IBM InfoSphere QualityStage
Versión 11 Release 3

Consulta de acciones de patrón

SC43-1216-00

IBM InfoSphere QualityStage
Versión 11 Release 3

Consulta de acciones de patrón

SC43-1216-00

Nota

Antes de utilizar esta información y el producto al que da soporte, lea la información del apartado “Avisos y marcas registradas” en la página 69.

Contenido

Consulta de acciones patrón de IBM InfoSphere QualityStage

Para obtener una estandarización correcta, es necesario entender los conceptos de coincidencia de patrón y las razones para realizar las coincidencias.

Este material de consulta describe el lenguaje de acción de patrón y los archivos de acción de patrón. Es para desarrolladores de aplicaciones. El archivo de patrón-acción consiste en una serie de patrones y de acciones asociadas. Los datos entrantes coinciden o no coinciden con un patrón. Si los datos de entrada coinciden con un patrón, las acciones asociadas con dicho patrón se ejecutan. Si los datos de entrada no coinciden con el patrón, las acciones se omiten.

Introducción al lenguaje de acción de patrón

Utilice el lenguaje de acción de patrón (Pattern Action) para manipular los datos. Puede descifrar e identificar patrones en los datos, y luego realizar acciones en estos datos de acuerdo con el patrón.

Un archivo de acción de patrón contiene una serie de conjuntos de acción de patrón. Cada conjunto contiene una condición de patrón seguida de sentencias de acción. Las acciones se toman contra datos de entrada que se han separado en señales y se clasificarán. Las acciones se basan en un determinado patrón de señales.

La condición de patrón puede contener los elementos siguientes:

Operandos
Representación de clase de datos de entrada. La representación de clase es una clase definida por el usuario o una clase predeterminada.

Variables de usuario
Definidas por el usuario, los nombres simbólicos asociados con valores que pueden cambiarse.

Campos de diccionario
Una colección de nombres de campo de salida como se definen en la tabla de diccionario ubicada en el archivo de definición de diccionario (.DCT).

Los patrones se ejecutan en el orden en el que aparecen en el archivo de acción de patrón. Un patrón coincide con los datos de entrada o no coincide. Si coincide, las acciones asociadas con dicho patrón se ejecutan. Si no, las acciones se omiten. El proceso a continuación prosigue con el siguiente patrón en el archivo.

El archivo de acción de patrón le permite utilizar la lógica que se convierte en una parte de los conjuntos de reglas. Los conjuntos de reglas, aplicados contra datos de entrada analizados y clasificados, estandarizan los datos.

Los archivos de acciones de patrón son archivos ASCII que puede crear o actualizar utilizando cualquier editor de texto estándar.

Descripción del formato de patrón

Puede describir el formato de patrón con operandos y sentencias condicionales.

Un patrón puede constar de uno o más operandos. Los operandos se separan mediante líneas verticales. Por ejemplo, el patrón siguiente tiene cuatro operandos:

```
^ | D | ? | T
```

Se hace referencia a éstos en las acciones como [1], [2], [3] y [4].

Puede utilizar espacios para separar operandos. Por ejemplo, los dos patrones siguientes son equivalentes:

```
^|D|?|T
^ | D | ? | T
```

Consejo: El uso de espacios para separar los operandos y conductos hace que la lectura y la depuración de patrones sean más fáciles. Los espacios se utilizan en las normas ya construidas proporcionadas con IBM® InfoSphere QualityStage.

Puede añadir comentarios precediéndolos con un punto y coma. Una línea entera o sólo el final de una línea puede ser un comentario. Por ejemplo:

```
;
;Direcciones estándar de proceso
;
^ | D | ? | T  ; 123 N MAPLE AVE
```

Puede hacer referencia a campos poniendo el nombre de columna entre llaves. Por ejemplo, {HouseNumber}, {StreetPrefixDirectional}, {StreetName} y {StreetSuffixType} hacen referencia a nombres de columna de diccionario que se definen en el archivo de definición del diccionario.

```
^ | D | ? | T | $ |
[ {HouseNumber} = "" & {StreetPrefixDirectional} = "" &
 {StreetName} = "" & {StreetSuffixType} = "" ] ;
 Patrón común encontrado: CALL Address_Type SUBROUTINE then EXIT
```

La coincidencia de patrón para una línea de patrón individual se detiene después de encontrar la primera coincidencia. Por ejemplo, en la dirección 123 MAPLE AVE y PLACE HILL 456, el siguiente patrón coincide con 123 MAPLE AVE pero no con PLACE HILL : 456

```
^ | ? | T
```

Un patrón no puede contener operandos. Por ejemplo, la sentencia condicional `[Required_Step ="TRUE"]` es un patrón que contiene una variable de usuario : *Required_Step*.

Los patrones más simples constan de sólo tipos de clasificación:

```
^ | D | ? | T
```

Son directos y no requieren más explicación. Los guiones y las barras inclinadas puede introducirse en los patrones. Por ejemplo:

```
123-45 coincide con ^ | - | ^

123 ½ coincide con ^ | ^ | / | ^
```

Elementos de análisis (PRAGMA)

El proceso de estandarización empieza por identificar señales dentro de los datos entrantes. Una señal puede ser un solo carácter, una palabra o varias palabras que no están separadas por espacios.

Los parámetros de análisis de la tabla en el archivo de acción de patrón definen las señales. Por ejemplo, para los idiomas latinos, 123-456 tiene tres señales: 123, el guión (-) y 456. Un guión separa palabras y se considera una señal en sí misma.

Los espacios son señales separadas. También se separan de la entrada. Por ejemplo, 123 MAIN ST consta de tres señales: 123, MAIN, y ST.

Utilización de SEPLIST y STRIPLIST

SEPLIST y **STRIPLIST** son las sentencias de especificación que se colocan entre las líneas **PRAGMA_START** y **PRAGMA_END** de un archivo de acción de patrón.

Puede alterar temporalmente las suposiciones predeterminadas especificando una de las sentencias siguientes o ambas:

- **SEPLIST**. Utiliza cualquier carácter de la lista para separar señales.
- **STRIPLIST**. Elimina cualquier carácter de la lista.

Cualquier carácter que se encuentra en ambas listas separa señales pero no aparece como señal en sí. El mejor ejemplo son los espacios. Uno o más espacios se separan pero el espacio indica dónde finaliza un palabra y empieza otra. Incluya el carácter de espacio en **SEPLIST** y **STRIPLIST**.

Si desea incluir **SEPLIST** y **STRIPLIST**, póngalos como el primer conjunto de sentencias en el archivo .pat, precedido de **PRAGMA_START** y seguido de **PRAGMA_END**. Por ejemplo:

```
\PRAGMA_START
SEPLIST " ,"
STRIPLIST " -"
\PRAGMA_END
```

Ponga los caracteres en la lista entre comillas.

Aplicación de reglas de análisis a una lista

La clase de señal especial (~) representa caracteres especiales que no están incluidos en SEPLIST y STRIPLIST. Estos caracteres (!, \, @, ~,%) requieren un manejo especial.

Cuando se añaden caracteres especiales, tenga en cuenta las siguientes reglas:

- No utilice las comillas en **SEPLIST** o **STRIPLIST** a menos que las preceda con el carácter de escape de barra invertida (\).
- La barra inclinada invertida (\) es el carácter de escape que se utiliza en un patrón, pero debe ser de escape (\\).

En este ejemplo, el espacio en ambas listas y el guión está en **STRIPLIST** pero no **SEPLIST**. Los guiones se eliminarán de forma que STRATFORD-ON-AVON se considera STRATFORDONAVON.

```
SEPLIST: " !?%$,.;:()/#&"
STRIPLIST: " !?*@$,.;:-\\''"
```

En este ejemplo, el guión está en ambas listas. Puesto que el **SEPLIST** se aplica antes que el **STRIPLIST**, STRATFORD-ON-AVON en los datos de entrada se analiza en tres señales: STRATFORD, ON y AVON.

```
SEPLIST: " !?%$,.;:-()/#&"
STRIPLIST: " !?*@$,.;:-\\''"
```

En este ejemplo, la coma separa las señales para que el nombre de la ciudad y el estado se puedan encontrar (SALT LAKE CITY, UTAH). Cualquier otro carácter especial se clasifica como un tipo especial.

```
SEPLIST: " !?%$,.;:()-/#&"
STRIPLIST: " !?*@$.;:\\''"
```

Cada conjunto de reglas tiene sus propias listas. Si no se codifica ninguna lista para un conjunto de reglas, se utilizan las listas por defecto siguientes:

```
SEPLIST: " !?%$,.;:()-/#&"
STRIPLIST: " !?*@$,.;:\\''"
```

Al alterar temporalmente los **SEPLIST** y **STRIPLIST** predeterminados, no provoque conflictos con los significados de clase predefinidos porque la clase de un carácter especial cambia si se incluye en **SEPLIST**.

Si se incluye un carácter especial en **SEPLIST** y no en **STRIPLIST**, la clase de señal para dicho carácter se convierte en el propio carácter.

Por ejemplo, ^ es el especificador de clase numérica. Si añade este carácter para **SEPLIST** y no para **STRIPLIST**, a cualquier señal que conste de ^ se le da ^ . Esta señal coincidiría con una clase numérica (^) en un archivo de acción de patrón.

Especificación del señalizador

En la sección PRAGMA del archivo de acción de patrón, puede utilizar el mandato **TOK** para especificar el valor regional que desea utilizar en un conjunto de reglas, y así indicar el modo en que desea manejar señales.

TOK es una sentencia de especificación opcional. Si no especifica un señalizador, el señalizador utilizado se basa en la configuración regional del sistema en el que se ejecuta una investigación o estandarización. Puede escogir uno de los señalizadores siguientes:

Señalizador	Descripción
Raíz latina	Para los idiomas como el inglés, español, francés y alemán. Las señales normalmente se separan con espacios.
CJK	Para los idiomas chino, japonés y coreano. Las señales no se suelen separar.

La sintaxis para **TOK** es la siguiente:

```
TOK locale
```

Locale es el estándar de entorno local de International Components for Unicode (ICU).

Durante la estandarización, el señalizador CJK se utiliza si el mandato TOK es seguido de una variable de entorno local que empieza con uno de los códigos siguientes:

- ja
- zh
- ko
- vi

Si la variable de entorno local es cualquier otro valor, se utiliza el señalizador de raíz latina.

Por ejemplo, si especifica `tok en_US`, el señalizador incluye consideraciones de idiomas latinos en el enfoque de señalización. Si especifica `TOK jp_JP`, el señalizador incluye consideraciones específicas de entorno local (CKJ) en el método de señalización.

Patrones incondicionales

Los patrones incondicionales no son sensibles a los valores individuales, son los más sencillos de codificar y los más generales. Puede especificar condiciones para hacer que los patrones coincidan sólo en circunstancias especificadas.

Identificación de clases de patrón simple

Las clases de patrón simple se utilizan para profundizar en la identificación de datos con un patrón significativo desde el cual se puede hacer coincidir con las acciones de patrón.

Las clases de patrón simple se representan mediante carácteres individuales.

Dentro de los patrones, debe utilizar el carácter de escape de barra inclinada invertida (\), para evitar que la sintaxis de las tablas interfiera con ciertas clases de caracteres de patrón único. Utilice el carácter de escape de barra invertida (\) con las siguientes clases de carácter único : el guión (-), barra inclinada (/), signo de número (#), paréntesis izquierdo y derecho () y ampersand (&).

Vaya con cuidado al especificar las entradas **SEPLIST** y **STRIPLIST**. Por ejemplo, para reconocer el carácter ampersand como una señal individual, inclúyalo en **SEPLIST** pero no en **STRIPLIST**. Si la barra inclinada invertida se encuentra en **SEPLIST**, su clase será \ (barra inclinada invertida). Si una barra inclinada invertida se utiliza en un patrón, debe tener un carácter de escape en un patrón como una barra inclinada invertida doble (\\). Consulte también “Aplicación de reglas de análisis a una lista” en la página 3

La clase **NULL** (0) no está incluida en esta lista de clases de carácter único. La clase **NULL** se utiliza en las clasificaciones (.CLS) o en la acción **RETYPE** para realizar una señal **NULL**. Dado que una clase **NULL** nunca coincide con nada, nunca se utiliza en un patrón.

Las clases de patrón simples son:

Tabla 1. Lista y descripción de clases de patrón simple

Clase	Descripción
A - Z	Clase suministrada por usuario a partir de las clasificaciones Las clases A-Z se corresponden con las clases que se codifican en las clasificaciones. Por ejemplo, si a APARTMENT se le asigna la clase U en las clasificaciones, entonces APARTMENT coincidirá con un patrón simple de U.
^	Numérico La clase ^ (signo de intercalación) representa un único número, por ejemplo el 123. Sin embargo, el número 1.230 utiliza tres señales: el número 1, una coma y el número 230.

Tabla 1. Lista y descripción de clases de patrón simple (continuación)

Clase	Descripción
?	Una o varias palabras consecutivas que no se encuentran en las clasificaciones. La clase ? (signo de interrogación) representa una o más palabras alfabéticas consecutivas. Por ejemplo, MAIN, CHERRY HILLy SATSUMA PLUM TREE HILL coinciden con una sola clase ? siempre que ninguna de estas palabras esté en las clasificaciones del conjunto de reglas. La clase ? es útil para nombres de calle cuando los nombres de calle de varias palabras o de una sola palabra deben tratarse de forma idéntica.

+	Una palabra alfabética que no está en clasificaciones
	
	La clase + (signo más) es útil para separar las partes de una serie desconocida. Por ejemplo, en un nombre como OWAIN LIAM JONES, copie las palabras individuales a columnas con el nombre de pila, un segundo nombre y apellido de la siguiente manera :
	`+ | + | +`
	`COPY [1] {GivenName}`
	`COPY [2] {MiddleName}`
	`COPY [3] {FamilyName}`

&	Una única señal de cualquier tipo. La clase & (ampersand) representa una sola señal de cualquier clase. Por ejemplo, un patrón para coincidir con una sola palabra después de un tipo de apartamento es: `U | &` Este patrón reconoce `SUITE 11`. Sin embargo, en un caso como `APT PRIMER FlOOR`, sólo `PRIMERA APT` es reconocido por este patrón.
\&	Escriba el carácter de escape de barra invertida (\) delante del ampersand para utilizarlo como literal. `< | \& | ? | T` Este patrón reconoce 1ST & MAIN ST.
>	Numérico inicial La clase > (símbolo mayor que) representa una señal con números seguidos de letras. Por ejemplo, un número de casa como 123A MAPLE AVE puede ser coincidente de la siguiente manera : `> | ? | T` Este patrón reconoce `123A`. La señal contiene números y caracteres alfabéticos, pero los números son líderes. En este ejemplo, `T` representa el tipo de calle.
<	Carácter alfabético inicial La clase < (símbolo menor que) se compara a sí mismo con letras alfabéticas iniciales. Es útil con los ejemplos siguientes: A123 ALPHA77 La señal contiene caracteres alfabéticos y números, pero los caracteres alfabéticos son lso principales.

Tabla 1. Lista y descripción de clases de patrón simple (continuación)

Clase	Descripción
@	Combinación compleja La clase @ (signo de arroba) representa señales que contienen una combinación compleja de caracteres alfabéticos y numéricos, por ejemplo: A123B, 345BCD789. Por ejemplo, la información de área como Hamilton ON L8N 2P1 puede coincidir como se indica a continuación : `+ | P | @ | @` En este ejemplo, P representa Provincia. El primer @ representa `L8N` y el segundo @ representa `2P1`.
~	Puntuación especial La clase ~ (tilde) representa caracteres especiales que no se encuentran en **`SEPLIST`**. Por ejemplo, si un **`SEPLIST`** no contiene el signo de dólar y signo de porcentaje, puede utilizar el patrón siguiente : `~ | +` En este ejemplo, $HELLO y % OFF coinciden con el patrón.
k	Uno o más caracteres numéricos chinos
/	Literal La clase / (barra inclinada) es útil para las direcciones de fracción como 123 ½ MAPLE AVE, que coincide con el patrón siguiente : `> | ^ | / | ^ | ? | T`
\/	Barra inclinada invertida, barra inclinada Puede utilizar el carácter de escape de barra invertida (\) con la barra inclinada en la misma forma que se utiliza la clase / (barra inclinada).
-	Literal La clase - (guión) se suele utilizar para rangos de direcciones, por ejemplo, un rango de direcciones como 123-127 coincide con el patrón siguiente : `^ | - | ^`
\-	Puede utilizar el carácter de escape de barra invertida (\) con el guión del mismo modo que utiliza la clase - (guión).
\#	Literal. Debe utilizar un carácter de escape de barra inclinada invertida (\), por ejemplo: \. La clase # (almohadilla) a menudo se utiliza como prefijo de unidad, por ejemplo, una dirección como suite #12 o la unidad #9A coincide con el patrón siguiente : `U | \# | &`

Tabla 1. Lista y descripción de clases de patrón simple (continuación)

Clase	Descripción

()	Literal
	
	Las clases (y) (paréntesis) se utilizan para incluir operandos o variables de usuario en una sintaxis de patrón. Un ejemplo de una sintaxis de patrón que incluye operadores numéricos iniciales y un operador de carácter final es el siguiente:
	`> | ? | T`
	`COPY [1](n) {HouseNumber}`
	`COPY [1](-c) {HouseNumberSuffix}`
	`COPY [2] {StreetName}`
	`COPY_A [3] {StreetSuffixType}`
	`EXIT`
	
	El ejemplo de sintaxis de patrón puede reconocer la dirección 123A MAPLE AVE. Los números 123 son reconocidos como el número de casa y la letra A se reconoce como un sufijo de número de casa.
	
