" label="Casilla de verificación 3 con una etiqueta mucho
más grande que la que teníamos">
     <Layout rowIncrement="0" gridWidth="1"/>
</CheckBoxControl>
<TextBoxControl property="string3" label="String 3" showLabel="false">
     <Layout gridColumn="1" fill="horizontal" columnWeight="0.001"/>
```

```
          <Activado>
               <Condition control="boolean3" op="equals" value="true"/>
          </Enabled>
</TextBoxControl>
```

De esta forma se asegura que el campo de texto sólo se activará si el valor boolenano asociado con la casilla de verificación es true.

Ejemplo: Activación de controles con una condición compleja: Para ilustrar la codificación de condiciones complejas, consultamos una de las pestañas del cuadro de diálogo del nodo de modelos lineales generalizados, que se ha desarrollado utilizando CLEF.

El cuadro de diálogo del nodo contiene la pestaña Experto, con opciones para usuarios con conocimientos avanzados de los modelos. Todas las opciones de la pestaña están desactivadas inicialmente.

Si define la casilla de verificación **Modo** a **Experto**, activará algunas de estas opciones.

Sin embargo, algunas aparecen desactivadas, como el control **Iteraciones** en la parte inferior del cuadro de diálogo. Este control sólo está desactivado si **las dos** condiciones siguientes son true:

- **Distribución** definida a **Normal**
- **Función de enlace** definida a **Identidad**

Esta combinación es en realidad el ajuste predeterminado del modo Experto y si modifica alguna de las casillas, activará **Iteraciones**.

El código necesario se incluye en una declaración `PropertiesSubPanel` del botón **Iteraciones**, como se indica a continuación:

```
<PropertiesSubPanel buttonLabel="Iteraciones..." buttonLabelKey= ...
     <Activado>
          <And>
               <Condition control="mode" op="equals" value="Expert"/>
                    <Not>
                         <And>
                              <Condition control="distribution" op="equals" value="NORMAL"/>
                              <Condition control="link_function" op="equals" value="IDENTITY"/>
                         </And>
                    </Not>
          </And>
     </Enabled>
      ...
</PropertiesSubPanel>
```

El elemento `Condición` de la sección `Y` especifica que **Modo** debe definirse a **Experto** antes de realizar cualquier modificación. Si esta condición se cumple, la sección `No` especifica que el botón no está activado (desactivado) sólo si se cumplen *ambas* condiciones de la sección `Y` interior. Además, en modo Experto, **Iteraciones** está activada si **Distribución** o **Función de enlace** no tienen su valor predeterminado.

Control de las características de diseño con el elemento Visible: También puede utilizar condiciones para que los controles se muestren u oculten en función de las circunstancias. Se realiza mediante el elemento `Visible`.

Formato

```
<Visible>
     <Condition .../>
     <And ... />
     <Or ... />
     <Not ... />
</Visible>
```

El elemento `Condition` especifica que una condición se debe comprobar para determinar si el control es visible.

Los elementos `And`, `Or` y `Not` permiten especificar condiciones compuestas.

Para obtener más información, consulte el tema "Condiciones" en la página 72.

Ejemplo

El ejemplo siguiente muestra el panel de la propiedad especificada sólo si se cumple la condición `idioma_origen`:

```
<PropertiesPanel>
     <Visible>
          <Condition control="source_language" op="equals" value="eng" />
     </Visible>
       ...
</PropertiesPanel>
```

Ventanas de resultados personalizadas

Para resultados de modelo, de documentos y de objetos de resultados interactivos (pero no nodos), es posible que una extensión sustituya completamente la ventana de resultados predeterminada por una ventana personalizada. Se aplica como una clase `java.awt.Frame` estándar.

Para proporcionar una ventana personalizada, especifique una clase de Java como el atributo `frameClass` del elemento `UserInterface`, de la siguiente forma:

```
<DocumentOutput id="mi_nodo_modelado" type="modelBuilder" ...>
     <Properties>
          <Property name="use_custom_type" valueType="boolean" .../>
           ...
     </Properties>
     <UserInterface frameClass="com.myextension.MyOutputFrame"/>
      ...
  </DocumentOutput>
```

La clase especificada debe implementar la interfaz `ExtensionObjectFrame` definida en la API de cliente de CLEF. Cubre el ciclo vital de la ventana:

- Acceso al `java.awt.Frame` subyacente
- Inicialización de la ventana, incluyendo acceso al objeto de resultado y sesión
- Sincronización de la ventana y del objeto subyacente si el objeto se va a guardar o eliminar
- Eliminación de la ventana

Para obtener más información, consulte el tema "Clases de API de cliente" en la página 175.

Capítulo 7. Adición del sistema de ayuda

Tipos de sistemas de ayuda

Si desarrolla una extensión CLEF, normalmente deseará incluir un sistema de ayuda en pantalla para explicar cómo utilizar la extensión. CLEF admite los siguientes tipos de sistemas de ayuda:

- Ayuda HTML
- JavaHelp

Ayuda HTML

Ayuda HTML es un formato propio desarrollado por Microsoft que sólo funciona en plataformas de Windows. Un sistema de ayuda HTML está compuesto por archivos `.htm` o `.html` individuales compilados en un formato comprimido como un único archivo con la extensión `.chm`. El sistema de ayuda de IBM SPSS Modeler se proporciona en el formato de ayuda HTML.

La ayuda HTML admite tablas de contenido, índices y características de búsqueda de texto completo (los términos del glosario se pueden mostrar como ventanas emergentes). Puede crear los archivos de temas de origen `.htm` o `.html` mediante un editor HTML o una herramienta de creación de ayuda comercial. Una opción para crear el archivo `.chm` consiste en utilizar HTML Help Workshop, disponible como descarga gratuita en el sitio Web de Centro de descarga de Microsoft (para obtener más información acerca de cómo crear archivos `.chm`, consulte el sistema de ayuda de HTML Help Workshop). También puede utilizar una herramienta de creación que admita el formato de ayuda HTML para compilar sus temas de ayuda y los archivos gráficos que utilice en un archivo `.chm`.

JavaHelp

JavaHelp es un formato de ayuda de código abierto desarrollado por Sun Microsystems que funciona en cualquier plataforma compatible con Java. Un sistema de JavaHelp se compone de los siguientes archivos:

- Los archivos de temas de origen `.htm` o `.html`
- Cualquier archivo de gráfico utilizado en los temas
- Un archivo helpset (con la extensión `.hs`) que controla el sistema de ayuda
- Un archivo `map.xml`, que se utiliza para asociar los ID de temas con archivos de temas y para definir la ventana que muestra los temas de ayuda
- Un archivo `index.xml`, que contiene las entradas de índice
- Un archivo `toc.xml`, que contiene las entradas de la tabla de contenido

JavaHelp es compatible con las características de tabla de contenido, índice, búsquedas de texto completo y glosario. Puede crear los archivos de origen `.htm` o `.html` mediante un editor HTML o una herramienta de creación de ayuda comercial. También necesitará el software de JavaHelp, disponible en forma de descarga gratuita en el sitio Web Sun Developer Network (para obtener más información, consulte el *Manual de usuario del sistema JavaHelp*, también disponible en el mismo sitio Web).

Implementación de un sistema de ayuda

Esta sección describe cómo definir los componentes relevantes del sistema de ayuda en el archivo de especificación.

Definición de la ubicación y el tipo del sistema de ayuda

El tipo de sistema de ayuda, si existe, utilizado para la extensión, se define en un elemento `HelpInfo` en la sección de recursos del archivo de especificación de la extensión.

Formato

```
<Resources>
 ...
     <HelpInfo id="nombre" type="tipo_ayuda" path="ruta_ayuda" helpset="loc_helpset"
      default="IDtema_predeterminado" />
 ...
</Resources>
```

donde:

`id` (necesario) es el identificador de la información de ayuda de esta extensión.

`type` (necesario) indica el tipo de ayuda y es uno de los siguientes:

- `htmlhelp`: Ayuda HTML, contenida en un archivo `.chm` identificado por el atributo `path`.
- `javahelp`: JavaHelp, uso de un archivo helpset (`.hs`) identificado por el atributo `helpset`, junto con el origen de la ayuda y los archivos asociados.

Si el tipo de ayuda es `htmlhelp`, se requiere el siguiente atributo adicional:

- `path`: la ubicación (relativa al archivo de especificación) y el nombre del archivo `.chm` que contiene el sistema de ayuda.

Si el tipo de ayuda es `javahelp`, se requerirán los siguientes atributos adicionales:

- `helpset`: la ubicación (relativa al archivo de especificación) y el nombre del archivo helpset `.hs` que se va a utilizar.
- `default`: el identificador del tema predeterminado que se mostrará si no se ha especificado ningún tema en una pestaña concreta.

Si no se ha especificado un elemento de `HelpInfo`, no se asocia ningún archivo con esta extensión.

Ejemplos

El primer ejemplo ilustra un elemento de `HelpInfo` de la ayuda HTML:

```
<HelpInfo id="ayuda" type="htmlhelp" path="help/mynode.chm" />
```

El equivalente en un sistema de JavaHelp es:

```
<HelpInfo id="ayuda" type="javahelp" helpset="help/minodo.hs"/>
```

Tenga en cuenta que en el caso de JavaHelp, los archivos asociados (imágenes, archivo de correlación, índice y archivos de contenido) se deben ubicar en la misma carpeta que el archivo helpset `.hs`.

Especificación de un tema de ayuda concreto para visualizar

Puede especificar un tema de ayuda concreto si el usuario activa la ayuda en un cuadro de diálogo de nodo, desde una pestaña o desde un subpanel de propiedades. Se realiza mediante el atributo `helpLink` del nodo, pestaña o subpanel de propiedades.

Si no se ha especificado ningún atributo `helpLink`, el tema predeterminado del sistema de ayuda se mostrará si el usuario activa la ayuda.

Si desea obtener más información, consulte las descripciones del atributo `helpLink` en “Nodo” en la página 48, “Pestañas” en la página 113 y en“Subpanel de propiedades” en la página 127.

Ejemplo

Este ejemplo asume que utiliza una ayuda HTML y le enseña cómo puede mostrar diferentes temas contextuales, dependiendo de la ventana seleccionada cuando el usuario activa la ayuda.

```
<Resources>
      ...
     <HelpInfo id="ayuda" type="htmlhelp" path="help/mynode.chm"/>
      ...
</Resources>
 ...
<Node id="minodo" scriptName="mi_nodo" type="transformadordatos" palette="Operacionesconregistro"
  label="Clasificador" description="Clasifica un archivo de datos" >
  ...
     <Tabs defaultTab="1">
          <Tab label="Controles básicos" labelKey="FichaControlesbásicos.LABEL"
            helpLink="controles_básicos.htm">
               <PropertiesPanel>
                ...
                    <PropertiesSubPanel buttonLabel="Configuración adicional..."
                      buttonLabelKey="AdditionalOptions.LABEL" dialogTitle="Configuración
                      adicional" dialogTitleKey="AdditionalOptionsDialog.LABEL" helpLink=
                      "addsettingsdlg.htm">
  ...
          </Tab>
          <Tab label="Controles de selector" labelKey="FichaControlesselector.LABEL"
           helpLink="controles_selector.htm">
  ...
          </Tab>
  ...
</Node>
```

Especifica que, si la pestaña Controles básicos está activada y el usuario acciona la ayuda, se muestra el tema de `controles_básicos.htm` del archivo de ayuda `minodo.chm`. Si el usuario pulsa en el botón **Configuración adicional** para abrir el cuadro de diálogo Configuración adicional y selecciona **Ayuda** en ese cuadro de diálogo, se muestra el tema de `configuraciónadicionaldlg.htm`. Si el usuario cancela el cuadro de diálogo Configuración adicional, selecciona la pestaña Controles de selector y vuelve a seleccionar **Ayuda**, se muestra el tema de `controles_selector.htm`.

En JavaHelp, el valor del atributo `helpLink` debe coincidir con el valor del atributo `target` en el archivo `map.xml`. Por ejemplo, si el archivo `map.xml` contiene la siguiente cadena:

```
<map version="1.0">
 ...
     <mapID target="controles_básicos" url="basic_controls.htm"/>
 ...
</map>
```

debe dar al atributo `helpLink` correspondiente el siguiente valor:

```
helpLink="controles_básicos"
```

Se debe a que cuando se activa JavaHelp, lee el valor del atributo `target` y lo correlaciona al valor `url` asociado para buscar el archivo correcto que se va a visualizar.

Capítulo 8. Localización y accesibilidad

Introducción

La **localización** se refiere al proceso de adaptación del software, la ayuda y la documentación a una configuración regional específica. Incluye la traducción de la interfaz de usuario, la ayuda y la documentación, así como la comprobación del sistema en la configuración regional adecuada. Si va a distribuir su extensión a usuarios de regiones diferentes a la suya, puede distribuir versiones localizadas de la extensión.

El término **accesibilidad** se refiere, en este contexto, a la inclusión de características en la interfaz de usuario que facilitan el acceso al sistema a usuarios con ciertas discapacidades, como problemas de visión o movilidad limitada en las manos.

Localización

IBM SPSS Modeler se ha localizado para varias regiones del mundo. Siempre que el usuario aplique su propia configuración regional en Windows, los componentes estándar de la IU de IBM SPSS Modeler aparecerán en ese idioma, si está admitido, por ejemplo:

- Menús del sistema y entradas de menú
- Botones del sistema (Generar, Aceptar, Ejecutar, Cancelar, Aplicar, Restablecer)
- Pestañas de cuadros de diálogo estándar (Anotaciones y Depurar, si se utilizan)
- Mensajes de error y del sistema (por ejemplo, "No se ha guardado este objeto.")

En aquellos lugares donde su extensión utilice estos componentes estándar de IBM SPSS Modeler, éstos aparecerán automáticamente en el idioma seleccionado si está admitido.

En el caso de los otros componentes de su extensión, CLEF ofrece una característica para ayudarle con la localización. Puede localizar:

- Nombres de nodos (en paleta y lienzo)
- Nombres de modelo (en la pestaña Modelos del panel del gestor)
- Nombres de documento (en la pestaña Resultados del panel del gestor)
- Ubicación de una imagen de icono asociada a una acción
- Texto de información sobre herramientas
- Sistemas de ayuda
- Cuadros de diálogo de nodos
  - Texto de barra de título
  - Menús personalizados y entradas de menú
  - Etiquetas de campo, propiedad, botón y pestaña
  - Texto estático
- Mensajes de error y del sistema

Las cadenas de texto deben tener una longitud razonablemente corta para permitir la introducción de texto más largo en la traducción.

Los mensajes de error y del sistema pueden localizarse mediante una combinación del archivo de especificación, archivos de propiedades y API de servidor. Para obtener más información, consulte el tema “Documento de detalles de estado” en la página 193.

Archivos de propiedades

Las cadenas de texto de los elementos que puede localizar se almacenan en archivos conocidos como **archivos de propiedades**, que utilizan un formato Java estándar para almacenar paquetes de recursos de localización. Cada archivo de propiedades está compuesto por una serie de registros, uno por cada elemento localizado de la extensión. Un campo de cada registro se corresponde con un atributo `labelKey` en el archivo de especificación, el cual permite que CLEF lea la cadena de texto correspondiente del archivo de propiedades y la muestre en el lugar correcto.

Un archivo de propiedades debe tener la extensión `.properties` y debe encontrarse en el mismo directorio que el archivo de especificación del nodo con el que está relacionado. IBM SPSS Modeler busca inicialmente el archivo de propiedades predeterminado, que se denomina:

ruta`.properties`

donde *ruta* es el valor del atributo `path` del elemento `Bundle` (en la sección Resources) que define el paquete de propiedades. Por ejemplo:

```
<Bundle id="paquete" path="mis_recursos"/>
```

Si no hay ningún archivo de propiedades predeterminado, IBM SPSS Modeler lee las cadenas de texto de las definiciones del archivo de especificación.

Debe haber un archivo de propiedades para cada idioma admitido por la localización. Los archivos de idiomas distintos del idioma predeterminado se distinguen por un sufijo en el nombre de archivo. Por ejemplo:

```
my_resources.properties
my_resources_de.properties
my_resources_fr.properties
```

Los sufijos cumplen con el estándar de dos caracteres ISO 639-1 para códigos de lenguaje.

Cada registro de un archivo de propiedades tiene el siguiente formato:

id=texto_cadena

donde:

id es el identificador de un atributo `buttonLabelKey`, `descriptionKey`, `dialogTitleKey`, `falseLabelKey`, `imagePathKey`, `labelKey`, `messageKey`, `textKey` o `trueLabelKey` en el archivo de especificación. Este identificador suele tener el sufijo `.LABEL` para poder distinguirlo fácilmente, aunque puede tener cualquier sufijo o ninguno, dependiendo de la forma en que aparezca en el archivo de especificación.

texto_cadena es el texto del elemento.

Ejemplo: Localización de la pestaña de un cuadro de diálogo

Este ejemplo de pestaña localizada de un cuadro de diálogo de nodo utiliza dos archivos de propiedades, la versión predeterminada (inglés) y la versión en francés, con las siguientes ubicaciones:

carpeta_extensión`\`**`mis_recursos`**`.properties`
carpeta_extensión`\`**`mis_recursos`**`_fr.properties`

donde *carpeta_extensión* es la carpeta con el archivo de especificación.

En el archivo de especificación, se hace referencia a los archivos de propiedades mediante el elemento `Bundle` de la sección `Resources`:

```
<Resources>
     <Bundle id="bundle" type="properties" path="mis_recursos"/>
</Resources>
```

Tenga en cuenta que el atributo `path` no debe incluir extensiones de lenguaje o el sufijo `.properties`.

Las otras partes relevantes del archivo de especificación son:

```
<Node id="PruebaIU" scriptName="prueba_iu" type="transformadordatos"
palette="Operacionesconregistro" label="Prueba de IU" ...>
     <Properties>
          <Property name="enum1" valueType="enum" defaultValue="valor4">
               <Enumeration>
                    <Enum value="value1" label="Value 1.1" labelKey="enum1.value1.LABEL"/>
                    <Enum value="value2" label="Value 1.2" labelKey="enum1.value2.LABEL"/>
                    <Enum value="value3" label="Value 1.3" labelKey="enum1.value3.LABEL"/>
                    <Enum value="value4" label="Value 1.4" labelKey="enum1.value4.LABEL"/>
                    <Enum value="value5" label="Value 1.5" labelKey="enum1.value5.LABEL"/>
               </Enumeration>
          </Property>
     </Properties>
 ...
     <UserInterface ...>
          <Tabs defaultTab="1">
               <Tab label="Basic Controls" labelKey="basicControlsTab.LABEL" ... >
      ...
     </UserInterface>
 ...
</Node>
```

El archivo de propiedades en su versión inglesa incluye los siguientes registros:

```
basicControlsTab.LABEL=Basic Controls
enum1.value1.LABEL=Value 1.1
enum1.value2.LABEL=Value 1.2
enum1.value3.LABEL=Value 1.3
enum1.value4.LABEL=Value 1.4
enum1.value5.LABEL=Value 1.5
```

Las partes del cuadro de diálogo a las que afectan estos registros se resaltan en la siguiente ilustración.

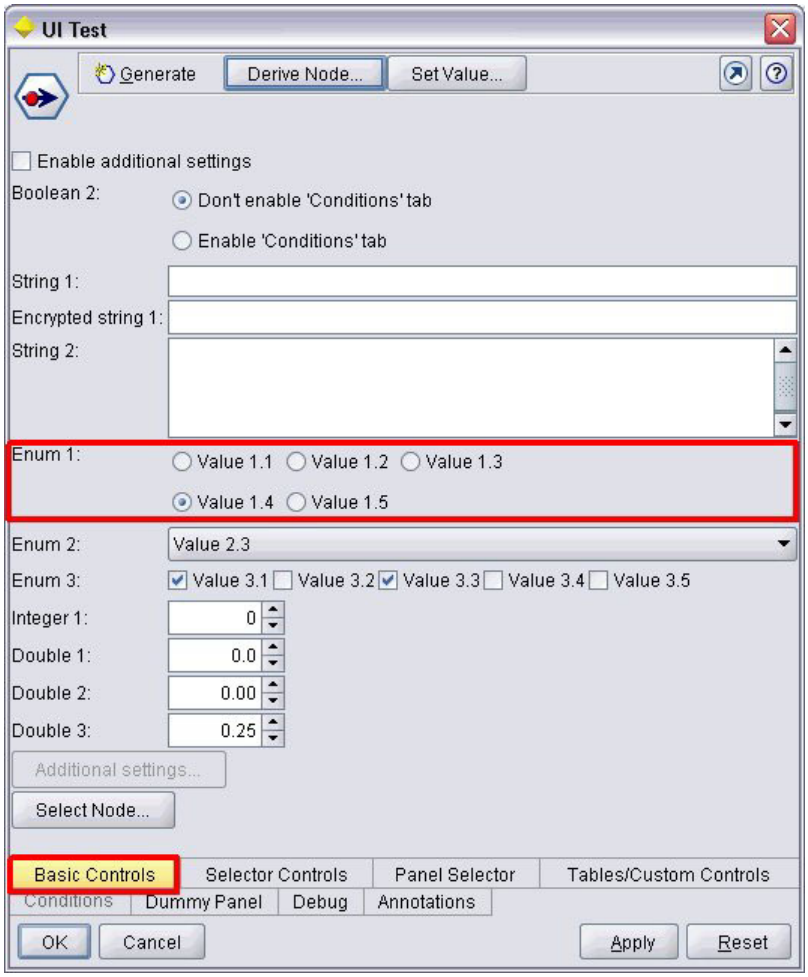

Figura 93. Pestaña no localizada

La sección correspondiente de la versión en francés del archivo de propiedades (`mis_recursos_fr.properties`) es:

```
basicControlsTab.LABEL=Contrôles de Base
enum1.value1.LABEL=Valeur 1,1
enum1.value2.LABEL=Valeur 1,2
enum1.value3.LABEL=Valeur 1,3
enum1.value4.LABEL=Valeur 1,4
enum1.value5.LABEL=Valeur 1,5
```

Estos registros hacen las partes relevantes de la pantalla muestren el texto traducido, como se muestra en la siguiente ilustración.

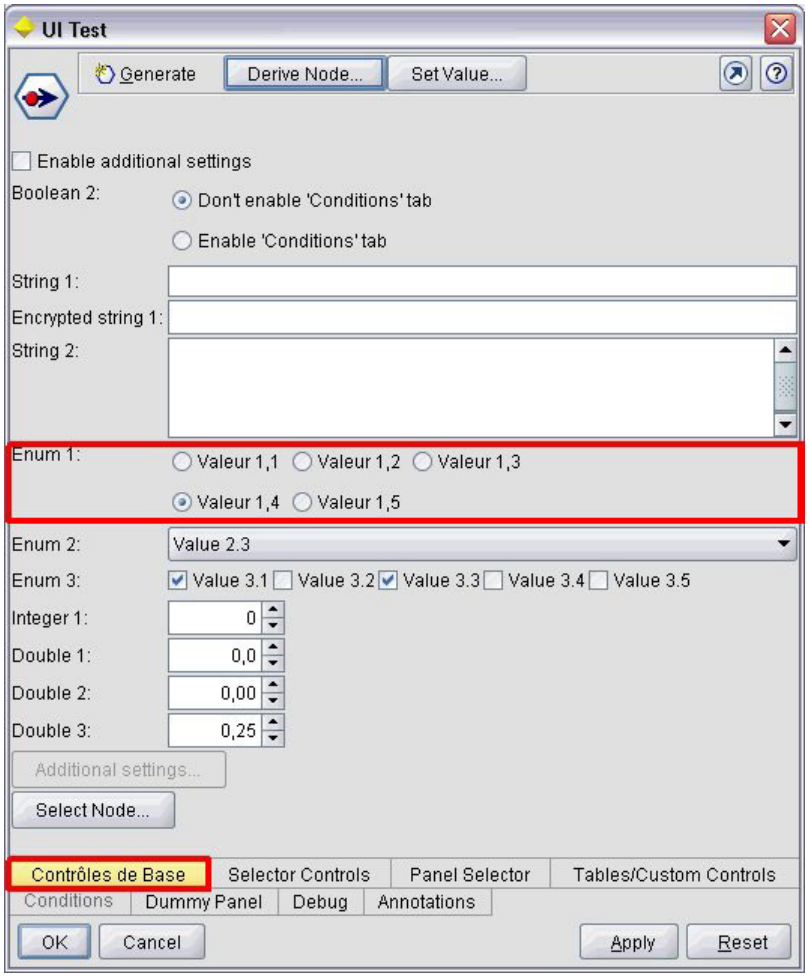

Figura 94. Pestaña localizada

Observe que la localización de los cuatro botones de la parte inferior de la pantalla se realiza mediante una característica estándar de IBM SPSS Modeler y no mediante CLEF.

Ejemplo: Uso de caracteres especiales

En el archivo de propiedades, deberá utilizar secuencias de escape Unicode para cualquier carácter especial en cadenas de texto ASCII estándar. Ésta es la parte de un archivo de propiedades localizada en francés:

```
Genlinnode.LABEL=Lin\u00e9aire g\u00e9n\u00e9ralis\u00e9

Fields.LABEL=Champs
Model.LABEL=Mod\u00e8le
Expert.LABEL=Expert

inputFields.LABEL=Entr\u00e9es
targetField.LABEL=Cible
 ...
```

En el caso de idiomas que utilizan caracteres no latinos, como el japonés o el chino, deberá utilizar secuencias de escape Unicode para las cadenas de texto completas. Éste es el mismo conjunto de registros localizado en japonés:

```
Genlinnode.LABEL=\u4e00\u822c\u5316\u7dda\u578b

Fields.LABEL=\u30d5\u30a3\u30fc\u30eb\u30c9
```

```
Model.LABEL=\u30e2\u30c7\u30eb
Expert.LABEL=\u30a8\u30ad\u30b9\u30d1\u30fc\u30c8

inputFields.LABEL=\u5165\u529b
targetField.LABEL=\u5bfe\u8c61
 ...
```

Archivos de ayuda

Si está localizando una extensión que tenga un sistema de ayuda, también debe proporcionar una versión localizada del sistema de ayuda. Debe proporcionar un sistema de ayuda localizado para cada extensión localizada.

Localización de la ayuda HTML

Si la extensión que está localizando utiliza un archivo de ayuda HTML (con el sufijo `.chm`), puede sustituir el archivo `.chm` predeterminado por una versión localizada. Para obtener más información sobre los sistemas de ayuda HTML, consulte "Ayuda HTML" en la página 163.

Para crear un archivo `.chm` localizado:

1. Cree versiones traducidas de los archivos de temas de ayuda individuales (`.htm` o `.html`) que conforman el sistema de ayuda, manteniendo los mismos nombres de archivo.
2. Si lo desea, utilice versiones localizadas de gráficos incluidos en el sistema de ayuda (por ejemplo, capturas de pantalla).
3. Utilice Microsoft HTML Help Workshop u otra herramienta de autoría de ayuda para compilar los archivos en el archivo `.chm` localizado.
4. Compruebe el sistema de ayuda con el nodo localizado. Para obtener más información, consulte el tema "Comprobación de un nodo localizado de CLEF".

Localización de JavaHelp

Si la extensión que está localizando utiliza un sistema JavaHelp, deberá proporcionar versiones localizadas de los archivos de origen de ayuda para cada idioma admitido. JavaHelp se encarga de mostrar la versión localizada correcta si existe. Para obtener más información, consulte el tema "JavaHelp" en la página 163.

Para crear un sistema JavaHelp localizado:

1. Cree versiones traducidas de los archivos de temas de ayuda individuales (`.htm` o `.html`) que conforman el sistema de ayuda, manteniendo los mismos nombres de archivo.
2. Si lo desea, utilice versiones localizadas de gráficos incluidos en el sistema de ayuda (por ejemplo, capturas de pantalla).
3. Genere el archivo helpset y otros archivos obligatorios (archivos de correlación, contenidos e índice).
4. Compruebe el sistema de ayuda con el nodo localizado. Para obtener más información, consulte el tema "Comprobación de un nodo localizado de CLEF".

Comprobación de un nodo localizado de CLEF

Para comprobar un nodo localizado y su sistema de ayuda:

1. En la sección Resources del archivo de especificación del nodo localizado, cambie el atributo `path` del elemento `HelpInfo` para hacer referencia al archivo localizado `.chm` o `.hs`. Por ejemplo, para la ayuda HTML puede utilizar:

   ```
   <Resources>
    ...
        <HelpInfo id="ayuda" type="AyudaHTML" path="help/minodo_fr.chm "/>
   </Resources>
   ```

   En JavaHelp puede utilizar:

```
<Resources>
 ...
     <HelpInfo id="ayuda" type="javahelp" helpset="help/minodo_fr.hs "/>
</Resources>
```

2. Copie el archivo localizado `.chm` o `.jar` en la ubicación indicada en el atributo `path`.
3. Aplique la configuración regional que desee en Windows:

   Panel de control > **Configuración regional y de idioma** > **Opciones regionales** > **Estándares y formatos** > **<idioma>**
4. Inicie IBM SPSS Modeler, asegurándose de que aparece en el idioma que desea.
5. Añada el nodo localizado a IBM SPSS Modeler. Para obtener más información, consulte el tema "Prueba de una extensión de CLEF" en la página 199.
6. Coloque una copia del nodo en el lienzo.

   Abra el cuadro de diálogo del nodo y compruebe que aparece en el idioma que desea correctamente.
7. Pulse en el botón Ayuda del cuadro de diálogo y asegúrese de que el tema de ayuda correcto aparece en el idioma que desea.

Accesibilidad

Los nodos de CLEF se benefician de todas las características estándar de accesibilidad de IBM SPSS Modeler, como los atajos de teclado para acciones del ratón y compatibilidad con lectores de pantallas.

Además, puede proporcionar un nodo de CLEF con texto personalizado de información sobre herramientas con fines de accesibilidad.

También puede especificar combinaciones de teclado para proporcionar a los usuarios finales acceso alternativo a varias características de interfaz de usuario que haya añadido en CLEF. Para obtener más información, consulte el tema "Claves de acceso y atajos de teclado" en la página 114.

En el caso de los botones de acción y los componentes de pantalla clasificados como controladores (como casillas de verificación o grupos de botones de radio), puede definir:

- label
- description

La **etiqueta** es el texto que aparecen en pantalla con el nombre del componente que el software de lector de pantallas puede leer. En el caso de usuarios con problemas de visión, puede cambiar el tamaño de fuente de la etiqueta a través de controles de la pestaña Representación dentro del cuadro de diálogo Opciones de usuario, que puede obtener desde:

Herramientas > **Opciones de usuario**

La **descripción** es el texto de información sobre herramientas que se muestra cuando el puntero del ratón pasa por encima del componente. La información sobre herramientas proporciona información acerca del componente que puede expresarse simplemente por el nombre. Un lector de pantallas también puede leer esta información sobre herramientas si está configurado adecuadamente.

Las etiquetas y las descripciones se definen mediante los atributos `label` y `description` del elemento que define el componente del archivo de especificación. Ambas pueden localizarse a través de los atributos `labelKey` y `descriptionKey`, respectivamente.

Ejemplo

Este ejemplo de un botón de acción muestra el uso de las características de etiqueta y descripción.

