



Plastic SCM Platform

# Manual de Triggers



## Contenidos

<b>1. Introducción</b> .....	<b>2</b>
1.1. Propósito .....	2
<b>2. Tipos de triggers</b> .....	<b>3</b>
2.1. Triggers de servidor .....	3
2.2. Triggers de cliente .....	4
<b>3. Uso de los triggers</b> .....	<b>5</b>
3.1. Creación del primer trigger .....	5
3.2. Listar, editar y eliminar triggers .....	6
<b>4. Referencias de triggers</b> .....	<b>7</b>
4.1. Listado de triggers .....	7
4.2. Operaciones de triggers .....	8
4.2.1. Crear un trigger .....	8
4.2.2. Mostrar tipos de triggers .....	9
4.2.3. Lista de triggers.....	10
4.2.4. Cambiar un trigger .....	10
4.2.5. Eliminar trigger.....	11
4.3. Comunicación de triggers .....	12
4.3.1. Entrada .....	12
4.3.2. Salida.....	12
4.3.3. Variables de entorno comunes .....	12
4.3.4. Variables Server.conf.....	13
4.4. Referencia detallada del trigger.....	13
4.4.1. Check in (Proteger) .....	15
4.4.2. Check out (Proteger) .....	18
4.4.3. Crear una rama .....	20
4.4.4. Crear una etiqueta .....	21
4.4.5. Crear un atributo .....	23
4.4.6. Crear un repositorio .....	24
4.4.7. Crear un espacio de trabajo.....	25
4.4.8. Configurar el selector .....	26
4.4.9. Obtener (update) .....	27
<b>5. Ejemplos</b> .....	<b>28</b>
5.1. Proteger .....	29
5.1.1. Aplicar embellecedor de código a ficheros .java .....	29
5.1.2. Aplicar acción de modificación a ítems en bloque .....	30
5.1.3. Comprobar que se han incluido comentarios en la operación de checkin	31
5.1.4. Generación de rss con el contenido del changeset .....	32
5.2. Crear una etiqueta .....	33
5.2.1. Validar que el nombre de la etiqueta comience con 'release'.....	33

## Acerca de este manual

Este manual describe el mecanismo de triggers de Plastic SCM.

## A quién está dirigido

Este manual está dirigida tanto a los desarrolladores como a los administradores del sistema, asumiendo que el lector está familiarizado Plastic SCM y conceptos de sistemas operativos.

## Documentación online

Además de este manual y el resto de guías, Plastic SCM proporciona una referencia online a través de sus diferentes clientes.

Desde la línea de comandos (tanto Windows como Unix) es posible teclear:

```
cm help
```

Para obtener información acerca de todos los comandos disponibles, y

```
cm help command
```

Para obtener información acerca de un comando específico.

Desde las herramientas gráficas es posible acceder a la ayuda Online en el menú de ayuda.

## Términos utilizados

En este manual se describe la sintaxis para diversos comandos. En estas descripciones se utilizan los siguientes términos:

- **Términos en negrita** son cadenas de texto literales
- *{argument}* requiere un argumento obligatorio
- [argument] requiere un argumento opcional

## Errores en la documentación

Si encuentra algún problema en cualquiera de las guías, o en alguna parte de la ayuda online, por favor, notifíquelo a la siguiente dirección de correo electrónico:

[support@codicesoftware.com](mailto:support@codicesoftware.com)

# 1. Introducción

## 1.1. Propósito

---

El sistema de triggers de Plastic SCM permite la ejecución de comandos de usuario en el workflow de ejecución tanto del cliente como del servidor, en la forma de scripts o ejecutable de cualquier otro sistema operativo.

Entre otras funcionalidades, el sistema de triggers de Plastic SCM permite que el desarrollador o el administrador realicen las siguientes tareas:

- Facilitar políticas de creación de ramas tales como convenciones para uso de nombres o asegurar que los nombres de las ramas siempre se refieren a una tarea asociada.
- Introducir reglas anteriores a la operación de checkin (protección) para crear reglas de formato.
- Inclusión de comentarios al realizar un checkin

Plastic SCM soporta la asociación de varios scripts a un trigger en concreto, siendo capaz de realizar diversas acciones seguidas. El usuario puede personalizar la secuencia en la que se ejecutan los scripts.

## 2. Tipos de triggers

### 2.1. Triggers de servidor

---

Este es el tipo más común de trigger. Cuando los clientes realizan operaciones tales como crear revisiones, ramas o cambiar las configuraciones del espacio de trabajo, el servidor es capaz de ejecutar los scripts de usuario antes y después de que se completen las operaciones.