	Utilice el carácter de escape de barra invertida (\) con los paréntesis de apertura o cierre para filtrar los comentarios entre paréntesis. Para eliminar un comentario entre paréntesis como (consulte Joe, Habitación 202), puede especificar este patrón:
	`\( | ** | \)`
	`RETYPE [1] 0`
	`RETYPE [2] 0`
	`RETYPE [3] 0`
	
	El ejemplo de código elimina los paréntesis y el contenido de la frase entre paréntesis. Además, cuando se vuelva a especificar estos campos en **NULL**, básicamente se hace que los patrones no tengan en cuenta la sentencia entre paréntesis que se encuentran más abajo en el archivo de acción de patrón.
	
	La clase **NULL** (0) no está incluida en esta lista de clases de carácter único. La clase **NULL** se utiliza en las clasificaciones (.CLS) o en la acción **RETYPE** para realizar una señal **NULL**. Dado que una clase **NULL** nunca coincide con nada, nunca se utiliza en un patrón.

\(y \)	Utilice el carácter de escape de barra invertida (\) con los paréntesis de apertura o cierre para filtrar los comentarios entre paréntesis. Para eliminar un comentario entre paréntesis como (consulte Joe, Habitación 202), puede especificar este patrón:
	`\( | ** | \)`
	`RETYPE [1] 0`
	`RETYPE [2] 0`
	`RETYPE [3] 0`
	
	El ejemplo de código elimina los paréntesis y el contenido de la frase entre paréntesis. Además, cuando se vuelva a especificar estos campos en **NULL**, básicamente se hace que los patrones no tengan en cuenta la sentencia entre paréntesis que se encuentran más abajo en el archivo de acción de patrón.

Aplicación de clases de subcampo (1 a 9, -1 y -9)

Las clases de subcampo de 1 a 9 y -1 a -9 se utilizan para analizar palabras individuales de una serie ? .

El número 1 representa la primera palabra, el número 2 representa la segunda, el número -1 representa la última palabra, y el número -2 representa la siguiente a la última palabra. Si la palabra a la que se hace referencia no existe, el patrón no coincide. Si está procesando los nombres de empresas y sólo quería la primera

palabra, un nombre de empresa como WILLFRED BIG SUPER CELL COMPANY coincide con los patrones siguientes (se presuponen que EMPRESA está en las clasificaciones (.CLS) como un tipo C).

+ | + | + | + | C	WILLIAMS es operando operand [1], BIG es operando [2], SUPER es operando [3] CELL es operando [4] y COMPANY es operando [5]
? | C	WILLIAMS BIG SUPER CELL es operando [1], COMPANY es operando [2]
1 | C	WILLIAMS es operando [1], COMPANY es operando [2]
2 | C	BIG es operando [1], COMPANY es operando [2]
–1 | C	CELL es operando [1], COMPANY es operando [2]
–2 | C	SUPER es operando [1], COMPANY es operando [2]

Puede combinar clases alfabéticas individuales (**+**) con clases de subcampo. Por ejemplo, en una serie de señales desconocidas consecutivas como CHERRY HILL SANDS, el patrón siguiente provoca la coincidencia siguiente:

```
+ | -1
```

+ coincide con la palabra CHERRY y **-1** coincide con SANDS. El operando [1] es CHERRY y el operando [2] es SANDS.

Especificación de rangos de subcampo

Al coincidir con un patrón, puede especificar un rango de palabras.

El formato es el siguiente:

```
(beg:end)
```

Ejemplos:

(1:3)	Especifica un rango de palabras 1 - 3
(–3:–1)	Especifica un rango de la tercera palabra de la última a la última palabra
(1:–1)	Especifica un rango de la primera palabra a la última palabra (tenga en cuenta que si utiliza ? para la última palabra realiza una acción más eficaz)

Si tiene la dirección 123-A Main St B, puede utilizar el patrón siguiente:

```
^ | - | (1:2)
COPY [3] {HouseNumberSuffix}
RETYPE [2] 0
RETYPE [3] 0
```

Esta patrón hace que A y B se muevan al campo {HouseNumberSuffix} (sufijo de número de casa). Este patrón también vuelve a especificar A y B para las señales NULL (y también de forma similar al volver escribir el guión) para que no se tengan en cuenta en más patrones del archivo.

Aplicación de clase universal

Puede combinar la clase (**) universal con otros operandos para restringir las señales adquiridas por la clase. La clase universal puede ser nulo, lo que significa ninguna señal.

La clase ** coincide con todas las señales. Por ejemplo, si utiliza un patrón de **, hace coincidir 123 MAIN ST y 123 ST MAIN, LOS ANGELES, CA 90016, etc. El siguiente patrón coincide con todas las señales antes del tipo (que puede ser sin señales) y el tipo:

```
** | T
```

Por lo tanto, 123 N MAPLE AVE coincide con el operando [1] siendo 123 N MAPLE y el operando [2] siendo AVE.

La clase universal puede ser nula. No es necesario colocar señales antes del tipo. AVENUE también coincide con este patrón con el operando [1] que es **NULL**.

En el patrón siguiente, ** hace referencia a todas las señales entre entre el tipo numérico y de calle:

```
^ | ** | T
```

En el ejemplo, la clase ^ (acento circunflejo) y la clase del tipo (T) definen el inicio y el final de la clase ** . La clase ** puede contener números adicionales a la clase ^ pero no cualquier señal de tipo de calle adicional.

Puede especificar un rango de señales para un operando **. Por ejemplo, el patrón siguiente coincide con un numérico seguido por al menos dos señales que no son del tipo calle seguidas por un tipo de calle :

```
^ | ** (1:2) | T
```

El operando [2] consta de exactamente dos señales que no son de tipo de calle. Esto coincide con 123 CHERRY DR TREE, pero no 123 DR ELM. Sólo una señal sigue al número. Puede especificar rangos desde una sola señal, como (1:1), para todas las señales, como por ejemplo (1:-1).

El patrón **(1:1) da lugar a un tiempo de procesamiento mucho más lento que el & equivalente para hacer coincidir con cualquier señal individual. Sin embargo, no utilice & en un patrón con **, como ** | &, porque la primera señal encontrada es utilizada por &. Comprobaciones de valor o condiciones adecuadas que se aplican utilizando & con ** pueden tener sentido. Por ejemplo:

```
** | & = "123", "ABC"
```

No se permiten valores o expresiones condicionales para los operandos con **.

Utilización del final del especificador de campo ($)

El especificador $ no coincide con ninguna señal real, pero indica el final del patrón.

Una condición de patrón sin el especificador $ puede representar una parte del campo como la información de ciudad, estado y código postal en una dirección en EE.UU. Por ejemplo, en Littleton, MA 01460y LITTLETON MA 01460-6245 coinciden con el patrón siguiente :

```
? | S | ^
```

Sin embargo, el guión (-) y el ZIP+4, 01460-6245, no son parte de la coincidencia. Para incluir el código postal, 01460-6245, como parte de la condición de coincidencia, utilice el patrón de la siguiente manera:

```
? | S | ^ | - | ^ | $
```

Los datos de entrada que siguen al código postal no forman parte de la coincidencia.

Utilización de especificador de posición flotante

Puede utilizar los especificadores de posicionamiento para modificar la ubicación de la coincidencia de patrón.

Para los patrones documentados hasta el momento, el patrón tenía que coincidir con la primera señal en el campo. Por ejemplo, el patrón siguiente coincide con MAPLE AVE y CHERRY HILL RD, pero no coincide con 123 MAPLE AVE, porque un número es la primera señal:

```
? | T
```

Puede utilizar especificadores flotantes para explorar el campo de entrada para un patrón determinado. El asterisco (*) es un especificador de posición y significa que se realiza una búsqueda en el patrón, de izquierda a derecha, hasta que haya explorado una coincidencia o el patrón completo. Puede utilizar el asterisco (*) para indicar que la clase inmediatamente siguiente es una clase flotante.

Si tiene números de apartamento de la dirección, para simplificar los datos, es recomendable explorar, procesar y volver a especificar los números de apartamento en NULL para que los patrones básicos pueden procesar la dirección principal. Por ejemplo, las direcciones como `123 MAIN ST APT 34` y 770 king ST FL 3 RM 101 contienen una dirección postal básica con información adicional. El siguiente patrón busca la información de unidad y planta, llena los campos de diccionario adecuados e impide que la información de unidad y planta se siga procesando. U es la clase de unidad y F es la clase de palabra.

```
*U | ^
COPY_A [1] {UnitType}
COPY [2] {UnitValue}
RETYPE [1] 0
RETYPE [2] 0


*F | ^
COPY_A [1] {FloorType}
COPY [2] {FloorValue}
RETYPE [1] 0
RETYPE [2] 0
```

Al volver a especificar las señales en **NULL** se hace que los patrones no tengan en cuenta las señales más adelante en el archivo de acción de patrón. Puede evitar volver contabilizar todas las combinaciones de posibilidades. Los datos que deben procesarse son `123 MAIN ST` y 770 KING ST. Ambas entradas tienen el patrón: `^ | ? | T`.

El procesamiento de una parte de los datos utilizando el especificador flotante simplifica los dos campos de entrada y hace que el patrón de entrada sea igual. La tarea de estandarización es más fácil.

Los especificadores de posición flotante operan explorando una señal hasta encontrar una coincidencia. Si todos los operandos coinciden, el patrón coincide. Si

los operandos no coinciden, el escáner avanza una señal a la derecha y repite el proceso. Esto es como mover una plantilla en la serie de entrada. Si la plantilla coincide, el proceso se lleva a cabo. De lo contrario, la plantilla avanza a la siguiente señal.

Nota: Sólo puede haber una coincidencia de un patrón en una serie de entrada. Después de que se hayan procesado las acciones, el control va al patrón siguiente, aunque pueda haber otras coincidencias en la línea.

El asterisco debe ir seguido de una clase. Por ejemplo, los operandos siguientes son válidos con un especificador de posición flotante seguido por una clase estándar:

```
* U
* ?
* ^
```

Puede haber más de un especificador de posición flotante en un patrón. Por ejemplo, los operandos siguientes coinciden con JOHN CHERRY HILL NORTH RD: 123.

```
*^ | ? | *T
```

El operando [1] es 123. El operando [2] es CHERRY HILL. El operando [3] es RD. NORTH se clasifica como direccional (D) de forma que no se incluye en la serie desconocida (?) .

Utilización de especificador de posición flotante inversa

El especificador de posición flotante inversa, indicado por un signo de número (), es similar al especificador de posición flotante de (*) excepto que continúa la exploración de derecha a izquierda en lugar de izquierda a derecha.

Puede utilizar este especificador para buscar elementos que aparecen al final de un campo, como por ejemplo código postal, estado y designaciones de apartamentos.

El especificador de posición flotante inversa sólo debe aparecer en el primer operando de un patrón, ya que se utiliza para colocar el patrón. Por ejemplo, si desea encontrar un código postal y ha dado el nombre del estado de clase S, el siguiente patrón realiza una exploración de derecha a izquierda para en busca de un nombre de estado seguido de un número:

```
#S | ^
```

Si tiene una serie de entrada CALIFORNIA 45 PRODUCTS, PHOENIX ARIZONA 12345 DEPT 45, la exploración de derecha a izquierda posiciona el patrón en ARIZONA. El número siguiente crea una coincidencia para el patrón.

Si no se encuentra ninguna coincidencia, la exploración continúa a la izquierda hasta que se encuentra un estado seguido de un número. Si está limitado al especificador de posición flotante de izquierda a derecha estándar (*S | ^), CALIFORNIA 45 se interpretará de forma incorrecta como nombre de estado y código postal.

Utilización del especificador de posición fija

El especificador de posición fija está situado en un operando en particular en la serie de entrada.

A veces es necesario colocar el patrón coincidente en un operando en particular en la serie de entrada. Este proceso es gestionado por el especificador de posición fija %*n*. Ejemplos de especificador de posición fija:

%1	Coincide con la primera señal
%2	Coincide con la segunda señal
%–1	Coincide con la última señal
%–2	Coincide con las señales de la segunda a la última

Las posiciones se pueden calificar mediante %*n* con un tipo de señal. A continuación se muestran algunos ejemplos:

%2^	Coincide con la segunda señal numérica
%-1^	Coincide con la última señal numérica
%3T	Coincide con la tercera señal de tipo de calle
%2?	Coincide con el segundo conjunto de dos o más señales alfabéticas desconocidas

Puede utilizar el especificador de posición fija (%) en sólo dos formas:

- Como el primer operando de un patrón
- Como los operandos primero y segundo de un patrón

El patrón siguiente está permitido y coincide con la segunda señal numérica como operando [1] y la tercera señal alfabética inicial que sigue como operando[2]:

```
%2^ | %3<
```

El especificador de posición fija trata cada señal según su clase. Los ejemplos siguientes ilustran cómo utilizar el especificador de posición fija para el campo de entrada :

```
John Doe
123 Martin Luther St
Salt Lake
```

%1 1	Coincide con la primera palabra de la primera serie: JUAN
%1 2	Coincide con la segunda palabra de la primera serie: DOE
%2 ?	Coincide con la segunda serie de las palabras desconocidas alfabéticas : MARTÍN LUTHER
%2 1	Coincide con la primera palabra de la segunda serie: MARTIN
%-2 –1	Coincide con la última palabra junto a la última serie: LUTHER
%3+	Coincide con la tercera palabra alfabética individual: MARTIN
%–1 ?	Coincide con la última serie de las palabras alfabéticas desconocidas: SALT LAKE
%–1+	Coincide con la última palabra alfabética individual: LAKE

El especificador de posición no continuar la exploración si un patrón de coincidencia falla (a diferencia de * y #).

Suponiendo que el valor de entrada S se clasifica como una D para la dirección, el patrón siguiente coincide con 789 S en la serie 123 A 456 B 789 S:

```
%3^ | D
```

Este mismo patrón no coincide con 123 A 456 B 789 C 124 S porque el tercer número (789) no está seguido por una dirección.

Calificador de clase de negación

El carácter de exclamación (!) se utiliza para indicar NOT.

La sintaxis del lenguaje de patrón siguiente muestra cómo utilizar el negativo para especificar coincidentes:

!T	Coincidencia con cualquier señal excepto un tipo de calle
!?	Coincidencia con cualquier señal

El ejemplo siguiente coincide con SUITE 3, APT molido pero no con SUITE CIRCLE porque CIRCLE está clasificado como un tipo de calle (T):

```
*U | !T
```

La frase RT 123 puede considerarse como un nombre de calle sólo si no hay ninguna palabra desconocida a continuación, como RT 123 MAPLE AVE. Puede utilizar el patrón siguiente para crear una no coincidencia con la palabra desconocida :

```
*T | ^ | !?
```

Este patrón coincide con RT 123, pero no con RT 123 MAPLE AVE debido a que un carácter alfabético desconocido sigue al operando numérico.

Puede combinar la clase de negación con la clase flotante (*) únicamente al principio de un patrón. Por ejemplo, al procesar direcciones de calle, es recomendable expandir ST a SAN donde corresponda.

Por ejemplo, cambie 123 ST CHARLES ST a 123 SAINT CHARLES ST, pero no convierta 123 MAIN ST POSTERIOR APT en 123 SAINT POSTERIOR APT MAIN. Puede utilizar el conjunto de patrones y acciones siguiente:

```
*!? | S | +
RETYPE [2] ? "SAINT"
```

El ejemplo anterior requiere que no haya ninguna clase desconocida antes del valor ST ya que las señales con este valor tiene su propia clase de S.

Patrones condicionales

Cuando se estandarizan direcciones, algunas direcciones requieren que las condiciones se adjunten a la coincidencia de patrón. Proporcionar valores condicionales en patrones permite el proceso correcto de casos específicos.

Valores condicionales simples

Una condición simple se expresa mediante el operando de patrón típico seguido de un signo igual y un valor.

Los valores alfabéticos deben entrecomillarse. Una condición para probar la presencia de una señal con el valor MAPLE seguido de un tipo de calle es:

```
*? = "MAPLE" | T
```

`*? = "MAPLE"` es una señal con una condición y también es un operando.