```
<Action id="ValorEstablecido" label="Valor establecido..." labelKey="ValorEstablecido.LABEL"
     description="Establece un valor" descriptionKey="ValorEstablecido.TOOLTIP"/>
```

Capítulo 9. Programación

Acerca de la programación de nodos CLEF

Para activar un nodo y realizar un procesamiento que no se define en el archivo de especificación, CLEF ofrece las siguientes interfaces de programación de la aplicación (API) a la que sus programas puede realizar llamadas:

- **API de cliente**. Una API de Java que pueden utilizar las extensiones que requieren controles adicionales, componentes de interfaz de usuario o interactividad que el archivo de especificación no proporciona directamente.
- **API de servidor Predictive (PSAPI)**. Una API de Java que contiene las funciones de IBM SPSS Modeler para el uso de aplicaciones que requieren minería de datos y capacidades de análisis predictivos. La PSAPI y el conjunto de programas de minería de datos de IBM SPSS Modeler comparten la misma tecnología subyacente.
- **API de servidor**. Una API basada en C trata aspectos como la configuración y ejecución de los parámetros, la persistencia de los parámetros, comentarios sobre la ejecución, control de trabajos (por ejemplo, interrupción de la ejecución), generación de SQL y objetos devueltos.

Documentación de la API CLEF

Las siguientes secciones proporcionan una visión general de las API de cliente y de servidor. Se incluye documentación más completa sobre la API en un archivo zip de la instalación de IBM SPSS Modeler que se debe extraer antes de poder utilizarlo.

Para extraer la documentación de la API:

1. Localice el archivo *clef_apidoc.zip* del producto DVD, en la carpeta *\Documentation\en*.
2. Con WinZip o una herramienta similar, extraiga el contenido del archivo zip en cualquier directorio conveniente. Así se creará una subcarpeta *clef_apidoc* en ese directorio que contenga toda la documentación de la API.

Para ver la documentación de la API:

1. Desplácese a la subcarpeta *clef_apidoc* y abra el archivo *clef_apidoc.htm*.
2. Seleccione la opción de servidor o de cliente/PSAPI que desee.

API de cliente

CLEF ofrece un número de clases de Java que contienen métodos que puede utilizar para el procesamiento de cliente. Por ejemplo, la clase `DataModelProvider` permite calcular el modelo de datos de salida donde los cambios del modelo de datos de entrada son demasiado complejos para utilizar las características que ofrece el archivo de especificación.

Clases de API de cliente

Las clases del lado del cliente son las siguientes.

Tabla 40. Clases de API del lado del cliente

Clase	Descripción
`ActionHandler`	Permite a la extensión gestionar acciones solicitadas por el usuario a través de las opciones del menú y los botones de la barra de herramientas

Tabla 40. Clases de API del lado del cliente (continuación)

Clase	Descripción
`DataModelProvider`	Permite nodos que realizan cambios complejos en el modelo de datos para utilizar Java en el cálculo del modelo de datos de salida
`ExtensionObjectFrame`	Define las funciones asociadas a las ventanas utilizadas pera visualizar los resultados de los documentos o modelos
`ExtensionObjectPanel`	Define las funciones asociadas a los paneles del objeto de extensión
`PropertyControl`	Define las funciones asociadas a un control de propiedad personalizado en un panel de propiedades

La documentación de la API de cliente incluye toda la información sobre estas clases. Para obtener más información, consulte el tema "Documentación de la API CLEF" en la página 175.

Uso de la API de cliente

Para incluir las llamadas de funciones de cliente en un nodo CLEF:

1. Cree los archivos de origen *.java* que contengan las llamadas de funciones.
2. Compile los archivos de origen en archivos *.class*.
3. Puede combinar varios archivos *.class* en un archivo *.jar* e incluir una referencia al archivo *.jar* en el archivo de especificación, por ejemplo:

   ```
   <Resources>
    ...
       <JarFile id="selfjar" path="selflearning.jar"/> ...
   </Resources>
   ```

   Algunos elementos CLEF permiten hacer referencia a una clase de forma explícita. Por ejemplo, puede incluir una referencia de clase en el atributo `controlClass` de un elemento `PropertyControl` en el archivo de especificación, como se muestra a continuación:

```
<PropertyControl property="target_field_values_specify" ...
     controlClass="com.spss.clef.selflearning.propertycontrols.list.CustomListControl" ... />
```

donde `CustomListControl` es el nombre de la clase que implementa el control de la propiedad y `com.spss.clef.selflearning.propertycontrols.list` es la ruta a una clase dentro del archivo *.jar* declarado en el elemento `JarFile`.

Para obtener más información, consulte el tema "Sección de recursos" en la página 34.

También puede encontrarlo útil para ver el código de origen de los nodos de ejemplo que se proporcionan con esta versión. Para obtener más información, consulte el tema "Evaluación del código fuente" en la página 28.

API de servidor Predictive (PSAPI)

La PSAPI proporciona una interfaz de programación a la tecnología subyacente del servidor Predictive. Los elementos principales de la PSAPI se especifican como interfaces de Java. La mayoría de estas interfaces se implementan a través de clases internas proporcionadas por la PSAPI pero que no forman parte de la especificación de la PSAPI. Este método tiene como objetivo proteger al usuario de la PSAPI de los cambios introducidos por la tecnología del servidor Predictive (como cambios de arquitectura, cambios de protocolo de cliente/servidor privado, etc.).

Se ofrecen todos los detalles de estas clases en la documentación de la PSAPI. Para obtener más información, consulte el tema "Documentación de la API CLEF" en la página 175.

API de servidor

La API de servidor se define como una API de lenguaje C pero admitirá implementaciones en C++. El desarrollador de un módulo de extensión puede seleccionar que se programe directamente con la API de lenguaje C programándolo en C o C++. Se pueden utilizar otros lenguajes si el desarrollador dispone de un método de enlace a la API de C. CLEF también proporciona varios archivos de origen de aplicación de ayuda C++ que funcionan como encapsuladores de algunas de las API basadas en C.

Arquitectura

Un **nodo** de extensión sobre el cliente se complementa con un **homólogo** de extensión en el servidor. Un homólogo viene definido por un **módulo de extensión**, que se implementa como una biblioteca compartida alojada en el servidor. La comunicación entre un nodo y su homólogo viene mediada por un **recurso** de extensión gestionado por el servidor. Un recurso llama las **funciones de servicio** definidas por el módulo de extensión para crear y manipular sus homólogos, mientras que el homólogo utiliza **funciones de devolución de llamada** para solicitar información y servicios del servidor.

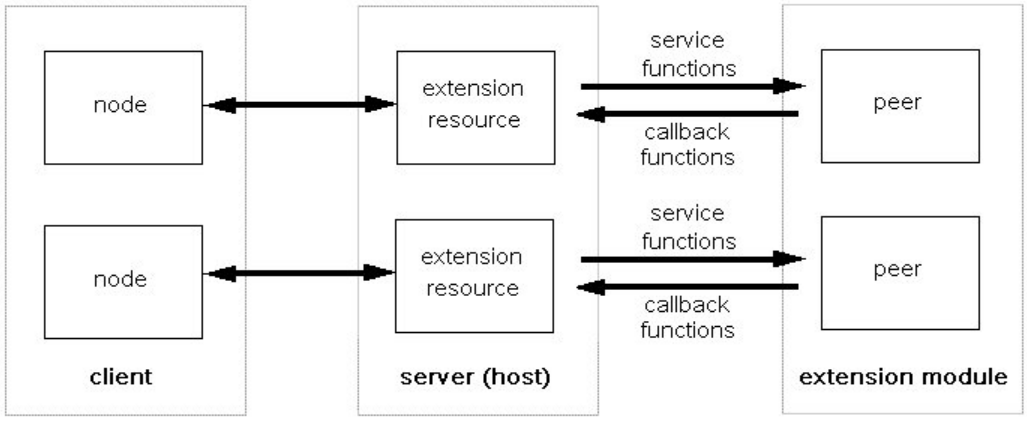

Figura 95. Arquitectura de la API CLEF

Funciones de servicio

El módulo de extensión implementa las funciones de servicio. Un módulo de extensión debe implementar todas las funciones marcadas como obligatorias y puede implementar una o todas las que no están marcadas como tal.

Hay dos tipos de funciones de servicio:

- Funciones de módulo
- Funciones de homólogo

Las siguientes secciones proporcionan conceptos básicos sobre las funciones de servicio. Las descripciones más detalladas de estas funciones se pueden encontrar en la documentación de la API de servidor, como se indica a continuación:

1. Desde la pantalla de documentación de la API CLEF, seleccione **Conceptos básicos de la API de servidor**.
2. Pulse en la pestaña **Módulos**.
3. Seleccione **Funciones del servicio de la API que el módulo de extensión debe implementar**.

Para obtener información sobre el acceso a la documentación de la API CLEF, consulte “Documentación de la API CLEF” en la página 175.

Funciones de módulo

Las funciones de módulo se llaman desde un subproceso individual.

Tabla 41. Funciones de módulo

Función	¿Necesario?	Descripción
`clemext_initialise`	Sí	Inicializa un módulo de extensión
`clemext_cleanup`	Sí	Elimina los recursos asignados por un módulo de extensión
`clemext_getModuleInformation`	No	Obtiene información sobre un módulo de extensión
`clemext_create_peer`	Sí	Crea una instancia de homólogo para un nodo de extensión
`clemext_destroy_peer`	Sí	Elimina una instancia de homólogo

Funciones de homólogo

Las funciones se aplican a un controlador de instancia de homólogo devuelto por una llamada previa a `clemext_create_peer`. Las funciones de homólogo se pueden llamar simultáneamente a partir de subprocesos individuales sólo cuando los controladores de homólogo son distintos. La única excepción es que la función `clemext_peer_cancelExecution` (si se ha definido) puede llamarse desde cualquier subproceso en cualquier momento para interrumpir una ejecución de larga duración en un subproceso diferente.

Tabla 42. Funciones de homólogo

Función	¿Necesario?	Descripción
`clemext_peer_configure`	Sí	Indica una instancia de homólogo para procesar sus parámetros y el modelo de datos
`clemext_peer_getDataModel`	Sí	Obtiene el modelo de datos de salida desde una instancia de homólogo
`clemext_peer_getCatalogueInformation`	No	Obtiene información de catálogo desde un módulo
`clemext_peer_getExecutionRequirements`	No	Obtiene los requisitos de ejecución de una instancia de homólogo
`clemext_peer_beginExecution`	Sí	Inicia la ejecución de una instancia de homólogo
`clemext_peer_nextRecord`	Sí	Se desplaza al siguiente registro en un conjunto de resultados de un homólogo
`clemext_peer_getRecordValue`	Sí	Devuelve el valor de un campo específico dentro del registro de resultados obtenido últimamente
`clemext_peer_endExecution`	Sí	Indica a una instancia de homólogo que la ejecución se ha completado
`clemext_peer_cancelExecution`	No	Se llama desde un subproceso individual para interrumpir una llamada de función de larga duración`beginExecution` o `nextRecord`
`clemext_peer_getSQLGeneration`	No	Genera SQL desde una instancia de homólogo
`clemext_peer_getErrorDetail`	Sí	Recupera información adicional sobre el último error específico del modulo en un homólogo

Las funciones de homólogo que se muestran en la tabla siguiente se designan para utilizarlas con un generador de modelos interactivos.

Tabla 43. Funciones de homólogo utilizadas para el uso con un generador de modelos interactivos

Función	¿Necesario?	Descripción
`clemext_peer_beginInteraction`	No	Inicia la interacción con una instancia de homólogo
`clemext_peer_request`	No	Realiza una solicitud interactiva en un homólogo
`clemext_peer_getRequestReply`	No	Recupera la respuesta desde la última solicitud ejecutada correctamente
`clemext_peer_endInteraction`	No	Indica a una instancia de homólogo que la interacción se ha completado

Funciones de devolución de llamada

Cuando un módulo de extensión requiere información o servicio del proceso del host, se debe realizar mediante una **devolución de llamada**. Una devolución de llamada se aplica a un **controlador**, que es un indicador que identifica el objetivo de la solicitud.

Las devoluciones de llamadas se invocan mediante el controlador del objeto de IBM SPSS Modeler al que va dirigida la llamada. Los controladores se introducen en un módulo de extensión como parámetros en funciones de servicio.

Si se produce un fallo en la función de devolución de llamada, se deberían devolver más detalles en el código de error específico del módulo asociado (que se indica por `CLEMEXTErrorCode`). El módulo puede gestionarlo por turnos indicando un error de devolución de llamada y pasando estos datos de manera que el host pueda inspeccionarlos.

Están disponibles los siguientes tipos de funciones de devolución de llamada:

- Funciones de host
- Funciones de nodo
- Funciones de iterador
- Funciones de progreso
- Funciones de canal (sólo para modelos interactivos)

Las siguientes secciones proporcionan conceptos básicos sobre las funciones de devolución de llamada. Las descripciones más detalladas de estas funciones se pueden encontrar en la documentación de la API de servidor, como se indica a continuación:

1. Desde la pantalla de documentación de la API CLEF, seleccione **Conceptos básicos de la API de servidor**.
2. Pulse en la pestaña **Módulos**.
3. Seleccione **Devoluciones de llamadas generales**.

Para obtener información sobre el acceso a la documentación de la API CLEF, consulte “Documentación de la API CLEF” en la página 175.

Funciones de host

Las funciones de host se definen en un controlador de host pasado por `clemext_initialise`.

Tabla 44. Funciones de host

Función	Descripción
`clemext_host_getHostInformation`	Obtiene información estática sobre el entorno de host

Funciones de nodo

Las funciones de nodo se definen en un controlador de nodo que pasa por `clemext_create_peer`.

Tabla 45. Funciones de nodo

Función	Descripción
`clemext_node_getNodeInformation`	Obtiene información estática sobre un nodo
`clemext_node_getParameters`	Obtiene parámetros desde un nodo
`clemext_node_getDataModel`	Obtiene el modelo de datos de entrada para un nodo
`clemext_node_getOutputDataModel`	Obtiene el modelo de datos de salida para un nodo
`clemext_node_getSQLGeneration`	Obtiene la información de generación de SQL anterior de la ruta para un nodo
`clemext_node_getPassword`	Recupera el texto plano de un identificador de contraseña
`clemext_node_getFilePath`	Recupera la ruta de un archivo que se ha intercambiado entre cliente y servidor durante la ejecución

Funciones de iterador

Las funciones de iterador se definen en un controlador de iterador pasado por `clemext_peer_beginExecution`. Un iterador expone un conjunto de datos de entrada para un módulo de extensión.

Tabla 46. Funciones de iterador

Función	Descripción
`clemext_iterator_nextRecord`	Se desplaza al siguiente registro en el conjunto de datos de entrada
`clemext_iterator_getRecordValue`	Devuelve el valor de un campo específico dentro del registro de entrada obtenido últimamente
`clemext_iterator_rewind`	Restaura el estado del conjunto de datos de entrada que la siguiente llamada a `nextRecord` desplazará al primer registro del conjunto

Funciones de progreso

Las funciones de progreso se definen en un controlador de progreso pasado por `clemext_peer_beginExecution`.

Tabla 47. Funciones de progreso

Función	Descripción
`clemext_progress_report`	Informa del progreso al host

Funciones de canal

Las funciones de canal se utilizan sólo con modelos interactivos y se definen en un controlador de canal pasado por `clemext_peer_beginInteraction`.

Tabla 48. Funciones de canal

Función	Descripción
`clemext_channel_send`	Envía un mensaje asíncrono al cliente, permitiendo que un subproceso de homólogo informe del progreso y el estado

Flujo de proceso

Un módulo de extensión llama a un número de servicio y a las funciones de devolución de llamada para realizar su procesamiento. Las funciones reales que se llaman dependen del procesamiento que el módulo debe realizar.

Ejemplo

El diagrama de secuencia de una ejecución de módulo típico es el que se muestra en la ilustración siguiente.

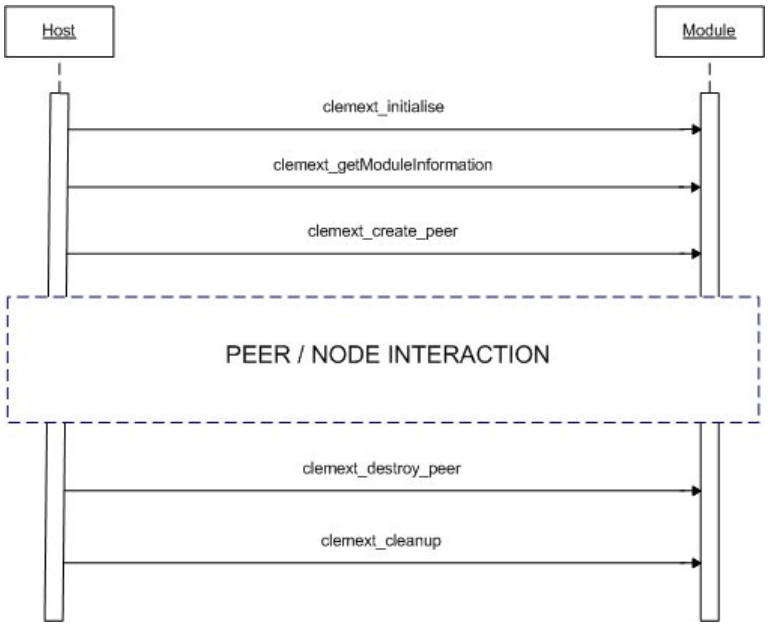

Figura 96. Flujo de proceso típico

El bloque de interacción de nodo/homólogo se muestra en la siguiente ilustración.

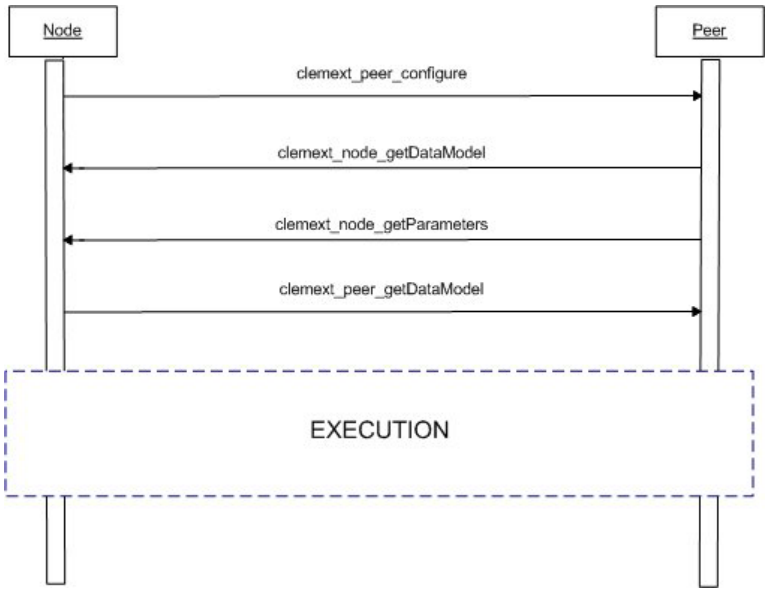

Figura 97. Bloque típico de interacción de nodo/homólogo

En la siguiente ilustración se muestra un bloque de ejecución típico.

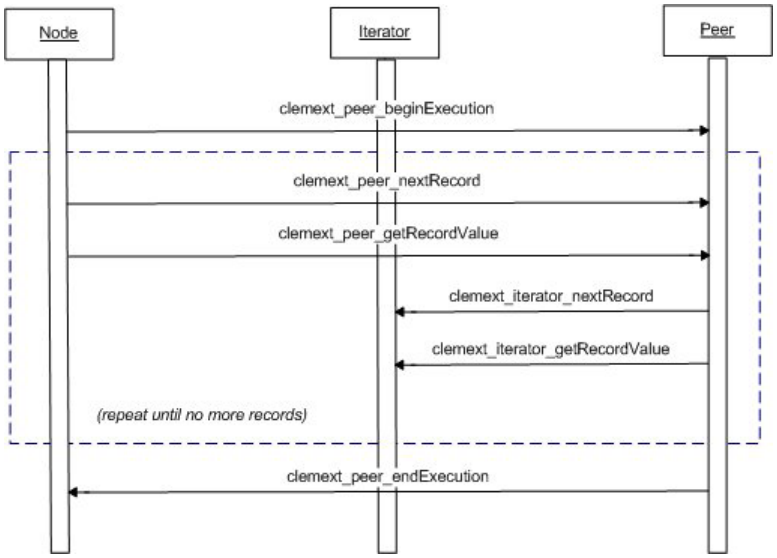

Figura 98. Bloque de ejecución típico

Notas:

- Se puede cargar un módulo en el proceso de servidor de IBM SPSS Modeler en el momento que se inicia el servidor o se puede cargar posteriormente bajo petición cuando se requiera su servicio por primera vez.
- Cuando el módulo se haya cargado, el host activa la función de servicio `clemext_initialise` una vez.
- Cuando el módulo se haya cargado e inicializado, es posible que el host consulte el módulo que utiliza la función de servicio `clemext_getModuleInformation`.
- Una vez cargado el módulo, se activan sus servicios a través de objetos de homólogo que proporciona el módulo. Dentro del módulo, el objeto de homólogo se crea por la función de servicio `clemext_create_peer` como homólogo al objeto del nodo del host para gestionar la ejecución de una tarea dirigida por la aplicación del host. Pueden existir varios objetos de homólogo del mismo tipo y ejecutarse simultáneamente dentro de un mismo proceso.
- Cuando se crea un objeto de homólogo, se puede configurar mediante la función de servicio `clemext_peer_configure`.
- En este punto, el homólogo puede ejecutar funciones de devolución de llamada para obtener información del cliente, como `clemext_node_getDataModel` y `clemext_node_getParameters`.
- IBM SPSS Modeler obtiene el modelo de datos de salida de la instancia de homólogo por medio de una función de servicio `clemext_peer_getDataModel`.
- La ejecución de la instancia de homólogo comienza por la función de servicio `clemext_peer_beginExecution`.
- La función de servicio `clemext_peer_nextRecord` desplaza el foco al siguiente registro en el conjunto de resultados del homólogo (o al primer registro si la función se llama por primera vez). Le sigue la función de servicio `clemext_peer_getRecordValue`, que devuelve el valor de un campo específico dentro del registro actual.
- Las funciones de devolución de llamada de iterador `clemext_iterator_nextRecord` y `clemext_iterator_getRecordValue` se pueden llamar por el módulo CLEF para secuenciar a través de registros de entrada y devolver valores de campo específicos.
- La ejecución de la instancia de homólogo finaliza con una llamada a la función de servicio `clemext_peer_endExecution`.
- La instancia de homólogo se elimina llamando a la función `clemext_destroy_peer`.

- Antes de descargar el módulo, el host activa la función de servicio `clemext_cleanup`.
- Es posible que se descargue un módulo cuando el proceso del servidor se desconecta o antes cuando ya no se requieren sus servicios.

Características de la API de servidor

Esta sección destaca algunas de las características de la API de servidor:

- Información de tipo de nodo
- Tipos de datos representando diferentes tipos de almacenamiento de datos
- Bibliotecas compartidas del servidor
- Filespaces y archivos temporales
- Retrotracción SQL para ejecutar instrucciones SQL en la base de datos
- Intercambio de información de modelo de datos entre IBM SPSS Modeler y la extensión
- Documentos de resultado
- Aplicaciones de ayuda C++

Tipos de nodo

En el archivo de especificación, una definición de nodo adopta el siguiente formato:

```
<Node id="identificador" type="tipo_nodo" .../>
```

El atributo `id` es una cadena que identifica de forma exclusiva al nodo.

El atributo `type` identifica el nodo como uno de los siguientes tipos:

- Lector de datos
- Escritor de datos
- Transformador de datos
- Generador de modelos
- Aplicador de modelos
- Generador de documentos

Para obtener más información, consulte el tema “Conceptos básicos sobre nodos” en la página 9.

La función `clemext_create_peer` incluye los valores de los atributos `id` y `type` del elemento `Nodo` como argumentos.

Un módulo de una sola extensión puede implementar nodos de varios tipos, ejecutando varias funciones dentro de cada tipo. Por ejemplo, un módulo puede implementar:

- Un lector de datos y un escritor de datos para un origen de datos
- Generadores de modelo y aplicadores de modelo de varios algoritmos de modelado
- Generadores de documentos para varios tipos de gráficos

Tipos de almacenamiento y datos

Una instancia de homólogo obtiene datos de entrada llamando `clemext_iterator_getRecordValue` en el iterador proporcionado al inicio de la ejecución y proporciona datos de resultados como respuesta a una solicitud de `clemext_peer_getRecordValue` del host. Los datos se transfieren en formato binario a la memoria y el homólogo y el host deben coincidir en el tipo de datos.

El tipo de datos binario se determina por el modelo de datos y está relacionado con el atributo de almacenamiento de un campo.

La siguiente tabla enumera los posibles tipos de almacenamiento junto con los tipos de datos utilizados para representarlos.

Tabla 49. Tipos de almacenamiento

Tipo de almacenamiento	Representado por:	Notas
cadena	`char *`	Las cadenas de caracteres siempre se codifican en UTF-8
real	`CLEMEXTReal`	Número de punto flotante de precisión doble
entero	`CLEMEXTInteger`	Entero con signo de 64 bits
fecha	`CLEMEXTDate`	Entero con signo de 64 bits que representa el número de días desde el 1 de enero de 1970
hora	`CLEMEXTTime`	Entero con signo de 64 bits que representa el número de segundos desde la medianoche
marca de tiempo	`CLEMEXTTimestamp`	Entero con signo de 64 bits que representa el número de segundos desde la medianoche del 1 de enero de 1970
unknown	-	Indica un tipo de datos desconocidos: los valores no se pueden representar

Bibliotecas

Una biblioteca compartida del servidor puede especificarse en el archivo de especificación para permitir la ejecución del nodo. La ruta de la biblioteca compartida se utiliza para localizar una biblioteca compartida que se carga dinámicamente en el proceso del host. La biblioteca compartida debe definir todas las funciones de la API necesarias. Para obtener más información, consulte el tema "Bibliotecas compartidas" en la página 36.

Si se proporciona un nombre de módulo en el archivo de especificación (en la sección `Execution` de la definición del nodo), el nombre se pasa al parámetro `nodeId` de la función de servicio `clemext_create_peer` para crear un objeto de homólogo. De esta forma, la extensión puede crear un módulo de homólogo de la clase apropiada. El valor parámetro `nodeType` también puede influir en la clase de homólogo que se crea. El nombre del módulo puede estar en blanco puesto que una biblioteca compartida no puede implementar más de un módulo de cada tipo.

Es posible que se necesiten bibliotecas dependientes para una biblioteca compartida que implemente un módulo de extensión. Se deben colocar en el mismo directorio que la biblioteca compartida de extensión.

Archivos temporales

El archivo de especificación de cliente y el módulo de extensión del servidor pueden especificar nombres de rutas relativos a **filespace**, un espacio privado temporal donde un homólogo puede crear archivos para utilizarlos durante su ejecución. Un filespace es un subdirectorio del directorio temporal del servidor creado por un homólogo. Se crea según sea necesario y se elimina cuando el homólogo se destruye.

El homólogo tiene control completo sobre el filespace mientras exista. El nombre de ruta completo del filespace se incluye en un **documento de información de nodo**. Esta información está en formato XML que se devuelve como resultado de la ejecución de una función de devolución de llamada `clemext_node_getNodeInformation`. Para obtener más información, consulte el tema "Documento de información de nodo" en la página 191.

Puntos de retrotracción de SQL

Donde una ruta de IBM SPSS Modeler lee datos de una base de datos SQL y realiza procesamiento de datos, los usuarios avanzados pueden mejorar la eficacia de esta operación con la retrotracción de instrucciones SQL para ejecutarla en la propia base de datos.

Varios nodos IBM SPSS Modeler estándar admiten la retrotracción SQL y la API de servidor incluye llamadas de funciones para que también la admitan los nodos CLEF.

La función de servicio clemext_peer_getSQLGeneration genera SQL a partir de una instancia de homólogo y se utiliza para retrotraer la ejecución de SQL a la base de datos. Para un nodo de lector de datos, el SQL generado debe ser suficiente por sí mismo para crear el conjunto de resultados del homólogo. Para cualquier otro tipo de nodo, el SQL generado dependerá con mayor probabilidad del SQL generado para los nodos anteriores de la ruta que proporcionan entrada al homólogo. Un homólogo puede obtener el SQL anterior de la ruta llamando la función de devolución de llamada clemext_node_getSQLGeneration en su controlador de nodo asociado.

Tratamiento del modelo de datos

Algunas de las llamadas de la API del servidor relacionadas con el intercambio de información de modelo de datos entre IBM SPSS Modeler y el módulo de extensión:

- clemext_node_getDataModel obtiene el modelo de datos de entrada para un nodo
- clemext_peer_getDataModel obtiene el modelo de datos de salida a partir de una instancia de homólogo
- clemext_node_getOutputDataModel obtiene el modelo de datos de salida para un nodo

Otras llamadas relacionadas con los métodos para leer datos dentro y fuera del módulo. El modelo de datos determina el índice utilizado para la búsqueda de valores de campo en las siguientes funciones, que devuelven un valor de un campo específico dentro del registro de entrada obtenido últimamente:

- clemext_peer_getRecordValue
- clemext_iterator_getRecordValue

IBM SPSS Modeler llama a clemext_node_getDataModel para obtener información sobre los campos del modelo de datos de entrada. La información se devuelve en formato XML; por ejemplo:

```
<DataModel>
     <Fields>
          <Field name="abc" storage="string" type="set" />
          <Field name="uvw" storage="integer" type="range" />
          <Field name="xyz" storage="real" type="range" />
     <Fields>
</DataModel>
```

Un módulo puede utilizar esta información para proporcionar el índice del campo cuando recupera los valores de un registro de entrada mediante la función clemext_iterator_getRecordValue.

La forma en la que el módulo afecta al modelo de datos de entrada se controla por el valor del atributo mode del elemento OutputDataModel en el archivo de especificación. El módulo puede:

- Ampliar el modelo añadiéndole nuevos campos.
- Modificar el modelo eliminando o cambiando el nombre a campos existentes.
- Sustituir el modelo existente con nuevos campos.
- Dejar el modelo sin modificar.

Los siguientes ejemplos muestran la ampliación y sustitución del modelo.

Ejemplo: ampliación del modelo de datos de entrada: Es el caso más fácil: permitir que un módulo añada nuevos campos y defina sus valores pero no elimine o cambie los valores de campos existentes.

Suponga que el archivo de especificación contiene las siguientes instrucciones en la definición del nodo:

```
<OutputDataModel mode="extend">
     <AddField name="field1" storage="string" ... />
     <AddField name="field2" storage="real" ... />
     ...
</OutputDataModel>
```

Aquí, el modelo de datos de salida se define como contenedor de todos los campos en el modelo de datos de entrada además de dos campos adicionales especificados en el elemento `OutputDataModel`. Así, el modelo de datos de salida consta de cinco campos.

La función `clemext_peer_getDataModel` devuelve la información sólo en los campos añadidos, por ejemplo:

```
<DataModel>
     <Fields>
          <Field name="campo1" storage="string" ... />
          <Field name="campo2" storage="real" ... />
     <Fields>
</DataModel>
```

La información obtenida debe coincidir con el tipo y los valores de número (pero no el nombre) de los elementos `<AddField>` en el archivo de especificación.

Un módulo puede utilizar la función de devolución de llamada `clemext_node_getOutputDataModel` para obtener los detalles de los campos que IBM SPSS Modeler espera que se añadan. Esta información puede volver a IBM SPSS Modeler como respuesta a una llamada a `clemext_peer_getDataModel`. Este procedimiento es útil para las situaciones donde la lógica del archivo de especificación para crear y nombrar campos de salida es complicada.

El módulo proporciona los nuevos valores de cada registro de salida cuando IBM SPSS Modeler llama a `clemext_peer_getRecordValue`. Los índices de campo para los nuevos campos empiezan después del último índice de los campos de entrada. En este ejemplo, el modelo de datos de entrada contiene tres campos (en las posiciones de índice 0, 1 y 2), de manera que los dos campos de salida se asignan a índices de campo 3 y 4. IBM SPSS Modeler no llamará a `clemext_peer_getRecordValue` con los índices de campo correspondientes a los campos de entrada porque el módulo no puede cambiar estos campos.

Ejemplo: sustitución del modelo de datos de entrada (1): En este ejemplo, el módulo de extensión descarta todos los campos de modelo de datos de entrada desde su salida, sustituyéndolos por nuevos campos.

El contenido del archivo de especificación es el siguiente:

```
<OutputDataModel mode="modify">
     <AddField name="key" storage="integer" ... />
     <AddField name="campo1" storage="real" ... />
     <AddField name="campo2" storage="real" ... />
     ...
</OutputDataModel>
```

Esta vez, los datos XML devueltos por una llamada a `clemext_peer_getDataModel` describen todos los campos del modelo de datos de salida:

```
<DataModel>
     <Fields>
          <Field name="key" storage="integer" ... />
          <Field name="campo1" storage="real" ... />
          <Field name="campo2" storage="real" ... />
     <Fields>
</DataModel>
```

Los índices de campo utilizados en las llamadas a `clemext_peer_getRecordValue` empiezan en 0 para el primer campo de salida (`key`), 1 para el siguiente campo (`campo1`) y así sucesivamente.

Ejemplo: sustitución del modelo de datos de entrada (2): En este ejemplo, el modelo de datos de salida proporcionado por la extensión también sustituye el modelo de datos de entrada, como en el ejemplo anterior. No obstante, en este caso, el modelo de datos de salida no se define en el archivo de

especificación pero, en su lugar, se calcula en el tiempo de procesamiento por el módulo de extensión del servidor. El contenido del archivo de especificación es el siguiente:

```
<OutputDataModel mode="modify" method="sharedLibrary" libraryId="myLibraryId" />
```

Para calcular el modelo de datos de salida, IBM SPSS Modeler primero llama a `clemext_peer_configure`, seguido de `clemext_peer_getDataModel`. Como en el ejemplo anterior, ninguno de los campos del modelo de datos de entrada se incluye automáticamente en el modelo de datos, que se define completamente por la respuesta de `clemext_peer_getDataModel`.

Note: En estos casos, donde el módulo de extensión define el modelo de datos de salida en el servidor, el módulo no puede utilizar `clemext_node_getOutputDataModel` para obtener el modelo de datos de salida porque resultaría en un error de operación no válida.

Documentos de resultado XML

Algunas funciones de devolución de llamada transfieren información entre el host y el módulo de extensión en forma de documentos de resultado XML. Hay diversos documentos diferentes disponibles, que se muestran en la tabla siguiente.

Tabla 50. Documentos de salida XML

Documento de resultado XML	**Notas**	**Devuelto por una llamada a...**
Documento de catálogo	Contiene la lista de valores de un control asociados con un catálogo.	`clemext_peer_getCatalogInformation`

Documento de modelo de datos	Describe el conjunto de campos entrantes y salientes de un nodo	`clemext_peer_getDataModel`
		`clemext_node_getDataModel`
		`clemext_node_getOutputDataModel`

Documento de detalles de error	Proporciona información sobre un error u otra condición	`clemext_peer_getErrorDetail`
Documento de requisitos de ejecución	Describe los requisitos de ejecución necesarios para una instancia de homólogo, como una caché de datos o campos de entrada obligatorios.	`clemext_peer_getExecutionRequirements`
Documento de información de host	Proporciona información sobre el entorno de host, entre la que se incluye: detalles de identificador de producto, descripción, versión, proveedor, copyright y plataforma	`clemext_host_getHostInformation`
Documento de información de módulo	Proporciona información sobre el módulo de extensión, entre la que se incluye: detalles de identificador de módulo, descripción, versión, proveedor, copyright y licencia	`clemext_getModuleInformation`
Documento de información de nodo	Proporciona información sobre el nodo asociado a una instancia de homólogo, entre la que se incluye: detalles de identificador de nodo, tipo y filespace	`clemext_node_getNodeInformation`
Documento de parámetros	Contiene parámetros de configuración del nodo de cliente; el contenido se determina por la extensión	`clemext_node_getParameters`
Documento de generación de SQL	Describe cómo se puede traducir la ejecución de un homólogo a SQL	`clemext_peer_getSQLGeneration` `clemext_node_getSQLGeneration`

Tabla 50. Documentos de salida XML (continuación)

Documento de resultado XML	Notas	Devuelto por una llamada a...
Documento de detalles de estado	Proporciona información adicional sobre el progreso y advertencias u otras condiciones surgidas durante la ejecución	`clemext_progress_report`

Documento de catálogo: Un documento de catálogo describe el contenido de un catálogo, que contiene una lista de valores que se pueden mostrar desde un control de IU.

El módulo de CLEF realiza una llamada a `getCatalogInformation` de la siguiente manera:

```
CLEMEXTStatus
getCatalogInformation(
          const char *catalogId,
            char* buffer,
            size_t buffer_size,
            size_t* data_size,
            CLEMEXTErrorCode* errorCode) {
      ...
}
```

donde `catalogId` es el identificador de un catálogo concreto, tal y como se define en el elemento `Catalog` del archivo de especificación.

Esta función devuelve el documento del catálogo.

Ejemplo

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<CatalogInformation>
     <CatalogEntry>
          <CatalogValue>apples</CatalogValue>
          <CatalogValue>0</CatalogValue>
     </CatalogEntry>
     <CatalogEntry>
          <CatalogValue>oranges</CatalogValue>
          <CatalogValue>1</CatalogValue>
     </CatalogEntry>
     <CatalogEntry>
          <CatalogValue>bananas</CatalogValue>
          <CatalogValue>2</CatalogValue>
     </CatalogEntry>
</CatalogInformation>
```

Documento de modelo de datos: Un documento de modelo de datos describe el **modelo de datos** (el conjunto de campos con sus nombres, tipos e información relacionada) entrante o saliente de un nodo. Incluye la información disponible en un nodo de tipo.

Un homólogo que no consume entrada (un nodo de origen) tiene un modelo de entrada vacío y un homólogo que no produce resultados (un nodo de terminal) tiene un modelo de salida vacío. Un homólogo que consume entrada y produce resultados (un nodo de proceso) debe saber cómo contribuye el modelo de salida a su entrada.

Un homólogo puede obtener el modelo de datos de entrada llamando a `clemext_node_getDataModel` en su controlador de nodo asociado. Un homólogo proporciona su modelo de datos de salida en respuesta a una solicitud `clemext_peer_getDataModel` del host.

Cualquier modelo de datos se puede expresar directamente como diccionario de datos que enumera todos los campos en el modelo de datos junto con sus propiedades. Un modelo de datos de entrada proporciona un nodo a un homólogo que tenga siempre la misma forma. Un modelo de datos de salida producido por un homólogo puede tener la misma forma o puede expresarse como una secuencia de operaciones (añadir campo, eliminar campo, modificar campo) aplicada al modelo de entrada. Lo que simplifica de forma significativa el modelo de salida de algunos nodos.

El orden de los campos de un documento de modelo de datos es significativo para determinar el orden en el que se presentan los datos dentro del conjunto de datos de entrada o salida correspondiente.

Un modelo de datos puede estar incompleto y proporcionar sólo una especificación parcial de los datos. Un modelo de entrada que se especifica de forma suficiente para permitir que un homólogo calcule un plan de ejecución se considera **ejecutable** de ese homólogo. Un modelo de datos ejecutable debe contener siempre el tipo binario para cada campo de manera que los datos de entrada y salida se puedan reunir correctamente.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<DataModel>
  <Fields>
    <Field name="Edad" type="range" storage="integer" direction="in">
      <Range minValue="15" maxValue="74"/>
    </Field>
    <Field name="Sexo" type="flag" storage="string">
      <Values>
        <Value value="F" flagValue="false" displayLabel="Female"/>
        <Value value="M" flagValue="true" displayLabel="Male"/>
      </Values>
    </Field>
    <Field name="BP" type="orderedSet" storage="integer">
      <Values>
        <Value value="-1" />
        <Value value="0" />
        <Value value="1" />
      </Values>
    </Field>
    <Field name="Colesterol" type="flag" storage="string">
      <Values>
        <Value value="NORMAL" flagValue="false"/>
        <Value value="HIGH" flagValue="true"/>
      </Values>
    </Field>
    <Field name="Na" type="range" storage="real" displayLabel="Sodio en sangre">
      <Range minValue="0.500517" maxValue="0.899774"/>
    </Field>
    <Field name="K" type="range" storage="real" displayLabel="Concentración de potasio">
      <Range minValue="0.020152" maxValue="0.079925"/>
    </Field>
    <Field name="Medicamento" type="set" storage="string" direction="out">
      <Values>
        <Value value="medicamentoA"/>
        <Value value="medicamentoB"/>
        <Value value="medicamentoC"/>
        <Value value="MedicamentoX"/>
        <Value value="medicamentoY"/>
      </Values>
    </Field>
  </Fields>
</DataModel>
```

Documento de detalles de error: Un documento de detalles de error se utiliza para volver a enviar mensajes (error, advertencia, información) a IBM SPSS Modeler y proporciona información sobre un error u otra condición. Un módulo de extensión puede proporcionar un documento de detalles de error para explicar un error específico de módulo en respuesta a una solicitud `clemext_peer_getErrorDetail` del host.

El detalle de error es un conjunto de uno o más elementos `Diagnostic` donde cada diagnóstico incluye al menos un código de error, un mensaje y un conjunto de uno o más parámetros que contienen más información para insertar en el mensaje. Los códigos de error coinciden con los valores en los elementos `StatusCode` del archivo de especificación.

El mensaje puede tener variantes de lenguajes diferentes o el cliente puede utilizar el código de error para seleccionar un mensaje de un paquete de recursos. Una secuencia de elementos de diagnóstico describe una cadena causal de errores.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<ErrorDetail>
  <Diagnostic code="123" severity="error">
    <Message>No puede hacerlo ({0})</Message>
    <Parameter>Permiso denegado</Parameter>
  </Diagnostic>
  <Diagnostic code="456" severity="warning">
    <Message>Fue ridículo</Message>
    <Message lang="fr">Quel idiot!</Message>
  </Diagnostic>
</ErrorDetail>
```

Documento de requisitos de ejecución: Un documento de requisitos de ejecución describe las características de ejecución necesarias para una instancia de homólogo. Una instancia de homólogo puede proporcionar un documento de requisitos de ejecución en respuesta a una solicitud `clemext_peer_getExecutionRequirements` del host. El host consulta el documento de requisitos antes de llamar a `clemext_peer_beginExecution` en el homólogo para proporcionar el entorno de ejecución adecuado.

El host puede proporcionar un servicio de caché de datos para activar el módulo para realizar muchas pasadas en los datos de entrada mediante la función `clemext_iterator_rewind`.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<ExecutionRequirements>
  <Cache/><!-- asegura que el módulo CLEF pueda realizar varias pasadas en los datos de entrada -->
</ExecutionRequirements>
```

Documento de información de host: Un documento de información de host describe el entorno de host. Un módulo de extensión puede obtener información de host llamado a `clemext_host_getHostInformation` en el controlador del host.

La información obtenida incluye detalles del identificador de producto, descripción, versión, proveedor, copyright y plataforma.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<HostInformation>
  <Host name="clemlocal" externalEncoding="cp1252" language="english_us"
        locale="English_United Kingdom.1252" provider="IBM Corp." version="16" platform=
        "Windows XP SP2" copyright="Copyright 1995-2011 IBM Corp. Todos los derechos reservados.">
```

```
    <VersionDetail major="12" minor="0"/>
    <PlatformDetail osType="windows" osName="WindowsNT" osMajor="5" osMinor="1"/>
    <LibraryDetail path="C:\Archivos de programa\IBM\SPSS\Modeler\16\ext\bin\my.module\myModule.dll"/>
  </Host>
</HostInformation>
```

Documento de información de módulo: Un documento de información de módulo describe un módulo de extensión. Un módulo de extensión debe proporcionar un documento de información de módulo en respuesta a una solicitud `clemext_getModuleInformation` del host.

La información obtenida incluye detalles del identificador de módulo, descripción, versión, proveedor, copyright y licencia.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<ModuleInformation>
  <Module name="MiModulo" provider="Mi Empresa S.A." version="10.1.0.329"
          copyright="Copyright 2006 Mi Empresa S.A. Todos los derechos reservados.">
    <VersionDetail major="10" minor="1" release="0" build="329"/>
    <Licence code="1234" type="mandatory"/>
    <Description>Proporciona una prueba exhaustiva del nuevo marco de extensiones.</Description>
  </Module>
</ModuleInformation>
```

Documento de información de nodo: Un documento de información de nodo describe el nodo asociado a una instancia de homólogo. Una instancia de homólogo puede obtener información llamando a `clemext_node_getNodeInformation` en un controlador de nodo. La información de nodo incluye detalles de identificador de nodo, tipo y filespace.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<NodeInformation>
  <Node name="databaseImport" type="dataReader">
    <FileSpace path="C:\Archivos de programaIBM SPSS Modeler Server
16\tmp\ext-8005-6711-01"/>
  </Node>
</NodeInformation>
```

Documento de parámetros: Un documento de parámetros contiene detalles de cada elemento `Property` definido en el archivo de especificación. Los detalles se devuelven en forma de parámetros de configuración, que un homólogo puede obtener llamando a `clemext_node_getParameters` en el controlador de nodo.

Un parámetro consta de un nombre y un valor, donde el valor puede ser:

- Valor simple (cadena).
- Valor con clave (clave y valor).
- Valor estructurado (conjunto de valores con nombre).
- Lista de valores

El contenido del documento Parámetros se determina completamente por el paquete de extensión. Los parámetros se definen en el archivo de especificación de cliente y se interpretan en el módulo de extensión de servidor.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<Parameters>
  <Parameter name="linesToScan" value="50"/>
  <Parameter name="useCaption" value="true"/>
  <Parameter name="caption" value="My Caption"/>
  <Parameter name="captionPosition" value="north"/>
  <Parameter name="defaultAggregation">
    <ListValue>
      <Value value="min"/>
      <Value value="max"/>
      <Value value="mean"/>
      <Value value="stddev"/>
    </ListValue>
  </Parameter>
</Parameters>
```

Documento de generación de SQL: Un documento de generación de SQL describe cómo se puede traducir la ejecución de un homólogo a SQL

Un homólogo puede proporcionar un documento de generación de SQL en respuesta a una solicitud `clemext_peer_getSQLGeneration` del host. El host intentará ejecutar el SQL en preferencia a la ejecución del homólogo de forma interna.

Un homólogo que consume entrada puede obtener su entrada SQL llamando a `clemext_node_getSQLGeneration` en su controlador de nodo asociado.

El componente principal de un documento de generación de SQL es una instrucción SQL que replica el comportamiento de ejecución de un nodo o fragmento de ruta. Para un nodo que produce datos (es decir, un lector de datos o nodo de transformador de datos), la instrucción debe ser `SELECT` y deber ir acompañada de un diccionario que correlacione nombres de campo en el modelo de datos a los nombres de columna en la instrucción `SELECT`.

Un documento de generación de SQL también puede incluir las propiedades de la conexión de la base de datos según la cual se ejecutará la instrucción, como el nombre de origen de datos y el nombre de producto. Un homólogo puede utilizar estas propiedades para ayudar a determinar el SQL que produce.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<SqlGeneration>
  <Propiedades
    datasourceName="SQL Server"
    databaseName="DataMining"
    serverName="GB1-RDUNCAN1"
    passwordKey="PW0"
    userName="fred"
    dbmsName="Microsoft SQL Server"
    dbmsVersion="09.00.1399"/>
  <Statement>
    <Bindings>
      <Binding columnName="C0" fieldName="ID"/>
      <Binding columnName="C1" fieldName="START_DATE"/>
    </Bindings>
    <TableParameters>
 <TableParameter name="${TABLE26}" value="dbo.DRUG4N"/>
    </TableParameters>
    <Sql>
      SELECT
      T0.ID AS C0,T0."START_DATE" AS C1
```

```
      FROM ${TABLE26} T0
      DONDE (T0."START_DATE" > '2003-01-01')
      ORDER BY 2 ASC
    </Sql>
  </Statement>
</SqlGeneration>
```

Documento de detalles de estado: Un documento de detalles de estado proporciona información sobre el progreso y advertencias u otras condiciones de menor importancia surgidas durante la ejecución. Un módulo de extensión puede enviar documentos de detalles de estado asincrónicamente mediante la función de devolución de llamada `clemext_progress_report`.

El documento de detalles de estado es un conjunto de uno o más elementos `Diagnostic` donde cada diagnóstico incluye al menos un código de condición, un mensaje (salvo que se proporcione en un archivo de propiedades) y un conjunto de uno o más parámetros que contienen más información para insertar en el mensaje. El elemento `StatusDetail` también puede tener un atributo `destination` opcional para indicar que el mensaje pase a uno de los siguientes:

- Un archivo de seguimiento local gestionado por IBM SPSS Modeler
- El cliente (para mensajes de interés al usuario).
- Todos (enviar a todos los destinos posibles)

El formato del elemento `Diagnostic` es:

```
<Diagnostic code="entero" severity="nivel_gravedad">
     <Message>mensaje_texto</Message>
     <Parameter>valor</Parameter>
</Diagnostic>
```

donde:

`code` (necesario) es un entero que indica el código del estado.

`severity` indica la gravedad del estado y puede ser uno de los siguientes: `desconocidos`, `información`, `warning`, `error` o `fatal`.

Ejemplo

```
<?xml version="1.0" encoding="utf-8"?>
<StatusDetail destination="cliente">
  <Diagnostic code="654" severity="information">
    <Message>{0} registros procesados</Message>
    <Parameter>10000</Parameter>
  </Diagnostic>
</StatusDetail>
```

Uso de mensajes localizados

Si desea utilizar los mensajes localizados de un archivo de propiedades, omita el elemento `Message` del documento de detalles de estado y utilice las teclas de mensaje del archivo de especificaciones, como en el ejemplo siguiente:

```
    ...
    <Execution ...>
    ...
        <StatusCodes>
            ...
            <StatusCode code="21" status="warning" messageKey="fieldIgnoredMsg.LABEL"/>
```

```
            ...
        </StatusCodes>
    </Execution>
    ...
```

El archivo messages.properties contendría:

```
fieldIgnoredMsg.LABEL=El campo "{0}" no se puede utilizar para generar modelos y se ignora
```

En el documento de detalles de estado, puede definir un parámetro (como un nombre de campo) para que se sustituya en el mensaje localizado, por ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<StatusDetail>
    <Diagnostic code="21">
        <Parameter>BP</Parameter>
    </Diagnostic>
</StatusDetail>
```

Aplicaciones de ayuda C++

Algunos de los nodos de ejemplo de CLEF incluyen un número predefinido de archivos de origen de origen C++, conocidos como **Aplicaciones de ayuda**. Funcionan como encapsuladores para algunas de las API de servidor basadas en C y se pueden compilar fácilmente en un complemento C++ de CLEF.

Tabla 51. Aplicaciones de ayuda C++

Aplicación de ayuda	Descripción
BufferHelper	Gestiona los búfer de memoria que pueden cambiar de tamaño utilizados en la API de C
DataHelper	Ayuda a ajustar las operaciones de lectura y escritura de datos
HostHelper	Ajusta el objeto CLEMEXT HostHandle
XMLHelper	Ajusta el procesamiento XML

Los ayudantes adoptan la forma de un par de archivos *.cpp* y *.h*: por ejemplo, *BufferHelper.cpp* y *BufferHelper.h*.

Para obtener información acerca de la visualización de estos archivos de aplicación de ayuda, consulte "Evaluación del código fuente" en la página 28.

Las descripciones más detalladas de estos archivos se pueden encontrar en la documentación de la API de servidor, como se indica a continuación:

1. Desde la pantalla de documentación de la API CLEF, seleccione **Conceptos básicos de la API de servidor**.
2. Pulse en la pestaña **Archivos**.
3. Pulse en el nombre del archivo `.h` que corresponda a la aplicación de ayuda de la que desea obtener información.
4. En **Estructuras de datos**, pulse en el nombre de clase correspondiente para visualizar la documentación.

Para obtener información sobre el acceso a la documentación de la API CLEF, consulte "Documentación de la API CLEF" en la página 175.

Tratamiento de errores

Cada función de la API devuelve un código de estado (`CLEMEXTStatus`) y un código de error opcional específico del módulo (`CLEMEXTErrorCode`). El código de estado puede ser `rendimiento` (sin error) o uno

de los códigos de error enumerados para la función de la API; éstos casi siempre incluyen errores específicos del módulo. El código de error específico del módulo puede ser 0, que significaría que no contiene error específico del módulo.

IBM SPSS Modeler proporciona los mensajes de código de estado. Los detalles de códigos de estado común se pueden encontrar en la documentación de la API de servidor, como se indica a continuación:

1. Desde la pantalla de documentación de la API CLEF, seleccione **Conceptos básicos de la API de servidor**.
2. Pulse en la pestaña **Módulos**.
3. Seleccione **Códigos de estado comunes**.

Para obtener información sobre el acceso a la documentación de la API CLEF, consulte "Documentación de la API CLEF" en la página 175.

Los mensajes de error específicos del módulo se pueden encontrar:

- En el archivo de especificación (en el elemento `StatusCode` de la sección Module).
- En un paquete de recursos al que se hace referencia en el archivo de especificación
- En el módulo de extensión

Para un código de error específico del módulo, un módulo puede ofrecer detalles de error adicionales que consten de un mensaje de error predeterminado (errores que no se tienen en cuenta en el archivo de especificación o paquete de recursos) y parámetros que se insertan en el mensaje. Los mensajes de error múltiples pueden describir cadenas de errores causales.

Un informe de errores del cliente tiene el siguiente formato:

node_label:*message*

donde:

- *node_label* es el valor del atributo `label` del elemento `Node` en el que se especifica el módulo.
- *message* es el texto del mensaje, que puede proporcionar el servidor o que se defina en el archivo de especificación (o archivo *.properties* para localización).

API de análisis XML

IBM SPSS Modeler incluye el analizador XML Xerces-C de Apache, ofreciendo un número de devoluciones de llamadas que permiten que un módulo lea y escriba datos XML. Puede sustituirlo por su propio analizador de XML, si lo prefiere.

Uso de la API de servidor

Para incluir las llamadas de funciones de servidor en un nodo:

1. Cree los archivos de origen *.cpp* y *.h* que contengan las llamadas de funciones.
2. Compile los archivos de origen en un archivo de biblioteca de enlace dinámica (*.dll*).
3. Incluye una referencia al archivo *.dll* del archivo de especificación, por ejemplo:

```
<Resources>
.
     <SharedLibrary id="mylib1" path="mycorp.mynode/mylib" /> .
</Resources>
```

Para obtener más información, consulte el tema "Bibliotecas compartidas" en la página 36.

También puede encontrarlo útil para ver el código de origen de los nodos de ejemplo que se proporcionan con esta versión. Para obtener más información, consulte el tema "Evaluación del código fuente" en la página 28.

Directrices de programación del servidor

La parte de la biblioteca de enlace dinámica (DLL) de servidor de un módulo de CLEF debe estar seguido de una serie de directrices para asegurar que el módulo funcione correctamente y para evitar afectar el funcionamiento de IBM SPSS Modeler. Un módulo de CLEF debe:

- garantizar que ejecución de homólogos es independiente
- admitir múltiples instancias de homólogos en un proceso simple
- garantizar la seguridad del subproceso
- evitar alterar el subproceso o el entorno del proceso
- restringir el uso del subproceso en el módulo
- gestionar correctamente las solicitudes para cancelar la ejecución
- reiniciar llamadas del sistema interrumpidas (UNIX)
- garantizar la activación de CoInitialize o CoUninitialize (Windows)
- evitar realizar suposiciones sobre cuándo se descargará el módulo
- garantizar el inicio de los subprocesos
- evitar escribir en resultados o errores estándar

Las siguientes secciones tratan de estas áreas detalladamente.

Garantizar que ejecución de homólogos es independiente

Las instancias de homólogos no deben hacer suposiciones acerca de la existencia de otras instancias de homólogos en el proceso del servidor de IBM SPSS Modeler. IBM SPSS Modeler puede programar la ejecución para que las instancias de homólogos correspondientes a los nodos adyacentes en un subproceso se ejecuten en fases diferentes, para que la existencia y ejecución de las instancias no se solapen.

Las instancias de homólogos deben ser independientes y no comunicarse con otras directamente, por ejemplo, mediante tuberías o sockets. Toda la comunicación entre las diferentes instancias de homólogos se deben ejecutar directamente (leyendo y escribiendo datos en el subproceso) o indirectamente, mediante algún agente externo(por ejemplo, un servidor de base de datos que gestione datos compartidos entre homólogos).

Admitir múltiples instancias de homólogos en un proceso simple

Los usuarios finales pueden crear múltiples instancias de un módulo de CLEF concreto (por ejemplo, más de un nodo del mismo tipo) en un proceso de servidor al ejecutar un subproceso. Además, los datos estáticos del módulo CLEF se comparten en múltiples instancias de homólogos y no se deben utilizar para guardar datos privados de un objeto de homólogo. Ejemplos de datos estáticos son miembros estáticos de clases de C++ y variables estáticas o globales en unidades de compilación de C.

Las funciones API del módulo de CLEF deben ser recurrentes y evitar activaciones no recurrentes del sistema. Por ejemplo, si una instancia de homólogo obtiene datos de entrada de su iterador utilizando `clemext_iterator_nextRecord`, es posible que active `clemext_peer_nextRecord` en una segunda instancia de homólogo anterior del primer homólogo y que el primer homólogo utilice lo que produce los datos.

Las activaciones del sistema como `strtok` no son recurrentes y no se pueden utilizar. Para obtener información acerca de las alternativas recurrentes, consulte la documentación del sistema operativo de la plataforma que utilice.

Garantizar la seguridad del subproceso

IBM SPSS Modeler puede intercalar la ejecución de múltiples instancias de homólogos de diferentes subprocesos de ejecución. El acceso a los recursos compartidos entre objetos homólogos debe estar protegida, por ejemplo, mediante la sincronización de objetos de exclusión mutua o servicios de bibliotecas de subprocesos similares.

Los módulos de CLEF deben evitar realizar activaciones del sistema que no sean seguras para los subprocesos. Para obtener más información, consulte la documentación de su sistema operativo o las páginas de man de UNIX.

Evitar alterar el subproceso o el entorno del proceso

Evite utilizar activaciones del sistema que puedan cambiar el entorno del subproceso o proceso de activación.

Algunos ejemplos de estas activaciones se enumeran aquí. La lista no es exhaustiva:

- `setlocale` se utiliza para cambiar la configuración local en lugar de la información de configuración local de lectura
- `SetCurrentDirectory` (Windows) o `chdir` (UNIX)
- `LogonUser` (Windows) o `seteuid` (UNIX)
- `putenv`
- `exit`
- `signal`

Nota: En Windows, una activación que cambia el entorno de un subproceso que puede ser necesario es `CoInitialize`. Para obtener más información, consulte el tema "Garantizar la activación de CoInitialize o CoUninitialize (Windows)" en la página 198.

Restringir el uso del subproceso en el módulo

En general, los módulos se pueden utilizar libremente subprocesos internos. Sin embargo, las funciones de devolución de llamada sólo se deben activar en un subproceso que utilice IBM SPSS Modeler para activar las funciones de modelo de CLEF (salvo para `clemext_peer_cancelExecution`).

Las siguientes funciones de devolución de llamada se puede activar de forma asíncrona desde cualquier subproceso en ejecución en el módulo:

- `clemext_progress_report`
- `clemext_channel_send`

Una instancia de homólogo debe garantizar que múltiples activaciones no activan estas llamadas de forma simultánea.

Gestionar correctamente las solicitudes para cancelar la ejecución

Si un usuario final solicita cancelar la ejecución de una instancia de homólogo, IBM SPSS Modeler realiza una activación asíncrona de la función `clemext_peer_cancelExecution` del módulo. Los desarrolladores deben intentar implementar esta activación. Tenga en cuenta que la función se debe activar de forma asíncrona y activarse mientras otra función API de CLEF se está ejecutando por el módulo.

Reiniciar llamadas del sistema interrumpidas (UNIX)

En UNIX, la aplicación IBM SPSS Modeler utiliza señales y controladores de señales. Algunas activaciones de sistemas UNIX pueden devolver el código EINTR si el proceso recibe una señal durante la ejecución de la activación (compruebe las páginas de man para obtener información de la activación del sistema de su plataforma UNIX).

Si esto ocurre, el código de llamada debería comprobar el código de retorno EINTR y reiniciar la activación. Una forma de hacerlo es crear una simple función de encapsulador (`open_safe`) y que su aplicación la ejecute:

```
int
open_safe(const char* path, int oflag, mode_t mode) {
  int res;
  while ((res = ::open(path, oflag, mode)) == -1
```

```
                                    && errno == EINTR) {
  }
  return res;
}
```

Garantizar la activación de CoInitialize o CoUninitialize (Windows)

En Windows, los subprocesos que necesitan utilizar los servicios de la biblioteca Windows Component Object Model (COM) deben activar la función API `CoInitialize` del sistema antes de utilizar los servicios COM y, a continuación activar `CoUninitialize`. Los subprocesos desde los que IBM SPSS Modeler activa la API de un módulo de CLEF pueden haber activado previamente `CoInitialize` o no.

Un módulo de CLEF que prefiera utilizar los servicios COM de estos subprocesos deben activar `CoInitialize`, normalmente en una función `clemext_create_peer` o `clemext_peer_beginExecution`. En ese caso, el módulo también debe activar `CoUninitialize` posteriormente cuando el subproceso complete la ejecución, normalmente en `clemext_destroy_peer` o `clemext_peer_endExecution`, respectivamente.

Para obtener más información acerca de `CoInitiaize`, consulte la documentación del sitio de Microsoft Developer Network (MSDN) en http://msdn.microsoft.com.

Evitar realizar suposiciones sobre cuándo se descargará el módulo

Actualmente, un módulo de CLEF permanece cargado hasta que finaliza una sesión (por ejemplo, los módulos no se pueden descargar y cargar bajo demanda). La función `clemext_cleanup` no se activa nunca, incluso al salir del proceso del servidor de IBM SPSS Modeler en el que se carga el módulo. Además, los desarrolladores no deben asumir que un módulo se descargará y que sus recursos se liberarán en cualquier punto.

Garantizar el inicio de los subprocesos

Iniciar un subproceso, mediante `CreateProcess` (Windows) o `fork` (UNIX), puede suponer diferentes complicaciones de forma que los procesos padres e hijos interactúen y de forma que el proceso hijo herede los recursos abiertos del padre.

Si un módulo CLEF necesita activar una ejecución fuera de un proceso, utilice una arquitectura alternativa adecuada. Por ejemplo, el módulo de CLEF puede utilizar los servicios de un servidor de aplicaciones para ejecutar la tarea requerida.

En concreto, los procesos de Windows deben evitar iniciar subprocesos que utilicen la función `CreateProcess` con el parámetro `bInheritHandles` definido como TRUE. De esta forma hará que los procesos hijo hereden todos los descriptores de archivo abiertos en el proceso padre (servidor de IBM SPSS Modeler).

Evitar escribir en resultados o errores estándar

Si un módulo de CLEF escribe en el resultado o subproceso de error estándar de un proceso (por motivos de depuración, posiblemente), no será visible para el usuario final. Sin embargo, si un subproceso con nodos de CLEF se despliega utilizando IBM SPSS Modeler Solution Publisher y se ejecuta desde la línea de comandos (en Windows o UNIX), este resultado será visible y puede confundir a los usuarios.

Los módulos de CLEF pueden activar un servicio de rastreo mediante la función de devolución de llamada `clemext_host_trace` y pasar el mensaje en forma de cadena. También es necesario activar el rastreo en la instalación de IBM SPSS Modeler mediante el ajuste del archivo de opciones de configuración de IBM SPSS Modeler Server (*/config/options.cfg* en el directorio de instalación de IBM SPSS Modeler):

```
trace_extension, 1
```

Los mensajes rastreados se incluyen en el archivo*/log/trace-<process_ID>-<process_ID>.log* en el directorio de instalación de IBM SPSS Modeler, donde *process_ID* es el identificador del proceso de IBM SPSS Modeler Server. Evite rastrear múltiples sesiones a la vez, ya que comparten el mismo archivo de registro.

Capítulo 10. Comprobación y distribución

Comprobación de las extensiones de CLEF

Es muy recomendable que compruebe una nueva extensión de forma local antes de distribuirla a otros usuarios.

Después de crear un archivo de especificación y cualquier paquete de recursos asociados, archivos .jar, bibliotecas compartidas y archivos de ayuda de usuario, puede probar la extensión organizando los archivos en la estructura de archivos necesaria y copiándolos a su instalación local de IBM SPSS Modeler. La próxima vez que inicie IBM SPSS Modeler, debería ver la nueva extensión en la interfaz de usuario de IBM SPSS Modeler.

Prueba de una extensión de CLEF

1. Cierre IBM SPSS Modeler si está abierto.
2. Si la extensión define un nodo o salida de CLEF, se recomienda activar la pestaña Depurar del diálogo de extensión hasta que esté satisfecho de que la extensión está funcionando correctamente. Para obtener más información, consulte el tema "Uso de la pestaña Depurar" en la página 200.
3. Organice los archivos del cliente y del servidor en la estructura necesaria. Compruebe que el archivo de especificaciones y los recursos asociados que el nodo necesite (como archivos `.jar` o `.dll`) se copian en las ubicaciones correctas. Para obtener más información, consulte el tema "Estructura de archivos" en la página 5.
4. Copie el directorio del cliente en la carpeta *\ext\lib* del directorio de instalación de IBM SPSS Modeler.
5. Copie el directorio del servidor en la carpeta *\ext\bin* del directorio de instalación de IBM SPSS Modeler.
6. Inicie IBM SPSS Modeler.
7. Si la extensión define un menú o un elemento de menú, asegúrese de que aparecen correctamente en el sistema de menú principal. Si la extensión define un nuevo nodo, asegúrese de que el nodo aparece en la posición deseada en la paleta de nodo correcta tal como se ha definido en el archivo de especificación.
8. Compruebe la extensión en profundidad.

   Por ejemplo, compruebe que:

   - el rendimiento del nodo no se ve afectado a medida que se aumenta el número de campos y de registros
   - los valores nulos se gestionan de forma coherente
   - se admiten diferentes configuraciones regionales (p. ej. europea o asiática)
9. Una vez que haya añadido una extensión, puede seguir realizando modificaciones en su archivo de especificación. No obstante, los cambios que realice no entrarán en vigor hasta que se reinicie IBM SPSS Modeler.

Depuración de una extensión de CLEF

CLEF contiene las siguientes características que facilitan la depuración de una extensión:

- Mensajes de error de sintaxis XML
- ejecución externa
- Pestaña Depurar

Errores de sintaxis XML

La sintaxis XML incorrecta de un archivo de especificación se marca con un mensaje de error del analizador XML.

El mensaje muestra el número de línea aproximado del error, junto con un comentario.

Para solucionar este problema:

1. Corrija el error del archivo.
2. Vuelva a comprobar el archivo mediante el procedimiento de "Prueba de una extensión de CLEF" en la página 199.
3. Repita este procedimiento hasta que el archivo de especificación no contenga más errores.

Ejecución externa

Normalmente, una extensión escrita por el usuario de CLEF se ejecuta en sus propios procesos, que son diferentes del proceso de IBM SPSS Modeler. Puede ser de gran ayuda cuando realice una depuración, en caso de error en el proceso, ya que no exige la ejecución del proceso IBM SPSS Modeler Server por completo.

Nota: Es posible sustituir este parámetro predeterminado. Para obtener más información, consulte el tema "Modificación de las opciones de ejecución" en la página 201.

Uso de la pestaña Depurar

En cualquier cuadro de diálogo o marco asociado con un nodo o resultado de CLEF, puede activar una pestaña Depurar que le permita inspeccionar la configuración de las propiedades del objeto. También puede visualizar el contenido de cualquier contenedor definido en la extensión y guardarlo en un archivo para su inspección posterior. Para obtener más información, consulte el tema "Containers" en la página 53.

Active la pestaña Depurar definiendo como `true` el valor del atributo `debug` del elemento `Extension` del archivo de especificación. Para obtener más información, consulte el tema "Elemento Extension" en la página 33.

Los campos de la pestaña son los siguientes:

ID de elemento. El identificador exclusivo de la extensión; el valor del atributo `id` del elemento `ExtensionDetails` del archivo de especificación.

Nombre de script. El identificador exclusivo del nodo cuando se referencia en un script; el valor del atributo `scriptName` del elemento `Node`.

ID de extensión. El nombre de la carpeta de extensión en la que se ubican el archivo y el directorio de recursos. Este valor se forma concatenando los atributos `providerTag` e `id` (separados por un carácter "`.`") del elemento `ExtensionDetails`. Tenga en cuenta que en la parte `providerTag` del identificador, el valor no debe incluir la cadena `spss`, que está reservada para uso interno.

Propiedades. Esta tabla muestra detalles seleccionados de las instrucciones de `Propiedad` para el nodo:

- **Propiedad**. El identificador exclusivo de la propiedad; el valor del campo `nombre` del elemento `Property`.
- **Nombre de script**. El identificador exclusivo de la propiedad cuando se referencia en un script; el valor del atributo `scriptName` del elemento `Property`.
- **Tipo de valor**. El tipo de valor que esta propiedad puede adoptar, tal y como define el atributo `valueType` del elemento `Property`.
- **¿Es una lista?** Indica si la propiedad es una lista de valores del tipo de valor especificado; el valor del atributo `isList` del elemento `Property`.

- **¿Está compartido?** Si está activado, indica que esta propiedad se utiliza en más de una ubicación de la extensión (por ejemplo, en el nodo generador de modelos, en los resultados del modelo o en el aplicador de modelos).
- **Value**. El valor predeterminado, si existe, de la propiedad.

Contenedores. Muestra el contenido del contenedor seleccionado (por ejemplo, datos del modelo). Pulse en el campo para mostrar una lista del resto de contenedores definidos para la extensión y seleccione otros diferentes para mostrar su contenido. Pulse en el botón **Guardar contenedor** para guardar el contenido en el contenedor seleccionado en formato XML para su posterior inspección.

Seguimiento. Muestra un cuadro de diálogo que permite visualizar el resultado del seguimiento cuando se ejecute el nodo.

Modificación de las opciones de ejecución

De forma predeterminada, un módulo de extensión de CLEF escrito por el usuario se ejecuta en su propio proceso diferentes del proceso principal de IBM SPSS Modeler. De esta forma, un fallo en el proceso de extensión no provoca un fallo del proceso de IBM SPSS Modeler. En contraste, los módulos de IBM Corp. proporcionados se ejecutan en el proceso principal de forma predeterminada.

Se incluyen dos opciones de configuración del servidor para que el administrador del sistema pueda modificar estos casos en uno o más módulos nombrados. Ambas opciones incluyen una lista separada por comas de los identificadores del módulo para indicar los módulos a los que afecta el cambio.

Nota: Normalmente, estas opciones sólo se modifican a solicitud de un representante del servicio de atención al cliente.

Las opciones son las siguientes:

Opción In-Process Execution (Ejecución en proceso)

Permite cargar directamente en IBM SPSS Modeler los módulos de extensión que normalmente se cargarían en un proceso externo (normalmente, módulos escritos por el usuario). El formato es:

`clef_inprocess_execution, "`*`IDmódulo1`*`[,`*`IDmódulo2`*`[,...`*`IDmódulon`*`]]"`

donde *IDmódulo* es el valor del atributo `id` del elemento `ExtensionDetails` en el archivo de especificación correspondiente. A continuación se incluye un ejemplo:

```
clef_inprocess_execution, "prueba.ejemplo_lectorarchivo"
```

Opción External Execution (Ejecución externa)

Permite cargar directamente en IBM SPSS Modeler los módulos de extensión que normalmente se cargarían en un proceso externo (normalmente, módulos de IBM Corp.). El formato es:

`clef_external_execution, "`*`IDmódulo1`*`[,`*`IDmódulo2`*`[,...`*`IDmódulon`*`]]"`

donde *IDmódulo* tiene el mismo valor que en `clef_inprocess_execution`. A continuación se incluye un ejemplo ficticio:

```
clef_external_execution, "spss.naivebayes,spss.terminador"
```

Opción Adding or Changing an Execution (Adición o modificación de una ejecución)

Para añadir o modificar una opción de ejecución, realice el procedimiento que aparece en "Uso del archivo options.cfg" en la *Guía de Administración de servidores y rendimiento de IBM SPSS Modeler 16*.

Distribución de las extensiones de CLEF

Cuando haya terminado de comprobar la nueva extensión y esté lista para su distribución:

1. Desactive la pestaña Depurar si se ha sido activado. Para obtener más información, consulte el tema "Uso de la pestaña Depurar" en la página 200.
2. Cree una estructura de archivos que refleje exactamente la forma en la que desea instalar los archivos de extensión. Para obtener más información, consulte el tema "Estructura de archivos" en la página 5.
3. Comprima la estructura de archivos en un archivo `.zip`. Puede que le resulte más fácil crear archivos `.zip` independientes para las instalaciones en el cliente y en el servidor.
4. Distribuya los archivos `.zip` a los usuarios finales.

Instalación de las extensiones de CLEF

Para instalar una extensión de CLEF:

1. Cuando reciba el archivo `.zip` con la estructura del archivo de extensión, extraiga los archivos de cliente a la carpeta `\ext\lib` del directorio de instalación de IBM SPSS Modeler.
2. Extraiga los archivos de servidor a la carpeta `\ext\bin` del directorio de instalación de IBM SPSS Modeler (o al directorio equivalente si utiliza IBM SPSS Modeler Server).
3. Inicie IBM SPSS Modeler y compruebe que los nuevos nodos aparecen en la ubicación correspondiente en la paleta de nodos.

Desinstalación de las extensiones de CLEF

Para desinstalar una extensión de CLEF:

1. Busque la carpeta de extensión en el directorio `\ext\lib` en el directorio de instalación de IBM SPSS Modeler.

   Si la extensión también ha instalado una carpeta de extensión del servidor, busque esta carpeta en el directorio `\ext\bin` en el directorio de instalación de IBM SPSS Modeler o IBM SPSS Modeler Server.
2. Elimine la carpeta o carpetas de extensión.

El cambio surtirá efecto la próxima vez que inicie IBM SPSS Modeler.

Apéndice. Esquema XML de CLEF

Referencia del elemento de CLEF

Esta sección ofrece una referencia para todos los elementos de CLEF.

Cada tema enumera los atributos válidos de un elemento y sus elementos padre e hijo. El diagrama muestra todos los elementos hijo. Tenga en cuenta que las flechas del diagrama indican elementos que se pueden compartir con otros elementos. Estos elementos se enumeran en la tabla de contenido como un hijo de este tema ("Referencia del elemento de CLEF") en lugar de como hijo del tema padre.

Elementos

Elemento Action

Tabla 52. Atributos de Action

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
id	**obligatorio**		*cadena*
imagePath	opcional		*cadena*
imagePathKey	opcional		*cadena*
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
shortcut	opcional		*cadena*
shortcutKey	opcional		*cadena*

Representación de XML

```
<xs:element name="Action">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="imagePath" type="xs:string" use="optional"/>
  <xs:attribute name="imagePathKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="shortcut" type="xs:string" use="optional"/>
  <xs:attribute name="shortcutKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Actions

Elemento ActionButton

Tabla 53. Atributos de ActionButton

Atributo	Uso	Descripción	Valores válidos
action	**obligatorio**		*cadena*
showIcon	opcional		*booleano*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="ActionButton">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="action" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

ComboBoxControl, ExtensionObjectPanel, ModelViewerPanel, SelectorPanel, StaticText, SystemControls, TabbedPanel, TextBrowserPanel

Elemento Actions

Representación XML

```
<xs:element name="Actions">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Action"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

Elementos padre

CommonObjects

Elementos hijo

Action

Elemento AddField

Tabla 54. Atributos de AddField

Atributo	Uso	Descripción	Valores válidos
direction	opcional		**in** **out** **both** **none** **partition**
directionRef	opcional		
fieldRef	opcional		
group	opcional		*cadena*
label	opcional		*cadena*
missingValuesRef	opcional		
name	**obligatorio**		
prefix	opcional		*cadena*
role	opcional		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
storage	opcional		**unknown** **integer** **real** **string** **date** **time** **timestamp**
storageRef	opcional		
tag	opcional		*cadena*
targetField	opcional		*cadena*
type	opcional		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless**
typeRef	opcional		
value	opcional		*cadena*

Representación XML

```
<xs:element name="AddField">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Range" minOccurs="0"/>
      <xs:element ref="Values" minOccurs="0"/>
      <xs:element ref="NumericInfo" minOccurs="0"/>
      <xs:element name="MissingValues" minOccurs="0">
        <xs:sequence>
          <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="Range" minOccurs="0"/>
        </xs:sequence>
      </xs:element>
      <xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
      </xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
  <xs:attribute name="storage" type="FIELD-STORAGE">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
  </xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE">
    <xs:enumeration value="auto"/>
    <xs:enumeration value="range"/>
    <xs:enumeration value="discrete"/>
    <xs:enumeration value="set"/>
    <xs:enumeration value="orderedSet"/>
    <xs:enumeration value="flag"/>
    <xs:enumeration value="typeless"/>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string"/>
  <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="storageRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="typeRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="directionRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="missingValuesRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="optional">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="predictedValue"/>
    <xs:enumeration value="predictedDisplayValue"/>
    <xs:enumeration value="probability"/>
    <xs:enumeration value="residual"/>
    <xs:enumeration value="standardError"/>
    <xs:enumeration value="entityId"/>
    <xs:enumeration value="entityAffinity"/>
    <xs:enumeration value="upperConfidenceLimit"/>
    <xs:enumeration value="lowerConfidenceLimit"/>
    <xs:enumeration value="propensity"/>
    <xs:enumeration value="value"/>
    <xs:enumeration value="supplementary"/>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string" use="optional"/>
  <xs:attribute name="value" type="xs:string" use="optional"/>
  <xs:attribute name="group" type="xs:string" use="optional"/>
  <xs:attribute name="tag" type="xs:string" use="optional"/>
  <xs:attribute name="prefix" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

ForEach, ModelFields

Elementos hijo

MissingValues, ModelField, NumericInfo, Range, Range, Values, Values

Elementos relacionados

ChangeField

Elemento MissingValues:

Tabla 55. Atributos de MissingValues

Atributo	Uso	Descripción	Valores válidos
treatNullAsMissing	opcional		*booleano*
treatWhitespaceAsMissing	opcional		*booleano*

Representación XML

```
<xs:element name="MissingValues" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Range" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

Elementos padre

Field

Elementos hijo

Range, Range, Values, Values

Elemento ModelField:

Tabla 56. Atributos de ModelField

Atributo	Uso	Descripción	Valores válidos
group	opcional		
role	**obligatorio**		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
tag	opcional		*cadena*
targetField	opcional		*cadena*
value	opcional		*cadena*

Representación XML

```
<xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
  <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="required">
    <xs:enumeration value="unknown"/>
```

```
    <xs:enumeration value="predictedValue"/>
    <xs:enumeration value="predictedDisplayValue"/>
    <xs:enumeration value="probability"/>
    <xs:enumeration value="residual"/>
    <xs:enumeration value="standardError"/>
    <xs:enumeration value="entityId"/>
    <xs:enumeration value="entityAffinity"/>
    <xs:enumeration value="upperConfidenceLimit"/>
    <xs:enumeration value="lowerConfidenceLimit"/>
    <xs:enumeration value="propensity"/>
    <xs:enumeration value="value"/>
    <xs:enumeration value="supplementary"/>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string"/>
  <xs:attribute name="value" type="xs:string"/>
  <xs:attribute name="group" type="MODEL-FIELD-GROUP"/>
  <xs:attribute name="tag" type="xs:string"/>
</xs:element>
```

Elementos padre

Field

Elemento And

Representación de XML

```
<xs:element name="And">
  <xs:sequence minOccurs="2" maxOccurs="unbounded">
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

Elementos padre

And, Command, Constraint, CreateDocument, CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModel, CreateModelApplier, CreateModelOutput, Enabled, Not, Option, Or, Run, Visible

Elementos hijo

And, Condition, Not, Or

Elemento Attribute

Tabla 57. Atributos de Attribute

Atributo	Uso	Descripción	Valores válidos
defaultValue	opcional		
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
isList	opcional		*booleano*
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*
name	**obligatorio**		*cadena*

Tabla 57. Atributos de Attribute (continuación)

Atributo	Uso	Descripción	Valores válidos
valueType	opcional		**cadena** **encryptedString** **integer** **double** **boolean** **date** **enum**

Representación de XML

```
<xs:element name="Attribute">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="valueType" type="ATTRIBUTE-VALUE-TYPE">
    <xs:enumeration value="string"/>
    <xs:enumeration value="encryptedString"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="boolean"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="enum"/>
  </xs:attribute>
  <xs:attribute name="defaultValue" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

Elementos padre

Catalog, Structure

Elemento BinaryFormat

Representación de XML

```
<xs:element name="BinaryFormat"/>
```

Elementos padre

FileFormatType

Elemento Catalog

Tabla 58. Atributos de Catalog

Atributo	Uso	Descripción	Valores válidos
id	**obligatorio**		*cadena*
valueColumn	**obligatorio**		*entero*

Representación XML

```
<xs:element name="Catalog">
  <xs:sequence minOccurs="1" maxOccurs="unbounded">
    <xs:element ref="Attribute"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="valueColumn" type="xs:integer" use="required"/>
</xs:element>
```

Elementos padre

Catalogs

Elementos hijo

Attribute

Elemento Catalogs

Representación XML

```
<xs:element name="Catalogs">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Catalog"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

Elementos padre

CommonObjects

Elementos hijo

Catalog

Elemento ChangeField

Tabla 59. Atributos de ChangeField

Atributo	Uso	Descripción	Valores válidos
direction	opcional		**in** **out** **both** **none** **partition**
directionRef	opcional		
fieldRef	**obligatorio**		
label	opcional		*cadena*
missingValuesRef	opcional		
name	**obligatorio**		
storage	opcional		**unknown** **integer** **real** **string** **date** **time** **timestamp**
storageRef	opcional		

Tabla 59. Atributos de ChangeField (continuación)

Atributo	Uso	Descripción	Valores válidos
type	opcional		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless**
typeRef	opcional		

Representación XML

```
<xs:element name="ChangeField">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Range" minOccurs="0"/>
      <xs:element ref="Values" minOccurs="0"/>
      <xs:element ref="NumericInfo" minOccurs="0"/>
      <xs:element name="MissingValues" minOccurs="0">
        <xs:sequence>
          <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="Range" minOccurs="0"/>
        </xs:sequence>
      </xs:element>
      <xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
      </xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
  <xs:attribute name="storage" type="FIELD-STORAGE">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
  </xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE">
    <xs:enumeration value="auto"/>
    <xs:enumeration value="range"/>
    <xs:enumeration value="discrete"/>
    <xs:enumeration value="set"/>
    <xs:enumeration value="orderedSet"/>
    <xs:enumeration value="flag"/>
    <xs:enumeration value="typeless"/>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string"/>
  <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="required"/>
  <xs:attribute name="storageRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="typeRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="directionRef" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="missingValuesRef" type="EVALUATED-STRING" use="optional"/>
</xs:element>
```

Elementos padre

ForEach, ModelFields

Elementos hijo

MissingValues, ModelField, NumericInfo, Range, Range, Values, Values

Elementos relacionados

AddField

Elemento MissingValues:

Tabla 60. Atributos de MissingValues

Atributo	Uso	Descripción	Valores válidos
treatNullAsMissing	opcional		*booleano*
treatWhitespaceAsMissing	opcional		*booleano*

Representación XML

```
<xs:element name="MissingValues" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Range" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

Elementos padre

Field

Elementos hijo

Range, Range, Values, Values

Elemento ModelField:

Tabla 61. Atributos de ModelField

Atributo	Uso	Descripción	Valores válidos
group	opcional		
role	**obligatorio**		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
tag	opcional		*cadena*
targetField	opcional		*cadena*
value	opcional		*cadena*

Representación XML

```
<xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
  <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="required">
    <xs:enumeration value="unknown"/>
```

```
    <xs:enumeration value="predictedValue"/>
    <xs:enumeration value="predictedDisplayValue"/>
    <xs:enumeration value="probability"/>
    <xs:enumeration value="residual"/>
    <xs:enumeration value="standardError"/>
    <xs:enumeration value="entityId"/>
    <xs:enumeration value="entityAffinity"/>
    <xs:enumeration value="upperConfidenceLimit"/>
    <xs:enumeration value="lowerConfidenceLimit"/>
    <xs:enumeration value="propensity"/>
    <xs:enumeration value="value"/>
    <xs:enumeration value="supplementary"/>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string"/>
  <xs:attribute name="value" type="xs:string"/>
  <xs:attribute name="group" type="MODEL-FIELD-GROUP"/>
  <xs:attribute name="tag" type="xs:string"/>
</xs:element>
```

Elementos padre

Field

Elemento CheckBoxControl

Tabla 62. Atributos de CheckBoxControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
invert	opcional		*booleano*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="CheckBoxControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="invert" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento CheckBoxGroupControl

Tabla 63. Atributos de CheckBoxGroupControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
layoutByRow	opcional		*booleano*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
rows	opcional		*positiveInteger*
showLabel	opcional		*booleano*
useSubPanel	opcional		*booleano*

Representación XML

```
<xs:element name="CheckBoxGroupControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="layoutByRow" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="useSubPanel" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento ClientDirectoryChooserControl

Tabla 64. Atributos de ClientDirectoryChooserControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
mode	**obligatorio**		**open** **save** **import** **export**
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="ClientDirectoryChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"/>
    <xs:enumeration value="save"/>
    <xs:enumeration value="import"/>
    <xs:enumeration value="export"/>
  </xs:attribute>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento ClientFileChooserControl

Tabla 65. Atributos de ClientFileChooserControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
mode	**obligatorio**		**open** **save** **import** **export**
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="ClientFileChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"/>
    <xs:enumeration value="save"/>
```

```
    <xs:enumeration value="import"/>
    <xs:enumeration value="export"/>
  </xs:attribute>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento ComboBoxControl

Tabla 66. Atributos de ComboBoxControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="ComboBoxControl" type="CONTROLLER">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
</xs:element>
```

Tabla 67. Tipos ampliados

Tipo	Descripción
ItemChooserControl	

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

ActionButton, ExtensionObjectPanel, ModelViewerPanel, SelectorPanel, StaticText, SystemControls, TabbedPanel, TextBrowserPanel

Elemento Command

Tabla 68. Atributos de Command

Atributo	Uso	Descripción	Valores válidos
path	**obligatorio**		

Representación de XML

```
<xs:element name="Command">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

Elementos padre

Run

Elementos hijo

And, Condition, Not, Or

Elemento CommonObjects

Proporciona una ubicación para las definiciones que son globales para la extensión

Tabla 69. Atributos de CommonObjects

Atributo	Uso	Descripción	Valores válidos
extensionListenerClass	opcional		*cadena*

Representación de XML

```
<xs:element name="CommonObjects">
  <xs:all>
    <xs:element ref="PropertyTypes" minOccurs="0"/>
```

```
    <xs:element ref="PropertySets" minOccurs="0"/>
    <xs:element ref="FileFormatTypes" minOccurs="0"/>
    <xs:element ref="ContainerTypes" minOccurs="0"/>
    <xs:element ref="Actions" minOccurs="0"/>
    <xs:element ref="Catalogs" minOccurs="0"/>
  </xs:all>
  <xs:attribute name="extensionListenerClass" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Extension

Elementos hijo

Actions, Catalogs, ContainerTypes, FileFormatTypes, PropertySets, PropertyTypes

Elemento Condition

Tabla 70. Atributos de Condition

Atributo	Uso	Descripción	Valores válidos
container	opcional		*cadena*
control	opcional		*cadena*
expression	opcional		
listMode	opcional		**todos** **cualquiera**

Tabla 70. Atributos de Condition (continuación)

Atributo	Uso	Descripción	Valores válidos
op	**obligatorio**		**equals** **notEquals** **isEmpty** **isNotEmpty** **lessThan** **lessOrEquals** **greaterThan** **greaterOrEquals** **equalsIgnoreCase** **isSubstring** **startsWith** **endsWith** **startsWithIgnoreCase** **endsWithIgnoreCase** **isSubstring** **hasSubstring** **isSubstringIgnoreCase** **hasSubstringIgnoreCase** **in** **countEquals** **countLessThan** **countLessOrEquals** **countGreaterThan** **countGreaterOrEquals** **contains** **storageEquals** **typeEquals** **directionEquals** **isMeasureDiscrete** **isMeasureContinuous** **isMeasureTypeless** **isMeasureUnknown** **isStorageString** **isStorageNumeric** **isStorageDatetime** **isStorageUnknown** **isModelOutput** **modelOutputRoleEquals** **modelOutputTargetField Equals** **modelOutputHasValue** **modelOutputTagEquals** **enumRestriction**
property	opcional		*cadena*
value	opcional		

Representación XML

```
<xs:element name="Condition">
  <xs:attribute name="expression" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="control" type="xs:string" use="optional"/>
  <xs:attribute name="property" type="xs:string" use="optional"/>
  <xs:attribute name="container" type="xs:string" use="optional"/>
  <xs:attribute name="op" type="CONDITION-TEST" use="required">
    <xs:enumeration value="equals"/>
    <xs:enumeration value="notEquals"/>
    <xs:enumeration value="isEmpty"/>
    <xs:enumeration value="isNotEmpty"/>
```