Los triggers previos a la ejecución de una operación suelen permitir cancelar dicha operación, dependiendo del código resultante en el script.

En el servidor se consideran dos tipos diferentes de triggers, dependiendo de qué servidor gestione la operación. Para una descripción detallada de las diferencias entre los servidores de espacio de trabajo y de repositorios puede ver la sección "Servidores independientes de espacios de trabajo y repositorios" en el Manual del Administrador.

- Los triggers de espacios de trabajo los ejecuta el servidor de espacios de trabajo:
  - Crear un espacio de trabajo.
  - Configurar el selector, incluyendo las operaciones "Cambiar a ...".
  - Añadir items al control de código fuente.
  - Check-in.
  - Check-out.
- Triggers de repositorio, ejecutados por el servidor de repositorios:
  - Crear atributo.
  - Crear rama.
  - Crear marcador.
  - Crear repositorio.

## 2.2. Triggers de cliente

---

Algunos eventos que tienen lugar en el cliente pueden tener scripts o programas asociados. Actualmente, los triggers se limitan a una sólo operación:

- Obtner (antes y después)

Por su naturaleza esta operación se gestiona de manera más sencilla desde el contexto de cliente que desde el de servidor.

## 3. Uso de los triggers

Este apartado describe los pasos básicos para comenzar a trabajar con los triggers así como los patrones de uso recomendados. Para una referencia detallada de los parámetros de los triggers y las características soportadas, por favor ver el apartado de Referencias de triggers.

### 3.1. Creación del primer trigger

---

Para asociar un script de usuario con un evento de cliente o servidor se debe de crear un trigger. Se puede obtener un listado de posibles eventos con el siguiente comando:

```
cm showtriggertypes
```

Para crear un trigger que valide los nombres de

Para poder crear un trigger que valide los nombres de las etiquetas, estos deben de crearse según un standard determinado, el usuario puede lanzar un commando como el siguiente (en servidor Windows)

```
cm maketrigger before-mklabel "check label name" "ruby  
c:\plastic\triggers\validate-label.rb"
```

Este es un ejemplo de script (validate-label.rb) que comprueba el nombre de la etiqueta que comienza con 'release'. De otro modo devolvería 1, indicando que el trigger falla y no permite que la operación mklabel termine:

```
if (ENV['PLASTIC_LABEL_NAME'] !~ /^release/) then exit(1) end
```

El script toma el nombre de la etiqueta de la variable de entorno `PLASTIC_LABEL_NAME` y comprueba su contenido contra la expresión `^release`, que significa 'unirlo a un string que comience con 'release'. Si este no es el caso (`!~` operator), la respuesta sera un código de salida indicando 1 fallo en la señalización del trigger.

## 3.2. Listar, editar y eliminar triggers

---

Para ver un trigger que acaba de ser creado, listar los triggers del tipo utilizado:

```
cm listtriggers before-mklabel
1 Validate label c:\tmp\triggers\validate-label.bat dave
```

Para modificar el script al que está señalando este trigger se utilice el comando `changetrigger`, indicando el tipo de trigger que se va a modificar y la posición del mismo, que es la primera parte que se incluye en la lista de comandos de triggers (1 en este caso):

```
cm changetrigger before-mklabel 1 --script="c:\tmp\other-
script.bat"
```

Para eliminar el trigger que se acaba de crear, utilizar el `remove`, indicando el tipo de trigger y su posición:

```
cm removetrigger before-mklabel 1
```

## 4. Referencias de triggers

### 4.1. Listado de triggers

---

Este es un listado completo de todos los eventos disponibles a los que se puede asociar un trigger:

before-add after-add	Lanzado al añadir un item, solo una vez por cada commando 'add' o "añadir". Proporciona un listado de los items añadidos al script del trigger.
before-checkout after-checkout	Lanzado al realizar una desprotección. Proporciona un listado de items desprotegidos.
before-checkin after-checkin	Lanzado al realizar una protección. Proporciona un listado de los elementos que pueden ser protegidos.
before-mkbranch after-mkbranch	Lanzado al crear una rama.
before-mklabel after-mklabel	Lanzado al crear una etiqueta.
before-mkattribute after-mkattribute	Lanzado al crear un atributo.

before-mkrep after-mkrep	Lanzado al crear un repositorio.
before-mkworkspace after-mkworkspace	Lanzado al crear un espacio de trabajo.
before-setselector after-setselector	Lanzado al realizar cualquier cambio en el selector del espacio de trabajo, incluyendo la configuración del selector y el comando "cambiar a..."
before-update after-update	Este es un trigger de cliente. El script recibe la ruta que se debe de actualizar. En el futuro, 'after-update'.