Si la palabra SOUTH está en las clasificaciones, puede probar de forma explícita SOUTH mediante D = "SOUTH" o cualquier dirección con la abreviación estándar S utilizando D = "S" para el operando.

Puede especificar valores numéricos sin comillas. El siguiente conjunto de acción de patrón coincide con 1000 MAIN pero no con 1001 MAIN.

```
* ^ = 1000 | ?
```

El operador de igualdad (=) comprueba tanto la abreviatura normalizada (si la señal se encuentra en el archivo .cls y tiene una abreviatura) y el valor de la señal original para la igualdad para el operando. Dos operadores adicionales están disponibles para probar la igualdad para la abreviatura solamente o el valor de señal original únicamente. Los operadores son los siguientes:

=A=	Sólo prueba la abreviatura del archivo .cls
=T=	Sólo comprueba el valor de señal original.

Por ejemplo, para gestionar correctamente AVE MARIA LANE, pruebe la igualdad con el valor de señal original:

```
*T =T= "AVE" | + | T
RETYPE [1] ?
```

Como operando, `*T =T= "AVE"` garantiza que AVE se codifica y no otro valor que se correlaciona con la misma abreviatura dejando entrada AVENUE MARIA inalterada. De forma similar, utilice =A= si sólo desea probar la abreviatura.

Expresiones condicionales

La expresión condicional se incluye entre corchetes inmediatamente después del operando de patrón.

Si los valores simples o tablas de valores no son suficientes para calificar coincidencias de patrón, puede utilizar expresiones condicionales. Estas expresiones tienen el formato siguiente:

operando [*expresión condicional*]

Una expresión condicional simple consta de un operando, un operador relacional y un segundo operando. Los siguientes son operadores relacionales:

<	Señal original es menos que
>	Señal original es mayor que
=	Abreviatura o señal original es igual a
=A=	Abreviatura es igual a
=T=	Señal original es igual a
<=	Señal original es menor o igual a
>=	Señal original es mayor o igual a

!=	Abreviatura o señal original no es igual a
!=A=	Abreviatura no es igual a
!=T=	Señal original no es igual a

El operando izquierdo puede ser cualquiera de los siguientes elementos:
- Un nombre de variable
- El contenido de campo del operando actual
- El contenido de cualquier campo de diccionario

El operando derecho puede ser cualquiera de los siguientes elementos:
- Un nombre de variable
- Un literal
- Una constante

El formato completo de una expresión condicional es:
operando-izquierdo operador-relacional operando-derecho

La tabla siguiente explica la expresión:

Operación	**Valores válidos**
operando-izquierdo	{} {} PICT {} LEN {*field-name*} {*field-name*} PICT {*field-name*} LEN *nombre-variable* *nombre-variable* PICT *nombre-variable* LEN <*expresión-aritmética*>
operador-relacional	< > <= >= != =
operando-derecho	Literal Constante Nombre de la variable

Contenido de operando actual

El contenido del operando actual está en las llaves izquierda y derecha ({ }).

Un ejemplo que utiliza fechas del calendario ilustra mejor este concepto. Al verificar fechas, verifique la longitud de los números. Un ejemplo es el siguiente:

```
^ [{}LEN=4] | - | ^ [{}LEN=2] | - | ^ [{}LEN=2]
; formato para ccyy-mm-dd
```

Este patrón coincide en 2009-07-08 pero no en 08.07.2009.

Los operandos de patrón del ejemplo anterior tienen el siguiente significado:

Operando [1]	^	Un número de 4 dígitos

Operando [2]	-	Un guión
Operando [3]	^	Un número de 2 dígitos
Operando [4]	-	Un guión
Operando [5]	^	Un número de 2 dígitos

Cuando los literales de caracteres se encuentran en una prueba de igualdad, la abreviatura estándar se prueba si hay uno disponible. Si esto falla, la entrada original se prueba. Los ejemplos siguientes muestran operandos de patrón cuando hay ROAD RD T en las clasificaciones:

`T [ {} = "RD" ]`	Compara la abreviatura del operando actual (RD) con el RD literal
`T [ {} = "ROAD" ]`	Compara el operando de entrada entero con el literal (porque no puede compararlo con la abreviatura)
`T [ {} =A= "RD"]`	Compara sólo la abreviatura del operando actual con RD
`T [ {} =T= "ROAD"]`	Compara el valor original de la señal con ROAD y no la abreviatura
`T [ {} <= "RD" ]`	Cuando las comparaciones (que no sean los operadores de igualdad) se especifican, la entrada original se utiliza en lugar de la abreviatura. Esto es cierto para cualquier comparación con un valor literal. Si el valor original es RD, el resultado es verdadero, pero, si el valor original es ROAD, el resultado es falso.

Contenido del campo de diccionario

El campo de diccionario se identifica mediante el nombre de campo que se lista en la primera columna de la tabla en el archivo de definición de diccionario (.DCT).

A veces es necesario probar un valor colocado en un campo de diccionario desde un proceso anterior o a partir de un conjunto de acción de patrón que se ha ejecutado anteriormente. Esto se puede conseguir especificando el nombre de campo delimitado entre llaves.

Por ejemplo, tiene dos calles denominadas EAST WEST HIGHWAY. En los códigos postales 20100 - 20300, el nombre de la calle es EAST WEST, y en los códigos postales 80000 - 90000, WEST es el nombre de la calle y EAST es la dirección. Si el campo del código postal se llena mediante un proceso anterior y se denomina {ZipCode}, puede utilizar los siguientes conjuntos de acción de patrón:

```
^ | D = "E" | D = "W" | T = "HWY" | [ {ZipCode} >= 20100 & {ZipCode} <= 20300 ]
COPY [1] {HouseNumber}   ; mover número de casa al campo {HouseNumber}
COPY [2] temp   ; concatenar EAST WEST y mover
CONCAT [3] temp   ;   a campo de nombre de calle
COPY temp {StreetName}
COPY_A [4] {StreetSuffixType}   ; mover HWY a campo de tipo de calle

^ | D  = "E" | D = "W" | T = "HWY" | [ {ZipCode} >= 80000 & {ZipCode} <= 90000 ]
COPY [1] {HouseNumber}   ; mover número de casa al campo {HouseNumber}
```

```
COPY_A [2] {StreetPrefixDirectional}   ; mover EAST a dirección
COPY [3] {StreetName}   ; mover WEST a nombre de calle
COPY_A [4] {StreetSuffixType}   ; mover HWY a campo de tipo de calle
```

En este patrón, el operando [{ZipCode} >= 20100 & {ZipCode} <= 20300] indica:

{ZipCode}	El valor en el campo de código postal
>=	es mayor o igual que
20100	el número 20100
&	y
{ZipCode}	el valor en el campo de código postal
<=	es menor o igual que
20300	el número 20300

El operador lógico & se utiliza para conectar dos expresiones condicionales. La condición se coloca en un operando diferente. También puede colocarse en el operando para HWY sería; por ejemplo:

```
^ | D = "E" | D = "W" | T = "HWY"  [ {ZipCode} >= 80000 & {ZipCode} <= 90000 ]
```

Estas dos formas son idénticas en cuanto a su función. Sin embargo, la primera forma es más fácil de leer porque las condiciones se colocan en distintos operandos de patrón.

Cuando las condiciones sólo hacen referencia a contenido de campo de diccionario (y no a cualquier operando de patrón), como en el ejemplo anterior con {ZipCode}, la condición debe seguir todos los operandos de patrón. El ejemplo siguiente no es válido porque el segundo operando no hace referencia a un campo de entrada y un tercer operando (T) sigue a la condición:

```
^ | [{ZipCode} = 80000] | T
```

El formato correcto es:

```
^ | T | [{ZipCode} = 80000]
```

Si se utiliza en una condición un nombre de campo de diccionario que no está definido en el archivo de definición de diccionario, cualquier prueba en su valor siempre devuelve FALSE. Si está probando contenido NULL, la prueba devuelve TRUE; por ejemplo:

```
{ZZ} = ""
```

Este recurso permite el uso de los mismos archivos de acción de patrón generales en proyectos que se proporcionan con determinados campos (en lugar de finalizar el programa con un error de campo no válido).

Literales: constantes de caracteres y variables de usuario

Los literales son las constantes de tipo carácter. Están representados una serie entre comillas.

Para hacer referencia a constantes numéricas se codifica un número. Los números negativos y puntos decimales no están permitidos en constantes numéricas. El operando de patrón siguiente coincide con un número igual a 10000:

```
^ [{} = 10000]
```

El ejemplo siguiente coincide con el texto MAIN:

```
? [ {} = "MAIN" ]
```

Si un operando desconocido (?) se ha especificado, varias palabras se concatenan con una sola palabra. Para coincidir con CHERRY HILL, utilice el siguiente patrón:

```
? [ {} = "CHERRYHILL" ]
```

Un valor nulo o vacío se indica mediante dos comillas consecutivas.

```
[ user_variable = ""]
```

Puede utilizar cualquiera de los operadores relacionales ara las series de caracteres. El ejemplo siguiente tiene coincidencias en todas las cadenas que empiezan por MA o superior, incluyendo MA, MAIN, NAN, PAT, pero no ADAMS, M o LZ:

```
? [ {} > "MA" ]
```

La igualdad (=) se ha probado en la abreviatura para el primer operando si existe uno, y, a continuación, el operando completo. Los demás operadores relacionades se prueba en el operando completo y no la Los operadores relacionales otra prueba en el operando y no la abreviatura.

Puede definir variables a las que asignar valores específicos utilizando las acciones. Puede probar variables para valores específicos dentro de condiciones. Las variables deben denominarse de acuerdo con los convenios siguientes :

- El primer carácter debe ser alfabético.
- El nombre no puede sobrepasar los 32 caracteres.

Después del primer carácter alfabético, puede utilizar cualquier combinación de caracteres alfanuméricos o el carácter de subrayado (_).

Por ejemplo, si establece la variable *postcode* para el código postal, puede hacer una prueba para ver si el código postal es 12345 como se indica a continuación :

```
[postcode = 12345]
```

Este tipo de condición puede ser un operando de patrón por separado o combinado con una clase estándar. Por ejemplo, los dos patrones siguientes producen resultados idénticos:

```
^ | [postcode = 12345]
^ [postcode = 12345]
```

Si una variable de usuario se establece en un valor numérico, su tipo es numérico. Si se establece en un valor literal, su tipo es carácter.

Si una condición sólo hace referencia a variables o campos de diccionario y no a operandos de entrada actual, la condición debe seguir todos los operandos. El ejemplo siguiente no es válido porque el segundo operando no hace referencia a un campo de entrada y un tercer operando le sigue:

```
^ | [postcode = 12345] | T
```

Debe reemplazarlo por:

```
^ | T | [postcode = 12345]
```

Referencia a longitudes de operandos (LEN)

LEN representa la longitud de un operando.

Una función LEN especial se encuentra disponible. Puede utilizar la expresión en una de las tres formas siguientes :

{} LEN	La longitud del operando actual
variable LEN	La longitud del contenido de una variable
{*field-name*} LEN	La longitud del contenido de un campo de diccionario

Si desea buscar un código postal de nueve dígitos de 12345-6789, busque un número de cinco dígitos seguido de un guión y un número de cuatro dígitos, por ejemplo:

```
^ [ {} LEN = 5] | - | ^ [{} LEN = 4]
```

Si los números no coinciden con la longitud, el patrón no coincide.

De forma similar para probar la longitud de una variable, el siguiente patrón coincide si la variable contiene cinco caracteres :

```
? [ temp LEN = 5]
```

Por último, para probar la longitud de un campo de diccionario, utilice el nombre de campo dentro de las llaves:

```
[ {StreetName} LEN = 20 ]
```

Los espacios en blanco finales se pasan por alto en el cálculo de la longitud. Los espacios en blanco iniciales se cuentan. Por ejemplo, si una variable de usuario o un campo de diccionario está establecido en " XY" (dos espacios iniciales), la longitud de cualquiera de los dos es 4.

Referencia a formatos de operandos (PICT)

La imagen define cómo los caracteres numéricos y alfabéticos se formatean.

En algunos casos, debe probar formatos especiales. Puede utilizar la función PICT (imagen). Por ejemplo, los códigos postales de Canadá tienen el formato carácter-número-carácter (espacio) número-carácter-número; por ejemplo, `K1A 3H4`. Puede utilizar la función PICT para representar estas secuencias:

```
@ [{} PICT = "cnc"] | @ [{} PICT = "ncn"]
```

La @ (arroba) coincide con un tipo complejo (caracteres numéricos y alfabéticos mixtos).

cnc	Para carácter-número-carácter
ncn	Para número-carácter-número

Algunos códigos postales británicos utilizan el formato carácter-carácter-número (espacio) número-carácter-carácter; por ejemplo, `AB3 5 NW`.

```
< [{} PICT = "ccn"] | > [{} PICT = "ncc"]
```

La cláusula PICT funciona para los valores de campo de diccionario:

```
[ {ZipCode} PICT = "ccn" ]
```

El operando PICT funciona para las variables de usuario:

```
[ temp PICT = "ncncn" ]
```

Sólo los operadores de igualdad (=) y desigualdad (!=) pueden utilizarse con comparaciones PICT.

Referencia a subseries de operandos

Un formulario especial se proporciona en los patrones para probar una parte de un operando.

Estas partes se llaman subseries. Las formas siguientes son válidas:

{} (beg:end)	Subserie del operando actual
variable (beg:end)	Subserie del contenido de la variable de usuario
field-name (beg:end)	Subserie del contenido del campo de diccionario

(beg:end) especifica el carácter de inicio y fin que debe extraerse. El primer carácter de la serie es 1, el último carácter es -1, etc.

Por ejemplo, las direcciones de estilo alemán tiene el tipo de calle añadido al final del nombre de la calle. Por lo tanto, HESSESTRASSE significa HESSE CALLE. El formato de subserie puede utilizarse para probar estos sufijos. Considere una dirección de entrada de HESSESTRASSE 15. El siguiente patrón coincide con todas las palabras que terminan en STRASSE que van seguidas de un valor numérico:

```
+ [{} (-7:-1) = "STRASSE"]  | ^
```

Del mismo modo, las variables y los campos se pueden probar:

```
[temp(2:4) = "BCD"]
[{StreetName}(1:4) = "FORT"]
```

Al realizar pruebas de subserie en valores de varias señales (?), recuerde que los espacios de separación se eliminan. Para probar MARTIN LUTHER KING, especifique el patrón como se indica a continuación :

```
? [{} (1:12) = "MARTINLUTHERKING"]
```

Serie de valores condicionales

Puede especificar una serie de valores condicionales delimitando las entradas con espacios o comas.

Los ejemplos siguientes son equivalentes y significan que un tipo de calle o abreviatura estandarizada es RD o AV o PL:

```
T = "RD", "AV", "PL"
T = "RD" "AV" "PL"
```

Las series numéricas se pueden representar en la misma forma, excepto sin comillas. Opcionalmente, puede utilizar el operador de igualdad de abreviatura =A= o el operador de valor original =T=, como por ejemplo:

```
T =A= "RD", "AV", "PL"
T =T= "RD", "AV", "PL"
```

Una serie de valores puede probarse en un campo de diccionario en forma parecida:

```
^ | T | [ {StreetName} = "MAIN", "ELM", "COLLEGE" ]
```

El patrón siguiente prueba código de país (especificado como CC) para determinar el proceso que se va a utilizar:

```
; Armenia, Azerbaijan, China, Georgia, Russia, Tajikstan
;
[ {CC}="ARM" ,"AZE" ,"CHN" ,"GEO" ,"RUS" ,"TJK" ]
CALL Area_Format_B
```

En el caso de que pruebe el valor de campo de diccionario en lugar de un operando patrón normal, la prueba debe seguir todos los operandos de patrón incluyendo el fin de campo.

```
^ | ? | T | $ | [ {StreetName} = "MAIN", "ELM", "COLLEGE" ]
```

La prueba `[ {StreetName} = "MAIN", "ELM", "COLLEGE" ]` sigue todos los operandos de patrón, incluido el final del especificador de campo ($).

Tablas de valores condicionales

Puede especificar muchos valores condicionales mediante la creación de un archivo de tabla de valores.

Puede utilizar una tabla de archivos de valores cuando tenga muchos valores que sería difícil incluir en una sentencia condicional, o cuando prefiera actualizar valores en una tabla en lugar de modificar el archivo de acción de patrón.

Las tablas se pueden especificar de la forma siguiente:

```
@TABLE_FILE_NAME.TBL
```

Por ejemplo, si desea probar un número para ver si es uno de la una serie de códigos postales. En primer lugar, edite el archivo de acción de patrón para hacer referenica al archivo que lista códigos postales. Un marcador se crea para el archivo en InfoSphere DataStage and QualityStage Designer. A continuación, edite el archivo que lista códigos postales. Asegúrese de que tiene una línea para cada código postal. Como ilustración, este archivo se denomina POSTCODE.TBL y tiene el aspecto siguiente:

```
90016
90034
90072
90043
...
```

Un patrón que coincide con ciudad, estado y código postal podría parecerse a:

```
? | S | ^ = @POSTCODE.TBL
```

Este patrón se concibe para coincidir con los casos con nombre de ciudad, estado y código postal. Si el operando numérico está en la lista, el patrón coincide; de lo contrario, no. LOS ANGELES CA 90016 coincide, pero CHICAGO IL 12345 no porque el código postal no está en la tabla.