```
    <xs:enumeration value="lessThan"/>
    <xs:enumeration value="lessOrEquals"/>
    <xs:enumeration value="greaterThan"/>
    <xs:enumeration value="greaterOrEquals"/>
    <xs:enumeration value="equalsIgnoreCase"/>
    <xs:enumeration value="isSubstring"/>
    <xs:enumeration value="startsWith"/>
    <xs:enumeration value="endsWith"/>
    <xs:enumeration value="startsWithIgnoreCase"/>
    <xs:enumeration value="endsWithIgnoreCase"/>
    <xs:enumeration value="isSubstring"/>
    <xs:enumeration value="hasSubstring"/>
    <xs:enumeration value="isSubstringIgnoreCase"/>
    <xs:enumeration value="hasSubstringIgnoreCase"/>
    <xs:enumeration value="in"/>
    <xs:enumeration value="countEquals"/>
    <xs:enumeration value="countLessThan"/>
    <xs:enumeration value="countLessOrEquals"/>
    <xs:enumeration value="countGreaterThan"/>
    <xs:enumeration value="countGreaterOrEquals"/>
    <xs:enumeration value="contains"/>
    <xs:enumeration value="storageEquals"/>
    <xs:enumeration value="typeEquals"/>
    <xs:enumeration value="directionEquals"/>
    <xs:enumeration value="isMeasureDiscrete"/>
    <xs:enumeration value="isMeasureContinuous"/>
    <xs:enumeration value="isMeasureTypeless"/>
    <xs:enumeration value="isMeasureUnknown"/>
    <xs:enumeration value="isStorageString"/>
    <xs:enumeration value="isStorageNumeric"/>
    <xs:enumeration value="isStorageDatetime"/>
    <xs:enumeration value="isStorageUnknown"/>
    <xs:enumeration value="isModelOutput"/>
    <xs:enumeration value="modelOutputRoleEquals"/>
    <xs:enumeration value="modelOutputTargetFieldEquals"/>
    <xs:enumeration value="modelOutputHasValue"/>
    <xs:enumeration value="modelOutputTagEquals"/>
    <xs:enumeration value="enumRestriction"/>
  </xs:attribute>
  <xs:attribute name="value" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="listMode" use="optional" default="all">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="all"/>
        <xs:enumeration value="any"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:element>
```

Elementos padre

And, Command, Constraint, CreateDocument, CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModel, CreateModelApplier, CreateModelOutput, Enabled, ExpertSettings, Not, Option, Or, Run, Visible

Elemento Constraint

Tabla 71. Atributos de Constraint

Atributo	Uso	Descripción	Valores válidos
property	**obligatorio**		*cadena*
singleSelection	opcional		*booleano*

Representación XML

```
<xs:element name="Constraint">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
```

```
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="singleSelection" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

Elementos padre

AutoModeling

Elementos hijo

And, Condition, Not, Or

Elemento Constructors

Representación de XML

```
<xs:element name="Constructors">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="CreateModelOutput"/>
      <xs:element ref="CreateDocumentOutput"/>
      <xs:element ref="CreateInteractiveModelBuilder"/>
      <xs:element ref="CreateInteractiveDocumentBuilder"/>
      <xs:element ref="CreateModelApplier"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

Elementos padre

DocumentOutput, Execution, InteractiveDocumentBuilder, InteractiveModelBuilder, ModelOutput, Node

Elementos hijo

CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModelApplier, CreateModelOutput

Elemento Container

Tabla 72. Atributos de Container

Atributo	Uso	Descripción	Valores válidos
name	**obligatorio**		*cadena*
type	opcional		*cadena*

Representación de XML

```
<xs:element name="Container">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Containers, Containers, Containers, Containers, Containers

Elemento ContainerFile

Tabla 73. Atributos de ContainerFile

Atributo	Uso	Descripción	Valores válidos
container	opcional		
containerType	opcional		

Tabla 73. Atributos de ContainerFile (continuación)

Atributo	Uso	Descripción	Valores válidos
path	**obligatorio**		

Representación de XML

```
<xs:element name="ContainerFile" type="SERVER-CONTAINER-FILE">
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
  <xs:attribute name="container" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="containerType" type="EVALUATED-STRING" use="optional"/>
</xs:element>
```

Elementos padre

InputFiles, OutputFiles

Elemento ContainerTypes

Representación de XML

```
<xs:element name="ContainerTypes">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="DocumentType"/>
      <xs:element ref="ModelType"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

Elementos padre

CommonObjects

Elementos hijo

DocumentType, ModelType

Elemento Controls

Representación de XML

```
<xs:element name="Controls">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="Menu"/>
      <xs:element ref="MenuItem"/>
      <xs:element ref="ToolbarItem"/>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

Elementos padre

UserInterface

Elementos hijo

Menu, MenuItem, ToolbarItem

Elemento CreateDocument

Tabla 74. Atributos de CreateDocument

Atributo	Uso	Descripción	Valores válidos
sourceFile	**obligatorio**		*cadena*

Tabla 74. Atributos de CreateDocument (continuación)

Atributo	Uso	Descripción	Valores válidos
target	**obligatorio**		*cadena*
type	opcional		*cadena*

Representación XML

```
<xs:element name="CreateDocument">
  <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
    <xs:choice>
      <xs:element ref="Condition"/>
      <xs:element ref="And"/>
      <xs:element ref="Or"/>
      <xs:element ref="Not"/>
    </xs:choice>
  </xs:group>
  <xs:attribute name="sourceFile" type="xs:string" use="required"/>
  <xs:attribute name="target" type="xs:string" use="required"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModelApplier, CreateModelOutput

Elementos hijo

And, Condition, Not, Or

Elementos relacionados

CreateModel

Elemento CreateDocumentOutput

Tabla 75. Atributos de CreateDocumentOutput

Atributo	Uso	Descripción	Valores válidos
type	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="CreateDocumentOutput">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

Constructors

Elementos hijo

And, Condition, CreateDocument, CreateModel, Not, Or, SetContainer, SetProperty

Elementos relacionados

CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModelApplier, CreateModelOutput

Elemento CreateInteractiveDocumentBuilder

Tabla 76. Atributos de CreateInteractiveDocumentBuilder

Atributo	Uso	Descripción	Valores válidos
type	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="CreateInteractiveDocumentBuilder">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

Constructors

Elementos hijo

And, Condition, CreateDocument, CreateModel, Not, Or, SetContainer, SetProperty

Elementos relacionados

CreateDocumentOutput, CreateInteractiveModelBuilder, CreateModelApplier, CreateModelOutput

Elemento CreateInteractiveModelBuilder

Tabla 77. Atributos de CreateInteractiveModelBuilder

Atributo	Uso	Descripción	Valores válidos
type	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="CreateInteractiveModelBuilder">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

Constructors

Elementos hijo

And, Condition, CreateDocument, CreateModel, Not, Or, SetContainer, SetProperty

Elementos relacionados

CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateModelApplier, CreateModelOutput

Elemento CreateModel

Tabla 78. Atributos de CreateModel

Atributo	Uso	Descripción	Valores válidos
signatureFile	opcional		*cadena*
sourceFile	**obligatorio**		*cadena*
target	**obligatorio**		*cadena*
type	opcional		*cadena*

Representación XML

```
<xs:element name="CreateModel">
  <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
    <xs:choice>
      <xs:element ref="Condition"/>
      <xs:element ref="And"/>
      <xs:element ref="Or"/>
      <xs:element ref="Not"/>
    </xs:choice>
  </xs:group>
  <xs:attribute name="sourceFile" type="xs:string" use="required"/>
  <xs:attribute name="target" type="xs:string" use="required"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
  <xs:sequence>
    <xs:element name="ModelDetail" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="signatureFile" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModelApplier, CreateModelOutput

Elementos hijo

And, Condition, ModelDetail, Not, Or

Elementos relacionados

CreateDocument

Elemento ModelDetail:

Tabla 79. Atributos de ModelDetail

Atributo	Uso	Descripción	Valores válidos
algorithm	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="ModelDetail" maxOccurs="unbounded">
  <xs:attribute name="algorithm" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

CreateModel

Elemento CreateModelApplier

Tabla 80. Atributos de CreateModelApplier

Atributo	Uso	Descripción	Valores válidos
type	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="CreateModelApplier">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

Constructors

Elementos hijo

And, Condition, CreateDocument, CreateModel, Not, Or, SetContainer, SetProperty

Elementos relacionados

CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModelOutput

Elemento CreateModelOutput

Tabla 81. Atributos de CreateModelOutput

Atributo	Uso	Descripción	Valores válidos
type	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="CreateModelOutput">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="SetProperty"/>
        <xs:element ref="SetContainer"/>
        <xs:element ref="CreateModel"/>
        <xs:element ref="CreateDocument"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

Constructors

Elementos hijo

And, Condition, CreateDocument, CreateModel, Not, Or, SetContainer, SetProperty

Elementos relacionados

CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModelApplier

Elemento DatabaseConnectionValue

Un valor que especifica los detalles de una conexión de base de datos.

Tabla 82. Atributos de DatabaseConnectionValue

Atributo	Uso	Descripción	Valores válidos
connectionType	**obligatorio**		*cadena*
datasourceName	**obligatorio**		*cadena*
password	**obligatorio**		*cadena*
userName	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="DatabaseConnectionValue" type="DATABASE-CONNECTION-VALUE">
  <xs:attribute name="connectionType" type="xs:string" use="required"/>
  <xs:attribute name="datasourceName" type="xs:string" use="required"/>
  <xs:attribute name="userName" type="xs:string" use="required"/>
  <xs:attribute name="password" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

Attribute, Attribute, ListValue, ListValue, ListValue, Parameter

Elemento DataFile

Tabla 83. Atributos de DataFile

Atributo	Uso	Descripción	Valores válidos
path	**obligatorio**		

Representación XML

```
<xs:element name="DataFile" type="SERVER-DATA-FILE">
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
  <xs:choice>
    <xs:element ref="DelimitedDataFormat"/>
  </xs:choice>
</xs:element>
```

Elementos padre

InputFiles, OutputFiles

Elementos hijo

DelimitedDataFormat

Elemento DataFormat

Representación de XML

```
<xs:element name="DataFormat">
  <xs:group ref="DATA-FORMAT-TYPE">
    <xs:choice>
      <xs:element ref="DelimitedDataFormat"/>
      <xs:element ref="SPSSDataFormat"/>
    </xs:choice>
  </xs:group>
</xs:element>
```

Elementos padre

FileFormatType

Elementos hijo

DelimitedDataFormat, SPSSDataFormat

Elemento DataModel

El modelo de datos entrantes o salientes de un nodo. Un modelo de datos de entrada/salida es un conjunto de campos.

Representación XML

```
<xs:element name="DataModel" type="DATA-MODEL">
  <xs:sequence>
    <xs:element name="FieldFormats" type="FIELD-FORMATS" minOccurs="0">
```

```
      <xs:sequence>
        <xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="FieldGroups" type="FIELD-GROUPS" minOccurs="0">
      <xs:sequence>
        <xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
          <xs:sequence>
            <xs:element name="FieldName">
            </xs:element>
          </xs:sequence>
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="Fields" type="FIELDS">
      <xs:sequence>
        <xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
          <xs:group ref="FIELD-CONTENT">
            <xs:sequence>
              <xs:element ref="DisplayLabel"/>
              <xs:choice minOccurs="0">
                <xs:element ref="Range"/>
                <xs:element ref="Values"/>
              </xs:choice>
              <xs:element ref="MissingValues"/>
            </xs:sequence>
          </xs:group>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

Elementos hijo

FieldFormats, FieldGroups, Fields

Elemento FieldFormats: Define los formatos de campo predeterminados. Los formatos de campo se utilizan al visualizar los valores de salida, como por ejemplo el formato general (número estándar, formatos científico o de moneda), el número de posiciones decimales que se van a mostrar, el separador decimal, etc. Actualmente, los formatos de campo sólo se utilizan para campos numéricos, aunque esto puede cambiar en versiones futuras.

Tabla 84. Atributos de FieldFormats

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*
defaultNumberFormat	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="FieldFormats" type="FIELD-FORMATS" minOccurs="0">
  <xs:sequence>
    <xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="defaultNumberFormat" type="xs:string" use="required"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos padre

DataModel

Elementos hijo

NumberFormat

Elemento NumberFormat: Define la información de formato para un campo numérico.

Tabla 85. Atributos de NumberFormat

Atributo	Uso	Descripción	Valores válidos
decimalPlaces	**obligatorio**		*nonNegativeInteger*
decimalSymbol	**obligatorio**		**period** **comma**
formatType	**obligatorio**		**standard** **scientific** **currency**
groupingSymbol	**obligatorio**		**none** **period** **comma** **space**
name	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
    <xs:enumeration value="standard"/>
    <xs:enumeration value="scientific"/>
    <xs:enumeration value="currency"/>
  </xs:attribute>
  <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
  </xs:attribute>
  <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
    <xs:enumeration value="none"/>
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
    <xs:enumeration value="space"/>
  </xs:attribute>
</xs:element>
```

Elementos padre

FieldFormats

Elemento FieldGroups: Define los grupos de campos. Los grupos de campos se utilizan para asociar campos relacionados. Por ejemplo, una pregunta de una encuesta que solicita a un encuestado que seleccione qué ubicaciones ha visitado entre un conjunto de opciones se representará como un conjunto de campos de distintivo (flag). Un grupo de campos puede utilizarse para identificar qué campos están asociados con la pregunta de la encuesta.

Tabla 86. Atributos de FieldGroups

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*

Representación XML

```
<xs:element name="FieldGroups" type="FIELD-GROUPS" minOccurs="0">
  <xs:sequence>
    <xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="FieldName">
        </xs:element>
      </xs:sequence>
```

```
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos padre

DataModel

Elementos hijo

FieldGroup

Elemento FieldGroup: Define un grupo de campos. Un grupo de campos consta de una lista de nombres de campo e información sobre el grupo de campos, como por ejemplo el nombre del grupo y la etiqueta opcional, el tipo de grupo y, para grupos multidicotomía, el valor contabilizado, es decir, el valor que representa "true".

Tabla 87. Atributos de FieldGroup

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*
countedValue	opcional		*cadena*
displayLabel	opcional		*cadena*
groupType	**obligatorio**		**fieldGroup** **multiCategorySet** **multiDichotomySet**
name	**obligatorio**		

Representación XML

```
<xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="FieldName">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-GROUP-NAME" use="required"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
  <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
    <xs:enumeration value="fieldGroup"/>
    <xs:enumeration value="multiCategorySet"/>
    <xs:enumeration value="multiDichotomySet"/>
  </xs:attribute>
  <xs:attribute name="countedValue" type="xs:string"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos padre

FieldGroups

Elementos hijo

FieldName

Elemento FieldName:

Tabla 88. Atributos de FieldName

Atributo	Uso	Descripción	Valores válidos
name	**obligatorio**		

Representación XML

```
<xs:element name="FieldName">
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
</xs:element>
```

Elementos padre

FieldGroup

Elemento Fields:

Tabla 89. Atributos de Fields

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*

Representación de XML

```
<xs:element name="Fields" type="FIELDS">
  <xs:sequence>
    <xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="FIELD-CONTENT">
        <xs:sequence>
          <xs:element ref="DisplayLabel"/>
          <xs:choice minOccurs="0">
            <xs:element ref="Range"/>
            <xs:element ref="Values"/>
          </xs:choice>
          <xs:element ref="MissingValues"/>
        </xs:sequence>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos padre

DataModel

Elementos hijo

Field

Elemento Field:

Tabla 90. Atributos de Field

Atributo	Uso	Descripción	Valores válidos
direction	opcional		**in** **out** **both** **none** **partition**
displayLabel	opcional		*cadena*
name	**obligatorio**		*cadena*

Tabla 90. Atributos de Field (continuación)

Atributo	Uso	Descripción	Valores válidos
storage	opcional		**unknown** **integer** **real** **string** **date** **time** **timestamp**
type	opcional		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless**

Representación XML

```
<xs:element name="Field" type="FIELD" minOccurs="0" maxOccurs="unbounded">
  <xs:group ref="FIELD-CONTENT">
    <xs:sequence>
      <xs:element ref="DisplayLabel"/>
      <xs:choice minOccurs="0">
        <xs:element ref="Range"/>
        <xs:element ref="Values"/>
      </xs:choice>
      <xs:element ref="MissingValues"/>
    </xs:sequence>
  </xs:group>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="type" type="FIELD-TYPE" default="auto">
    <xs:enumeration value="auto"/>
    <xs:enumeration value="range"/>
    <xs:enumeration value="discrete"/>
    <xs:enumeration value="set"/>
    <xs:enumeration value="orderedSet"/>
    <xs:enumeration value="flag"/>
    <xs:enumeration value="typeless"/>
  </xs:attribute>
  <xs:attribute name="storage" type="FIELD-STORAGE" default="unknown">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION" default="in">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
  <xs:attribute name="displayLabel" type="xs:string"/>
</xs:element>
```

Elementos padre

Fields

Elementos hijo

DisplayLabel, MissingValues, Range, Range, Values, Values

Elemento DBConnectionChooserControl

Tabla 91. Atributos de DBConnectionChooserControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="DBConnectionChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento DBTableChooserControl

Tabla 92. Atributos de DBTableChooserControl

Atributo	Uso	Descripción	Valores válidos
connectionProperty	**obligatorio**		*cadena*

Tabla 92. Atributos de DBTableChooserControl (continuación)

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="DBTableChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="connectionProperty" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento DefaultValue

Representación de XML

```
<xs:element name="DefaultValue">
  <xs:choice>
    <xs:element name="ServerTempFile">
    </xs:element>
    <xs:element name="ServerTempDir">
```

```
    </xs:element>
    <xs:element name="Identifier">
    </xs:element>
  </xs:choice>
</xs:element>
```

Elementos padre

Property, PropertyType

Elementos hijo

Identifier, ServerTempDir, ServerTempFile

Elemento ServerTempFile:

Tabla 93. Atributos de ServerTempFile

Atributo	Uso	Descripción	Valores válidos
basename	**obligatorio**		

Representación de XML

```
<xs:element name="ServerTempFile">
  <xs:attribute name="basename" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

Elementos padre

DefaultValue

Elemento ServerTempDir:

Tabla 94. Atributos de ServerTempDir

Atributo	Uso	Descripción	Valores válidos
basename	**obligatorio**		

Representación de XML

```
<xs:element name="ServerTempDir">
  <xs:attribute name="basename" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

Elementos padre

DefaultValue

Elemento Identifier:

Tabla 95. Atributos de Identifier

Atributo	Uso	Descripción	Valores válidos
basename	**obligatorio**		

Representación de XML

```
<xs:element name="Identifier">
  <xs:attribute name="basename" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

Elementos padre

DefaultValue

Elemento DelimitedDataFormat

Tabla 96. Atributos de DelimitedDataFormat

Atributo	Uso	Descripción	Valores válidos
delimeter	opcional		**tab** **comma** **semicolon** **colon** **verticalBar** **other**
eol	opcional		**cr** **crlf** **lf** **other**
includeFieldNames	opcional		*booleano*
otherDelimiter	opcional		*cadena*
otherEol	opcional		*cadena*
quoteStrings	opcional		*booleano*
stringQuote	opcional		*cadena*

Representación XML

```
<xs:element name="DelimitedDataFormat">
  <xs:attribute name="delimeter" use="optional" default="tab">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="tab"/>
        <xs:enumeration value="comma"/>
        <xs:enumeration value="semicolon"/>
        <xs:enumeration value="colon"/>
        <xs:enumeration value="verticalBar"/>
        <xs:enumeration value="other"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="otherDelimiter" type="xs:string" use="optional"/>
  <xs:attribute name="eol" use="optional" default="cr">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="cr"/>
        <xs:enumeration value="crlf"/>
        <xs:enumeration value="lf"/>
        <xs:enumeration value="other"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="otherEol" type="xs:string" use="optional"/>
  <xs:attribute name="includeFieldNames" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="quoteStrings" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="stringQuote" type="xs:string" use="optional" default="""/>
</xs:element>
```

Elementos padre

DataFile, DataFormat

Elemento DisplayLabel

Una etiqueta de visualización para un campo o valor en un idioma especificado. El atributo displayLabel puede utilizarse cuando no hay ninguna etiqueta para un idioma determinado.

Tabla 97. Atributos de DisplayLabel

Atributo	Uso	Descripción	Valores válidos
lang	opcional		*NMTOKEN*
value	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="lang" type="xs:NMTOKEN" default="en"/>
</xs:element>
```

Elementos padre

Field

Elemento DocumentBuilder

Representación de XML

```
<xs:element name="DocumentBuilder">
  <xs:sequence>
    <xs:element name="DocumentGeneration">
    </xs:element>
  </xs:sequence>
</xs:element>
```

Elementos padre

Node

Elementos hijo

DocumentGeneration

Elemento DocumentGeneration:

Tabla 98. Atributos de DocumentGeneration

Atributo	Uso	Descripción	Valores válidos
controlsId	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="DocumentGeneration">
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

DocumentBuilder

Elemento DocumentOutput

Tabla 99. Atributos de DocumentOutput

Atributo	Uso	Descripción	Valores válidos
deprecatedScriptNames	opcional		*cadena*
id	**obligatorio**		*cadena*
scriptName	opcional		*cadena*

Representación de XML

```
<xs:element name="DocumentOutput">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Extension

Elementos hijo

Constructors, Containers, ModelProvider, Properties, UserInterface

Elementos relacionados

InteractiveDocumentBuilder, InteractiveModelBuilder, ModelOutput, Node

Elemento Containers:

Representación XML

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

Node

Elementos hijo

Container

Elemento DocumentType

Define un tipo de documento nuevo.

Tabla 100. Atributos de DocumentType

Atributo	Uso	Descripción	Valores válidos
format	**obligatorio**		**utf8** **binary**
id	**obligatorio**		*cadena*
type	opcional		**unknown** **rowSet** **report** **graph**

Representación XML

```
<xs:element name="DocumentType">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="format" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="utf8"/>
        <xs:enumeration value="binary"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="type" type="DOCUMENT-TYPE" use="optional">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="rowSet"/>
    <xs:enumeration value="report"/>
    <xs:enumeration value="graph"/>
  </xs:attribute>
</xs:element>
```

Elementos padre

ContainerTypes

Elementos relacionados

ModelType

Elemento Enabled

Representación de XML

```
<xs:element name="Enabled">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

Elementos padre

ActionButton, CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, ComboBoxControl, DBConnectionChooserControl, DBTableChooserControl, ExtensionObjectPanel, ItemChooserControl, ModelViewerPanel, MultiFieldChooserControl, MultiItemChooserControl, PasswordBoxControl, PropertiesPanel, PropertiesSubPanel, PropertyControl,

RadioButtonGroupControl, SelectorPanel, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SingleItemChooserControl, SpinnerControl, StaticText, SystemControls, TabbedPanel, TableControl, TextAreaControl, TextBoxControl, TextBrowserPanel

Elementos hijo

And, Condition, Not, Or

Elemento Enumeration

Representación de XML

```
<xs:element name="Enumeration">
  <xs:sequence>
    <xs:element name="Enum" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

Elementos padre

PropertyType

Elementos hijo

Enum

Elemento Enum:

Tabla 101. Atributos de Enum

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
imagePath	opcional		*cadena*
imagePathKey	opcional		*cadena*
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*
value	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="Enum" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="imagePath" type="xs:string" use="optional"/>
  <xs:attribute name="imagePathKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Enumeration

Elemento ErrorDetail

Información complementaria sobre un error u otra condición.

Representación de XML

```
<xs:element name="ErrorDetail" type="ERROR-DETAIL">
  <xs:sequence>
    <xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
        </xs:element>
        <xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

Elementos hijo

Diagnostic

Elemento Diagnostic:

Tabla 102. Atributos de Diagnostic

Atributo	Uso	Descripción	Valores válidos
code	**obligatorio**		*entero*
severity	opcional		**unknown** **information** **warning** **error** **fatal**
source	opcional		*cadena*
subCode	opcional		*entero*

Representación XML

```
<xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
    </xs:element>
    <xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="subCode" type="xs:integer" default="0"/>
  <xs:attribute name="severity" type="DIAGNOSTIC-SEVERITY" default="error">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="information"/>
    <xs:enumeration value="warning"/>
    <xs:enumeration value="error"/>
    <xs:enumeration value="fatal"/>
  </xs:attribute>
  <xs:attribute name="source" type="xs:string"/>
</xs:element>
```

Elementos padre

ErrorDetail

Elementos hijo

Message, Parameter

Elemento Message:

Tabla 103. Atributos de Message

Atributo	Uso	Descripción	Valores válidos
lang	opcional		*NMTOKEN*

Representación de XML

```
<xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
  <xs:attribute name="lang" type="xs:NMTOKEN"/>
</xs:element>
```

Elementos padre

Diagnostic

Elemento Parameter:

Representación de XML

```
<xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

Elementos padre

Diagnostic

Elemento Executable

Representación XML

```
<xs:element name="Executable">
  <xs:sequence>
    <xs:element ref="Run" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

Execution

Elementos hijo

Run

Elemento Execution

Representación de XML

```
<xs:element name="Execution">
  <xs:sequence>
    <xs:element ref="Properties" minOccurs="0"/>
    <xs:element ref="InputFiles"/>
    <xs:element ref="OutputFiles"/>
    <xs:choice>
      <xs:element ref="Executable"/>
      <xs:element ref="Module"/>
    </xs:choice>
    <xs:element ref="Constructors" minOccurs="0"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

Node

Elementos hijo

Constructors, Executable, InputFiles, Module, OutputFiles, Properties

Elemento Extension

Define el contenedor de extensión de nivel superior.

Tabla 104. Atributos de Extension

Atributo	Uso	Descripción	Valores válidos
debug	opcional		*booleano*
version	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="Extension">
  <xs:sequence>
    <xs:element ref="ExtensionDetails"/>
    <xs:element ref="Resources"/>
    <xs:element ref="License" minOccurs="0"/>
    <xs:element ref="CommonObjects"/>
    <xs:element ref="UserInterface" minOccurs="0"/>
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:choice>
        <xs:element ref="Node"/>
        <xs:element ref="ModelOutput"/>
        <xs:element ref="DocumentOutput"/>
        <xs:element ref="InteractiveModelBuilder"/>
        <xs:element ref="InteractiveDocumentBuilder"/>
      </xs:choice>
    </xs:sequence>
  </xs:sequence>
  <xs:attribute name="version" type="xs:string" use="required"/>
  <xs:attribute name="debug" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

Elementos hijo

CommonObjects, DocumentOutput, ExtensionDetails, InteractiveDocumentBuilder, InteractiveModelBuilder, License, ModelOutput, Node, Resources, UserInterface

Elemento ExtensionDetails

Define información sobre la extensión, como el ID de extensión, el proveedor de extensión e información sobre la versión.

Tabla 105. Atributos de ExtensionDetails

Atributo	Uso	Descripción	Valores válidos
copyright	opcional		*cadena*
description	opcional		*cadena*
id	**obligatorio**		*cadena*
label	**obligatorio**		*cadena*
provider	opcional		*cadena*
providerTag	**obligatorio**		*cadena*
version	opcional		*cadena*

Representación de XML

```
<xs:element name="ExtensionDetails">
  <xs:attribute name="providerTag" type="xs:string" use="required"/>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="version" type="xs:string"/>
```

```
  <xs:attribute name="provider" type="xs:string" use="optional" default="(unknown)"/>
  <xs:attribute name="copyright" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Extension

Elemento ExtensionObjectPanel

Tabla 106. Atributos de ExtensionObjectPanel

Atributo	Uso	Descripción	Valores válidos
id	opcional		*cadena*
panelClass	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="ExtensionObjectPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="panelClass" type="xs:string" use="required"/>
  <xs:attribute name="id" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel, Tab

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

ActionButton, ComboBoxControl, ModelViewerPanel, SelectorPanel, StaticText, SystemControls, TabbedPanel, TextBrowserPanel

Elemento Field

Tabla 107. Atributos de Field

Atributo	Uso	Descripción	Valores válidos
direction	opcional		**in** **out** **both** **none** **partition**
label	opcional		*cadena*
name	**obligatorio**		

Tabla 107. Atributos de Field (continuación)

Atributo	Uso	Descripción	Valores válidos
storage	opcional		**unknown** **integer** **real** **string** **date** **time** **timestamp**
type	opcional		**auto** **range** **discrete** **set** **orderedSet** **flag** **typeless**

Representación de XML

```
<xs:element name="Field" type="FIELD-DECLARATION">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Range" minOccurs="0"/>
      <xs:element ref="Values" minOccurs="0"/>
      <xs:element ref="NumericInfo" minOccurs="0"/>
      <xs:element name="MissingValues" minOccurs="0">
        <xs:sequence>
          <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="Range" minOccurs="0"/>
        </xs:sequence>
      </xs:element>
      <xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
      </xs:element>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
  <xs:attribute name="storage" type="FIELD-STORAGE">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="real"/>
    <xs:enumeration value="string"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="time"/>
    <xs:enumeration value="timestamp"/>
  </xs:attribute>
  <xs:attribute name="type" type="FIELD-TYPE">
    <xs:enumeration value="auto"/>
    <xs:enumeration value="range"/>
    <xs:enumeration value="discrete"/>
    <xs:enumeration value="set"/>
    <xs:enumeration value="orderedSet"/>
    <xs:enumeration value="flag"/>
    <xs:enumeration value="typeless"/>
  </xs:attribute>
  <xs:attribute name="direction" type="FIELD-DIRECTION">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string"/>
</xs:element>
```

Elementos hijo

MissingValues, ModelField, NumericInfo, Range, Range, Values, Values

Elemento MissingValues:

Tabla 108. Atributos de MissingValues

Atributo	Uso	Descripción	Valores válidos
treatNullAsMissing	opcional		*booleano*
treatWhitespaceAsMissing	opcional		*booleano*

Representación XML

```
<xs:element name="MissingValues" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Values" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Range" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

Elementos padre

Field

Elementos hijo

Range, Range, Values, Values

Elemento ModelField:

Tabla 109. Atributos de ModelField

Atributo	Uso	Descripción	Valores válidos
group	opcional		
role	**obligatorio**		**unknown** **predictedValue** **predictedDisplayValue** **probability** **residual** **standardError** **entityId** **entityAffinity** **upperConfidenceLimit** **lowerConfidenceLimit** **propensity** **value** **supplementary**
tag	opcional		*cadena*
targetField	opcional		*cadena*
value	opcional		*cadena*

Representación XML

```
<xs:element name="ModelField" type="MODEL-FIELD-INFORMATION" minOccurs="0">
  <xs:attribute name="role" type="MODEL-FIELD-ROLE" use="required">
    <xs:enumeration value="unknown"/>
```

```
    <xs:enumeration value="predictedValue"/>
    <xs:enumeration value="predictedDisplayValue"/>
    <xs:enumeration value="probability"/>
    <xs:enumeration value="residual"/>
    <xs:enumeration value="standardError"/>
    <xs:enumeration value="entityId"/>
    <xs:enumeration value="entityAffinity"/>
    <xs:enumeration value="upperConfidenceLimit"/>
    <xs:enumeration value="lowerConfidenceLimit"/>
    <xs:enumeration value="propensity"/>
    <xs:enumeration value="value"/>
    <xs:enumeration value="supplementary"/>
  </xs:attribute>
  <xs:attribute name="targetField" type="xs:string"/>
  <xs:attribute name="value" type="xs:string"/>
  <xs:attribute name="group" type="MODEL-FIELD-GROUP"/>
  <xs:attribute name="tag" type="xs:string"/>
</xs:element>
```

Elementos padre

Field

Elemento FieldFormats

Define los formatos de campo predeterminados. Los formatos de campo se utilizan al visualizar los valores de salida, como por ejemplo el formato general (número estándar, formatos científico o de moneda), el número de posiciones decimales que se van a mostrar, el separador decimal, etc. Actualmente, los formatos de campo sólo se utilizan para campos numéricos, aunque esto puede cambiar en versiones futuras.