Se puede obtener un listado de todos los triggers soportados en el cliente en línea de comandos utilizando el commando:

```
cm showtriggertypes
```

## 4.2. Operaciones de triggers

### 4.2.1. Crear un trigger

Los triggers se crean desde el cliente en línea de comandos (cm). Esta es la sintáxis para el comando de creación de un trigger:

```
cm maketrigger {type} {name} {script}
  [--position=value]
  [--server=server:port]
```

En el cual:

- Type es el tipo de trigger, como viene listado en la tabla del apartado anterior, como por ejemplo before-mkbranch, or mkbranch-before. Este argumento es obligatorio.
- Name es el nombre que se le da al nuevo trigger. Es una denominación informal y se puede asignar el mismo nombre a más de un trigger (los triggers se identifican por su tipo y posición). Este argumento es obligatorio.
- Script es la ruta completa para el script o programa que sera ejecutado. La ruta a un fichero en el servidor de Plastic, por lo que debe de ser una especificación de fichero válida que el servidor (sea Windows o Unix) comprenda. Este argumento es obligatorio.
- Position se refiere a la posición en la lista de ejecución de un tipo de trigger en concreto. Este parámetro determina el orden de ejecución en el caso de que haya varios scripts registrados para un tipo de trigger determinado. Si la posición que se asigna ya está siendo utilizada por otro

script, dará un error y no se podrá crear el trigger. Este argumento es opcional y en el caso de que se omita el trigger se añadirá al final de la lista actual de scripts.

- Server es el servidor en el que se crea el trigger. En el caso de que se omita, ya que es un argumento opcional, el trigger se creará en el servidor configurado por defecto. La sintáxis especifica un nombre de servidor ':', y un puerto, por defecto 8084.

La posición del script en el tipo de trigger es única, con lo que se mantiene una lista para cada tipo de trigger y las posiciones en dicha lista pueden o no ser utilizadas por un trigger, pero sólo un trigger se puede asignar a cada posición. Si no se especifica una posición el trigger se añadirá al final de la lista. El usuario podrá cambiar la posición en la lista utilizando el comando 'changetrigger'.

Si el script del trigger no existe, se dará un error al ejecutar la operación afectada por el mismo (que evitará que en cualquier caso se complete).

Aquí hay algunos ejemplos de uso:

Para crear un trigger que sea lanzado después de configurar el selector de espacios de trabajo, localizado en /home/scripts/plastic-backup en el servidor, y con el nombre "backup":

```
cm maketrigger after-setselector backup /home/scripts/plastic-backup
Trigger created on position1.
```

Para crear un trigger que se lance antes de que se cree una etiqueta, denominado "validate-label.bat" en el servidor "myserver" en el puerto 8084, con el nombre "Validate label":

```
cm mktrigger before-mklabel "Validate label"
"c:\tmp\triggers\validate-label.bat" --server=myserver:8084
Trigger created on position 1.

cm listtriggers before-mklabel
2 Validate label c:\tmp\triggers\validate-label.bat dave
```

Para crear un trigger que valide el contenido del checkin antes de que se realice la protección en el repositorio en un servidor Windows:

```
cm maketrigger before-checkin ensure-code-stds
"c:\plastic\triggers\checkcode.bat"
Trigger created on position 3.
```

Se pueden encontrar más ejemplos de scripts de triggers en la sección "Ejemplos" al final de este documento.

#### 4.2.2. Mostrar tipos de triggers

Para conseguir un listado de los tipos de triggers disponibles se utilice el commando 'showtriggertypes'. Esta es la sintáxis completa:

```
cm showtriggertypes
```

Este commando es puramente informativo y simplemente devuelve un listado de los tipos de triggers posibles, por lo que es independiente tanto del servidor como del cliente.

### 4.2.3. Lista de triggers

Es posible obtener un listado de los triggers registrados para cualquier tipo de trigger. La sintaxis para el comando es la siguiente:

```
cm listtriggers {type}
  [--server=server:port]
  [--format=formatstring]
```

En dónde:

- Type es el tipo de trigger que se utiliza para obtener la lista de los scripts asociados, como se explica en el apartado de "Listado de triggers". Este argumento es obligatorio.
- Server es el servidor en el que se crea el trigger. En el caso de que se omita, el trigger se creará en el servidor configurado por defecto. La sintaxis especifica un nombre de servidor, ':', y un puerto, por 8084.
- Format es el formato utilizado en los commandos de Plastic. Debajo se puede ver una referencia a los valores disponibles.