El nombre de archivo de tabla puede contener información de vía de acceso completa o relativa, incluyendo una variable de entorno. Si un conjunto de reglas tiene una vía de acceso especificada en una línea de mandatos, se presupone que la vía de acceso para los archivos de valor utilizado en el patrón de archivo para el conjunto de reglas y la vía de acceso no se pueden especificar por segunda vez.

El contenido del campo de diccionario también puede probarse contra las tablas de valores:

```
^ | T | {StreetName} = @STRTNAME.TBL
```

Si el contenido del campo de diccionario no incluye operandos de patrón, la prueba en una tabla de valores debe seguir todos los operandos de patrón, incluyendo operando de fin de campo. El ejemplo siguiente no es válido, ya que un operando de patrón sigue la prueba de tabla:

```
^ | {StreetName} = @STRTNAME.TBL | T
```

Utilización de expresiones aritméticas

Puede incluir expresiones aritméticas como el operando izquierdo de una expresión condicional.

Durante la estandarización, los conjuntos de reglas sólo prueban la salida de expresiones aritméticas. Los conjuntos de reglas no generan la respuesta a las expresiones aritméticas para llenar la salida.

Los operadores aritméticos disponibles son:

+	Suma
–	Resta
*	Multiplicación
/	División
%	Módulo

La aritmética está limitada a una operación por expresión. Los paréntesis no se permiten. La operación de módulo es el resto de una división de enteros. Por ejemplo, x% 2 es cero si el número es divisible por dos. Es uno si el número es impar.

Una expresión aritmética es

left-arithmetic-operand arithmetic-operator right-arithmetic-operand

Operación	**Valor válido**
left-arithmetic-operand	variable-name {*field-name*} {}
arithmetic-operator	+ – * / %
right-arithmetic-operand	variable-name constante

Ejemplos de expresiones aritméticas:

`temp –2`	El valor de temp –2
`{} % 2`	El valor de operando actual del módulo 2

La expresión condicional siguiente se puede utilizar para la coincidencia con casas pares.

```
^ [{} % 2 = 0]
```

Los números pares son divisibles por dos, por lo tanto el módulo de número de casa dos es cero. La expresión aritmética aparece en el lado izquierdo del operador relacional (el signo igual).

La sintaxis siguiente es una expresión condicional para ver si el operando actual dividido por tres es mayor que el contenido de la variable temporal:

```
^ [{} / 3 > temp]
```

Una vez más, observe que las referencias de campo y la expresión aritmética se encuentran a la izquierda del operador relacional. Otros ejemplo son:

```
[ temp * temp2 > temp3 ]
[ {ZipCode} + 4 > 12345]
```

Combinación de expresiones condicionales

Puede combinar expresiones condicionales utilizando operadores lógicos.

&	AND
|	OR

Nota: El operador OR es incluyente, si parte de la sentencia es TRUE, el resultado es TRUE.

Para comprobar las casas de número par mayor que 1000:

```
^ [{} % 2 = 0 & {} > 1000]
```

Para comprobar las casas en el rango de 1000 a 10000:

```
^ [{} >= 1000 & {} <= 10000]
```

Para comprobar las casas menores que 100 o mayores que 1000:

```
^[{} < 100 | {} > 1000]
```

Para comprobar las casas pares y la mitad del valor en temp superior a 50, y con un código postal mayor que 12345:

```
^ [{} % 2 = 0 & temp / 2 > 50 & {ZipCode} > 12345]
```

Los paréntesis no se permiten. Todas las operaciones se ejecutan de izquierda a derecha. Dentro de una condición delimitada por un único corchete, los operadores AND y OR no se pueden mezclar. Un mensaje de error se imprimen si este caso se encuentra. La precedencia del operador se puede obtener utilizando operandos de patrón independientes si es posible. La expresión aritmética ((a | b) & (c | d)) se puede representar mediante:

```
[a | b] | [c | d]
```

Las líneas verticales (|) dentro de los corchetes son operaciones OR lógicas y las líneas verticales fuera de los corchetes son separadores de operando. En el ejemplo anterior, a, b, c y d representan expresiones condicionales.

Las sentencias de acción

Las sentencias de acción ejecutan las reglas de estandarización que son las acciones asociadas con el patrón de señales en el archivo de acción de patrón.

Cualquier acción que hace referencia a un campo que no se ha definido en el archivo de definición de diccionario se pasa por alto, y un mensaje de aviso se añade al registro de IBM InfoSphere DataStage and QualityStage Director.

Copia de información

La acción COPY copia la información de un origen a un destino.

El formato es:

```
COPY source target
```

source puede ser cualquiera de los elementos siguientes:

Tipo	Descripción
operando	Un operando de patrón ([1], [2], ...)
operando de subserie	Una subserie de operando de patrón
operando mixto	Subconjunto numérico o de caracteres inicial
variable de usuario	Una variable definida por usuario
nombre del campo	Una referencia clave ({StreetName}, ...)
literal	Un literal de serie entre comillas ("SAINT")
constante	Un valor numérico

El *target* puede ser:

Tipo	Descripción
nombre del campo	Un campo de diccionario ({StreetName}, ...)
variable de usuario	Una variable definida por usuario

Por ejemplo, un patrón de dirección de los Estados Unidos que coincide con 123 N MAPLE AVE es:

```
^ | D | + | T
```

Esto se consigue mediante el conjunto de acciones de patrón siguiente:

```
^ | D | + | T
COPY [1] {HouseNumber}
COPY [2] {StreetPrefixDirectional}
COPY [3] {StreetName}
COPY [4] {StreetSuffixType}
EXIT
```

Se producen las operaciones siguientes:

- El operando de número [1] se mueve al campo de número de casa {HouseNumber}
- El operando de clase D (dirección) se mueve al campo dirección de prefijo {StreetPrefixDirectional}
- El operando alfabético desconocido se mueve al campo de nombre de calle {StreetName}
- El tipo de calle (clase T) se mueve al campo {StreetSuffixType}.

Copia de subseries

Una subserie de un operando se copia utilizando el formato de operando de subserie.

La forma más sencilla de la acción COPY es copiar operando en un valor de campo de diccionario. Por ejemplo:

```
COPY [2] {StreetName}
```

El operando de subserie sólo funciona en operandos o variables de usuario estándar. El formato es:

```
COPY source(b:e) target
```

b es la columna inicial de la serie y e es la columna final. El ejemplo siguiente copia el primer carácter (1:1) del operando 2 al campo de nombre de calle :

```
COPY [2](1:1) {StreetName}
```

El ejemplo siguiente copia los caracteres del segundo al cuarto del contenido de la variable temp al campo {StreetName} :

```
COPY temp(2:4) {StreetName}
```

Puede utilizar un uno negativo (-1) para indicar el último carácter. Un dos negativo (-2) indica el último carácter, etc. El ejemplo siguiente copia los tres últimos caracteres del operando 2 al campo de nombre de calle :

```
COPY [2](-3:-1) {StreetName}
```

Cipa de caracteres iniciales y finales

Al gestionar señales de clase alfabéticas, númericas o alfanuméricas iniciales, puede aislar las subseries según el tipo de carácter.

Los cuatro especificadores de operando mixtos posibles son:

(n)	Todos los caracteres numéricos iniciales
(–n)	Todos los caracteres numéricos finales
(c)	Todos los caracteres alfabéticos iniciales
(–c)	Todos los caracteres alfabéticos finales

Estos especificadores se pueden utilizar para operandos estándar o variables de usuario.

Por ejemplo, con la dirección 123A MAPLE AVE, desea que los números del 123 ser reconozcan como el número de casa y la letra A como un sufijo de número de casa. Puede lograr esto con el patrón siguiente:

```
> | ? | T
COPY [1](n) {HouseNumber}
COPY [1](-c) {HouseNumberSuffix}
COPY [2] {StreetName}
COPY_A [3] {StreetSuffixType}
EXIT
```

Tenga en cuenta que el primer operando > es la clase adecuada (numérico inicial). Estos especificadores iniciales y finales se utilizan principalmente con los operadores > OR <. Sin embargo, los especificadores iniciales y finales se pueden utilizar para separar las variables de usuario también. En el ejemplo siguiente, XYZ se copia en StreetName:

```
COPY "456XYZ" tmp2
COPY tmp2(-c) {StreetName}
```

Copia de variables de usuario

El tipo de una variable de usuario de destino está determinado por el tipo de origen.

Una variable de usuario puede ser el destino y el origen de un **COPY**. A continuación se muestran algunos ejemplos:

Ejemplos de copia	Descripción
`COPY [1] temp`	El operando 1 se copia en una variable denominada *temp*.
`COPY "SAINT" temp`	El literal "SAN" se copia en la variable *temp*.
`COPY temp1 temp2`	El contenido de la variable *temp1* se copia en *temp2*.
`COPY temp1(1:3) temp2`	Los primeros tres caracteres de *temp1* se copian en *temp2*.

Las variables de usuario pueden constar de 1 a 32 caracteres donde el primer carácter es alfabético y los demás caracteres son alfabéticos, numéricos o un carácter de subrayado (_).

Copia de columnas de diccionario

Las columnas de diccionario se pueden copiar a otras columnas de diccionario o a variables de usuario.

El ejemplo siguiente muestra las columnas de diccionario que se copian en otras columnas de diccionario o variables de usuario:

```
COPY {HouseNumber} {HC}
COPY {HouseNumber} temp
```

Copia de abreviaturas estandarizadas

La acción **COPY_A** copia la abreviatura estandarizada para un operando a un destino. Mientras que la acción **COPY** copia la entrada a un destino.

Las abreviaturas estandarizadas están codificadas para las entradas en las clasificaciones para el conjunto de reglas. No están disponibles para las clases predeterminadas, como por ejemplo un número, un carácter alfabético desconocido, etc.

Puede utilizar la acción **COPY_A** para copiar la abreviatura de un operando a la columna de diccionario o una variable de usuario. A continuación se muestra un ejemplo:

```
^ | ? | T
COPY [1] {HouseNumber}
COPY [2] {StreetName}
COPY_A [3] {StreetSuffixType}
```

La tercera línea de la abreviatura de operando tres a la columna de tipo de calle. Del mismo modo, el ejemplo siguiente copia la abreviación estándar del operando tres a la variable denominada temp:

```
COPY_A [3] temp
```

Las abreviaturas están limitadas a un máximo de 25 caracteres.

La acción **COPY_A** copia la abreviatura estandarizada al campo de diccionario en lugar de la señal original. **COPY_A** puede incluir un rango de subserie, en cuyo caso la subserie hace referencia a la abreviatura estándar y no la señal original, como en **COPY**.

Copia con espacios

Puede conservar espacios entre las palabras mediante la acción **COPY_S**.

Cuando utilice COPY para copiar un operando alfabético (?) o un rango de señales (**) para una columna de diccionario o una variable de usuario, las palabras individuales se concatenan entre sí.

COPY_S requiere un operando como origen y una columna de diccionario o una variable de usuario como destino. Por ejemplo, con la serie de entrada siguiente:

```
123 OLD CHERRY HILL RD
```

Una acción **COPY** estándar produce OLDCHERRYHILL, pero en el patrón siguiente, **COPY_S** se puede utilizar como se muestra a continuación:

```
^ | ? | T
COPY [1] {HouseNumber}
COPY_S [2] {StreetName}
COPY_A [3] {StreetSuffixType}
```

La columna {StreetName} contiene: OLD CHERRY HILL.

Si utiliza el operando de coincidencia universal, se copian todas las señales en el rango especificado. Por ejemplo, considere la posibilidad de eliminar comentarios entre paréntesis a una columna denominada {AdditionalInformation}. Si tiene la dirección de entrada siguiente:

```
123 MAIN ST (CORNER OF 5TH ST) APARTMENT 6
```

Puede utilizar el patrón siguiente para mover CORNER OF 5TH ST a la columna {AdditionalInformation}. La segunda acción mueve la misma información a la variable de usuario temp:

```
\( | ** | \)
COPY_S [2] {AdditionalInformation}
COPY_S [2] temp
```

Sólo cuando copie el contenido de un operando **COPY** eliminará espacios; por lo tanto, la restricción de que el origen de una acción **COPY_S** sólo pueda ser un operando. Para todas las otras fuentes (literales, columnas formateadas, y las variables de usuario), **COPY** conserva los espacios.

Copia de la señal más cercana

La acción **COPY_C** copia los valores de entrada corregidos tal como coinciden con las entradas en las clasificaciones (.CLS). Cuando una señal tiene un umbral de incertidumbre y se encuentra en las clasificaciones, puede copiar una versión corregida de la entrada en lugar del valor de abreviatura.

Al crear coincidencias en circunstancias inciertas para entradas en las clasificaciones, es recomendable que utilice la acción **COPY_C** para que todas la señal se escriba corretamente en lugar de copiar una abreviatura.

Por ejemplo, si tiene la tabla de nombres de estado con una entrada como:

```
MASSACHUSETTS   MA   S   800.0
```

Si Massachusetts se escribe incorrectamente en un registro de entrada (por ejemplo, Masssachusettts), copie la ortografía correcta en la columna de diccionario. La acción siguiente coloca la señal completa escrita correctamente MASSACHUSETTS en la columna adecuada:

```
COPY_C [operand-number] {column name}
```

En el caso de COPY_C, el origen sólo puede ser un operando porque COPY_C utiliza la señal más cercana de las clasificaciones.

Copia de iniciales

La acción **COPY_I** copia el carácter inicial (el primer carácter de las señales relevantes) desde un origen a la columna de diccionario en lugar del valor completo.

La columna de diccionario puede ser un campo de diccionario o una variable de usuario. Puede utilizar COPY_I dentro de un conjunto de acciones patrón o como una acción **POST**.

El valor MAPLE coloca la M en {NameAcronym}, si utiliza **COPY_I** de la manera siguiente :

```
?
COPY_I [1] {NameAcronym}
```

Para una serie alfabética de varias señales como John Henry Smith, el valor de salida depende del destino. Si el destino es una variable de usuario, el valor de salida de la acción **COPY_I** es JAL.

Si utiliza **COPY_I** como una acción **POST**, el origen debe ser una columna de diccionario y el destino debe ser una columna de diccionario. Generalmente utilice **COPY_I** para facilitar la coincidencia de matriz.

Por ejemplo, el nombre de empresa INTERNATIONAL BUSINESS MACHINES se distribuye en columnas de diccionario C1 a C5 (de tal manera que contiene C1 contiene INTERNATIONAL, C2 BUSINESS, C3 MACHINES y C4 y C5 están en blanco). El siguiente conjunto de acciones **POST** pone el valor de IBM en la columna CI de diccionario de iniciales de la empresa.

```
\POST_START
COPY_I {C1} {CI}
CONCAT_I {C2} {CI}
CONCAT_I {C3} {CI}
CONCAT_I {C4} {CI}
CONCAT_I {C5} {CI}
\POST_END
```

Cuando se utiliza como una acción POST, **COPY_I** tiene sólo el primer carácter de la columna de origen. Por ejemplo, si, en el ejemplo anterior, C1 INTERNATIONAL DIVISION, el resultado es todavía IBM.

Referencia a campos de diccionario desde otro conjunto de reglas

Puede utilizar los campos de diccionario de un conjunto de reglas en otro conjunto de reglas. Es posible que desee hacer referencia a otros conjuntos de normas para crear un caso de prueba que le ayude a determinar qué patrones utilizar.

La sintaxis de referencia es la siguiente: {*<DictionaryFieldName>* DE *<RuleSetName>*}.

En la etapa de estandarización, puede comprobar las condiciones y acciones de los campos de diccionario de un conjunto de reglas anteriormente procesado y determinar si un conjunto diferente de acciones es necesario para el mismo patrón dentro del conjunto de reglas actual. Debe asegurarse de que tanto el conjunto de reglas y el conjunto de reglas de referencia se especifican en la misma etapa de estandarización. Asegúrese de que el conjunto de reglas de referencia precede al conjunto de reglas de referencia en la lista de propiedades de la etapa de conjuntos de reglas. Por ejemplo, si USADDR hace referencia a USAREA, el conjunto de reglas USAREA debe continuar con USADDR en la lista de conjuntos de reglas en la ventana Propiedades de la etapa de estandarización.

Los dominios, área y dirección de los dos conjuntos de reglas pueden proporcionar más ejemplos. La dirección se procesa de forma independiente del área. En este ejemplo, debe comprobar si el conjunto de reglas CAAREA genera Quebec como la provincia asociada. Si Quebec es la provincia canadiense, CAADDR deberá manejar la dirección de forma diferente que para otras provincias canadienses.