Tabla 110. Atributos de FieldFormats

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*
defaultNumberFormat	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="FieldFormats" type="FIELD-FORMATS">
  <xs:sequence>
    <xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="defaultNumberFormat" type="xs:string" use="required"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos hijo

NumberFormat

Elemento NumberFormat: Define la información de formato para un campo numérico.

Tabla 111. Atributos de NumberFormat

Atributo	Uso	Descripción	Valores válidos
decimalPlaces	**obligatorio**		*nonNegativeInteger*
decimalSymbol	**obligatorio**		**period** **comma**
formatType	**obligatorio**		**standard** **scientific** **currency**

Tabla 111. Atributos de NumberFormat (continuación)

Atributo	Uso	Descripción	Valores válidos
groupingSymbol	**obligatorio**		**none** **period** **comma** **space**
name	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
    <xs:enumeration value="standard"/>
    <xs:enumeration value="scientific"/>
    <xs:enumeration value="currency"/>
  </xs:attribute>
  <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
  </xs:attribute>
  <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
    <xs:enumeration value="none"/>
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
    <xs:enumeration value="space"/>
  </xs:attribute>
</xs:element>
```

Elementos padre

FieldFormats

Elemento FieldGroup

Define un grupo de campos. Un grupo de campos consta de una lista de nombres de campo e información sobre el grupo de campos, como por ejemplo el nombre del grupo y la etiqueta opcional, el tipo de grupo y, para grupos multidicotomía, el valor contabilizado, es decir, el valor que representa "true".

Tabla 112. Atributos de FieldGroup

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*
countedValue	opcional		*cadena*
displayLabel	opcional		*cadena*
groupType	**obligatorio**		**fieldGroup** **multiCategorySet** **multiDichotomySet**
name	**obligatorio**		

Representación de XML

```
<xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION">
  <xs:sequence>
    <xs:element name="FieldName">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-GROUP-NAME" use="required"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
  <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
    <xs:enumeration value="fieldGroup"/>
```

```
    <xs:enumeration value="multiCategorySet"/>
    <xs:enumeration value="multiDichotomySet"/>
  </xs:attribute>
  <xs:attribute name="countedValue" type="xs:string"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos hijo

FieldName

Elemento FieldName:

Tabla 113. Atributos de FieldName

Atributo	Uso	Descripción	Valores válidos
name	**obligatorio**		

Representación XML

```
<xs:element name="FieldName">
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
</xs:element>
```

Elementos padre

FieldGroup

Elemento FieldGroups

Define los grupos de campos. Los grupos de campos se utilizan para asociar campos relacionados. Por ejemplo, una pregunta de una encuesta que solicita a un encuestado que seleccione qué ubicaciones ha visitado entre un conjunto de opciones se representará como un conjunto de campos de distintivo (flag). Un grupo de campos puede utilizarse para identificar qué campos están asociados con la pregunta de la encuesta.

Tabla 114. Atributos de FieldGroups

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*

Representación XML

```
<xs:element name="FieldGroups" type="FIELD-GROUPS">
  <xs:sequence>
    <xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="FieldName">
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos hijo

FieldGroup

Elemento FieldGroup: Define un grupo de campos. Un grupo de campos consta de una lista de nombres de campo e información sobre el grupo de campos, como por ejemplo el nombre del grupo y la etiqueta opcional, el tipo de grupo y, para grupos multidicotomía, el valor contabilizado, es decir, el valor que representa "true".

Tabla 115. Atributos de FieldGroup

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*
countedValue	opcional		*cadena*
displayLabel	opcional		*cadena*
groupType	**obligatorio**		**fieldGroup** **multiCategorySet** **multiDichotomySet**
name	**obligatorio**		

Representación XML

```
<xs:element name="FieldGroup" type="FIELD-GROUP-DECLARATION" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="FieldName">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="FIELD-GROUP-NAME" use="required"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
  <xs:attribute name="groupType" type="FIELD-GROUP-TYPE" use="required">
    <xs:enumeration value="fieldGroup"/>
    <xs:enumeration value="multiCategorySet"/>
    <xs:enumeration value="multiDichotomySet"/>
  </xs:attribute>
  <xs:attribute name="countedValue" type="xs:string"/>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos padre

FieldGroups

Elementos hijo

FieldName

Elemento FieldName:

Tabla 116. Atributos de FieldName

Atributo	Uso	Descripción	Valores válidos
name	**obligatorio**		

Representación XML

```
<xs:element name="FieldName">
  <xs:attribute name="name" type="FIELD-NAME" use="required"/>
</xs:element>
```

Elementos padre

FieldGroup

Elemento FileFormatType

Tabla 117. Atributos de FileFormatType

Atributo	Uso	Descripción	Valores válidos
name	opcional		

Representación de XML

```
<xs:element name="FileFormatType">
  <xs:sequence>
    <xs:group ref="FILE-FORMAT">
      <xs:choice>
        <xs:element ref="UTF8Format"/>
        <xs:element ref="BinaryFormat"/>
        <xs:element ref="DataFormat"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="name" type="EVALUATED-STRING" use="optional"/>
</xs:element>
```

Elementos padre

FileFormatTypes

Elementos hijo

BinaryFormat, DataFormat, UTF8Format

Elemento FileFormatTypes

Representación XML

```
<xs:element name="FileFormatTypes">
  <xs:sequence>
    <xs:element ref="FileFormatType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

CommonObjects

Elementos hijo

FileFormatType

Elemento ForEach

Tabla 118. Atributos de ForEach

Atributo	Uso	Descripción	Valores válidos
container	opcional		*cadena*
from	opcional		*cadena*
inFields	opcional		*cadena*
inFieldValues	opcional		*cadena*
inProperty	opcional		*cadena*
step	opcional		*cadena*
to	opcional		*cadena*
var	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="ForEach">
  <xs:sequence maxOccurs="unbounded">
    <xs:group ref="DATA-MODEL-EXPRESSION">
      <xs:choice>
        <xs:element ref="ForEach"/>
        <xs:element ref="AddField"/>
        <xs:element ref="ChangeField"/>
```

```
        <xs:element ref="RemoveField"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="var" type="xs:string" use="required"/>
  <xs:attribute name="inProperty" type="xs:string" use="optional"/>
  <xs:attribute name="inFields" type="xs:string" use="optional"/>
  <xs:attribute name="inFieldValues" type="xs:string" use="optional"/>
  <xs:attribute name="from" type="xs:string" use="optional"/>
  <xs:attribute name="to" type="xs:string" use="optional"/>
  <xs:attribute name="step" type="xs:string" use="optional"/>
  <xs:attribute name="container" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

ForEach, ModelFields

Elementos hijo

AddField, ChangeField, ForEach, RemoveField

Elemento Icon

Tabla 119. Atributos de Icon

Atributo	Uso	Descripción	Valores válidos
imagePath	**obligatorio**		*cadena*
resourceID	opcional		*cadena*
type	**obligatorio**		**standardNode** **smallNode** **standardWindow**

Representación XML

```
<xs:element name="Icon">
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="standardNode"/>
        <xs:enumeration value="smallNode"/>
        <xs:enumeration value="standardWindow"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="imagePath" type="xs:string" use="required"/>
  <xs:attribute name="resourceID" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Icons, Palette

Elemento Icons

Representación XML

```
<xs:element name="Icons">
  <xs:sequence>
    <xs:element ref="Icon" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

UserInterface

Elementos hijo

Icon

Elemento InputFiles

Representación de XML

```
<xs:element name="InputFiles">
  <xs:group ref="RUNTIME-FILES">
    <xs:sequence>
      <xs:element ref="DataFile"/>
      <xs:element ref="ContainerFile" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
</xs:element>
```

Elementos padre

Execution, Module

Elementos hijo

ContainerFile, DataFile

Elemento InteractiveDocumentBuilder

Tabla 120. Atributos de InteractiveDocumentBuilder

Atributo	Uso	Descripción	Valores válidos
deprecatedScriptNames	opcional		*cadena*
id	**obligatorio**		*cadena*
scriptName	opcional		*cadena*

Representación de XML

```
<xs:element name="InteractiveDocumentBuilder">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Extension

Elementos hijo

Constructors, Containers, ModelProvider, Properties, UserInterface

Elementos relacionados

DocumentOutput, InteractiveModelBuilder, ModelOutput, Node

Elemento Containers:

Representación XML

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

Node

Elementos hijo

Container

Elemento InteractiveModelBuilder

Tabla 121. Atributos de InteractiveModelBuilder

Atributo	Uso	Descripción	Valores válidos
deprecatedScriptNames	opcional		*cadena*
id	**obligatorio**		*cadena*
scriptName	opcional		*cadena*

Representación de XML

```
<xs:element name="InteractiveModelBuilder">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Extension

Elementos hijo

Constructors, Containers, ModelProvider, Properties, UserInterface

Elementos relacionados

DocumentOutput, InteractiveDocumentBuilder, ModelOutput, Node

Elemento Containers:

Representación XML

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

Node

Elementos hijo

Container

Elemento Layout

Tabla 122. Atributos de Layout

Atributo	Uso	Descripción	Valores válidos
anchor	opcional		**north** **northeast** **east** **southeast** **south** **southwest** **west** **northwest** **center**
columnWeight	opcional		*double*
fill	opcional		**horizontal** **vertical** **both** **none**
gridColumn	opcional		*nonNegativeInteger*
gridHeight	opcional		*nonNegativeInteger*
gridRow	opcional		*nonNegativeInteger*
gridWidth	opcional		*nonNegativeInteger*
leftIndent	opcional		*nonNegativeInteger*
rowIncrement	opcional		*nonNegativeInteger*
rowWeight	opcional		*double*

Representación XML

```
<xs:element name="Layout">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Cell">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="gridRow" type="xs:nonNegativeInteger" use="optional"/>
```

```
  <xs:attribute name="gridColumn" type="xs:nonNegativeInteger" use="optional"/>
  <xs:attribute name="rowIncrement" type="xs:nonNegativeInteger" use="optional"/>
  <xs:attribute name="gridWidth" type="xs:nonNegativeInteger" use="optional" default="1"/>
  <xs:attribute name="gridHeight" type="xs:nonNegativeInteger" use="optional" default="1"/>
  <xs:attribute name="rowWeight" type="xs:double" use="optional"/>
  <xs:attribute name="columnWeight" type="xs:double" use="optional"/>
  <xs:attribute name="fill" type="UI-COMPONENT-FILL" use="optional" default="none">
    <xs:enumeration value="horizontal"/>
    <xs:enumeration value="vertical"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
  </xs:attribute>
  <xs:attribute name="anchor" type="UI-COMPONENT-ANCHOR" use="optional" default="west">
    <xs:enumeration value="north"/>
    <xs:enumeration value="northeast"/>
    <xs:enumeration value="east"/>
    <xs:enumeration value="southeast"/>
    <xs:enumeration value="south"/>
    <xs:enumeration value="southwest"/>
    <xs:enumeration value="west"/>
    <xs:enumeration value="northwest"/>
    <xs:enumeration value="center"/>
  </xs:attribute>
  <xs:attribute name="leftIndent" type="xs:nonNegativeInteger" use="optional"/>
</xs:element>
```

Elementos padre

ActionButton, CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, ComboBoxControl, DBConnectionChooserControl, DBTableChooserControl, ExtensionObjectPanel, ItemChooserControl, ModelViewerPanel, MultiFieldChooserControl, MultiItemChooserControl, PasswordBoxControl, PropertiesPanel, PropertiesSubPanel, PropertyControl, RadioButtonGroupControl, SelectorPanel, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SingleItemChooserControl, SpinnerControl, StaticText, SystemControls, TabbedPanel, TableControl, TextAreaControl, TextBoxControl, TextBrowserPanel

Elementos hijo

Cell

Elemento Cell:

Tabla 123. Atributos de Cell

Atributo	Uso	Descripción	Valores válidos
column	**obligatorio**		*nonNegativeInteger*
row	**obligatorio**		*nonNegativeInteger*
width	**obligatorio**		*nonNegativeInteger*

Representación XML

```
<xs:element name="Cell">
  <xs:attribute name="row" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="column" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="width" type="xs:nonNegativeInteger" use="required"/>
</xs:element>
```

Elementos padre

Layout

Elemento License

Reservado para uso del sistema.

Representación de XML

```
<xs:element name="License">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element ref="OptionCode"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

Extension

Elementos hijo

OptionCode

Elemento ListValue

Una secuencia de valores. Todos los valores deben tener el mismo tipo de contenido, pero esto no se comprueba.

Representación de XML

```
<xs:element name="ListValue" type="LIST-VALUE">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
</xs:element>
```

Elementos padre

Attribute, Attribute, ListValue, ListValue, ListValue, Parameter

Elementos hijo

DatabaseConnectionValue, ListValue, MapValue, StructuredValue, Value

Elemento MapValue

Un conjunto de entradas de correlación, cada una formada por una clave y un valor.

Representación de XML

```
<xs:element name="MapValue" type="MAP-VALUE">
  <xs:sequence>
    <xs:element name="MapEntry" type="MAP-ENTRY" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="KeyValue" type="KEY-VALUE">
        </xs:element>
        <xs:element name="StructuredValue" type="STRUCTURED-VALUE">
          <xs:sequence>
            <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
              <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
                <xs:choice>
                  <xs:element ref="MapValue"/>
                  <xs:element ref="StructuredValue"/>
                  <xs:element ref="ListValue"/>
                  <xs:element ref="Value"/>
                  <xs:element ref="DatabaseConnectionValue"/>
                </xs:choice>
              </xs:group>
              <xs:sequence>
                <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
                  <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
                    <xs:choice>
                      <xs:element ref="MapValue"/>
                      <xs:element ref="StructuredValue"/>
                      <xs:element ref="ListValue"/>
```

```
                    <xs:element ref="Value"/>
                    <xs:element ref="DatabaseConnectionValue"/>
                  </xs:choice>
                </xs:group>
              </xs:element>
            </xs:sequence>
          </xs:element>
        </xs:sequence>
      </xs:element>
    </xs:sequence>
  </xs:element>
</xs:element>
```

Elementos padre

Attribute, Attribute, ListValue, ListValue, ListValue, Parameter

Elementos hijo

MapEntry

Elemento MapEntry: Una entrada de una correlación de propiedades por clave. Cada entrada consta de una clave y un valor asociado.

Representación XML

```
<xs:element name="MapEntry" type="MAP-ENTRY" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="KeyValue" type="KEY-VALUE">
    </xs:element>
    <xs:element name="StructuredValue" type="STRUCTURED-VALUE">
      <xs:sequence>
        <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
          <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
            <xs:choice>
              <xs:element ref="MapValue"/>
              <xs:element ref="StructuredValue"/>
              <xs:element ref="ListValue"/>
              <xs:element ref="Value"/>
              <xs:element ref="DatabaseConnectionValue"/>
            </xs:choice>
          </xs:group>
          <xs:sequence>
            <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
              <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
                <xs:choice>
                  <xs:element ref="MapValue"/>
                  <xs:element ref="StructuredValue"/>
                  <xs:element ref="ListValue"/>
                  <xs:element ref="Value"/>
                  <xs:element ref="DatabaseConnectionValue"/>
                </xs:choice>
              </xs:group>
            </xs:element>
          </xs:sequence>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

Elementos padre

MapValue

Elementos hijo

KeyValue, StructuredValue

Elemento KeyValue: El valor de clave de una entrada de correlación.

Tabla 124. Atributos de KeyValue

Atributo	Uso	Descripción	Valores válidos
value	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="KeyValue" type="KEY-VALUE">
  <xs:attribute name="value" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

MapEntry

Elemento StructuredValue: Una secuencia de valores definidos ("atributos").

Representación de XML

```
<xs:element name="StructuredValue" type="STRUCTURED-VALUE">
  <xs:sequence>
    <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
      <xs:sequence>
        <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
          <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
            <xs:choice>
              <xs:element ref="MapValue"/>
              <xs:element ref="StructuredValue"/>
              <xs:element ref="ListValue"/>
              <xs:element ref="Value"/>
              <xs:element ref="DatabaseConnectionValue"/>
            </xs:choice>
          </xs:group>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

Elementos padre

MapEntry

Elementos hijo

Attribute

Elemento Attribute:

Tabla 125. Atributos de Attribute

Atributo	Uso	Descripción	Valores válidos
name	**obligatorio**		*cadena*
value	opcional		*cadena*

Representación de XML

```
<xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
  <xs:sequence>
    <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string"/>
</xs:element>
```

Elementos padre

StructuredValue

Elementos hijo

DatabaseConnectionValue, ListValue, ListValue, MapValue, StructuredValue, Value

Elemento ListValue: Una secuencia de valores. Todos los valores deben tener el mismo tipo de contenido, pero esto no se comprueba.

Representación de XML

```
<xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
</xs:element>
```

Elementos padre

Attribute

Elementos hijo

DatabaseConnectionValue, ListValue, MapValue, StructuredValue, Value

Elemento Menu

Tabla 126. Atributos de Menu

Atributo	Uso	Descripción	Valores válidos
id	**obligatorio**		*cadena*
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*

Tabla 126. Atributos de Menu (continuación)

Atributo	Uso	Descripción	Valores válidos
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
offset	opcional		*nonNegativeInteger*
separatorAfter	opcional		*booleano*
separatorBefore	opcional		*booleano*
showIcon	opcional		*booleano*
showLabel	opcional		*booleano*
systemMenu	**obligatorio**		**file** **edit** **insert** **view** **tools** **window** **help** **generate** **file.project** **file.outputs** **file.models** **edit.stream** **edit.node** **edit.outputs** **edit.models** **edit.project** **tools.repository** **tools.options** **tools.streamProperties**

Representación de XML

```
<xs:element name="Menu">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="systemMenu" type="STANDARD-MENU" use="required">
    <xs:enumeration value="file"/>
    <xs:enumeration value="edit"/>
    <xs:enumeration value="insert"/>
    <xs:enumeration value="view"/>
    <xs:enumeration value="tools"/>
    <xs:enumeration value="window"/>
    <xs:enumeration value="help"/>
    <xs:enumeration value="generate"/>
    <xs:enumeration value="file.project"/>
    <xs:enumeration value="file.outputs"/>
    <xs:enumeration value="file.models"/>
    <xs:enumeration value="edit.stream"/>
    <xs:enumeration value="edit.node"/>
    <xs:enumeration value="edit.outputs"/>
    <xs:enumeration value="edit.models"/>
    <xs:enumeration value="edit.project"/>
    <xs:enumeration value="tools.repository"/>
    <xs:enumeration value="tools.options"/>
    <xs:enumeration value="tools.streamProperties"/>
  </xs:attribute>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"/>
</xs:element>
```

Elementos padre

Controls

Elemento MenuItem

Tabla 127. Atributos de MenuItem

Atributo	Uso	Descripción	Valores válidos
action	**obligatorio**		*cadena*
customMenu	opcional		*cadena*
offset	opcional		*nonNegativeInteger*
separatorAfter	opcional		*booleano*
separatorBefore	opcional		*booleano*
showIcon	opcional		*booleano*
showLabel	opcional		*booleano*
systemMenu	opcional		**file** **edit** **insert** **view** **tools** **window** **help** **generate** **file.project** **file.outputs** **file.models** **edit.stream** **edit.node** **edit.outputs** **edit.models** **edit.project** **tools.repository** **tools.options** **tools.streamProperties**

Representación XML

```
<xs:element name="MenuItem">
  <xs:attribute name="action" type="xs:string" use="required"/>
  <xs:attribute name="systemMenu" type="STANDARD-MENU" use="optional">
    <xs:enumeration value="file"/>
    <xs:enumeration value="edit"/>
    <xs:enumeration value="insert"/>
    <xs:enumeration value="view"/>
    <xs:enumeration value="tools"/>
    <xs:enumeration value="window"/>
    <xs:enumeration value="help"/>
    <xs:enumeration value="generate"/>
    <xs:enumeration value="file.project"/>
    <xs:enumeration value="file.outputs"/>
    <xs:enumeration value="file.models"/>
    <xs:enumeration value="edit.stream"/>
    <xs:enumeration value="edit.node"/>
    <xs:enumeration value="edit.outputs"/>
    <xs:enumeration value="edit.models"/>
    <xs:enumeration value="edit.project"/>
    <xs:enumeration value="tools.repository"/>
    <xs:enumeration value="tools.options"/>
    <xs:enumeration value="tools.streamProperties"/>
  </xs:attribute>
  <xs:attribute name="customMenu" type="xs:string" use="optional"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="false"/>
```

```
  <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"/>
</xs:element>
```

Elementos padre

Controls

Elemento MissingValues

Tabla 128. Atributos de MissingValues

Atributo	Uso	Descripción	Valores válidos
treatNullAsMissing	opcional		*booleano*
treatWhitespaceAsMissing	opcional		*booleano*

Representación de XML

```
<xs:element name="MissingValues" type="MISSING-VALUES" minOccurs="0">
  <xs:sequence>
    <xs:element name="Range" type="RANGE">
    </xs:element>
    <xs:element name="Values" type="FIELD-VALUES">
      <xs:sequence>
        <xs:element name="Value" type="FIELD-VALUE" minOccurs="0" maxOccurs="unbounded">
          <xs:sequence>
            <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
            </xs:element>
          </xs:sequence>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="treatNullAsMissing" type="xs:boolean" default="true"/>
  <xs:attribute name="treatWhitespaceAsMissing" type="xs:boolean" default="false"/>
</xs:element>
```

Elementos padre

Field

Elementos hijo

Range, Values

Elemento Range:

Tabla 129. Atributos de Range

Atributo	Uso	Descripción	Valores válidos
maxValue	**obligatorio**		*cadena*
minValue	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="Range" type="RANGE">
  <xs:attribute name="minValue" type="xs:string" use="required"/>
  <xs:attribute name="maxValue" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

MissingValues

Elemento Values:

Tabla 130. Atributos de Values

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*

Representación XML

```
<xs:element name="Values" type="FIELD-VALUES">
  <xs:sequence>
    <xs:element name="Value" type="FIELD-VALUE" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos padre

MissingValues

Elementos hijo

Value

Elemento Value:

Tabla 131. Atributos de Value

Atributo	Uso	Descripción	Valores válidos
code	**obligatorio**		*entero*
displayLabel	opcional		*cadena*
flagValue	opcional		*booleano*
value	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="Value" type="FIELD-VALUE" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="flagValue" type="xs:boolean"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
</xs:element>
```

Elementos padre

Values

Elementos hijo

DisplayLabel

Elemento DisplayLabel: Una etiqueta de visualización para un campo o valor en un idioma especificado. El atributo displayLabel puede utilizarse cuando no hay ninguna etiqueta para un idioma determinado.

Tabla 132. Atributos de DisplayLabel

Atributo	Uso	Descripción	Valores válidos
lang	opcional		*NMTOKEN*
value	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="lang" type="xs:NMTOKEN" default="en"/>
</xs:element>
```

Elementos padre

Values

Elemento ModelBuilder

Tabla 133. Atributos de ModelBuilder

Atributo	Uso	Descripción	Valores válidos
allowNoInputs	opcional		*booleano*
allowNoOutputs	opcional		*booleano*
miningFunctions	**obligatorio**		*cualquiera*
nullifyBlanks	opcional		*booleano*

Representación de XML

```
<xs:element name="ModelBuilder">
  <xs:sequence>
    <xs:element name="Algorithm">
    </xs:element>
    <xs:element name="ModelingFields" minOccurs="0">
      <xs:sequence>
        <xs:element name="InputFields" minOccurs="0">
        </xs:element>
        <xs:element name="OutputFields" minOccurs="0">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="ModelGeneration">
    </xs:element>
    <xs:element name="ModelFields">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:group ref="DATA-MODEL-EXPRESSION">
          <xs:choice>
            <xs:element ref="ForEach"/>
            <xs:element ref="AddField"/>
            <xs:element ref="ChangeField"/>
            <xs:element ref="RemoveField"/>
          </xs:choice>
        </xs:group>
      </xs:sequence>
    </xs:element>
    <xs:element name="ModelEvaluation" minOccurs="0">
      <xs:sequence>
        <xs:element name="RawPropensity" minOccurs="0">
        </xs:element>
        <xs:element name="AdjustedPropensity" minOccurs="0">
        </xs:element>
        <xs:element name="VariableImportance" minOccurs="0">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element name="AutoModeling" minOccurs="0">
      <xs:sequence>
```

```
        <xs:element name="SimpleSettings">
          <xs:sequence>
            <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:element>
        <xs:element name="ExpertSettings" minOccurs="0">
          <xs:sequence>
            <xs:element ref="Condition" minOccurs="0"/>
            <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:element>
        <xs:element name="PropertyMap" minOccurs="0">
          <xs:sequence>
            <xs:element name="PropertyMapping" maxOccurs="unbounded">
            </xs:element>
          </xs:sequence>
        </xs:element>
        <xs:element ref="Constraint" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="miningFunctions" use="required"/>
  <xs:attribute name="allowNoInputs" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="allowNoOutputs" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="nullifyBlanks" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

Elementos padre

Node

Elementos hijo

Algorithm, AutoModeling, ModelEvaluation, ModelFields, ModelGeneration, ModelingFields

Elemento Algorithm:

Tabla 134. Atributos de Algorithm

Atributo	Uso	Descripción	Valores válidos
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*
value	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="Algorithm">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

ModelBuilder

Elemento ModelingFields:

Tabla 135. Atributos de ModelingFields

Atributo	Uso	Descripción	Valores válidos
controlsId	**obligatorio**		*cadena*
ignoreBOTH	opcional		*booleano*

Representación de XML

```
<xs:element name="ModelingFields" minOccurs="0">
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
  <xs:sequence>
    <xs:element name="InputFields" minOccurs="0">
    </xs:element>
    <xs:element name="OutputFields" minOccurs="0">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="ignoreBOTH" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

Elementos padre

ModelBuilder

Elementos hijo

InputFields, OutputFields

Elemento InputFields:

Tabla 136. Atributos de InputFields

Atributo	Uso	Descripción	Valores válidos
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*
multiple	**obligatorio**		*booleano*
onlyDatetime	opcional		*booleano*
onlyDiscrete	opcional		*booleano*
onlyNumeric	opcional		*booleano*
onlyRanges	opcional		*booleano*
onlySymbolic	opcional		*booleano*
property	**obligatorio**		*cadena*
storage	opcional		*cadena*
types	opcional		*cadena*

Representación de XML

```
<xs:element name="InputFields" minOccurs="0">
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="multiple" type="xs:boolean" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

ModelingFields

Elemento OutputFields:

Tabla 137. Atributos de OutputFields

Atributo	Uso	Descripción	Valores válidos
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*
multiple	**obligatorio**		*booleano*
onlyDatetime	opcional		*booleano*
onlyDiscrete	opcional		*booleano*
onlyNumeric	opcional		*booleano*
onlyRanges	opcional		*booleano*
onlySymbolic	opcional		*booleano*
property	**obligatorio**		*cadena*
storage	opcional		*cadena*
types	opcional		*cadena*

Representación de XML

```
<xs:element name="OutputFields" minOccurs="0">
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="multiple" type="xs:boolean" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

ModelingFields

Elemento ModelGeneration:

Tabla 138. Atributos de ModelGeneration

Atributo	Uso	Descripción	Valores válidos
controlsId	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="ModelGeneration">
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

ModelBuilder

Elemento ModelFields:

Representación de XML

```
<xs:element name="ModelFields">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:group ref="DATA-MODEL-EXPRESSION">
      <xs:choice>
        <xs:element ref="ForEach"/>
```

```
        <xs:element ref="AddField"/>
        <xs:element ref="ChangeField"/>
        <xs:element ref="RemoveField"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

Elementos padre

ModelBuilder

Elementos hijo

AddField, ChangeField, ForEach, RemoveField

Elemento ModelEvaluation:

Tabla 139. Atributos de ModelEvaluation

Atributo	Uso	Descripción	Valores válidos
container	**obligatorio**		*cualquiera*
controlsId	**obligatorio**		*cadena*
outputContainer	opcional		*cualquiera*

Representación de XML

```
<xs:element name="ModelEvaluation" minOccurs="0">
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
  <xs:sequence>
    <xs:element name="RawPropensity" minOccurs="0">
    </xs:element>
    <xs:element name="AdjustedPropensity" minOccurs="0">
    </xs:element>
    <xs:element name="VariableImportance" minOccurs="0">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="container" use="required"/>
  <xs:attribute name="outputContainer" use="optional"/>
</xs:element>
```

Elementos padre

ModelBuilder

Elementos hijo

AdjustedPropensity, RawPropensity, VariableImportance

Elemento RawPropensity:

Tabla 140. Atributos de RawPropensity

Atributo	Uso	Descripción	Valores válidos
availabilityProperty	opcional		*cadena*
defaultValue	opcional		*booleano*
property	opcional		*cadena*

Representación de XML

```
<xs:element name="RawPropensity" minOccurs="0">
  <xs:attribute name="property" type="xs:string" use="optional"/>
  <xs:attribute name="availabilityProperty" type="xs:string" use="optional"/>
  <xs:attribute name="defaultValue" type="xs:boolean" use="optional"/>
</xs:element>
```

Elementos padre

ModelEvaluation

Elemento AdjustedPropensity:

Tabla 141. Atributos de AdjustedPropensity

Atributo	Uso	Descripción	Valores válidos
availabilityProperty	opcional		*cadena*
defaultValue	opcional		*booleano*
property	opcional		*cadena*

Representación de XML

```
<xs:element name="AdjustedPropensity" minOccurs="0">
  <xs:attribute name="property" type="xs:string" use="optional"/>
  <xs:attribute name="availabilityProperty" type="xs:string" use="optional"/>
  <xs:attribute name="defaultValue" type="xs:boolean" use="optional"/>
</xs:element>
```