Ejemplo de uso para listar los scripts asociados al evento before-checkin:

```
cm listtriggers before-checkin
  1 checkstyle c:\tmp\triggers\checkin-checkstyle.bat dave
```

Este commando devuelve una línea para cada trigger definido. Este es el significado de las columnas de salida:

- 0.- Posición del trigger
- 1.- Nombre del trigger
- 2.- Script del trigger
- 3.- Propietario del trigger

Se pueden indexar las columnas con el argumento --format para obtener salidas personalizadas como en el siguiente ejemplo:

```
cm listtriggers before-mklabel --format="{0} = {2}"
  1 = c:\tmp\triggers\validate-label.bat
  2 = c:\tmp\triggers\loglabels.bat
```

### 4.2.4. Cambiar un trigger

Una vez que se ha creado un trigger se pueden cambiar sus opciones sin tener que volver a recrearlo a través del commando changetrigger command. La sintaxis es:

```

cm changetrigger {type} {existing-trigger-position}
  [--position=value]
  [--name=value]
  [--script=value]
  [--server=server:port]

```

Dónde:

- Type es el tipo de trigger para obtener el listado de los scripts asociados, como se describe en la tabla del apartado "Listado de triggers". Este argumento es obligatorio.
- Existing trigger position es el nombre por el cual el trigger se identifica en la lista de triggers asociados al tipo de trigger. Este valor, junto con el tipo de trigger, identifica de manera única el script que se va a editar. Este argumento es obligatorio.
- Position es la nueva posición del trigger en la lista de triggers. Hay que tener en cuenta que la posición de destino no debe de utilizarse en otro trigger o dará error. Este argumento es opcional.
- Name es el nuevo nombre del trigger. Hay que tener en cuenta que el nombre se utiliza por trazabilidad. Este argumento es opcional.
- Script es la nueva ruta para que se ejecute el programa o script. Aquí se aplican las mismas restricciones que se describían para la creación del trigger. Este argumento es opcional.
- Server es el servidor en el que se edita el trigger. En el caso de que se omita, el trigger se podrá editar en el servidor configurado por defecto. La sintaxis especifica un nombre de servidor, ':', y un puerto, por defecto 8084.

Ejemplo para cambiar el script del nombre del trigger y el objetivo:

```

cm listtriggers before-checkin
  1 checkstyle c:\tmp\triggers\checkin-checkstyle.bat dave

cm changetrigger before-checkin 1 --name="codestyle"

cm listtriggers before-checkin
  1 codestyle c:\tmp\triggers\checkin-checkstyle.bat dave

```

#### 4.2.5. Eliminar trigger

Se puede eliminar un trigger de Plastic. Al eliminar un trigger no se elimina el script o el programa del sistema de ficheros asociado al mismo, simplemente Plastic recibe la orden de no ejecutar ese script. Esta es la sintaxis del comando:

```

cm removetrigger {type} {existing-trigger-position}
  [--server=server:port]

```

En dónde:

- Type es el tipo de trigger para obtener la lista de scripts asociados, según la lista en la tabla de "Listado de triggers". Este argumento es obligatorio.

- Existing trigger position es el índice por el que se denomina el trigger en el listado de triggers asociados a ese tipo determinado. Este valor, junto con el tipo de trigger, identifica el script que será eliminado. Este argumento es obligatorio.

Ejemplo:

```
cm listtriggers before-mklabel
1 Validate label c:\tmp\triggers\validate-label.bat dave
2 log labels c:\tmp\triggers\loglabels.bat dave

cm rmtrigger before-mklabel 2

cm listtriggers before-mklabel
1 Validate label c:\tmp\triggers\validate-label.bat dave
```

### 4.3. Comunicación de triggers

---

#### 4.3.1. Entrada

Plastic SCM enviará información al script del trigger para realizar la operación de ejecución. Se pueden utilizar dos enfoques diferentes:

- Entrada estándar: normalmente las referencias a objetos tales como las especificaciones de revisions referents a la operación de trigger se incluirán en el script del trigger a través de entrada estándar.
- Variables de entorno: información general como qué usuario de Plastic comenzó la operación o la máquina del cliente. Para una descripción detallada de las variables que se utilizan en una operación en concreto, comprobar la operación específica en la referencia del apartado Variables de entorno comunes.

#### 4.3.2. Salida

El script del trigger comunicará el resultado de la ejecución utilizando el código resultante. Plastic interpretará los siguientes códigos resultantes:

- Cero (0): el trigger terminó correctamente, la operación puede continuar.
- No-cero (>0): el trigger ha fallado, la operación no puede continuar.

Si el resultado no es cero en un trigger 'before', la operación será cancelada y el cliente recibirá un error.

Si el resultado no es cero para un trigger 'after', la operación ya se ha realizado y no se puede reanudar, pero el cliente también recibirá un error.