Puede hacer referencia a los campos de diccionario CAAREA en el conjunto de reglas CAADDR. En la etapa de estandarización, asegúrese de que CAAREA se define y lista antes de CAADDR.

En el archivo CAADDR.PAT, el patrón realiza comprobaciones para ver el contenido del campo de diccionario, `ProvinceAbbreviation of CAAREA` es `QC`. Los patrones siguientes muestran cómo puede crear la referencia en CAADDR:

```
^ | T | ? | $  [ {ProvinceAbbreviation OF CAAREA} ="QC" ]
COPY [1] {CivicNumber}
COPY_A [2] {StreetSuffixType}
COPY_S [3] {StreetName}
RETYPE [1] 0
RETYPE [2] 0
RETYPE [3] 0
RETURN
```

El patrón anterior especifica que, en la provincia de Quebec, el tipo de calle (como se define en CAADDR.CLS y representado por T), se desplaza a StreetSuffixType, para facilitar la comparación.

A continuación, para todas los demás provincias canadienses, el tipo de calle se pasa a StreetPrefixType.

```
^ | T | ? | $
COPY [1] {CivicNumber}
COPY_A [2] {StreetPrefixType}
COPY_S [3] {StreetName}
RETYPE [1] 0
RETYPE [2] 0
RETYPE [3] 0
RETURN
```

Una dirección postal canadiense podría ser: 498 Rue Burdeos. La salida de los conjuntos de reglas sería la siguiente:

CAAREA	CivicNumber	StreetPrefixType	StreetName	StreetSuffixType
Quebec	498		Bordeaux	Rue
Otras provincias canadienses	498	Rue	Bordeaux	

Desplazamiento de información

Puede utilizar **MOVE** para mover una variable de usuario o una columna de diccionario a una columna del diccionario.

La acción **MOVE** copia información, de origen a destino y, a continuación, borra el valor del origen. Los únicos orígenes válidos para la acción **MOVE** es una variable de usuario o una columna de diccionario. No puede utilizar la acción MOVE con operando. Ejemplos de la acción **MOVE**:

```
MOVE Country_Code {ISOCountryCode}
```

y

```
MOVE {HouseNumber} {HouseNumberSuffix}
```

Concatenación de información

La acción **CONCAT** y la acción **PREFIX** proporcionan dos acciones para concatenar información en un conjunto de reglas.

Acción CONCAT

CONCAT concatena información a una variable de usuario o una columna de diccionario. El origen puede ser un operando, un literal o una variable de usuario. Por ejemplo, las fracciones, como ½ y ¼, se pueden copiar en una sola columna en el diccionario mediante el conjunto de acciones de patrón siguiente:

```
^ | / | ^
COPY [1] temp
CONCAT [2] temp
CONCAT [3] temp
COPY temp {Fractions}
```

La columna {Fracciones} contiene toda la fracción (½).

Las variables de usuario se encuentra en la mayoría de casos con la acción **CONCAT** para formar un campo. Si desea copiar las dos direcciones con espacios (por ejemplo: EAST WEST) en una sola columna de diccionario, cree las acciones siguientes:

```
D =T= "EAST"| D =T= "WEST" | T
COPY [1] temp
CONCAT " " temp
CONCAT [2] temp
COPY temp {StreetName}
```

Este patrón prueba las direcciones específicas de EAST y WEST. El primer operando se copia en la variable de usuario temporal. El contenido de temp es ahora EAST. La línea siguiente concatena un espacio para la variable temporal. La segunda **CONCAT** añade WEST a la variable temporal. La variable contiene ahora EAST WEST. A continuación, el contenido de temp se copia en el campo de nombre de la calle.

Nota: Un literal con un solo espacio se concatena con la variable. Los mismos resultados no se pueden obtener concatenando directamente en la columna del diccionario. Las columnas de diccionario son nulas en el momento de la inicialización. La adición de un espacio no cambia el contenido de la columna y no altera acciones futuras.

CONCAT permite rangos de subserie. Por ejemplo:

```
CONCAT [1](3:-2) {StreetName}
```

Desde la posición 3 de la segunda a la última posición del primer operando se concatena con la columna de nombre de calle de la columna del diccionario.

CONCAT_A concatena la abreviatura estándar en lugar de los datos de la señal original. El origen sólo puede ser un operando. **CONCAT_A** permite rangos de subserie. Sin embargo, la subserie hace referencia a la abreviatura estándar y no la señal original.

CONCAT_I concatena las iniciales en lugar de los datos de la señal original. Puede utilizar **CONCAT_I** como una acción **POST**, donde el origen debe ser una columna de diccionario y el destino debe ser una columna de diccionario.

CONCAT_I, cuando no se utiliza como una acción **POST**, permite rangos de subserie con la subserie que hace referencia a las iniciales y no la señal original. En la mayoría de los casos, hay una sola inicial, pero para series de varias señales, como por ejemplo John Henry Smith, las iniciales son JAL, y otros rangos de subserie distintos de (1:1) tienen sentido.

Más información sobre la acción CONCAT y los espacios: Cuando el origen es una variable de usuario, la acción CONCAT conserva los espacios dentro de la señal. Cuando el origen es un operando, la acción CONCAT no preserva espacios dentro de la señal.

CONCAT no proporciona la posibilidad de conservar los espacios entre las señales que coinciden hasta ? o ** en una sentencia de patrón (como por ejemplo la acción **COPY_S**). Para conservar los espacios entre las señales, debe utilizar la acción **COPY_S** para copiar las señales a una variable de usuario y añadir prefijo o concatenar dicha variable de usuario. Para seleccionar texto de atención en una línea de entrada, consulte los ejemplos siguientes:

Tabla 2. Ejemplos de cómo copiar señales en variables de usuario

Líneas dentro de un conjunto de acciones de patrón	**Descripción**
+="SEE" | **	`; tomar "SEE JOHN DOE"`
COPY [1] temp	`; colocar "SEE" en la variable de usuario` *temp*
CONCAT " " temp	`; añadir un espacio al final de la variable de usuario` *temp*
COPY_S [2] temp2	`; colocar "JOHN DOE" (espacio incluido) en temp2`
CONCAT temp2 temp	`; concatenar temp2 en temp`
COPY temp {AdditionalNameInformation}	`; colocar "SEE JOHN DOE" en la columna AdditionalNameInformation`

Acción PREFIX

La acción **PREFIX** añade datos al principio de una serie. El origen de **PREFIX** puede ser un operando, un literal o una variable de usuario. El destino puede ser una variable de usuario o una columna de diccionario.

```
COPY "CHARLES" temp
PREFIX "SAINT" temp
```

En el ejemplo anterior, la variable *temp* contiene SAINTCHARLES.

PREFIX permite rangos de subserie. Por ejemplo, en la muestra siguiente:

```
PREFIX [1](3:-2) {StreetName}
```

En posición 3 a las posiciones segunda y última del primer operado se utiliza el prefijo para la columna de nombre de calle.

PREFIX_A se utiliza como prefijo de la abreviatura estándar en lugar de los datos de la señal original. El origen sólo puede ser un operando. **PREFIX_A** permite rangos de subserie; sin embargo, la subserie hace referencia a la abreviatura estándar y no la señal original.

PREFIX_I se utiliza como prefijo para las iniciales en lugar de los datos de señal original. Puede utilizar **PREFIX_I** como una acción **POST**, donde el origen debe ser una columna de diccionario y el destino debe ser una columna de diccionario.

PREFIX_I, cuando no se utiliza como una acción **POST**, permite rangos de subserie con la subserie que hace referencia a las iniciales y no la señal original. En la mayoría de los casos, hay una sola inicial, pero para series de varias señales, como por ejemplo John Henry Smith, las iniciales son JAL, y otros rangos de subserie distintos de (1:1) tienen sentido.

Más información sobre la acción PREFIX y los espacios: Cuando el origen es una variable de usuario, la acción PREFIX conserva los espacios dentro de la señal. Cuando el origen es un operando, la acción PREFIX no preserva espacios dentro de la señal.

PREFIX no le permite conservar los espacios entre señales de coincidencia en ? o ** en una sentencia de patrón (como por ejemplo la acción **COPY_S**). Para preservar espacios entre las señales, debe utilizar **COPY_S** para copiar las señales de una variable de usuario y el prefijo o concatenar esa variable de usuario. Consulte el ejemplo de conjunto de acciones de patrón precedente.

Conversión de información

Puede utilizar la acción CONVERT para convertir los datos de acuerdo con una tabla de búsqueda o un literal que suministre.

Puede realizar las acciones siguientes:

CONVERT
: Cambia señales.

CONVERT_S
: Concatena un sufijo en señales especificadas.

CONVERT_P
: Concatena el primer prefijo en una señal que coincide con un valor de una tabla de búsqueda en señales especificadas.

CONVERT_PL
: Concatena el prefijo más largo en una señal que coincide con un valor de una tabla de búsqueda en señales especificadas.

CONVERT_R
: Ejecuta señales a través del proceso de señalización de nuevo cuando se implementan cambios en las señales.

Puede utilizar las siguientes acciones para convertir caracteres de idiomas no originados en el latín:

TRANS_KH
Convierte caracteres Katakana en caracteres Hiragana.

TRANS_HK
Convierte caracteres Hiragana en caracteres Katakana.

TRANS_WN
Convierte caracteres de anchura total en caracteres de anchura media.

TRANS_NW
Convierte caracteres de anchura media en caracteres de anchura total.

Conversión de códigos de lugar

Puede utilizar la conversión con registros de entrada que utilizan códigos numéricos para nombres de lugares.

Los códigos se convierte en nombres de lugar reales. Primero debe crear un archivo de tabla con dos columnas. La primera columna es el valor de entrada y la segunda columna es el valor de sustitución. Por ejemplo, el archivo CODES.TBL contiene:

```
001 "SILVER SPRING"
002 BURTONSVILLE   800.0
003 LAUREL
...
```

Varias palabras deben ir entre comillas (""). El segundo operando puede ir seguido de ponderaciones opcionales en el ejemplo anterior para indicar que se pueden llegar a utilizar comparaciones de incertidumbre. La rutina de comparación de serie se utiliza en BURTONSVILLE, y cualquier puntuación de 800 o mayor es aceptable. El patrón siguiente convierte señales de acuerdo a la tabla anterior:

```
&
CONVERT [1] @CODES.TBL TKN
```

Las señales permanecen convertidas para todos los patrones que siguen, como si el código se cambiara permanentemente a texto.

Los archivos convertidos no deben contener valores de entrada duplicados (primera señal o primer conjunto de señales entre comillas). Si se detectan entradas duplicadas, la etapa Standardize emite un mensaje de error y se detiene.

Conversión temporal

Con la acción **CONVERT**, puede especificar TEMP para aplicar una conversión a sólo el conjunto de acciones actual.

La modalidad TEMP es una conversión temporal. El ejemplo siguiente convierte el sufijo del primer operando de acuerdo con las entradas en la tabla SUFFIX.TBL.

```
CONVERT_S [1] @SUFFIX.TBL TEMP
```

Si tiene un valor de operando de HESSESTRASSE y una entrada de tabla en SUFFIX.TBL de:

STRASSE	STRASSE	800.0

El operando [1] se sustituye con el valor:

```
HESSE STRASSE
```

Ahora hay un espacio entre las palabras. Las acciones subsiguientes en este conjunto de acción de patrón funcionan como se esperaba. Por ejemplo, COPY_S copia las dos palabras HESSE STRASSE al destino. COPY, CONCAT y PREFIX copian la serie sin espacios. Por ejemplo, si la entrada de tabla es:

STRASSE	STR	800.0

El resultado de la conversión es HESSE STR. **COPY_S** conserva ambas palabras, pero **COPY** copia HESSESTR como una sola palabra. El origen de una acción CONVERT_P, CONVERT_PL o CONVERT_Spuede ser un operando (como en el ejemplo, un campo de diccionario o una variable de usuario con resultados equivalentes.

Conversión permanente

La modalidad de TKN proporciona conversión permanente de una señal.

Generalmente, cuando se está realizando una conversión permanente, se especifica un argumento de nueva especificación que se aplica al sufijo con **CONVERT_S** o el prefijo con **CONVERT_P** o **CONVERT_PL**. Por ejemplo, suponga que está utilizando la siguiente sentencia **CONVERT_S** :

```
CONVERT_S [1] @SUFFIX.TBL TKN T
```

También tiene un valor de operando de HESSESTRASSE y una entrada de tabla en SUFFIX.TBL de:

STRASSE	STRASSE	800.0

HESSE retiene la clase ? porque no ha especificado un quinto argumento para volver a especificar el cuerpo o la raíz de la palabra, y a STRASSE se le ha dado el tipo para el sufijo, como por ejemplo T, para el tipo de calle. Para realizar acciones adicionales sobre estas dos señales, necesita un patrón de:

```
? | T
```

Si no se proporciona ninguna clase de nueva especificación, ambas señales conservaran la clase original ?.

Es recomendable que vuelva a especificar el sufijo y el prefijo del cuerpo. Al comprobar los espacios eliminados, una señal como APT234 puede producirse. En este caso, la señal se ha encontrado con una clase de < (carácter alfabético inicial) y un cuarto argumento opcional puede volver a especificar el prefijo APT a U para multiunit y un quinto argumento opcional puede volver a especificar el cuerpo 234 en ^ para el valor numérico. En el ejemplo siguiente, la tabla PREFIX.TBL contiene una entrada APT:

```
CONVERT_P [1] @PREFIX.TBL TKN U ^
```

Si desea volver a especificar sólo el cuerpo, debe especificar un cuarto argumento ficticio que repita la clase original.

Conversión de operandos de varias señales

Si está convirtiendo operandos de varias señales que coinciden con los patrones ** o ?, el formato de la tabla de conversión depende de si el tercer argumento para CONVERT es TKN o TEMP.

Si el tercer argumento es TKN, cada señal se convierte por separado.

Por lo tanto, para convertir SOLANO BEACH a MALIBU SHORES, la tabla de conversión debe tener las dos líneas siguientes :

SOLANO	MALIBU
BEACH	SHORES

Esto puede producir efectos secundarios no deseados, pues cualquier ocurrencia de SOLANO se convierte a MALIBU y cualquier ocurrencia de BEACH se convierte a SHORES.

Para evitar esta situación, la opción *TEMP* de **CONVERT** debe ser utilizada. Las señales combinadas se tratan como una serie individual sin espacios. Por lo tanto, SOLANOBEACH se convierte en la representación de un patrón ? que contiene las señales SOLANO y BEACH. La siguiente entrada en la tabla **CONVERTIR** realiza el cambio apropiado:

SOLANOBEACH	"MALIBU SHORES"

En esta tabla de conversión, no puede haber espacios que separen las señales concatenadas originales. Al copiar el valor convertido en un campo de diccionario, **COPY** no conserva los espacios. Por lo tanto, utilice **COPY_S** si necesita conservar los espacios.

Asignación de valores fijos

Puede utilizar **CONVERTIR** para asignar un valor fijo a un operando o columna de diccionario.

Esto se consigue:

```
CONVERT operand literal TEMP | TKN
CONVERT dictionary-field literal
```

Por ejemplo, para asignar un nombre de ciudad de LOS ANGELES a una columna del diccionario, puede utilizar cualquiera de las acciones siguientes:

```
COPY "LOS ANGELES" {CT}
CONVERT {CT} "LOS ANGELES"
```

Más importante, puede utilizarse para convertir un operando en un valor fijo:

```
CONVERT [1] "LOS ANGELES" TKN
```

TKN hace que el cambio sea permanente para todas las acciones que impliquen a este registro, y TEMP hace que el cambio sea temporal para el conjunto actual de las acciones.

Una clase opcional puede seguir a TKN. Puesto que la conversión a un valor literal siempre es satisfactoria, siempre se lleva a cabo la nueva especificación.

Conversión de prefijos y sufijos

Cuando una única señal puede estar compuesta por dos entidades diferentes, una acción de **CONVERTIR**-type puede utilizarse para separar y estandarizar ambas partes.

Un ejemplo de esto es en direcciones alemanas, donde el sufijo STRASSE puede concatenarse en el nombre adecuado de la calle, como HESSESTRASSE.

Si una lista de direcciones estadounidense tiene una tasa de error importante, es posible que tenga que buscar ocurrencias de espacios eliminados como en MAINSTREET. Para manejar casos como estos, puede utilizar la acción **CONVERT_P** o **CONVERT_PL** para examinar la señal para un prefijo y **CONVERT_S** para un sufijo.

Al igual que **CONVERT_P**, la acción **CONVERT_PL** examina la señal para un prefijo. Sin embargo, **CONVERT_P** adopta el primer prefijo que coincida con un valor en la tabla de búsqueda y **CONVERT_PL** adopta el prefijo más largo que coincida.