Elementos padre

ModelEvaluation

Elemento VariableImportance:

Tabla 142. Atributos de VariableImportance

Atributo	Uso	Descripción	Valores válidos
availabilityProperty	opcional		*cadena*
defaultValue	opcional		*booleano*
property	opcional		*cadena*

Representación de XML

```
<xs:element name="VariableImportance" minOccurs="0">
  <xs:attribute name="property" type="xs:string" use="optional"/>
  <xs:attribute name="availabilityProperty" type="xs:string" use="optional"/>
  <xs:attribute name="defaultValue" type="xs:boolean" use="optional"/>
</xs:element>
```

Elementos padre

ModelEvaluation

Elemento AutoModeling:

Tabla 143. Atributos de AutoModeling

Atributo	Uso	Descripción	Valores válidos
enabledByDefault	opcional		*cualquiera*

Representación de XML

```
<xs:element name="AutoModeling" minOccurs="0">
  <xs:sequence>
    <xs:element name="SimpleSettings">
      <xs:sequence>
        <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
    <xs:element name="ExpertSettings" minOccurs="0">
      <xs:sequence>
        <xs:element ref="Condition" minOccurs="0"/>
        <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:element>
    <xs:element name="PropertyMap" minOccurs="0">
      <xs:sequence>
        <xs:element name="PropertyMapping" maxOccurs="unbounded">
        </xs:element>
      </xs:sequence>
    </xs:element>
    <xs:element ref="Constraint" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="enabledByDefault" use="optional"/>
</xs:element>
```

Elementos padre

ModelBuilder

Elementos hijo

Constraint, ExpertSettings, PropertyMap, SimpleSettings

Elemento SimpleSettings:

Representación XML

```
<xs:element name="SimpleSettings">
  <xs:sequence>
    <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

AutoModeling

Elementos hijo

PropertyGroup

Elemento ExpertSettings:

Representación XML

```
<xs:element name="ExpertSettings" minOccurs="0">
  <xs:sequence>
    <xs:element ref="Condition" minOccurs="0"/>
    <xs:element ref="PropertyGroup" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

AutoModeling

Elementos hijo

Condition, PropertyGroup

Elemento PropertyMap:

Representación XML

```
<xs:element name="PropertyMap" minOccurs="0">
  <xs:sequence>
    <xs:element name="PropertyMapping" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

Elementos padre

AutoModeling

Elementos hijo

PropertyMapping

Elemento PropertyMapping:

Tabla 144. Atributos de PropertyMapping

Atributo	Uso	Descripción	Valores válidos
property	**obligatorio**		*cadena*
systemProperty	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="PropertyMapping" maxOccurs="unbounded">
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="systemProperty" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

PropertyMap

Elemento ModelOutput

Tabla 145. Atributos de ModelOutput

Atributo	Uso	Descripción	Valores válidos
deprecatedScriptNames	opcional		*cadena*
helpLink	opcional		*cadena*
id	**obligatorio**		*cadena*
label	opcional		*cadena*
labelKey	opcional		*cadena*
scriptName	opcional		*cadena*

Representación XML

```
<xs:element name="ModelOutput">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
```

```
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="helpLink" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Extension

Elementos hijo

Constructors, Containers, ModelProvider, Properties, UserInterface

Elementos relacionados

DocumentOutput, InteractiveDocumentBuilder, InteractiveModelBuilder, Node

Elemento Containers:

Representación XML

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

Node

Elementos hijo

Container

Elemento ModelProvider

Tabla 146. Atributos de ModelProvider

Atributo	Uso	Descripción	Valores válidos
container	**obligatorio**		*cadena*
isPMML	opcional		*booleano*

Representación XML

```
<xs:element name="ModelProvider">
  <xs:attribute name="container" type="xs:string" use="required"/>
  <xs:attribute name="isPMML" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

Elementos padre

DocumentOutput, InteractiveDocumentBuilder, InteractiveModelBuilder, ModelOutput, Node

Elemento ModelType

Define un nuevo tipo de modelo.

Tabla 147. Atributos de ModelType

Atributo	Uso	Descripción	Valores válidos
format	**obligatorio**		**utf8** **binary**
id	**obligatorio**		*cadena*
inputDirection	opcional		*cadena*
outputDirection	opcional		*cadena*

Representación XML

```
<xs:element name="ModelType">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="format" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="utf8"/>
        <xs:enumeration value="binary"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="inputDirection" type="xs:string" use="optional" default="[in]"/>
  <xs:attribute name="outputDirection" type="xs:string" use="optional" default="[out]"/>
</xs:element>
```

Elementos padre

ContainerTypes

Elementos relacionados

DocumentType

Elemento ModelViewerPanel

Tabla 148. Atributos de ModelViewerPanel

Atributo	Uso	Descripción	Valores válidos
container	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="ModelViewerPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="container" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

Tab

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

ActionButton, ComboBoxControl, ExtensionObjectPanel, SelectorPanel, StaticText, SystemControls, TabbedPanel, TextBrowserPanel

Elemento Module

Tabla 149. Atributos de Module

Atributo	Uso	Descripción	Valores válidos
libraryId	opcional		*cadena*
name	opcional		*cadena*
systemModule	opcional		**SmartScore**

Representación de XML

```
<xs:element name="Module">
  <xs:sequence>
    <xs:element ref="InputFiles"/>
    <xs:element ref="OutputFiles"/>
    <xs:element ref="StatusCodes" minOccurs="0" maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="systemModule" use="optional" default="SmartScore">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="SmartScore"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="libraryId" type="xs:string" use="optional"/>
  <xs:attribute name="name" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Execution

Elementos hijo

InputFiles, OutputFiles, StatusCodes

Elemento MultiFieldChooserControl

Tabla 150. Atributos de MultiFieldChooserControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
onlyDatetime	opcional		*booleano*
onlyDiscrete	opcional		*booleano*
onlyNumeric	opcional		*booleano*

Tabla 150. Atributos de MultiFieldChooserControl (continuación)

Atributo	Uso	Descripción	Valores válidos
onlyRanges	opcional		*booleano*
onlySymbolic	opcional		*booleano*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*
storage	opcional		*cadena*
types	opcional		*cadena*

Representación XML

```
<xs:element name="MultiFieldChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento MultiItemChooserControl

Tabla 151. Atributos de MultiItemChooserControl

Atributo	Uso	Descripción	Valores válidos
catalog	**obligatorio**		*cadena*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*

Tabla 151. Atributos de MultiItemChooserControl (continuación)

Atributo	Uso	Descripción	Valores válidos
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="MultiItemChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="catalog" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

SingleItemChooserControl

Elemento Node

Tabla 152. Atributos de Node

Atributo	Uso	Descripción	Valores válidos
deprecatedScriptNames	opcional		*cadena*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
helpLink	opcional		*cadena*
id	**obligatorio**		*cadena*
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*

Tabla 152. Atributos de Node (continuación)

Atributo	Uso	Descripción	Valores válidos
palette	opcional		**import** **fieldOp** **recordOp** **modeling** **dbModeling** **graph** **output** **export** **modeling.classification** **modeling.association** **modeling.segmentation** **modeling.auto**
relativePosition	opcional		*cadena*
relativeTo	opcional		*cadena*
scriptName	opcional		*cadena*
type	**obligatorio**		**dataReader** **dataWriter** **dataTransformer** **modelApplier** **modelBuilder** **documentBuilder**

Representación de XML

```
<xs:element name="Node">
  <xs:sequence maxOccurs="unbounded">
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Properties"/>
      <xs:element name="Containers" minOccurs="0">
        <xs:sequence maxOccurs="unbounded">
          <xs:element ref="Container"/>
        </xs:sequence>
      </xs:element>
      <xs:element ref="UserInterface"/>
      <xs:element ref="Constructors" minOccurs="0"/>
      <xs:element ref="ModelProvider" minOccurs="0"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
  <xs:sequence>
    <xs:element ref="ModelBuilder" minOccurs="0"/>
    <xs:element ref="DocumentBuilder" minOccurs="0"/>
    <xs:element ref="Execution"/>
    <xs:element ref="OutputDataModel" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="type" type="NODE-TYPE" use="required">
    <xs:enumeration value="dataReader"/>
    <xs:enumeration value="dataWriter"/>
    <xs:enumeration value="dataTransformer"/>
    <xs:enumeration value="modelApplier"/>
    <xs:enumeration value="modelBuilder"/>
    <xs:enumeration value="documentBuilder"/>
  </xs:attribute>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="palette" type="SYSTEM-PALETTE" use="optional">
    <xs:enumeration value="import"/>
    <xs:enumeration value="fieldOp"/>
    <xs:enumeration value="recordOp"/>
    <xs:enumeration value="modeling"/>
    <xs:enumeration value="dbModeling"/>
```

```
    <xs:enumeration value="graph"/>
    <xs:enumeration value="output"/>
    <xs:enumeration value="export"/>
    <xs:enumeration value="modeling.classification"/>
    <xs:enumeration value="modeling.association"/>
    <xs:enumeration value="modeling.segmentation"/>
    <xs:enumeration value="modeling.auto"/>
  </xs:attribute>
  <xs:attribute name="helpLink" type="xs:string" use="optional"/>
  <xs:attribute name="relativeTo" type="xs:string" use="optional"/>
  <xs:attribute name="relativePosition" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Extension

Elementos hijo

Constructors, Containers, DocumentBuilder, Execution, ModelBuilder, ModelProvider, OutputDataModel, Properties, UserInterface

Elementos relacionados

DocumentOutput, InteractiveDocumentBuilder, InteractiveModelBuilder, ModelOutput

Elemento Containers:

Representación XML

```
<xs:element name="Containers" minOccurs="0">
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Container"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

Node

Elementos hijo

Container

Elemento Not

Representación de XML

```
<xs:element name="Not">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

Elementos padre

And, Command, Constraint, CreateDocument, CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModel, CreateModelApplier, CreateModelOutput, Enabled, Not, Option, Or, Run, Visible

Elementos hijo

And, Condition, Not, Or

Elemento NumberFormat

Define la información de formato para un campo numérico.

Tabla 153. Atributos de NumberFormat

Atributo	Uso	Descripción	Valores válidos
decimalPlaces	**obligatorio**		*nonNegativeInteger*
decimalSymbol	**obligatorio**		**period** **comma**
formatType	**obligatorio**		**standard** **scientific** **currency**
groupingSymbol	**obligatorio**		**none** **period** **comma** **space**
name	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="NumberFormat" type="NUMBER-FORMAT-DECLARATION">
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="formatType" type="NUMBER-FORMAT-TYPE" use="required">
    <xs:enumeration value="standard"/>
    <xs:enumeration value="scientific"/>
    <xs:enumeration value="currency"/>
  </xs:attribute>
  <xs:attribute name="decimalPlaces" type="xs:nonNegativeInteger" use="required"/>
  <xs:attribute name="decimalSymbol" type="DECIMAL-SYMBOL" use="required">
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
  </xs:attribute>
  <xs:attribute name="groupingSymbol" type="NUMBER-GROUPING-SYMBOL" use="required">
    <xs:enumeration value="none"/>
    <xs:enumeration value="period"/>
    <xs:enumeration value="comma"/>
    <xs:enumeration value="space"/>
  </xs:attribute>
</xs:element>
```

Elemento NumericInfo

Tabla 154. Atributos de NumericInfo

Atributo	Uso	Descripción	Valores válidos
mean	opcional		*double*
standardDeviation	opcional		*double*

Representación de XML

```
<xs:element name="NumericInfo">
  <xs:attribute name="mean" type="xs:double"/>
  <xs:attribute name="standardDeviation" type="xs:double"/>
</xs:element>
```

Elementos padre

AddField, ChangeField, Field

Elemento Option

Tabla 155. Atributos de Option

Atributo	Uso	Descripción	Valores válidos
ifProperty	opcional		*cadena*
unlessProperty	opcional		*cadena*
value	**obligatorio**		

Representación de XML

```
<xs:element name="Option">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
  <xs:attribute name="value" type="EVALUATED-STRING" use="required"/>
  <xs:attribute name="ifProperty" type="xs:string" use="optional"/>
  <xs:attribute name="unlessProperty" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Run

Elementos hijo

And, Condition, Not, Or

Elemento OptionCode

Tabla 156. Atributos de OptionCode

Atributo	Uso	Descripción	Valores válidos
code	opcional		*long*
description	opcional		*cadena*
type	opcional		**mandatory** **opcional**

Representación de XML

```
<xs:element name="OptionCode">
  <xs:attribute name="code" type="xs:long"/>
  <xs:attribute name="type" type="LicenseType">
    <xs:enumeration value="mandatory"/>
    <xs:enumeration value="optional"/>
  </xs:attribute>
  <xs:attribute name="description" type="xs:string"/>
</xs:element>
```

Elementos padre

License

Elemento Or

Representación de XML

```
<xs:element name="Or">
  <xs:sequence minOccurs="2" maxOccurs="unbounded">
    <xs:group ref="CONDITION-EXPRESSION">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

Elementos padre

And, Command, Constraint, CreateDocument, CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModel, CreateModelApplier, CreateModelOutput, Enabled, Not, Option, Or, Run, Visible

Elementos hijo

And, Condition, Not, Or

Elemento OutputDataModel

Tabla 157. Atributos de OutputDataModel

Atributo	Uso	Descripción	Valores válidos
libraryId	opcional		*cadena*
method	opcional		**xml** **dataModelProvider** **sharedLibrary**
mode	opcional		**fixed** **modify** **extend** **replace**
providerClass	opcional		*cadena*

Representación XML

```
<xs:element name="OutputDataModel">
  <xs:attribute name="mode" use="optional" default="fixed">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="fixed"/>
        <xs:enumeration value="modify"/>
        <xs:enumeration value="extend"/>
        <xs:enumeration value="replace"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="method" use="optional" default="xml">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="xml"/>
        <xs:enumeration value="dataModelProvider"/>
        <xs:enumeration value="sharedLibrary"/>
      </xs:restriction>
    </xs:simpleType>
```

```
  </xs:attribute>
  <xs:attribute name="providerClass" type="xs:string" use="optional"/>
  <xs:attribute name="libraryId" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Node

Elemento OutputFiles

Representación de XML

```
<xs:element name="OutputFiles">
  <xs:group ref="RUNTIME-FILES">
    <xs:sequence>
      <xs:element ref="DataFile"/>
      <xs:element ref="ContainerFile" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
</xs:element>
```

Elementos padre

Execution, Module

Elementos hijo

ContainerFile, DataFile

Elemento Palette

Tabla 158. Atributos de Palette

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
id	**obligatorio**		*cadena*
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*
position	opcional		**atStart** **atEnd** **before** **after**
systemPalette	opcional		**import** **fieldOp** **recordOp** **modeling** **dbModeling** **graph** **output** **export** **modeling.classification** **modeling.association** **modeling.segmentation** **modeling.auto**

Representación de XML

```
<xs:element name="Palette">
  <xs:sequence>
    <xs:element ref="Icon"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="position" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="atStart"/>
        <xs:enumeration value="atEnd"/>
        <xs:enumeration value="before"/>
        <xs:enumeration value="after"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="systemPalette" type="SYSTEM-PALETTE" use="optional">
    <xs:enumeration value="import"/>
    <xs:enumeration value="fieldOp"/>
    <xs:enumeration value="recordOp"/>
    <xs:enumeration value="modeling"/>
    <xs:enumeration value="dbModeling"/>
    <xs:enumeration value="graph"/>
    <xs:enumeration value="output"/>
    <xs:enumeration value="export"/>
    <xs:enumeration value="modeling.classification"/>
    <xs:enumeration value="modeling.association"/>
    <xs:enumeration value="modeling.segmentation"/>
    <xs:enumeration value="modeling.auto"/>
  </xs:attribute>
</xs:element>
```

Elementos hijo

Icon

Elemento Parameters

Parámetros de configuración del nodo de extensión.

Tabla 159. Atributos de Parameters

Atributo	Uso	Descripción	Valores válidos
count	opcional		*nonNegativeInteger*

Representación de XML

```
<xs:element name="Parameters" type="PARAMETERS">
  <xs:sequence>
    <xs:element name="Parameter" type="PARAMETER" minOccurs="0" maxOccurs="unbounded">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="count" type="xs:nonNegativeInteger"/>
</xs:element>
```

Elementos hijo

Parameter

Elemento Parameter: Un parámetro tiene un nombre y un valor. Un valor simple puede expresarse con el atributo value; un valor compuesto utiliza el modelo de contenido descrito por ParameterContent. Esta combinación de atributo y contenido se repite para los valores anidados.

Tabla 160. Atributos de Parameter

Atributo	Uso	Descripción	Valores válidos
name	**obligatorio**		*cadena*
value	opcional		*cadena*

Representación XML

```
<xs:element name="Parameter" type="PARAMETER" minOccurs="0" maxOccurs="unbounded">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string"/>
</xs:element>
```

Elementos padre

Parameters

Elementos hijo

DatabaseConnectionValue, ListValue, MapValue, StructuredValue, Value

Elemento PasswordBoxControl

Tabla 161. Atributos de PasswordBoxControl

Atributo	Uso	Descripción	Valores válidos
columns	opcional		*positiveInteger*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="PasswordBoxControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
```

```
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento Properties

Representación de XML

```
<xs:element name="Properties">
  <xs:sequence>
    <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

DocumentOutput, Execution, InteractiveDocumentBuilder, InteractiveModelBuilder, ModelOutput, Node

Elementos hijo

Property

Elemento PropertiesPanel

Tabla 162. Atributos de PropertiesPanel

Atributo	Uso	Descripción	Valores válidos
id	opcional		*cadena*
label	opcional		*cadena*
labelKey	opcional		*cadena*

Representación de XML

```
<xs:element name="PropertiesPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
```

```
    </xs:choice>
  </xs:sequence>
  <xs:sequence maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="CheckBoxControl"/>
      <xs:element ref="TextBoxControl"/>
      <xs:element ref="PasswordBoxControl"/>
      <xs:element ref="TextAreaControl"/>
      <xs:element ref="RadioButtonGroupControl"/>
      <xs:element ref="CheckBoxGroupControl"/>
      <xs:element ref="ComboBoxControl"/>
      <xs:element ref="SpinnerControl"/>
      <xs:element ref="ServerFileChooserControl"/>
      <xs:element ref="ServerDirectoryChooserControl"/>
      <xs:element ref="ClientFileChooserControl"/>
      <xs:element ref="ClientDirectoryChooserControl"/>
      <xs:element ref="TableControl"/>
      <xs:element ref="SingleFieldChooserControl"/>
      <xs:element ref="MultiFieldChooserControl"/>
      <xs:element ref="SingleFieldValueChooserControl"/>
      <xs:element ref="SingleItemChooserControl"/>
      <xs:element ref="MultiItemChooserControl"/>
      <xs:element ref="DBConnectionChooserControl"/>
      <xs:element ref="DBTableChooserControl"/>
      <xs:element ref="PropertyControl"/>
      <xs:element ref="StaticText"/>
      <xs:element ref="SystemControls"/>
      <xs:element ref="ActionButton"/>
      <xs:element ref="PropertiesPanel"/>
      <xs:element ref="PropertiesSubPanel"/>
      <xs:element ref="SelectorPanel"/>
      <xs:element ref="ExtensionObjectPanel"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel, Tab

Elementos hijo

ActionButton, CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, ComboBoxControl, DBConnectionChooserControl, DBTableChooserControl, Enabled, ExtensionObjectPanel, Layout, MultiFieldChooserControl, MultiItemChooserControl, PasswordBoxControl, PropertiesPanel, PropertiesSubPanel, PropertyControl, RadioButtonGroupControl, SelectorPanel, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SingleItemChooserControl, SpinnerControl, StaticText, SystemControls, TableControl, TextAreaControl, TextBoxControl, Visible

Elementos relacionados

PropertiesSubPanel

Elemento PropertiesSubPanel

Tabla 163. Atributos de PropertiesSubPanel

Atributo	Uso	Descripción	Valores válidos
buttonLabel	opcional		*cadena*
buttonLabelKey	opcional		*cadena*
dialogTitle	opcional		*cadena*
dialogTitleKey	opcional		*cadena*
helpLink	opcional		*cadena*

Tabla 163. Atributos de PropertiesSubPanel (continuación)

Atributo	Uso	Descripción	Valores válidos
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*

Representación XML

```
<xs:element name="PropertiesSubPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:sequence maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="CheckBoxControl"/>
      <xs:element ref="TextBoxControl"/>
      <xs:element ref="PasswordBoxControl"/>
      <xs:element ref="TextAreaControl"/>
      <xs:element ref="RadioButtonGroupControl"/>
      <xs:element ref="CheckBoxGroupControl"/>
      <xs:element ref="ComboBoxControl"/>
      <xs:element ref="SpinnerControl"/>
      <xs:element ref="ServerFileChooserControl"/>
      <xs:element ref="ServerDirectoryChooserControl"/>
      <xs:element ref="ClientFileChooserControl"/>
      <xs:element ref="ClientDirectoryChooserControl"/>
      <xs:element ref="TableControl"/>
      <xs:element ref="SingleFieldChooserControl"/>
      <xs:element ref="MultiFieldChooserControl"/>
      <xs:element ref="SingleFieldValueChooserControl"/>
      <xs:element ref="SingleItemChooserControl"/>
      <xs:element ref="MultiItemChooserControl"/>
      <xs:element ref="DBConnectionChooserControl"/>
      <xs:element ref="DBTableChooserControl"/>
      <xs:element ref="PropertyControl"/>
      <xs:element ref="StaticText"/>
      <xs:element ref="SystemControls"/>
      <xs:element ref="ActionButton"/>
      <xs:element ref="PropertiesPanel"/>
      <xs:element ref="PropertiesSubPanel"/>
      <xs:element ref="SelectorPanel"/>
      <xs:element ref="ExtensionObjectPanel"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="buttonLabel" type="xs:string" use="optional"/>
  <xs:attribute name="buttonLabelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="dialogTitle" type="xs:string" use="optional"/>
  <xs:attribute name="dialogTitleKey" type="xs:string" use="optional"/>
  <xs:attribute name="helpLink" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

ActionButton, CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, ComboBoxControl, DBConnectionChooserControl, DBTableChooserControl, Enabled, ExtensionObjectPanel, Layout, MultiFieldChooserControl, MultiItemChooserControl, PasswordBoxControl, PropertiesPanel, PropertiesSubPanel, PropertyControl, RadioButtonGroupControl, SelectorPanel, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SingleItemChooserControl, SpinnerControl, StaticText, SystemControls, TableControl, TextAreaControl, TextBoxControl, Visible

Elementos relacionados

PropertiesPanel

Elemento Property

Tabla 164. Atributos de Property

Atributo	Uso	Descripción	Valores válidos
defaultValue	opcional		
defaultValueKey	opcional		*cadena*
deprecatedScriptNames	opcional		*cadena*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
isList	opcional		*booleano*
label	opcional		*cadena*
labelKey	opcional		*cadena*
max	opcional		*cadena*
min	opcional		*cadena*
name	**obligatorio**		*cadena*
scriptName	opcional		*cadena*
type	opcional		*cadena*
valueType	opcional		**cadena** **encryptedString** **fieldName** **integer** **double** **boolean** **date** **enum** **structure** **databaseConnection**

Representación de XML

```
<xs:element name="Property">
  <xs:choice>
    <xs:element ref="DefaultValue" minOccurs="0"/>
  </xs:choice>
  <xs:attribute name="valueType" type="PROPERTY-VALUE-TYPE">
    <xs:enumeration value="string"/>
    <xs:enumeration value="encryptedString"/>
    <xs:enumeration value="fieldName"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="boolean"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="enum"/>
    <xs:enumeration value="structure"/>
    <xs:enumeration value="databaseConnection"/>
  </xs:attribute>
  <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="min" type="xs:string" use="optional"/>
  <xs:attribute name="max" type="xs:string" use="optional"/>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="scriptName" type="xs:string" use="optional"/>
  <xs:attribute name="deprecatedScriptNames" type="xs:string" use="optional"/>
  <xs:attribute name="type" type="xs:string" use="optional"/>
  <xs:attribute name="defaultValue" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="defaultValueKey" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
```

```
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Properties, PropertySets

Elementos hijo

DefaultValue

Elementos relacionados

PropertyType

Elemento PropertyControl

Tabla 165. Atributos de PropertyControl

Atributo	Uso	Descripción	Valores válidos
controlClass	**obligatorio**		*cadena*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="PropertyControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="controlClass" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento PropertyGroup

Tabla 166. Atributos de PropertyGroup

Atributo	Uso	Descripción	Valores válidos
label	opcional		*cadena*
labelKey	opcional		*cadena*
propiedades	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="PropertyGroup">
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="properties" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

ExpertSettings, SimpleSettings

Elemento PropertySets

Representación XML

```
<xs:element name="PropertySets">
  <xs:sequence>
    <xs:element ref="Property" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

CommonObjects

Elementos hijo

Property

Elemento PropertyType

Tabla 167. Atributos de PropertyType

Atributo	Uso	Descripción	Valores válidos
id	**obligatorio**		*cadena*
isKeyed	opcional		*booleano*
isList	opcional		*booleano*
max	opcional		*cadena*

Tabla 167. Atributos de PropertyType (continuación)

Atributo	Uso	Descripción	Valores válidos
min	opcional		*cadena*
valueType	opcional		**cadena** **encryptedString** **fieldName** **integer** **double** **boolean** **date** **enum** **structure** **databaseConnection**

Representación XML

```
<xs:element name="PropertyType">
  <xs:choice>
    <xs:element ref="DefaultValue" minOccurs="0"/>
  </xs:choice>
  <xs:attribute name="valueType" type="PROPERTY-VALUE-TYPE">
    <xs:enumeration value="string"/>
    <xs:enumeration value="encryptedString"/>
    <xs:enumeration value="fieldName"/>
    <xs:enumeration value="integer"/>
    <xs:enumeration value="double"/>
    <xs:enumeration value="boolean"/>
    <xs:enumeration value="date"/>
    <xs:enumeration value="enum"/>
    <xs:enumeration value="structure"/>
    <xs:enumeration value="databaseConnection"/>
  </xs:attribute>
  <xs:attribute name="isList" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="min" type="xs:string" use="optional"/>
  <xs:attribute name="max" type="xs:string" use="optional"/>
  <xs:choice>
    <xs:element ref="Enumeration" minOccurs="0"/>
    <xs:element ref="Structure" minOccurs="0"/>
  </xs:choice>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="isKeyed" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

Elementos padre

PropertyTypes

Elementos hijo

DefaultValue, Enumeration, Structure

Elementos relacionados

Property

Elemento PropertyTypes

Representación XML

```
<xs:element name="PropertyTypes">
  <xs:sequence>
    <xs:element ref="PropertyType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

CommonObjects

Elementos hijo

PropertyType

Elemento RadioButtonGroupControl

Tabla 168. Atributos de RadioButtonGroupControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
falseLabel	opcional		*cadena*
falseLabelKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
layoutByRow	opcional		*booleano*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
rows	opcional		*positiveInteger*
showLabel	opcional		*booleano*
trueFirst	opcional		*booleano*
trueLabel	opcional		*cadena*
trueLabelKey	opcional		*cadena*
useSubPanel	opcional		*booleano*

Representación XML

```
<xs:element name="RadioButtonGroupControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="layoutByRow" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="useSubPanel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="falseLabel" type="xs:string" use="optional"/>
  <xs:attribute name="falseLabelKey" type="xs:string" use="optional"/>
```

```
  <xs:attribute name="trueLabel" type="xs:string" use="optional"/>
  <xs:attribute name="trueLabelKey" type="xs:string" use="optional"/>
  <xs:attribute name="trueFirst" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento Range

Tabla 169. Atributos de Range

Atributo	Uso	Descripción	Valores válidos
max	opcional		*cadena*
min	opcional		*cadena*

Representación XML

```
<xs:element name="Range">
  <xs:attribute name="min" type="xs:string"/>
  <xs:attribute name="max" type="xs:string"/>
</xs:element>
```

Elementos padre

AddField, ChangeField, Field, Field, MissingValues, MissingValues, MissingValues

Elemento Range

Tabla 170. Atributos de Range

Atributo	Uso	Descripción	Valores válidos
maxValue	**obligatorio**		*cadena*
minValue	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="Range" type="RANGE">
  <xs:attribute name="minValue" type="xs:string" use="required"/>
  <xs:attribute name="maxValue" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

AddField, ChangeField, Field, Field, MissingValues, MissingValues, MissingValues

Elemento RemoveField

Tabla 171. Atributos de RemoveField

Atributo	Uso	Descripción	Valores válidos
fieldRef	**obligatorio**		

Representación XML

```
<xs:element name="RemoveField">
  <xs:attribute name="fieldRef" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

Elementos padre

ForEach, ModelFields

Elemento Resources

Define recursos comunes tales como bibliotecas y paquetes de recursos del lado del cliente y bibliotecas del lado del servidor.

Representación de XML

```
<xs:element name="Resources">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element name="Bundle" minOccurs="0">
      </xs:element>
      <xs:element name="JarFile" minOccurs="0">
      </xs:element>
      <xs:element name="SharedLibrary" minOccurs="0">
      </xs:element>
      <xs:element name="HelpInfo" minOccurs="0">
      </xs:element>
    </xs:choice>
  </xs:sequence>
</xs:element>
```

Elementos padre

Extension

Elementos hijo

Bundle, HelpInfo, JarFile, SharedLibrary

Elemento Bundle:

Tabla 172. Atributos de Bundle

Atributo	Uso	Descripción	Valores válidos
id	**obligatorio**		*cadena*
path	**obligatorio**		
type	**obligatorio**		**list** **properties**

Representación XML

```
<xs:element name="Bundle" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="list"/>
```

```
      <xs:enumeration value="properties"/>
    </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

Elementos padre

Resources

Elemento JarFile:

Tabla 173. Atributos de JarFile

Atributo	Uso	Descripción	Valores válidos
id	**obligatorio**		*cadena*
path	**obligatorio**		

Representación de XML

```
<xs:element name="JarFile" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

Elementos padre

Resources

Elemento SharedLibrary:

Tabla 174. Atributos de SharedLibrary

Atributo	Uso	Descripción	Valores válidos
id	**obligatorio**		*cadena*
path	**obligatorio**		

Representación de XML