#### 4.3.3. Variables de entorno comunes

Esta tabla muestra las variables de entorno disponibles para cada script de trigger:

PLASTIC_USER	El usuario que ha comenzado la operación en el cliente.
--------------	---

PLASTIC_CLIENTMACHINE	La máquina de cliente que comenzó la operación.
PLASTIC_SERVER	El nombre del servidor de Plastic.

#### 4.3.4. Variables Server.conf

Las variables se pueden definir en el fichero "server.conf". Se pasa su valor al script del trigger o al programa como variables de entorno. Para definir estas variables se debe de incluir una sección denominada 'TriggerVariables' en el fichero server.conf que se encuentra en la carpeta de instalación del servidor. El siguiente ejemplo muestra una opción de uso para dicho fichero:

```
<?xml version="1.0"?>
<ServerConfigData>
  <Language>en</Language>
  <WorkingMode>UPWorkingMode</WorkingMode>
  <ServerType>ServerTypeAll</ServerType>

  <TriggerVariables>
    <TriggerVariable name="TRIGGERS_PATH" value="c:\triggers" />
  </TriggerVariables>

</ServerConfigData>
```

Este ejemplo define una variable denominada 'TRIGGERS\_PATH' con el valor 'c:\triggers'. Esta variable se puede utilizar al crear un trigger, en el campo 'script' como en el siguiente ejemplo:

```
cm createtrigger before-checkin "code checker"
"@TRIGGERS_PATH\stylecheck.bat"
```

Esta variable también pasa al script del trigger como variable de entorno, por lo que se podría utilizar dentro del mismo script como en el siguiente ejemplo:

```
@echo off

set OUTPUT_FILE=%TRIGGERS_PATH%\label.log.txt

echo %PLASTIC_REPOSITORY_NAME% %PLASTIC_LABEL_NAME% >>
%OUTPUT_FILE%

exit 0
```

## 4.4. Referencia detallada del trigger

Los siguientes apartados proporcionan referencias detalladas de los triggers así como parámetros tanto de entrada como de salida. Se incluyen ejemplos par alas acciones más comunes. Add

## Nombres de los triggers

```
before-add
after-add
```

### Descripción

Ejecutan los scripts de usuarios cuando se añaden los items al control de código fuente.

### Funcionamiento en

Servidor de repositorios

### Entrada estándar

No hay entrada estándar para estos.

### Resultado de salida

El código resultante del script del trigger o el ejecutable determina que la operación tenga éxito o falle:

0	El trigger concluyó satisfactoriamente. La rama sera creada.
No cero	El trigger da un error. Si el trigger es before-add, lanzará un error y los items no se añadirán al repositorio. Si el trigger es after-add lanzará un error. De todos modos los ítems ya han sido añadidos al repositorio.

### Variables de entorno

Además de las variables definidas en las secciones "Variables de entorno comunes" y "Variables Server.conf" las siguientes variables están disponibles:

PLASTIC_COMMENT	El comentario incluido por el usuario al realizar la operación "añadir".
-----------------	--

### 4.4.1. Check in (Proteger)

#### Nombres de los triggers

```
before-checkin
after-checkin
```

#### Descripción

Ejecutan scripts de usuario al realizar en cualquier cliente una operación de protección.

#### Comentarios

Los triggers previos y posteriores a la operación de protección o check-in se invocan una sola vez para todos los items involucrados en la operación. La entrada estándar del trigger recibirá una lista de los items involucrados.

Este es uno de los trigger más complejos y útiles. Los ejemplos de uso incluyen comprobación del código antes de hacer protección en el repositorio contra algún tipo de validación o herramienta de formato o envío de notificaciones al proteger el código en el repositorio.

Se puede acceder al contenido de las revisiones a través del comando 'cm cat', incluyendo la especificación de la revisión que se da en la entrada estándar. Se pueden validar, modificar y volver a almacenar en el servidor a través del comando 'cm shelve'. En caso de que el comando 'shelve' actualice el contenido de una revisión en el repositorio, el cliente que realice la operación de protección **actualizará estos ítems de manera automática** por lo que el contenido del espacio de trabajo siempre será el adecuado.

#### Funcionamiento en

Servidor de espacios de trabajo

#### Entrada estándar

La entrada estándar recibe identificadores de revisión para todos los ítems incluidos en la operación de check in, uno por cada línea. Cada uno de ellos tiene una cadena de formato específico que contiene la ruta del servidor para cada ítem (independiente del espacio de trabajo) y la especificación de la revisión, por lo que se puede obtener fácilmente el contenido utilizando el comando 'cm cat'.