Por ejemplo, suponga que una tabla de búsqueda contiene entradas para NORTH y NORTHWEST. Para la señal NORTHWESTPOINT, la acción **CONVERT_P** tiene el prefijo NORTH y la acción **CONVERT_PL** tiene el prefijo NORTHWEST.

CONVERT_P, **CONVERT_PL** y **CONVERT_S** utilizan casi la misma sintaxis que **CONVERT**. La primera diferencia es que debe utilizar una tabla de búsqueda con estas acciones. La segunda diferencia es que tiene un quinto argumento opcional.

```
CONVERT_P source @table_name TKN | TEMP retype1 retype2
CONVERT_PL source @table_name TKN | TEMP retype1 retype2
CONVERT_S source @table_name TKN | TEMP retype1 retype2
```

Argumento	Descripción
origen	Puede ser un operando, un campo de diccionario o una variable de usuario.
retype1	Hace referencia a la clase de señal que desea asignar al prefijo con una acción **CONVERT_P** o **CONVERT_PL** o el sufijo con un **CONVERT_S**. Este argumento es opcional.
retype2	Hace referencia a la clase de señal que ha asignado al resto de la señal después de la conversión, si el origen es un operando.

Conversión con reseñalización

Puede utilizar la acción **CONVERT_R** para forzar las nuevas señales a través del proceso de señalización para que las clases y las abreviaturas sean correctos.

Utilice la acción **CONVERT_R** cuando desee convertir una sola señal en dos o más señales, algunas de los cuales pueden ser de distintas clases de la clase original.

La sintaxis de **CONVERT_R** es más sencilla que la sintaxis de **CONVERTIR** ya que un operando es el destino de la acción y el argumento de TKN se presupone. Utilice una tabla de conversión y el proceso de señalización vuelve a especificar automáticamente:

```
CONVERT_R source @table_name
```

Con **CONVERT_R**, el origen es el operando.

Un ejemplo de utilizar **CONVERT_R** es para los alias de calle. Por ejemplo, el archivo ST_ALIAS.TBL contiene las entradas siguientes:

```
OBT "ORANGE BLOSSOM TRL"
SMF "SANTA MONICA FREEWAY"
WBN "WILSHIRE BLVD NORTH"
WBS "WILSHIRE BLVD SOUTH"
```

El conjunto de acciones de patrón tiene el aspecto siguiente:

```
*+
CONVERT_R [1] @ST_ALIAS.TBL
```

El alias se expande y sus señales individuales se clasifican correctamente. Utilizando el ejemplo anterior, WBN se expande en tres señales con las clases ?, T y D. Los conjuntos de patrones y acciones restantes funcionan como se tenía pensado en la nueva serie de dirección.

Cómo volver a especificar señales:

Una clase de señal opcional puede segior a la palabra clave TKN.

La señal se vuelve a especificar para esta clase si la conversión es satisfactoria. Si no se especifica ninguna clase, la señal mantiene su clase original. El ejemplo siguiente convierte una sola señal alfabética desconocido en función de las entradas de los tres archivos. Si existe una conversión satisfactoria, la señal se vuelve a especificar en C, T o U.

```
+
CONVERT [1] @CITIES.TBL TKN C
CONVERT [1] @TOWNS.TBL TKN T
CONVERT [1] @UNINCORP.TBL TKN U
```

Acciones de conversión para idiomas que no son latinos

Puede utilizar la conversión con registros que contienen caracteres de idiomas que no están basados en el latín. Estas acciones pueden utilizarse en etapas de calidad de datos tales como la etapa de investigación y de estandarización.

Estas acciones de conversión se pueden aplicar a cualquier tipo de carácter. Sin embargo, las acciones son más útiles cuando se aplican a idiomas que son procesados por el señalizador CJK, como los idiomas que se hablan en China, Japón y Corea.

Conversión de caracteres Katakana y hiragana:

Puede convertir caracteres Katakana y hiragana mediante las acciones **TRANS_KH** y **TRANS_HK**.

La acción **TRANS_KH** convierte caracteres Katakana en Hiragana. La acción **TRANS_HK** convierte caracteres Hiragana en Katakana.

Si utiliza estas acciones, especifique el señalizador CJK utilizando el mandato **TOK** en la sección PRAGMA del archivo de acción de patrón. El señalizador CKJ maneja señales en función de los convenios de idiomas que no están basados en el latín.

Aunque la sintaxis de estas acciones es similar a la sintaxis para la acción **CONVERT**, estas acciones no utilizan tablas de búsqueda y no pueden convertir un objeto a un literal. Por ejemplo, puede utilizar la sintaxis siguiente para la acción **TRANS_KH**:

```
TRANS_KH source TKN | TEMP retype1
```

Argumento	Descripción
origen	El objeto que se convierte. El origen puede ser un operando, un campo del diccionario o una variable de usuario.

Argumento	Descripción
retype1	La clase de señal que desea asignar a la señal convertida. Este argumento es opcional.

Conversión de ancho de caracteres:

Puede convertir la anchura de los caracteres utilizando las acciones **TRANS_WN** y **TRANS_NW**.

La acción **TRANS_WN** convierte los caracteres de anchura completa en caracteres de anchura media. La acción **TRANS_NW** convierte los caracteres de anchura media en caracteres de anchura total.

Si utiliza estas acciones, especifique el señalizador CJK utilizando el mandato **TOK** con una variable de entorno local adecuada en la sección PRAGMA del archivo de acción de patrón. El señalizador CKJ maneja señales en función de los convenios de idiomas que no están basados en el latín. Por ejemplo, para utilizar la acción **TRANS_WN** para convertir caracteres japoneses de anchura total en caracteres de anchura media, especifique `TOK ja_JP`.

Aunque la sintaxis de estas acciones es similar a la sintaxis para la acción **CONVERT**, estas acciones no utilizan tablas de búsqueda y no pueden convertir un objeto a un literal. Por ejemplo, puede utilizar la sintaxis siguiente para la acción **TRANS_WN**:

```
TRANS_WN source TKN | TEMP retype1
```

Argumento	Descripción
origen	El objeto que se convierte. El origen puede ser un operando, un campo del diccionario o una variable de usuario.
retype1	La clase de señal que desea asignar a la señal convertida. Este argumento es opcional.

Consideraciones sobre CONVERT

Un origen CONVERT puede ser un operando, una columna de diccionario o una variable de usuario, y si el origen es un operando, se requiere un tercer argumento. Los resultados de las acciones CONVERT_P, CONVERT_PL y CONVERT_S pueden variar en función de las clases de patrón con las que se utilizan las acciones.

Algunas consideraciones a tener en cuenta cuando se utiliza la acción **CONVERT** incluyen:

- El origen de un **CONVERT** puede ser un operando, un campo de diccionario o una variable de usuario. En el ejemplo siguiente, ambas acciones son válidas:

  ```
  CONVERT temp @CODES.TBL
  CONVERT {CT} @CODES.TBL
  ```

- Los nombres de vía de acceso completa pueden codificarse para la especificación de archivo de tabla de conversión:

  ```
  CONVERT {CT} @..\cnvrt\cnvrtfile.dat
  ```

- Si el origen de un **CONVERT** es un operando, se requiere un tercer argumento:

  ```
  CONVERT operand table TKN
  CONVERT operand table TEMP
  ```

TKN se utiliza para realizar el cambio permanente para todos los conjuntos de acción de patrón que siguen a la conversión y que implican este registro.

Si se incluye TEMP, la conversión se aplica sólo al conjunto actual de las acciones.

Si la conversión debe ser aplicada tanto a acciones más abajo del programa y al conjunto actual de las acciones, especifique dos acciones de **CONVERTIR** (una utilizando TKN para otros conjuntos de acciones y conjuntos de reglas, y otro utilizando TEMP para otras acciones en el conjunto de acciones actual).

Los resultados de las acciones de CONVERT_P, CONVERT_PL y CONVERT_S están afectados por las clases de patrón. Por ejemplo, la clase ? puede coincidir con más de una señal. Si SUFFIX.TBL tiene las líneas siguientes:

```
STRASSE STRASSE 800.
AVENUE AVE 800.
```

Si el patrón y las acciones son:

```
^ | ?
COPY [1] {HouseNumber}
CONVERT_S [2] @SUFFIX.TBL TEMP
COPY_S [2] {StreetName}
EXIT
```

La entrada siguiente:

```
123 AAVENUE BB CSTRASSE
```

tiene el siguiente resultado en el número de casa y los campos de nombre de la calle :

```
123 AAVENUEBBC STRASSE
```

Si el patrón y las acciones son:

```
^ | ?
CONVERT_S [2] @SUFFIX.TBL TKN T

^ | + | T | + | + | T
COPY [1] {HouseNumber}
COPY [5] {StreetName}
COPY [6] {StreetSuffixType}
EXIT
```

La entrada siguiente:

```
123 AAVENUE BB CSTRASSE
```

da como resultado:

123	C	STRASSE

Los campos {HouseNumber}, {StreetName} y {StreetSuffixType} son:

```
COPY [1] {HouseNumber}
COPY [2] {StreetName}
COPY [3] {StreetSuffixType}
```

lo que da como resultado el movimiento de:

123	A	AVENUE

Al concatenar caracteres alfabéticos con el patrón ?, la acción CONVERT_P, CONVERT_PL o CONVERT_S opera en la serie concatenada para todas las variables de usuario, los campos de diccionario y los operandos si la modalidad es TEMP.

Para los operandos con una modalidad de TKN, cada señal en la tabla de señales que incluye el operando es examinada individualmente y las nuevas señales correspondientes al prefijo o al sufijo se insertan en la tabla cada vez que el prefijo o sufijo en cuestión se encuentra.

Volver a especificar operandos

La acción **RETYPE** se utiliza para cambiar la clase, el valor de la señal y la abreviatura de un operando.

El formato del operando de nueva especificación es el siguiente conjunto de acción de patrón:

```
RETYPE operand class [variable | literal] [variable | literal]
```

Argumento	Descripción
operando	El número de operando en el patrón
clase	El nuevo valor de clase
variable | literal	El nuevo valor de señal, que puede ser una variable o un literal (opcional)
variable | literal	La nueva abreviación de señal puede ser una variable o un literal (opcional) **Nota:** Si desea cambiar la abreviación de señal pero dejar el valor de señal tal como está, puede copiar el valor de la señal en una variable de usuario y utilizar dicha variable como el tercer argumento.

Un concepto básico a la hora de escribir reglas de estandarización es el filtrado. Puede cambiar frases y cláusulas detectándolas, procesándolas o eliminándolas de la tabla de señales. Puede eliminarlas volviendo a especificarlas en la clase NULL (0). Las clases NULL se ignoran en toda la coincidencia de patrón.

Por ejemplo, si desea procesar apartamentos y impedir que se siga procesando la información de la unidad, puede utilizar el conjunto de acción de patrón siguiente:

```
*U | &
COPY_A [1] {UnitType}
COPY [2] {UnitValue}
RETYPE [1] 0
RETYPE [2] 0
```

La eliminación de la designación de apartamentos convierte la dirección 123 MAIN ST APT 56 a un formato estándar, como por ejemplo 123 MAIN ST. Un apartamento seguido por cualquier señal individual se detecta. Los campos se trasladan al campo de diccionario y se vuelven a especificar enNULL para que no coincidan con los patrones más adelante.

Puede utilizar un tercer operando para sustituir el texto de una señal. Por ejemplo, si desea reconocer los nombres de calle como ST CHARLES y sustituir ST con la palabra SAN, puede utilizar la siguiente regla:

```
*!? | T | + | T
RETYPE [2] ? "SAINT"
```

Este conjunto explora una señal de tipo T (el valor sólo para el tipo T es ST) precedida por cualquier tipo de señal excepto el carácter alfabético desconocido y seguido de una palabra alfabética individual. La acción **RETYPE** cambia el tipo T de operando a un carácter alfabético desconocido (?) y sustituye el texto con SAN. Si los datos de entrada son la dirección siguiente:

```
123 ST CHARLES ST
```

El resultado corrige la ambigüedad de las abreviaturas de SAN y CALLE, tal como se muestra en el siguiente ejemplo:

```
123 SAINT CHARLES ST
```

Esta regla coincide con el patrón ^ | ? | T estándar después de que el ST final se vuelva a especificar en T como se hace en el conjunto de reglas geocodificar. ? se utiliza para la clase de nueva especificación. Esto es importante porque debe considerar SAN para que sea la misma señal que las señales alfabéticas desconocidas vecinas.

El primer operando del patrón (*!?) garantiza que ST no está precedido por un carácter alfabético desconocido. Esto evita que la entrada MAIN JUNK ST se estandarice en MAIN JUNK SAN.

Un cuarto operando está disponible para la acción **RETYPE** para cambiar la abreviación estándar del operando. Si incluye este operando, también debe incluir el argumento de sustitución de valor de señal (tercer argumento). Si no desea sustituir el valor de señal, puede utilizar el valor original como valor de sustitución. Por ejemplo, la dirección ST 123 puede significar SUITE 123.

Si la abreviatura estándar de SUITE es STE, deberá cambiar la abreviatura de señal con el conjunto de acción de patrón siguiente:

```
S | ^
RETYPE [1] U "SUITE" "STE"
```

Esto se interpreta en patrones futuros como SUITE 123 y tiene la abreviatura STE.

En el caso de los argumentos opcionales tercer y cuarto, puede utilizar un rango de subserie. Por ejemplo, con un registro de entrada de 8 143⌂⌂ Ave y el siguiente conjunto de acción de patrón:

```
^ | > | T
COPY [2](n) temp
RETYPE [2] ^ temp(1:2)
```

El 143 se copia a la variable *temp*, el 14 sustituye el contenido de la segunda señal y su clase pasa a ser numérica (^).

RETYPE reclasifica todos los elementos de una clase alfabética posiblemente concatenada (?) o clase universal (**), si no se especifica ningún rango de subcampo (n:m). **RETYPE** reclasifica sólo las señales dentro del rango de subcampo si se especifica un rango. Por ejemplo, si tiene la entrada siguiente:

```
15 AA BB CC DD EE FF RD
```

Los patrones o acciones siguientes tienen el efecto descrito:

^ | ? | T	

RETYPE [2] 0	; Establece AA en AA para clase NULL
^ | 3 | T	
RETYPE [2] 0	; Establece CC a la clase NULL
^ | (2:3) | T	
RETYPE [2] 0	; Establece BB y CC en la clase NULL
^ | -2 | T	
RETYPE [2] 0	; Establece EE en la clase NULL

RETYPE opera de un modo similar a **CONVERT** con un argumento TKN. Los valores para **RETYPE** (clase, valor de señal, abreviatura) no están disponibles en el conjunto de acción de patrón acutal y sólo están disponibles para los siguientes conjuntos de acción de patrón. Por lo tanto, un conjunto de acciones de patrón no genera los resultados buscados y por lo tanto, se señala como error, como en el ejemplo siguiente:

```
...
RETYPE [1] ...
COPY [1] ...
```

Volver a especificar varias señales

Un conjunto de acciones de patrón especial está disponible para volver a especificar varias apariciones de una señal.

El patrón debe tener el formato siguiente seguido de una sentencia **RETYPE** estándar que hace referencia al operando[1]:

```
number*class
```

El *número* debe ser un entero de 0 a MAX_TOKENS (actualmente 40) e indicar el número de apariciones de referencia. Cero significa que se exploran todas las apariciones.

Por ejemplo, si ha procesado los números de casa separados por guiones, como 123-45 Main St., y desea eliminar todos los guiones de las señales que permanecen, utilice el conjunto de acciones de patrón siguiente:

```
0 * -
RETYPE [1] 0
```

De este modo se exploran todos los guiones y se vuelven a especificar en NULL. Si desea volver a especificar dos señales numéricas para un tipo desconocido con un valor de UKN, utilice el siguiente conjunto de acciones de patrón:

```
2 * ^
RETYPE [1] ? "UKN"
```

El patrón completo está restringido a este formato simple cuando se hace referencia a varias apariciones.

Patrones

La sentencia PATTERN genera el patrón para el conjunto de señales activo.

La acción de patrón siguiente muestra la sintaxis:

```
PATTERN Target
```

Argumento	Descripción
Destino	El nombre de una variable de usuario o columna de diccionario donde los patrones devueltos se almacenan.

Por ejemplo:

```
&
PATTERN {InputPattern}
```

En el ejemplo anterior, cuando la acción **PATTERN** se ejecuta, el patrón asociado con todas las señales activas se genera y se almacena en el campo de diccionario {InputPattern}.

Extensiones de conjunto de reglas en archivo de acción de patrón

Puede habilitar las extensiones de conjunto de reglas mediante la actualización del archivo de acción de patrón con referencias a grupos de reglas. El archivo de acción de patrón puede contener sólo una referencia para cada grupo de reglas.