```
<xs:element name="SharedLibrary" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="path" type="EVALUATED-STRING" use="required"/>
</xs:element>
```

Elementos padre

Resources

Elemento de HelpInfo:

Tabla 175. Atributos de HelpInfo

Atributo	Uso	Descripción	Valores válidos
default	opcional		*cadena*
helpset	opcional		
id	opcional		*cadena*
missing	opcional		*cadena*
path	opcional		

Tabla 175. Atributos de HelpInfo (continuación)

Atributo	Uso	Descripción	Valores válidos
type	**obligatorio**		**native** **javahelp** **html**

Representación XML

```
<xs:element name="HelpInfo" minOccurs="0">
  <xs:attribute name="id" type="xs:string" use="optional"/>
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="native"/>
        <xs:enumeration value="javahelp"/>
        <xs:enumeration value="html"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="path" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="helpset" type="EVALUATED-STRING" use="optional"/>
  <xs:attribute name="default" type="xs:string" use="optional"/>
  <xs:attribute name="missing" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Resources

Elemento Run

Representación de XML

```
<xs:element name="Run">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
    <xs:element ref="Command" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Option" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="StatusCodes" minOccurs="0"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

Executable

Elementos hijo

And, Command, Condition, Not, Option, Or, StatusCodes

Elemento SelectorPanel

Tabla 176. Atributos de SelectorPanel

Atributo	Uso	Descripción	Valores válidos
control	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="SelectorPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Selector">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="control" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Selector, Visible

Elementos relacionados

ActionButton, ComboBoxControl, ExtensionObjectPanel, ModelViewerPanel, StaticText, SystemControls, TabbedPanel, TextBrowserPanel

Elemento Selector:

Tabla 177. Atributos de Selector

Atributo	Uso	Descripción	Valores válidos
controlValue	**obligatorio**		*cadena*
panelId	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="Selector">
  <xs:attribute name="panelId" type="xs:string" use="required"/>
  <xs:attribute name="controlValue" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

SelectorPanel

Elemento ServerDirectoryChooserControl

Tabla 178. Atributos de ServerDirectoryChooserControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*

Tabla 178. Atributos de ServerDirectoryChooserControl (continuación)

Atributo	Uso	Descripción	Valores válidos
mnemonicKey	opcional		*cadena*
mode	**obligatorio**		**open** **save** **import** **export**
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="ServerDirectoryChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"/>
    <xs:enumeration value="save"/>
    <xs:enumeration value="import"/>
    <xs:enumeration value="export"/>
  </xs:attribute>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento ServerFileChooserControl

Tabla 179. Atributos de ServerFileChooserControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*

Tabla 179. Atributos de ServerFileChooserControl (continuación)

Atributo	Uso	Descripción	Valores válidos
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
mode	**obligatorio**		**open** **save** **import** **export**
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="ServerFileChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="mode" type="FILE-CHOOSER-MODE" use="required">
    <xs:enumeration value="open"/>
    <xs:enumeration value="save"/>
    <xs:enumeration value="import"/>
    <xs:enumeration value="export"/>
  </xs:attribute>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento SetContainer

Tabla 180. Atributos de SetContainer

Atributo	Uso	Descripción	Valores válidos
source	**obligatorio**		*cadena*
target	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="SetContainer">
  <xs:attribute name="source" type="xs:string" use="required"/>
  <xs:attribute name="target" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModelApplier, CreateModelOutput

Elementos relacionados

SetProperty

Elemento SetProperty

Tabla 181. Atributos de SetProperty

Atributo	Uso	Descripción	Valores válidos
source	**obligatorio**		*cadena*
target	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="SetProperty">
  <xs:attribute name="source" type="xs:string" use="required"/>
  <xs:attribute name="target" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

CreateDocumentOutput, CreateInteractiveDocumentBuilder, CreateInteractiveModelBuilder, CreateModelApplier, CreateModelOutput

Elementos relacionados

SetContainer

Elemento SingleFieldChooserControl

Tabla 182. Atributos de SingleFieldChooserControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*

Tabla 182. Atributos de SingleFieldChooserControl (continuación)

Atributo	Uso	Descripción	Valores válidos
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
onlyDatetime	opcional		*booleano*
onlyDiscrete	opcional		*booleano*
onlyNumeric	opcional		*booleano*
onlyRanges	opcional		*booleano*
onlySymbolic	opcional		*booleano*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*
storage	opcional		*cadena*
types	opcional		*cadena*

Representación XML

```
<xs:element name="SingleFieldChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="storage" type="xs:string" use="optional"/>
  <xs:attribute name="onlyNumeric" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlySymbolic" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDatetime" type="xs:boolean" use="optional"/>
  <xs:attribute name="types" type="xs:string" use="optional"/>
  <xs:attribute name="onlyRanges" type="xs:boolean" use="optional"/>
  <xs:attribute name="onlyDiscrete" type="xs:boolean" use="optional"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento SingleFieldValueChooserControl

Tabla 183. Atributos de SingleFieldValueChooserControl

Atributo	Uso	Descripción	Valores válidos
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
fieldControl	opcional		*cadena*
fieldDirection	opcional		**in** **out** **both** **none** **partition**
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="SingleFieldValueChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="fieldControl" type="xs:string" use="optional"/>
  <xs:attribute name="fieldDirection" type="FIELD-DIRECTION" use="optional">
    <xs:enumeration value="in"/>
    <xs:enumeration value="out"/>
    <xs:enumeration value="both"/>
    <xs:enumeration value="none"/>
    <xs:enumeration value="partition"/>
  </xs:attribute>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SpinnerControl, TableControl, TextAreaControl, TextBoxControl

Elemento SingleItemChooserControl

Tabla 184. Atributos de SingleItemChooserControl

Atributo	Uso	Descripción	Valores válidos
catalog	**obligatorio**		*cadena*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="SingleItemChooserControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="catalog" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

MultiItemChooserControl

Elemento SpinnerControl

Tabla 185. Atributos de SpinnerControl

Atributo	Uso	Descripción	Valores válidos
columns	opcional		*positiveInteger*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
maxDecimalDigits	opcional		*positiveInteger*
minDecimalDigits	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*
stepSize	opcional		*decimal*

Representación XML

```
<xs:element name="SpinnerControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="5"/>
  <xs:attribute name="stepSize" type="xs:decimal" use="optional" default="1.0"/>
  <xs:attribute name="minDecimalDigits" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="maxDecimalDigits" type="xs:positiveInteger" use="optional"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl,

SingleFieldChooserControl, SingleFieldValueChooserControl, TableControl, TextAreaControl, TextBoxControl

Elemento SPSSDataFormat

Representación XML

```
<xs:element name="SPSSDataFormat"/>
```

Elementos padre

DataFormat

Elemento StaticText

Tabla 186. Atributos de StaticText

Atributo	Uso	Descripción	Valores válidos
text	opcional		*cadena*
textKey	opcional		*cadena*

Representación XML

```
<xs:element name="StaticText">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="text" type="xs:string" use="optional"/>
  <xs:attribute name="textKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

ActionButton, ComboBoxControl, ExtensionObjectPanel, ModelViewerPanel, SelectorPanel, SystemControls, TabbedPanel, TextBrowserPanel

Elemento StatusCode

Tabla 187. Atributos de StatusCode

Atributo	Uso	Descripción	Valores válidos
code	**obligatorio**		*entero*
message	opcional		*cadena*
messageKey	opcional		*cadena*
status	opcional		**success** **warning** **error**

Representación XML

```
<xs:element name="StatusCode">
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="status" use="optional" default="success">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="success"/>
        <xs:enumeration value="warning"/>
        <xs:enumeration value="error"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="message" type="xs:string" use="optional"/>
  <xs:attribute name="messageKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

StatusCodes

Elemento StatusCode

Tabla 188. Atributos de StatusCodes

Atributo	Uso	Descripción	Valores válidos
defaultMessage	opcional		*cadena*
defaultMessageKey	opcional		*cadena*

Representación XML

```
<xs:element name="StatusCodes">
  <xs:sequence>
    <xs:element ref="StatusCode" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="defaultMessage" type="xs:string" use="optional"/>
  <xs:attribute name="defaultMessageKey" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Module, Run

Elementos hijo

StatusCode

Elemento StatusDetail

Información complementaria acerca de un progreso u otra condición.

Tabla 189. Atributos de StatusDetail

Atributo	Uso	Descripción	Valores válidos
destination	opcional		**client** **tracefile** **console**

Representación de XML

```
<xs:element name="StatusDetail" type="STATUS-DETAIL">
  <xs:sequence>
    <xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
      <xs:sequence>
        <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
        </xs:element>
        <xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

```
      </xs:sequence>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="destination" type="STATUS-DESTINATION" default="client">
    <xs:enumeration value="client"/>
    <xs:enumeration value="tracefile"/>
    <xs:enumeration value="console"/>
  </xs:attribute>
</xs:element>
```

Elementos hijo

Diagnostic

Elemento Diagnostic:

Tabla 190. Atributos de Diagnostic

Atributo	Uso	Descripción	Valores válidos
code	**obligatorio**		*entero*
severity	opcional		**unknown** **information** **warning** **error** **fatal**
source	opcional		*cadena*
subCode	opcional		*entero*

Representación XML

```
<xs:element name="Diagnostic" type="DIAGNOSTIC" minOccurs="0" maxOccurs="unbounded">
  <xs:sequence>
    <xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
    </xs:element>
    <xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="subCode" type="xs:integer" default="0"/>
  <xs:attribute name="severity" type="DIAGNOSTIC-SEVERITY" default="error">
    <xs:enumeration value="unknown"/>
    <xs:enumeration value="information"/>
    <xs:enumeration value="warning"/>
    <xs:enumeration value="error"/>
    <xs:enumeration value="fatal"/>
  </xs:attribute>
  <xs:attribute name="source" type="xs:string"/>
</xs:element>
```

Elementos padre

StatusDetail

Elementos hijo

Message, Parameter

Elemento Message:

Tabla 191. Atributos de Message

Atributo	Uso	Descripción	Valores válidos
lang	opcional		*NMTOKEN*

Representación de XML

```
<xs:element name="Message" type="DIAGNOSTIC-MESSAGE" minOccurs="0">
  <xs:attribute name="lang" type="xs:NMTOKEN"/>
</xs:element>
```

Elementos padre

Diagnostic

Elemento Parameter:

Representación de XML

```
<xs:element name="Parameter" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
```

Elementos padre

Diagnostic

Elemento Structure

Representación XML

```
<xs:element name="Structure">
  <xs:sequence>
    <xs:element ref="Attribute" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:element>
```

Elementos padre

PropertyType

Elementos hijo

Attribute

Elemento StructuredValue

Una secuencia de valores definidos ("atributos").

Representación de XML

```
<xs:element name="StructuredValue" type="STRUCTURED-VALUE">
  <xs:sequence>
    <xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
      <xs:sequence>
        <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
          <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
            <xs:choice>
              <xs:element ref="MapValue"/>
              <xs:element ref="StructuredValue"/>
              <xs:element ref="ListValue"/>
              <xs:element ref="Value"/>
              <xs:element ref="DatabaseConnectionValue"/>
            </xs:choice>
          </xs:group>
        </xs:element>
      </xs:sequence>
    </xs:element>
  </xs:sequence>
</xs:element>
```

Elementos padre

Attribute, Attribute, ListValue, ListValue, ListValue, Parameter

Elementos hijo

Atributo

Elemento Attribute:

Tabla 192. Atributos de Attribute

Atributo	Uso	Descripción	Valores válidos
name	**obligatorio**		*cadena*
value	opcional		*cadena*

Representación de XML

```
<xs:element name="Attribute" type="ATTRIBUTE" maxOccurs="unbounded">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
  <xs:sequence>
    <xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
      <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
        <xs:choice>
          <xs:element ref="MapValue"/>
          <xs:element ref="StructuredValue"/>
          <xs:element ref="ListValue"/>
          <xs:element ref="Value"/>
          <xs:element ref="DatabaseConnectionValue"/>
        </xs:choice>
      </xs:group>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required"/>
  <xs:attribute name="value" type="xs:string"/>
</xs:element>
```

Elementos padre

StructuredValue

Elementos hijo

DatabaseConnectionValue, ListValue, ListValue, MapValue, StructuredValue, Value

Elemento ListValue: Una secuencia de valores. Todos los valores deben tener el mismo tipo de contenido, pero esto no se comprueba.

Representación de XML

```
<xs:element name="ListValue" type="LIST-VALUE" minOccurs="0" maxOccurs="1">
  <xs:group ref="PARAMETER-CONTENT" minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="MapValue"/>
      <xs:element ref="StructuredValue"/>
      <xs:element ref="ListValue"/>
      <xs:element ref="Value"/>
      <xs:element ref="DatabaseConnectionValue"/>
    </xs:choice>
  </xs:group>
</xs:element>
```

Elementos padre

Attribute

Elementos hijo

DatabaseConnectionValue, ListValue, MapValue, StructuredValue, Value

Elemento SystemControls

Tabla 193. Atributos de SystemControls

Atributo	Uso	Descripción	Valores válidos
controlsId	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="SystemControls">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="controlsId" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

ActionButton, ComboBoxControl, ExtensionObjectPanel, ModelViewerPanel, SelectorPanel, StaticText, TabbedPanel, TextBrowserPanel

Elemento Tab

Tabla 194. Atributos de Tab

Atributo	Uso	Descripción	Valores válidos
helpLink	opcional		*cadena*
id	opcional		*cadena*
label	**obligatorio**		*cadena*
labelKey	opcional		*cadena*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*

Representación XML

```
<xs:element name="Tab">
  <xs:sequence minOccurs="0" maxOccurs="unbounded">
    <xs:choice>
      <xs:element ref="PropertiesPanel"/>
      <xs:element ref="ExtensionObjectPanel"/>
      <xs:element ref="TextBrowserPanel"/>
```

```
      <xs:element ref="ModelViewerPanel"/>
      <xs:element ref="TabbedPanel"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="optional"/>
  <xs:attribute name="label" type="xs:string" use="required"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="helpLink" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

Tabs

Elementos hijo

ExtensionObjectPanel, ModelViewerPanel, PropertiesPanel, TabbedPanel, TextBrowserPanel

Elemento TabbedPanel

Tabla 195. Atributos de TabbedPanel

Atributo	Uso	Descripción	Valores válidos
style	opcional		**standard** **sidebar**

Representación de XML

```
<xs:element name="TabbedPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:sequence maxOccurs="unbounded">
    <xs:element ref="Tabs"/>
  </xs:sequence>
  <xs:attribute name="style" use="optional">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="standard"/>
        <xs:enumeration value="sidebar"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:element>
```

Elementos padre

Tab

Elementos hijo

Enabled, Layout, Tabs, Visible

Elementos relacionados

ActionButton, ComboBoxControl, ExtensionObjectPanel, ModelViewerPanel, SelectorPanel, StaticText, SystemControls, TextBrowserPanel

Elemento TableControl

Tabla 196. Atributos de TableControl

Atributo	Uso	Descripción	Valores válidos
columns	opcional		*positiveInteger*
columnWidths	opcional		*cadena*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
rows	opcional		*positiveInteger*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="TableControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="8"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/>
  <xs:attribute name="columnWidths" type="xs:string" use="optional"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TextAreaControl, TextBoxControl

Elemento Tabs

Tabla 197. Atributos de Tabs

Atributo	Uso	Descripción	Valores válidos
defaultTab	opcional		*nonNegativeInteger*

Representación XML

```
<xs:element name="Tabs">
  <xs:sequence>
    <xs:element ref="Tab" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="defaultTab" type="xs:nonNegativeInteger" use="optional" default="0"/>
</xs:element>
```

Elementos padre

TabbedPanel, UserInterface

Elementos hijo

Tab

Elemento TextAreaControl

Tabla 198. Atributos de TextAreaControl

Atributo	Uso	Descripción	Valores válidos
columns	opcional		*positiveInteger*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
rows	opcional		*positiveInteger*
showLabel	opcional		*booleano*
wrapLines	opcional		*booleano*

Representación XML

```
<xs:element name="TextAreaControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
```

```
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="rows" type="xs:positiveInteger" use="optional" default="8"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/>
  <xs:attribute name="wrapLines" type="xs:boolean" use="optional" default="true"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextBoxControl

Elemento TextBoxControl

Tabla 199. Atributos de TextBoxControl

Atributo	Uso	Descripción	Valores válidos
columns	opcional		*positiveInteger*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación XML

```
<xs:element name="TextBoxControl">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="property" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="label" type="xs:string" use="optional"/>
  <xs:attribute name="labelKey" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonic" type="xs:string" use="optional"/>
  <xs:attribute name="mnemonicKey" type="xs:string" use="optional"/>
  <xs:attribute name="labelWidth" type="xs:positiveInteger" use="optional" default="1"/>
  <xs:attribute name="labelAbove" type="xs:boolean" use="optional" default="false"/>
```

```
  <xs:attribute name="description" type="xs:string" use="optional"/>
  <xs:attribute name="descriptionKey" type="xs:string" use="optional"/>
  <xs:attribute name="columns" type="xs:positiveInteger" use="optional" default="20"/>
</xs:element>
```

Elementos padre

PropertiesPanel, PropertiesSubPanel

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, DBConnectionChooserControl, DBTableChooserControl, MultiFieldChooserControl, PasswordBoxControl, PropertyControl, RadioButtonGroupControl, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SpinnerControl, TableControl, TextAreaControl

Elemento TextBrowserPanel

Tabla 200. Atributos de TextBrowserPanel

Atributo	Uso	Descripción	Valores válidos
columns	opcional		*cadena*
container	**obligatorio**		*cadena*
rows	opcional		*cadena*
textFormat	**obligatorio**		**plainText** **html** **rtf**
wrapLines	opcional		*booleano*

Representación XML

```
<xs:element name="TextBrowserPanel">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="container" type="xs:string" use="required"/>
  <xs:attribute name="textFormat" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="plainText"/>
        <xs:enumeration value="html"/>
        <xs:enumeration value="rtf"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="rows" type="xs:string" use="optional"/>
  <xs:attribute name="columns" type="xs:string" use="optional"/>
  <xs:attribute name="wrapLines" type="xs:boolean" use="optional" default="false"/>
</xs:element>
```

Elementos padre

Tab

Elementos hijo

Enabled, Layout, Visible

Elementos relacionados

ActionButton, ComboBoxControl, ExtensionObjectPanel, ModelViewerPanel, SelectorPanel, StaticText, SystemControls, TabbedPanel

Elemento ToolbarItem

Tabla 201. Atributos de ToolbarItem

Atributo	Uso	Descripción	Valores válidos
action	**obligatorio**		*cadena*
offset	opcional		*nonNegativeInteger*
separatorAfter	opcional		*booleano*
separatorBefore	opcional		*booleano*
showIcon	opcional		*booleano*
showLabel	opcional		*booleano*

Representación de XML

```
<xs:element name="ToolbarItem">
  <xs:attribute name="action" type="xs:string" use="required"/>
  <xs:attribute name="showLabel" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="showIcon" type="xs:boolean" use="optional" default="true"/>
  <xs:attribute name="separatorBefore" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="separatorAfter" type="xs:boolean" use="optional" default="false"/>
  <xs:attribute name="offset" type="xs:nonNegativeInteger" use="optional" default="0"/>
</xs:element>
```

Elementos padre

Controls

Elemento UserInterface

Tabla 202. Atributos de UserInterface

Atributo	Uso	Descripción	Valores válidos
actionHandler	opcional		*cualquiera*
frameClass	opcional		*cualquiera*

Representación de XML

```
<xs:element name="UserInterface">
  <xs:sequence>
    <xs:element ref="Icons" minOccurs="0"/>
    <xs:element ref="Controls" minOccurs="0"/>
    <xs:element ref="Tabs" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="frameClass" use="optional"/>
  <xs:attribute name="actionHandler" use="optional"/>
</xs:element>
```

Elementos padre

DocumentOutput, Extension, InteractiveDocumentBuilder, InteractiveModelBuilder, ModelOutput, Node

Elementos hijo

Controls, Icons, Tabs

Elemento UTF8Format

Representación de XML

```
<xs:element name="UTF8Format"/>
```

Elementos padre

FileFormatType

Elemento Value

Un valor simple.

Tabla 203. Atributos de Value

Atributo	Uso	Descripción	Valores válidos
value	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="Value" type="SIMPLE-VALUE">
  <xs:attribute name="value" type="xs:string" use="required"/>
</xs:element>
```

Elementos padre

Attribute, Attribute, ListValue, ListValue, ListValue, Parameter

Elemento Values

Representación de XML

```
<xs:element name="Values">
  <xs:sequence>
    <xs:element name="Value" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
</xs:element>
```

Elementos padre

AddField, ChangeField, Field, Field, MissingValues, MissingValues, MissingValues

Elementos hijo

Value

Elemento Value:

Tabla 204. Atributos de Value

Atributo	Uso	Descripción	Valores válidos
flagProperty	opcional		**trueValue** **falseValue**
value	**obligatorio**		*cadena*
valueLabel	opcional		*cadena*

Representación XML

```
<xs:element name="Value" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="valueLabel" type="xs:string" use="optional"/>
  <xs:attribute name="flagProperty">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="trueValue"/>
        <xs:enumeration value="falseValue"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
</xs:element>
```

Elementos padre

Values

Elemento Values

Tabla 205. Atributos de Values

Atributo	Uso	Descripción	Valores válidos
code	**obligatorio**		*entero*
displayLabel	opcional		*cadena*
flagValue	opcional		*booleano*
value	**obligatorio**		*cadena*

Representación XML

```
<xs:element name="Values" type="FIELD-VALUE">
  <xs:sequence>
    <xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
    </xs:element>
  </xs:sequence>
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="code" type="xs:integer" use="required"/>
  <xs:attribute name="flagValue" type="xs:boolean"/>
  <xs:attribute name="displayLabel" type="xs:string"/>
</xs:element>
```

Elementos padre

AddField, ChangeField, Field, Field, MissingValues, MissingValues, MissingValues

Elementos hijo

DisplayLabel

Elemento DisplayLabel: Una etiqueta de visualización para un campo o valor en un idioma especificado. El atributo displayLabel puede utilizarse cuando no hay ninguna etiqueta para un idioma determinado.

Tabla 206. Atributos de DisplayLabel

Atributo	Uso	Descripción	Valores válidos
lang	opcional		*NMTOKEN*
value	**obligatorio**		*cadena*

Representación de XML

```
<xs:element name="DisplayLabel" type="DISPLAY-LABEL" minOccurs="0" maxOccurs="unbounded">
  <xs:attribute name="value" type="xs:string" use="required"/>
  <xs:attribute name="lang" type="xs:NMTOKEN" default="en"/>
</xs:element>
```

Elementos padre

Values

Elemento Visible

Representación de XML

```
<xs:element name="Visible">
  <xs:sequence>
    <xs:group ref="CONDITION-EXPRESSION" minOccurs="0">
      <xs:choice>
        <xs:element ref="Condition"/>
        <xs:element ref="And"/>
        <xs:element ref="Or"/>
        <xs:element ref="Not"/>
      </xs:choice>
    </xs:group>
  </xs:sequence>
</xs:element>
```

Elementos padre

ActionButton, CheckBoxControl, CheckBoxGroupControl, ClientDirectoryChooserControl, ClientFileChooserControl, ComboBoxControl, DBConnectionChooserControl, DBTableChooserControl, ExtensionObjectPanel, ItemChooserControl, ModelViewerPanel, MultiFieldChooserControl, MultiItemChooserControl, PasswordBoxControl, PropertiesPanel, PropertiesSubPanel, PropertyControl, RadioButtonGroupControl, SelectorPanel, ServerDirectoryChooserControl, ServerFileChooserControl, SingleFieldChooserControl, SingleFieldValueChooserControl, SingleItemChooserControl, SpinnerControl, StaticText, SystemControls, TabbedPanel, TableControl, TextAreaControl, TextBoxControl, TextBrowserPanel

Elementos hijo

And, Condition, Not, Or

Tipos ampliados

Los tipos ampliados amplían los elementos en un documento XML añadiendo atributos y elementos secundarios. Para utilizar un tipo ampliado en un documento XML, se especifica el tipo ampliado con el atributo xsi:type para el elemento. A continuación podrá usar los atributos y elementos definidos por el tipo ampliado.

Tipo ItemChooserControl

Tabla 207. Atributos de ItemChooserControl

Atributo	Uso	Descripción	Valores válidos
catalog	**obligatorio**		*cadena*
description	opcional		*cadena*
descriptionKey	opcional		*cadena*
label	opcional		*cadena*
labelAbove	opcional		*booleano*
labelKey	opcional		*cadena*
labelWidth	opcional		*positiveInteger*

Tabla 207. Atributos de ItemChooserControl (continuación)

Atributo	Uso	Descripción	Valores válidos
mnemonic	opcional		*cadena*
mnemonicKey	opcional		*cadena*
property	**obligatorio**		*cadena*
showLabel	opcional		*booleano*

Representación de XML

```
<xs:complexType name="ItemChooserControl" mixed="false">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="Layout" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Enabled" minOccurs="0" maxOccurs="1"/>
      <xs:element ref="Visible" minOccurs="0" maxOccurs="1"/>
    </xs:choice>
  </xs:sequence>
</xs:complexType>
```

Amplía

ComboBoxControl

Elementos hijo

Enabled, Layout, Visible

Tipos relacionados

ItemChooserControl

Avisos

Esta información se ha desarrollado para los productos y servicios ofrecidos en todo el mundo.

Es posible que IBM no ofrezca los productos, servicios o características que se tratan en este documento en otros países. Póngase en contacto con el representante local de IBM si desea obtener información sobre los productos y servicios disponibles en su zona. Las referencias a programas, productos o servicios de IBM no pretenden establecer ni implicar que sólo puedan utilizarse dichos productos, programas o servicios de IBM. Se puede utilizar en su lugar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ningún derecho de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y comprobar el funcionamiento de todo producto, programa o servicio que no sea de IBM.

Puede que IBM tenga patentes o solicitudes de patente pendientes que afecten al objeto de este documento. Este documento no le otorga ninguna licencia para estas patentes. Puede enviar preguntas acerca de las licencias, por escrito, a:

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

Para consultas sobre licencias relativas a información de doble byte (DBCS), póngase en contacto con el Departamento de propiedad intelectual de IBM de su país o envíe sus consultas, por escrito, a:

Intellectual Property Licensing
Ley de Propiedad intelectual
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde dichas disposiciones entren en contradicción con la legislación local: INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGÚN TIPO, NI EXPLÍCITA NI IMPLÍCITA, INCLUYENDO, PERO NO LIMITÁNDOSE, A LAS GARANTÍAS IMPLÍCITAS DE NO VULNERABILIDAD, COMERCIALIZACIÓN O ADECUACIÓN A UN PROPÓSITO DETERMINADO. Algunos estados no permiten la renuncia a expresar o a garantías implícitas en determinadas transacciones , por lo tanto , esta declaración no se aplique a usted.

Esta información puede incluir imprecisiones técnicas o errores tipográficos. Periódicamente, se efectúan cambios en la información aquí y estos cambios 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 a sitios Web que no sean de IBM en esta información solamente es ofrecida por comodidad y de ningún modo sirve como aprobación de esos sitios Web. El material de esos sitios Web no forma parte del material de este producto de IBM y el uso de esos sitios Web es a cuenta y riesgo del usuario.

IBM puede utilizar o distribuir la información que se le proporcione de la forma que considere adecuada, sin incurrir en ninguna obligación con el cliente.

Los licenciatarios de este programa que deseen obtener información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) el uso mutuo de información que se haya intercambiado, deben ponerse en contacto con:

IBM Software Group
ATTN: Licensing
200 W. Madison St.
Chicago, IL; 60606
EE.UU.

Esta información estará disponible, bajo las condiciones adecuadas, incluyendo en algunos casos el pago de una cuota.

El programa bajo licencia descrito en este documento y todo el material bajo licencia disponible se proporcionan bajo los términos de IBM Customer Agreement, IBM International Program License Agreement o cualquier otro acuerdo equivalente entre IBM y el cliente.

Cualquier dato de rendimiento mencionado aquí ha sido determinado en un entorno controlado. Por lo tanto, los resultados obtenidos en otros entornos operativos pueden variar de forma significativa. Es posible que algunas mediciones se hayan realizado en sistemas en desarrollo y no existe ninguna garantía de que estas medidas sean las mismas en los sistemas comerciales. Además, es posible que algunas mediciones hayan sido estimadas a través de extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben consultar los datos que corresponden a su entorno específico.

Se ha obtenido información acerca de productos que no son de IBM de los proveedores de esos productos, de sus publicaciones anunciadas o de otros orígenes disponibles públicamente. IBM no ha probado dichos productos y no puede confirmar la precisión de su rendimiento, la compatibilidad ni contemplar ninguna otra reclamación relacionada con los productos que no son de IBM. Las preguntas acerca de las aptitudes de productos que no sean de IBM deben dirigirse a los proveedores de dichos productos.

Todas las declaraciones sobre el futuro del rumbo y la intención de IBM están sujetas a cambio o retirada sin previo aviso y representan únicamente metas y objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales diarias. Para ilustrarlos lo máximo posible, los ejemplos incluyen los nombres de las personas, empresas, marcas y productos. Todos esos nombres son ficticios y cualquier parecido con los nombres y direcciones utilizados por una empresa real es pura coincidencia.

Si está viendo esta información en copia electrónica, es posible que las fotografías y las ilustraciones en color no aparezcan.

Marcas comerciales

IBM, el logotipo de IBM e ibm.com son marcas registradas o marcas comerciales registradas de International Business Machines Corp., registrada en muchas jurisdicciones en todo el mundo. Otros nombres de productos y servicios pueden ser marcas registradas de IBM u otras empresas. Hay disponible una lista actual de marcas registradas de IBM en la web en “Copyright and trademark information” (Información de copyright y de marca registrada) en www.ibm.com/legal/copytrade.shtml.

Intel, el logotipo de Intel, Intel Inside, el logotipo de Intel Inside, Intel Centrino, el logotipo de Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium y Pentium son marcas comerciales o marcas registradas de Intel Corporation o sus filiales en Estados Unidos y otros países.

Linux es una marca registrada de Linus Torvalds en Estados Unidos, otros países o ambos.

Microsoft, Windows, Windows NT, y el logotipo de Windows son marcas comerciales de Microsoft Corporation en Estados Unidos, otros países o ambos.

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

Java y todas las marcas comerciales y logotipos con base Java son marcas comerciales o son marcas registradas de Oracle y/o sus filiales.

Otros productos y nombres de servicio pueden ser marcas comerciales de IBM u otras empresas.

Índice

E

F

G

H

I

J

L

M

N

O

P

R

S

T

V

X

Impreso en España