Este es el formato de las especificaciones de la revisión, uno por línea:

```
item_path#br:branch#rev_no;rev_id@rep:rep_name@repserver:server
```

En la siguiente tabla se puede ver el significado de las especificaciones en cursiva:

item_path	La ruta de la revision en <b>formato servidor</b> , que es independiente del espacio de trabajo del cliente y del sistema operativo.
-----------	--

branch	La rama en la que está la revisión.
rev_no	El número de revision dentro de la rama de la revisión.
rev_id	El identificador único de la revisión. Se puede utilizar para facilitar el análisis sintáctico al acceder a las revisiones con 'cm cat' o 'cm shelve' en el script del trigger ya que la serie a partir del punto y coma identifica de manera única la revision dentro del servidor.
rep_name	El nombre del repositorio al que pertenece la revisión.
rep_server	El servidor de repositorios en el que está el repositorio.

Ejemplo de entrada estándar proporcionada por un trigger de checkin:

```
/code/clean.bat#br:/main#CO;revid:12936@rep:default@repserver:DARK
TOWER:8084;wk:trigger_test@DARKTOWER
```

### Resultado de salida

El código resultante del script del trigger o el ejecutable determina el éxito o el error de la operación:

0	El trigger concluyó satisfactoriamente. Los ítems se protegerán en el repositorio.
No cero	El trigger da un error.  Si el trigger es before-checkin, la operación de checkin parará y no se protegen los ítems ni se crea el changeset. El cliente recibe un mensaje de error.  Si el trigger es after-checkin lanzará un error. De todos modos la operación ya se ha realizado.

### Variables de entorno

Además de las variables definidas en las secciones "Variables de entorno comunes" y "Variables Server.conf" las siguientes variables están disponibles:

PLASTIC_COMMENT	El comentario incluido por el usuario al proteger.
PLASTIC_CHANGESET	El changeset o los changesets que se crearon como resultado de la operación de checkin. Hay que tener en cuenta que esta variable tan solo está disponible en el trigger 'after-checkin'. Ver las notas sobre el formato de esta variable.

La variable PLASTIC\_CHANGESET contiene las especificaciones de los changesets creados, separados por punto y coma (;'). Este es un ejemplo del valor de una variable con changesets creados en dos repositorios diferentes:

```
cs:23@br:/main@rep:default@repserver:DARKTOWER:8084;  
cs:19@br:/main@rep:secondrep@repserver:DARKTOWER:8084
```

### Ejemplo en línea de comandos

```
cm mktrigger checkin-before "checkstyle" "c:\tmp\triggers\checkin-  
checkstyle.bat"  
Trigger created on position 1.
```

### Ejemplo de script de trigger

El siguiente script simplemente lee todas las entradas standard y las redirecciona al fichero 'c:\tmp\triggers\checkout.txt'. Aquí lo importante es el uso del commando find.exe para leer la entrada estándar en línea de comando Windows 'cmd.exe':

```
@echo off  
  
for /f "tokens=*" %%g in ('find /V ""') do (  
    echo %%g >> c:\tmp\triggers\checkout.txt  
)  
  
exit 0
```

## 4.4.2. Check out (Proteger)

### Nombre de los triggers

```
before-checkout
after-checkout
```

### Descripción

Ejecutan scripts de usuario al realizar en cualquier cliente una operación de desprotección.

### Comentarios

Los triggers previos y posteriores a la operación de desprotección o checkout se invocan una sola vez para todos los ítems involucrados en la operación. La entrada estándar del trigger recibirá una lista de los ítems involucrados.

### Funcionamiento en

Servidor de espacios de trabajo

### Entrada estándar

La entrada estándar recibe identificadores de revisión para todos los ítems incluidos en la operación, uno por cada línea. Cada uno de ellos tiene una cadena de formato específico que contiene la ruta del servidor para cada ítem (independiente del espacio de trabajo) y la especificación de la revisión, por lo que se puede obtener fácilmente el contenido utilizando el comando 'cm cat'.

Este es el formato de las especificaciones de la revisión, uno por línea:

```
item_path#br:branch#rev_no;rev_id@rep:rep_name@repserver:server
```

The meaning of the members in italic is detailed in the following table:

En la siguiente tabla se puede ver el significado de las especificaciones en cursiva:

<i>item_path</i>	La ruta de la revision en <b>formato servidor</b> , que es independiente del espacio de trabajo del cliente y del sistema operativo.
<i>branch</i>	La rama padre.
<i>rev_no</i>	El número de la revision padre.
<i>rev_id</i>	El identificador único de la revisión. Se puede utilizar para facilitar el análisis sintáctico al acceder a las revisiones con 'cm cat' o 'cm shelve' en el script del trigger ya que la serie a partir del punto y coma identifica de manera única la revision dentro del servidor.

rep_name	El nombre del repositorio al que pertenece la revisión.
rep_server	El servidor de repositorios en el que está el repositorio.