Puede utilizar la sintaxis siguiente:

`PROCESS_RC` *`Rule_Group_Name`* `RC_Return [`*`Rule_ID`*`]`

Argumento	Descripción
Rule_Group_Name	El nombre del grupo de reglas. El nombre sólo puede contener caracteres alfanuméricos y subrayados y debe empezar por un carácter alfanumérico.
Rule_ID	Nombre de una variable. Cuando la acción se ejecuta, el nombre de la regla que se aplica al registro actual se graba en la variable. Si ninguna regla se ha aplicado al registro, la variable está vacía. Puede añadir acciones al lenguaje de acción de patrón que utilizan esta variable. Por ejemplo, puede añadir una acción que añade el ID de regla a una columna de salida para fines de informes. Este argumento es opcional.

Cuando se añade una referencia a un grupo de reglas, seleccione una ubicación en el archivo de acción de patrón en función de cuándo desea que las acciones se ejecuten. Por ejemplo, para aplicar las reglas en un grupo de reglas antes de cualquier otro proceso, añada una referencia al principio de la primera subrutina que se llama.

Valores de código de retorno

Cuando se hace referencia a un grupo de reglas, puede añadir acciones que se basan en el valor del código de retorno para cada registro de entrada. El valor del código de retorno depende de si las reglas en el grupo de normas se aplican a un registro de entrada concreto.

Valor	Descripción
0	No hay reglas en el grupo de normas que se apliquen al registro de entrada.
1	Una regla en el grupo de reglas se aplica al registro de entrada.
2	Una regla en el grupo de reglas se aplica al registro de entrada, pero los registros se pasan a través de la regla. Como resultado, la regla no cambia el registro.

Examples

```
PROCESS_RC Input_Overrides RC_Return
[ RC_Return = "1" ]
; Una regla en el grupo de reglas se aplica al registro de entrada:
; CALL Post_Process SUBROUTINE then EXIT
COPY "CI" {UserOverrideFlag}
CALL Post_Process
EXIT
```

Este ejemplo hace referencia al grupo de reglas Input_Overrides. El ejemplo especifica una acción que ocurre cuando una regla del grupo de reglas se aplica al registro de entrada.

```
PROCESS_RC RuleGroup1 RC_Return RuleID
[ RC_Return = "1" ]
; Una regla en el grupo de reglas se aplica al registro de entrada:
; CALL Post_Process SUBROUTINE then EXIT
COPY RuleID {RuleID}
; Añadir el nombre de la regla que se aplica al registro a la columna de salida
; de RuleID COPY "RuleGroup1" {RuleGroupName}
; Añadir el nombre del grupo de normas para la columna de salida RuleGroupName
CALL Post_Process
EXIT
```

Este ejemplo hace referencia al grupo de reglas RuleGroup1 y utiliza el argumento RuleID. Una acción copia el nombre de cualquier regla en el grupo de reglas que se aplican a un registro de entrada a la columna de salida RuleID. Una acción diferente añade el nombre del grupo de normas para la columna de salida RuleGroupName. Antes de que se ejecuten las acciones, las columnas de salida que hacen referencia deben añadirse al archivo de diccionario.

Alteraciones temporales de usuario para conjuntos de reglas de preprocesador

Puede crear alteraciones temporales para conjuntos de reglas de preprocesador de dominio (PREP) utilizando las opciones de alteración temporal de usuario.

Valores de código de retorno

Las sentencias de alteración temporal deben estar en la tabla de acciones de patrón. Las sentencias permiten alteraciones temporales en el cliente Diseñador.

Sintaxis recomendada:

```
OVERRIDE_P Lookup_Value @Table_Name Return_Code [ CLASS | VALUE ]
```

Argumento	Descripción
Lookup_Value	El nombre de la variable de usuario que contiene un patrón o referencia de texto para el conjunto de señales activo que debe comprobarse para cualquier alteración temporal de usuario aplicable buscando en el *Nombre_tabla* proporcionado.
Table_Name	Nombre de la tabla de alteraciones temporales de usuario que debe utilizarse.
Return_Code	Nombre de una variable de usuario o campo de diccionario donde se almacena el valor de código de retorno.
[*CLASS* | *VALUE*]	*CLASS* indica que *Lookup_Value* contiene una referencia del patrón donde cada señal está representada por su clase de señal. *VALUE* indica que *Lookup_Value* contiene una referencia de texto donde cada señal está representada por su valor de señal.

Valor	Descripción
0 (cero) (valor por defecto)	No se ha ejecutado ninguna alteración temporal de usuario porque no se ha encontrado ninguna coincidencia para el *Lookup_Value* especificado en el *Table_Name* especificado.
1	Se ha ejecutado una alteración temporal de usuario correctamente sin errores.

Por ejemplo:

```
OVERRIDE_P Text @USPREPIP.TBL Return VALUE
OVERRIDE_P Pattern @USPREPIP.TBL Return CLASS
```

El *Lookup_Value* especificado se busca en el *Table_Name* especificado. Si el *Lookup_Value* se encuentra, el conjunto de señales activo es **RETYPE** que utiliza las instrucciones de alteración temporal que se encuentran en la entrada de la tabla coincidente. Un valor de código de retorno siempre se devuelve y se almacena en el *Return_Code* especificado.

El *Lookup_Value* puede contener un patrón o una serie de valores de señal literal, lo que representa el conjunto activo de señales.

Table_Name es una tabla de conversión STAN de dos columnas que utiliza los formatos siguientes:

- Formato de tabla de alteración temporal de patrón (para conjuntos de reglas de preprocesador)

  Columna 1: <[Classification for Token #1][Classification for Token #2]...>

  Columna 2: <[Domain Mask for Token #1][Domain Mask for Token #2]...>

  Por ejemplo (para conjuntos de reglas de preprocesador):

  ```
  N^D+TU^   AAAAAAA
  A++E      NNNN
  A+S^      RRRR
  ```

- Formato de tabla de alteraciones temporales de texto (para conjuntos de reglas de preprocesador)

  Columna 1: <["][Token #1][space][Token #2][space]...["]>

  Columna 2: <[Domain Mask for Token #1][Domain Mask for Token #2]...>

  Por ejemplo (para conjuntos de reglas de preprocesador):

```
"ZQNAMEZQ 456 SOUTH MAIN AVENUE APARTMENT 7"   AAAAAA
"ZQADDRZQ IBM CORPORATION"   NNNN
"ZQADDRZQ LITTLETON MA 01460"   RRRR
```

Para obtener más información sobre conjuntos de reglas de preprocesador de dominio, consulte la publicación *IBM InfoSphere QualityStage Guía del usuario*.

Validación de sintaxis de la tabla de alteraciones temporales

La validación de sintaxis de las tablas de alteración temporal se realiza durante la inicialización de la etapa de estandarización o investigación.

Si se encuentran errores de sintaxis, la ejecución de la etapa se detiene y se graba un mensaje de error en el archivo de registro incluyendo el nombre de tabla de alteración temporal, el contenido de las entradas no válidas y una explicación de la naturaleza de los errores.

Para cada señal activa debería haber una máscara de dominio correspondiente en la instrucción de alteración temporal. La máscara de dominio es la clase que se utilizará para **RETYPE** la señal correspondiente. Los únicos valores válidos para una máscara de dominio son A, N, y R.

Los dos principales errores que hay que comprovar son:

- Máscaras de dominio no válidas (no son A, N ni R)
- Instrucciones de alteración temporal incompletas (no había una instrucción válida para cada señal activa)

Alteraciones temporales de usuario para conjuntos de reglas

Puede crear alteraciones temporales mediante opciones de alteración temporal de usuario dentro de conjuntos de reglas de dominio.

Valores de código de retorno

Las sentencias de alteración temporal deben estar en la tabla de acciones de patrón. Las sentencias permiten alteraciones temporales en el cliente Diseñador.

Puede utilizar la sintaxis siguiente:

```
OVERRIDE_D Lookup_Value @Table_Name Return_Code [ CLASS | VALUE ]
```

Argumento	Descripción
Lookup_Value	El nombre de la variable de usuario que contiene un patrón o referencia de texto para el conjunto de señales activo que debe comprobarse para cualquier alteración temporal de usuario aplicable buscando en el *Nombre_tabla* proporcionado.
Table_Name	Nombre de la tabla de alteraciones temporales de usuario que debe utilizarse.

Argumento	Descripción
Return_Code	Nombre de una variable de usuario o campo de diccionario donde se almacena el valor de código de retorno.
[CLASS | VALUE]	CLASS indica que *Lookup_Value* contiene una referencia del patrón donde cada señal está representada por su clase de señal. VALUE indica que *Lookup_Value* contiene una referencia de texto donde cada señal está representada por su valor de señal.

Valor	Descripción
0 (cero) (valor por defecto)	No se ha ejecutado ninguna alteración temporal de usuario porque no se ha encontrado ninguna coincidencia para el *Lookup_Value* especificado en el *Table_Name* especificado.
1	Se ha ejecutado una alteración temporal de usuario correctamente sin errores.

Ejemplo de una acción de alteración temporal creada por usuario:

```
OVERRIDE_D Text @USNAME.UTO Return VALUE
OVERRIDE_D Pattern @USNAME.UPO Return CLASS
```

El *Lookup_Value* especificado se busca en el *Table_Name* especificado. Si se encuentra el *Lookup_Value*, el conjunto de señales activo se graba en campos de diccionario mediante las instrucciones de alteración temporal que se encuentran en la entrada de la tabla coincidente y, a continuación, se utiliza **RETYPE** para pasar al estado nulo. Todo el contenido existente del campo de diccionario se mantiene durante la ejecución de las instrucciones de alteración temporal. Un valor de código de retorno siempre se devuelve y se almacena en el *Return_Code* especificado.

El *Lookup_Value* puede contener un patrón o una serie de valores de señal literal, lo que representa el conjunto activo de señales.

Table_Name es una tabla de conversión STAN de dos columnas que utiliza los formatos siguientes:

- Formato de tabla de alteración temporal de patrón (para conjuntos de reglas)

  Columna 1: <[Classification for Token #1][Classification for Token #2]...>

  Columna 2: <[DCT Field for Token #1][Data Flag for Token #1] - [DCT Field for Token #2][Data Flag for Token #2] -...>

  Por ejemplo (para conjuntos de reglas):

  ```
  FIF   FirstName1-MiddleName1-PrimaryName1
  ^D+T   HouseNumber1
  -StreetPrefixDirectional2-StreetName1-StreetSuffixType2
  +++   ExceptionData5
  ```

- Formato de tabla de alteración temporal de texto (para conjuntos de reglas)

  Columna 1: <["][Token #1][space][Token #2][space]...["]>

  Columna 2: <[DCT Field for Token #1][Data Flag for Token #1] [DCT Field for Token #2] [Data Flag for Token #2]...>

  Ejemplos de alteraciones temporales de texto (para conjuntos de reglas)

```
"JAMES E HARRIS" FirstName1-MiddleName1-PrimaryName1
"123 N. MAIN STREET" HouseNumber1
-StreetPrefixDirectional2-StreetName1-StreetSuffixType2
"DO NOT MAIL"   ExceptionData5
```

Para obtener más información sobre las categorías de conjuntos de reglas, incluyendo las específicas del dominio, consulte la publicación *IBM InfoSphere QualityStage Guía del usuario*.

Validación de sintaxis de tabla de alteraciones temporales para conjuntos de reglas específicas de dominio

La validación de sintaxis de las tablas de alteración temporal se realiza durante la inicialización de la etapa de estandarización o investigación.

Si se encuentran errores de sintaxis, la ejecución de la etapa se detiene y se graba un mensaje de error en el archivo de registro incluyendo el nombre de tabla de alteración temporal, el contenido de las entradas no válidas y una explicación de la naturaleza de los errores.

Para cada señal activa hay una instrucción correspondiente en la sentencia de alteración temporal. La instrucción consiste en el campo denominado seguido de un distintivo de datos. Un distintivo de datos válido indica las acciones asociadas a realizarse en la señal correspondiente. Cada instrucción de alteración temporal se separa por un guión (-). La única situación en la que hay menos instrucciones de alteración temporal que señales activas es cuando se utiliza un distintivo de datos "mover todas las señales restantes" o "eliminar todas las señales restantes" (valores 5,6,7,8, y 9 en la tabla siguiente.

Los tres principales errores que hay que comprovar son:

- Los distintivos de datos no válidos (no un valor listado en la siguiente tabla)
- Los nombres de campo de diccionario no válidos (que no aparecen en el archivo de diccionario del conjunto de reglas)
- Instrucciones de alteración temporal incompletas (no había una instrucción válida para cada señal activa)

Tabla de distintivos de datos (que se utiliza para conjuntos de reglas específicos del dominio sólo):

Valor	Acciones asociadas
0 (cero)	Descartar la señal actual
1	Anexar un espacio de carácter inicial y, a continuación, anexar el Valor original de la señal correspondiente, al campo de diccionario especificado (no se puede añadir ningún espacio de carácter inicial si el campo de diccionario especificado está vacío)
2	Anexar un espacio de carácter inicial y, a continuación, anexar el valor estándar de la señal correspondiente, al campo de diccionario especificado (no se puede anexar ningún espacio de carácter inicial si el campo de diccionario especificado está vacío, también si no hay ningún valor estándar disponible para la señal correspondiente, utilizar el valor original)

Valor	Acciones asociadas
3	Anexar el valor original de la señal correspondiente, sin anexar un espacio de carácter inicial, al campo de diccionario especificado
4	Anexar un valor estándar de la señal correspondiente sin anexar un espacio de carácter inicial, al campo de diccionario especificado (si no hay ningún valor estándar disponible para la señal correspondiente, utilizar el valor original)
5	Mover todas las señales restantes que utilizan sus valores originales, dejando un espacio de carácter entre cada señal, en el campo de diccionario especificado
6	Mover todas las señales restantes que utilizan sus valores estándar, dejando un espacio de carácter entre cada señal, al campo de diccionario especificado (si no hay disponible ningún valor estándar para ninguna de las señales restantes, utilizar el valor original)
7	Mover todas las señales restantes que utilizan sus valores originales, sin espacio de carácter entre cada señal, en el campo de diccionario especificado
8	Mover todas las señales restantes que utilizan sus valores estándar, sin dejar espacio de carácter entre cada señal, al campo de diccionario especificado (si no hay disponible ningún valor estándar para ninguna de las señales restantes, utilizar el valor original)
9	Descartar todas las señales restantes

Establecimiento de márgenes

Puede controlar qué parte del patrón está disponible estableciendo los márgenes izquierdo y derecho.

Es posible que desee restringir los conjuntos de acciones de patrón a los rangos de señales de entrada. Por ejemplo, si la entrada consta de registros de cuatro líneas, dos líneas de información de dirección y cero, una o dos líneas de nombres de departamento o institución, es posible que desee buscar una línea de dirección con un número, nombre de calle, tipo de calle, procesar sólo dicha línea y volver a especificar las señales en NULL.

Si hizo lo mismo para las líneas de ciudad, estado y código postal, puede estar seguro de que todo el resto lo ocupan los nombres de departamento o institución.

Este tipo de proceso restringido puede realizarse estableciendo los márgenes izquierdo y derecho. Los valores por omisión son: margen izquierdo es la primera señal y el margen derecho es la última. Estos valores se pueden cambiar utilizando

las acciones **SET_L_MARGIN** y **SET_R_MARGIN**. Cada acción tiene una de las dos palabras clave seguida de un número o número dentro de corchetes. Puede utilizar la sintaxis siguiente:

```
SET_L_MARGIN LINE number | OPERAND number
SET_R_MARGIN LINE number | OPERAND number
```

Un ejemplo de restricción de conjuntos de acciones:

```
*^ | D | ?
SET_L_MARGIN OPERAND [3]
SET_R_MARGIN OPERAND [3]
```

En este ejemplo, tanto los márgenes izquierdo y derecho están configurados para el operando tres. Puesto que más de una señal puede contener el conjunto de señales correspondiente al tipo alfabético desconocido ?, el margen izquierdo siempre está establecido en la primera señal del conjunto y el margen derecho siempre se establece en la última señal del conjunto.

Lo mismo puede decirse para el tipo **. Si el operando tres se ha especificado otro tipo que sólo puede hacer referencia a una sola señal, como ^, establecer los márgenes izquierdo y derecho en el operando tres hará que en los próximos conjuntos de acciones de patrón sólo se vea dicha señal individual. Al igual que con cualquier conjunto de acciones de patrón, si el patrón asociado no coincide con el registro de entrada, las acciones se ignoran y los márgenes se dejan sin modificar.

Una vez finalizado el procesamiento en el conjunto de señales especificado por los márgenes izquierdo y derecho, deberá restablecer los márgenes a su valor más ancho para volver a colocar las otras líneas o señales. Utilice las palabras clave **BEGIN_TOKEN** para el margen izquierdo y **END_TOKEN** para el margen derecho:

```
SET_L_MARGIN OPERAND [BEGIN_TOKEN]
SET_R_MARGIN OPERAND [END_TOKEN]
```

Es importante tener en cuenta que si, después de establecer márgenes, debe volver a especificar todas las señales dentro de los márgenes en NULL, no tendrá acceso a ninguna otra señal. Para ejecutar las acciones que restauran los márgenes, debe utilizar un patrón que no requieraencontrar ninguna señal. El método normal es, al principio de un proceso, crear una condición que sea verdadera en cualquier punto de la secuencia patrón-acción. El primer conjunto de acciones de patrón puede ser:

```
&
COPY "TRUE" true
```

En algún momento después de que todas las señales visibles se hayan vuelto a especificar en NULL, puede ejecutar las acciones de restablecimiento de margen utilizando la siguiente secuencia de patrón-acción:

```
[true = "TRUE"]
SET_L_MARGIN OPERAND [BEGIN_TOKEN]
SET_R_MARGIN OPERAND [END_TOKEN]
```

Codificación fonética SOUNDEX

La acción **SOUNDEX** se utiliza para calcular el código **SOUNDEX** de un campo de diccionario y mover los resultados a otro campo de diccionario.

Los códigos **SOUNDEX** son claves fonéticas que son útiles para bloquear códigos en una operación coincidente.

SOUNDEX es una variable de bloqueo excelente porque no es muy discriminatoria y todavía se utiliza para particionar los archivos en un número razonable de subconjuntos. La acción está específicamente diseñado para el idioma inglés, pero quizás pueda ser útil para más idiomas.

La acción **SOUNDEX** tiene el formato siguiente:

```
SOUNDEX source-field target-field
```

Por ejemplo, la acción siguiente calcula **SOUNDEX** del campo de diccionario StreetName y pone el resultado en el campo: campo **StreetNameSOUNDEX**:

```
SOUNDEX {StreetName} {StreetNameSOUNDEX}
```

La acción **RSOUNDEX** (**SOUNDEX** inv.) es la misma que la acción **SOUNDEX**, salvo que el código fonético se genera a partir del último carácter no blanco del campo y procederá al primero. Esto es útil para bloquear los campos donde los caracteres iniciales son erróneos. Un ejemplo de sintaxis de patrón es el siguiente:

```
RSOUNDEX {StreetName} {StreetNameRVSNDX}
```

Las acciones **SOUNDEX** y **RSOUNDEX** se utilizan en la sección de acción **POST**, por lo que se ejecutan después de que se haya completado la coincidencia de patrón del registro.

Las acciones **POST** deben producirse antes de cualquier conjunto de acción de patrón y deben ir precedidas por la línea **POST_START** y seguidas por la línea **POST_END**.

Código NYSIIS

La acción NYSIIS se utiliza para calcular el código NYSIIS de un campo de diccionario y mover los resultados a otro campo de diccionario.

NYSIIS significa New York State Information and Intelligence Systems. NYSIIS es un algoritmo de codificación genética desarrollado después de SOUNDEX. Similar a SOUNDEX, NYSIIS está específicamente diseñado para el idioma inglés, pero quizás pueda ser útil para más idiomas.

La acción NYSIIS tiene el formato siguiente:

```
NYSIIS source-field target-field
```

Por ejemplo, la acción siguiente calcula el NYSIIS del campo MatchFirstName y coloca el resultado de ocho caracteres en el campo de diccionario MatchFirstNameNYSIIS :

```
NYSIIS {MatchFirstName} {MatchFirstNameNYSIIS}
```

La acción de RNYSIIS es NYSIIS inversa. Las acciones de NYSIIS y RNYSIIS se utilizan en la sección de acción **POST**, de modo que se ejecutandespués de que la coincidencia de patrón se haya completado para el registro.

Las acciones **POST** deben producirse antes de cualquier conjunto de acción de patrón y deben ir precedidas por la línea **POST_START** y seguidas por la línea **POST_END**.

Terminación de coincidencia de patrón

La acción **EXIT** se utiliza para salir del programa de coincidencia de patrón del proceso actual.

La acción **EXIT** impide que los conjuntos de acción de patrón se ejecuten; por ejemplo:

```
^ | D | ? | T
COPY [1] {HouseNumber}
COPY_A [2] {StreetPrefixDirectional}
COPY [3] {StreetName}
COPY [4] {StreetSuffixType}
EXIT
```

Si el registro de entrada coincide con el patrón, el proceso actual finaliza después de las acciones **COPY** (y después de cualquier acción **POST**).

Llamadas a subrutinas

Las subrutinas pueden utilizarse para mejorar el tiempo de proceso y controlar el número de conjutos de acciones de patrón que deben codificarse.

Las subrutinas se invocan con la sintaxis:

```
CALL <nombre subrutina>
```

Puesto que los conjuntos de acción de patrón se ejecutan secuencialmente, es más rápido probar un tipo genérico y llamar a una subrutina para procesar ese tipo. Esto se ilustra mejor mediante un ejemplo:

```
*U
CALL UNITS
```

Si se detecta un tipo de unidad (U) en cualquier lugar de la entrada, se llama a la subrutina Units para procesar los números de apartamento. Los nombres de las subrutinas están formados según las mismas normas que los nombres de variables (hasta un máximo de 32 caracteres con el primer carácter alfabético).

Si se llama a una subrutina que no existe, el conjunto de normas terminará anormalmente en tiempo de ejecución.

Devolución de una subrutina

Puede devolver el control de una subrutina al programa principal con una acción **RETURN**.

La acción **RETURN** está disponible para devolver el control de una subrutina al programa principal. Una **RETURN** no es necesaria inmediatamente antes de la sentencia \END_SUB.

Grabación de subrutinas

Una subrutina está delimitada como las acciones **POST**.

Las subrutinas tienen una cabecera y una línea de cola :

\SUB <name>

...

... cuerpo de subrutina

\END_SUB

Puede añadir tantos conjuntos **\SUB** y **\END_SUB** como requiera. El siguiente ejemplo ilustra una organización de archivo de reglas:

```
\POST_START
                                        acciones
POST_END
%1 U
CALL UNITS
%1 R
CALL ROUTES
...
...                                     Patrones y acciones adicionales
...
\SUB UNITS
...
...                                     Patrones y acciones de procesamiento de
                                        apartamento
...
\END_SUB
\SUB ROUTES
...
...                                     Patrones y acciones de procesamiento de ruta
...
\END_SUB
```

Todas las subrutinas están codificadas al final del archivo. El orden de las propias subrutinas no es importante. Las subrutinas contienen los conjuntos de acción de patrón estándar tal como se encuentran en las normas principales.

Las subrutinas se pueden anidar. Es decir, las acciones **CALL** están permitidas dentro de subrutinas.

El control se devuelve al nivel desde el que una rutina se ha llamado, cualquier otra subrutina o el programa principal, cuando se alcanza **\END_SUB** o cuando se ejecuta la acción **RETURN**.

Realización de acciones repetidamente

Utilice la acción **REPEAT** para ejecutar el patrón anterior y su conjunto de acciones asociado de manera repetitiva hasta que el patrón no coincida.

Cuando se utiliza la acción **REPEAT**, la sentencia **REPEAT** debe ser la última sentencia de acción en el conjunto de acciones. La sentencia **REPEAT** es válida después de una sentencia **CALL**. El conjunto de acciones debe incluir una o más acciones que cambian los objetos de la prueba de patrón o resultados de un bucle indefinido.

En el ejemplo siguiente, el conjunto de acciones de patrón cambia los tipos de todos los operandos con la clase A a la clase alpha desconocida (?) después de convertir el valor de entrada al valor estándar ::

```
*A
COPY_A [1] temp
RETYPE [1] ? temp temp
REPEAT
```

En este ejemplo, el conjunto de acciones de patrón llama a la subrutina Parsing_Rules hasta que los datos de variable de usuario están vacíos:

```
[data != ""]
CALL Parsing_Rules
REPEAT
```

Puede ver un ejemplo de la subrutina Parsing_Rules en el conjunto de reglas USNAME. En la vista del repositorio del Cliente del diseñador, expanda la carpeta **Reglas de estandarización**. Expanda la carpeta de dominio USA.

Resumen de orígenes y destinos

Las acciones pueden tener varios orígenes y destinos.

El siguiente es un resumen de los orígenes y destinos permitidos para todas las acciones.

Acción	Origen	Destino
CONCAT	operando operando literal literal variable de usuario variable de usuario	variable de usuario campo de diccionario variable de usuario campo de diccionario campo de diccionario variable de diccionario
CONCAT_A	operando operando	variable de usuario campo de diccionario
CONCAT_I	operando operando literal literal variable de usuario variable de usuario	variable de usuario campo de diccionario variable de usuario campo de diccionario campo de diccionario variable de usuario
CONVERT	campo de diccionario variable de usuario operando	Las sentencias CONVERT convierten el origen. El origen también es el destino.
CONVERT_P|PL|S	campo de diccionario variable de usuario operando	Las sentencias CONVERT convierten el origen. El origen también es el destino.
CONVERT_R	operando	Las sentencias CONVERT convierten el origen. El origen también es el destino.

Acción	Origen	Destino
COPY	operando operando campo con formato campo de diccionario campo de diccionario literal literal variable de usuario variable de usuario	campo de diccionario variable de usuario campo de diccionario campo de diccionario variable de usuario campo de diccionario variable de usuario campo de diccionario variable de usuario
COPY_A	operando operando	variable de usuario campo de diccionario
COPY_C	operando operando	campo de diccionario variable de usuario
COPY_I	operando operando campo con formato campo de diccionario campo de diccionario literal literal variable de usuario variable de usuario	campo de diccionario variable de usuario campo de diccionario campo de diccionario variable de usuario campo de diccionario variable de usuario campo de diccionario variable de usuario
COPY_S	operando operando	campo de diccionario variable de usuario
MOVE	variable de usuario campo de diccionario	campo de diccionario campo de diccionario
NYSIIS	campo de diccionario variable de usuario	campo de diccionario campo de diccionario
PREFIX	operando operando literal literal variable de usuario variable de usuario	variable de usuario campo de diccionario variable de usuario campo de diccionario campo de diccionario variable de usuario
PREFIX_A	operando operando	variable de usuario campo de diccionario
PREFIX_I	operando operando literal literal variable de usuario variable de usuario	variable de usuario campo de diccionario variable de usuario campo de diccionario campo de diccionario variable de usuario
REPEAT	(ninguno)	(ninguno)

Acción	Origen	Destino
RNYSIIS	campo de diccionario variable de usuario	campo de diccionario campo de diccionario
RSOUNDEX	campo de diccionario variable de usuario	campo de diccionario campo de diccionario
SOUNDEX	campo de diccionario variable de usuario	campo de diccionario campo de diccionario
TRANS_HK|KH	campo de diccionario variable de usuario operando	Estas sentencias convierten el origen. El origen también es el destino.
TRANS_NW|WN	campo de diccionario variable de usuario operando	Estas sentencias convierten el origen. El origen también es el destino.

Las acciones de establecimiento de margen no tienen origen o destino, sino que toman un primer argumento de palabra clave OPERAND/LINE y un segundo argumento de número de operando (entre corchetes []) o número de línea, por ejemplo:

```
SET_L_MARGIN LINE 5
SET_R_MARGIN OPERAND [3]
```

Apéndice A. Accesibilidad de los productos

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

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

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

Documentación sobre accesibilidad

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

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

IBM y la accesibilidad

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

Apéndice B. Lectura de la sintaxis de línea de mandatos

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

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

[ ] Identifica un argumento opcional. Los argumentos no especificados entre corchetes son obligatorios.

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

| Indica información mutuamente excluyente. Puede utilizar el argumento de la izquierda o bien el argumento de la derecha del separador. No puede utilizar ambos argumentos en un uso único del mandato.

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

Nota:

- El número máximo de caracteres en un argumento es de 256.
- Especifique los valores de argumento que tengan espacios intercalados entre comillas simples o dobles,

Por ejemplo:

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

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

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

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

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

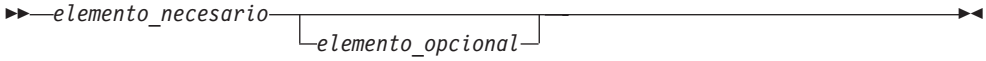
En este ejemplo, los argumentos rule_pack y rule_set se excluyen mutuamente, pero debe especificarse uno de ellos. Además, los puntos suspensivos (...) indican que puede especificarvarios paquetes de reglas o conjuntos de reglas.

Apéndice C. Cómo leer los diagramas de sintaxis

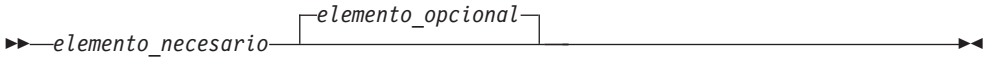
Las reglas siguientes se aplican a los diagramas de sintaxis que se utilizan en esta información:

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

- Los elementos opcionales aparecen debajo de la línea principal.

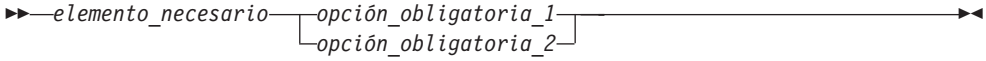


  Si aparece un elemento opcional sobre la línea principal, dicho elemento no tendrá efecto sobre el elemento de sintaxis y sólo se utilizará para facilitar la lectura.

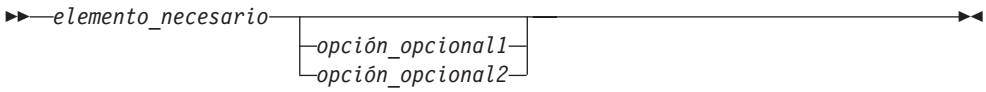


- Si se puede elegir entre dos o más elementos, éstos aparecerán apilados verticalmente.

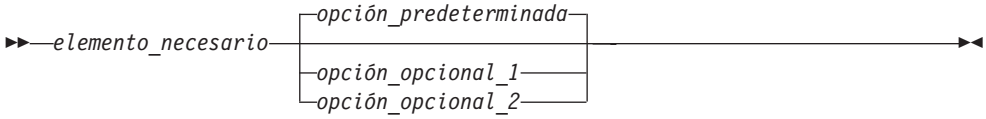
  Si se debe elegir uno de los elementos, un elemento de la pila aparece en la línea principal.



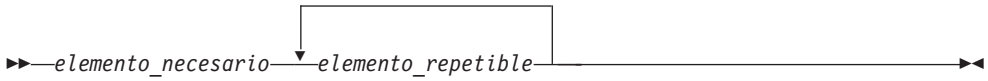
  Si la elección de uno de los elementos es opcional, la pila completa aparece debajo de la trayectoria principal.



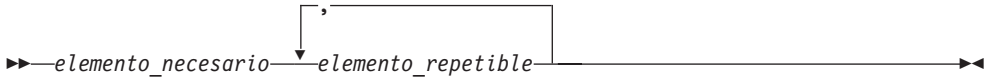
  Si uno de los elementos es el predeterminado, aparecerá por encima de la línea principal y las opciones restantes se mostrarán por debajo.



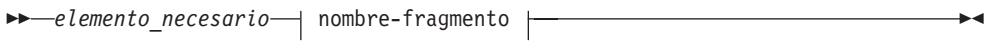
- Una flecha que vuelve hacia la izquierda, sobre la línea principal, indica un elemento que se puede repetir.



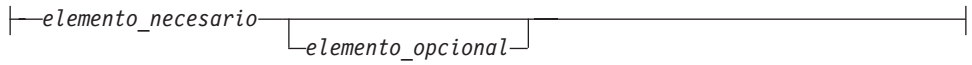
  Si la flecha de repetición contiene una coma, los elementos repetidos se deben separar mediante una coma.



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



  Nombre_fragmento:



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

Apéndice D. Cómo ponerse en contacto con IBM

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

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

Tabla 3. Recursos de IBM

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

Apéndice E. Acceso a la documentación del producto

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

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

Cómo acceder a IBM Knowledge Center

Existen varias maneras de acceder a la documentación en línea:

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

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

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

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

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

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

  Consejo:

  El Knowledge Center tiene también un URL corto:

  `http://ibm.biz/knowctr`

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

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

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

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

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

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

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

Windows

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

AIX Linux

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

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

Obtener la documentación en PDF y en copia impresa

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

Avisos y marcas registradas

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

Avisos

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

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

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

Para realizar consultas relativas a la información de juego de caracteres de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM de su país o envíe las consultas, por escrito, a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokio 103-8510, Japón

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

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

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

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

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

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

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

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

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

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

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

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

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

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

LICENCIA DE COPYRIGHT:

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

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

© (el nombre de su empresa) (año). Partes de este código provienen de programas de ejemplo de IBM Corp. © Copyright IBM Corp. _escriba el año o años_. Reservados todos los derechos.

Si está viendo esta información en copia software, es posible que las fotografías y las ilustraciones en color no aparezcan.

Consideraciones sobre la política de privacidad

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

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

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

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

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

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

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

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

Marcas registradas

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

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

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

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

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

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

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

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

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

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

Índice

F

G

L

M

O

P

R

S

T

V

Impreso en España

SC43-1216-00