Ejemplo de entrada estándar proporcionada por un trigger de checkin:

```
/code/clean.bat#br:/main#1;revid:12936@rep:default@repserver:DARKTOWER:8084;wk:trigger_test@DARKTOWER
```

## Resultado de salida

El código resultante del script del trigger o el ejecutable determina el éxito o el error de la operación:

0	El trigger concluyó satisfactoriamente. Los ítems se desprotegerán en el repositorio.
No cero	El trigger da un error.  Si el trigger es before-checkout, la operación parará y no se desprotegerán los ítems. El cliente recibe un mensaje de error.  Si el trigger es after-checkout lanzará un error. De todos modos la operación ya se ha realizado.

## Variables de entorno

Además de las variables definidas en las secciones "Variables de entorno comunes" y "Variables Server.conf" la siguiente variable está disponible:

PLASTIC_COMMENT	El comentario incluido por el usuario al proteger.
-----------------	--

## Ejemplo en línea de comandos

```
cm mktrigger checkout-before "checkstyle"
"c:\tmp\triggers\checkout-checkstyle.bat"
Trigger created on position 1.
```

## Ejemplo de script de trigger

El siguiente script simplemente lee todas las entradas standard y las redirecciona al fichero 'c:\tmp\triggers\checkinout.txt'. Aquí lo importante es el uso del commando find.exe para leer la entrada estándar en línea de comando Windows 'cmd.exe':

```
@echo off
for /f "tokens=*" %%g in ('find /V ""') do (
    echo %%g >> c:\tmp\triggers\checkinout.txt
)
exit 0
```

### 4.4.3. Crear una rama

#### Nombres de los triggers

```
before-mkbranch
after-mkbranch
```

#### Descripción

Ejecutan los scripts de usuario al crear una rama.

#### Comentarios

Estos triggers se lanzan al crear una rama.

#### Funcionamiento en

Servidor de repositorios

#### Entrada estándar

No hay entrada estándar para estos triggers.

#### Resultado de salida

El código resultante del script del trigger o del ejecutable determina el éxito o error de la operación:

0	El trigger concluyó satisfactoriamente. La rama será creada.
No cero	El trigger da un error. Si el trigger es before-mkbranch, la operación parará y no se creará la rama. Si el trigger es after-mkbranch lanzará un error. De todos modos la operación ya se ha realizado.

#### Variables de entorno

Además de las variables definidas en las secciones "Variables de entorno comunes" y "Variables Server.conf" la siguiente variable está disponible:

PLASTIC_COMMENT	El comentario incluido por el usuario al crear una rama.
PLASTIC_BRANCH_NAME	El nombre de la rama que se está creando.
PLASTIC_REPOSITORY_NAME	El nombre del repositorio en el que se crea la rama.

#### 4.4.4. Crear una etiqueta

##### Nombres de los triggers

```
before-mklabel
after-mklabel
```

##### Descripción

Ejecuta los scripts de usuario al crear una etiqueta.

##### Comentarios

Este trigger se lanza al crear una etiqueta.

##### Funcionamiento en

Servidor de repositorios

##### Entrada estándar

Estos triggers no tienen entrada estándar.

##### Resultado de salida

El código resultante del script del trigger o del ejecutable determina el éxito o error de la operación:

0	El trigger concluyó satisfactoriamente. La etiqueta será creada.
No cero	El trigger da un error. Si el trigger es before-mklabel, la operación parará y no se creará la etiqueta. Si el trigger es after-mklabel lanzará un error. De todos modos la operación ya se ha realizado.

##### Variables de entorno

Además de las variables definidas en las secciones "Variables de entorno comunes" y "Variables Server.conf" la siguiente variable está disponible:

PLASTIC_COMMENT	El comentario incluido por el usuario al crear una rama.
PLASTIC_LABEL_NAME	La etiqueta que se ha creado.
PLASTIC_REPOSITORY_NAME	El nombre del repositorio en el que se ha creado la etiqueta.

## Ejemplo en línea de comandos

```
cm maketrigger before-mklabel "validate label"  
"c:\plastic\triggers\Validate-label.bat"  
Trigger created on position 1.  
  
cm listtriggers before-mklabel  
2 Validate label c:\tmp\triggers\validate-label.bat dave
```

## El ejemplo de script del trigger

El siguiente script guarda un registro de las ramas creadas en el fichero c:\plastic\triggers\labels.log.txt.

```
@echo off  
  
echo %PLASTIC_REPOSITORY_NAME% %PLASTIC_LABEL_NAME% >>  
c:\plastic\triggers\labels.log.txt  
  
exit 0
```

#### 4.4.5. Crear un atributo

##### Nombres de los triggers

```
before-mkattribute
after-mkattribute
```

##### Descripción

Ejecuta los scripts de usuario cuando se crea un atributo.

##### Comentarios

Estos triggers se lanzan al crearse un atributo.

##### Funcionamiento en

Servidor de repositorios

##### Entrada estándar

No hay entrada estándar para estos atributos.

##### Resultado de salida

El código resultante del script del trigger o del ejecutable determina el éxito o error de la operación:

0	El trigger concluyó satisfactoriamente. El atributo será creado.
No cero	El trigger da un error. Si el trigger es before-mkattribute, la operación parará y el atributo no será creado. Si el trigger es after-mkattribute lanzará un error. De todos modos la operación ya se ha realizado.

##### Variables de entorno

Además de las variables definidas en las secciones "Variables de entorno comunes" y "Variables Server.conf" la siguiente variable está disponible:

PLASTIC_COMMENT	El comentario incluido por el usuario al crear el atributo.
PLASTIC_ATTRIBUTE_NAME	El atributo que se ha creado.
PLASTIC_REPOSITORY_NAME	El nombre del repositorio en el que se ha creado el atributo.

#### 4.4.6. Crear un repositorio

##### Nombres de los triggers

```
before-mkrepository
after-mkrepository
```

##### Descripción

Ejecuta los scripts de usuario al crear un repositorio.

##### Comentarios

Estos triggers se lanzan al crear un repositorio

##### Funcionamiento en

Servidor de repositorios

##### Entrada estándar

No hay entrada estándar para estos atributos.

##### Resultado de salida

El código resultante del script del trigger o del ejecutable determina el éxito o error de la operación:

0	El trigger concluyó satisfactoriamente. El repositorio será creado.
No cero	El trigger da un error. Si el trigger es before-mkrepository, la operación parará y no se creará el repositorio. Si el trigger es after-mkrepository lanzará un error. De todos modos la operación ya se ha realizado.

##### Variables de entorno

Además de las variables definidas en las secciones "Variables de entorno comunes" y "Variables Server.conf" la siguiente variable está disponible:

PLASTIC_REPOSITORY_NAME	El nombre del repositorio en el que se crea el atributo.
-------------------------	--

#### 4.4.7. Crear un espacio de trabajo

##### Nombres de los triggers

```
before-mkworkspace
after-mkworkspace
```

##### Descripción

Ejecuta los triggers de usuario al crear un espacio de trabajo

##### Comentarios

Estos triggers se lanzan al crearse un espacio de trabajo.

##### Funcionamiento en

Servidor de espacios de trabajo

##### Entrada estándar

No hay entrada estándar para estos atributos.

##### Resultado de salida

El código resultante del script del trigger o del ejecutable determina el éxito o error de la operación:

0	El trigger concluyó satisfactoriamente por lo que se creará el espacio de trabajo.
No cero	El trigger da un error. Si el trigger es before-mkworkspace, la operación parará y no se creará el espacio de trabajo. Si el trigger es after-mkworkspace lanzará un error. De todos modos, ya se ha creado el espacio de trabajo.

##### Variables de entorno

Además de las variables definidas en las secciones "Variables de entorno comunes" y "Variables Server.conf" la siguiente variable está disponible:

PLASTIC_WORKSPACE_NAME	El nombre dado al nuevo espacio de trabajo.
PLASTIC_WORKSPACE_PATH	La ruta del espacio de trabajo en la máquina del cliente.

#### 4.4.8. Configurar el selector

##### Nombres de los triggers

```
before-setselector
after-setselector
```

##### Descripción

Ejecutan scripts de usuarios al cambiar el selector del espacio de trabajo.

##### Comentarios

Los selectores se modifican con el comando 'setselector' o con 'Switch workspace to branch / label', tanto en línea de comandos como en el cliente gráfico.

##### Funcionamiento en

Servidor de espacios de trabajo

##### Entrada estándar

La entrada estándar para estos triggers recibe el contenido del selector que configura el cliente.

##### Resultado de salida

El código resultante del script del trigger o el ejecutable determina el éxito o error de la operación:

0	El trigger concluyó satisfactoriamente por lo que se cambiará el selector.
No cero	El trigger da un error. Si el trigger es before-setselector, la operación parará y no se modificará el selector. Si el trigger es after-setselector lanzará un error. De todos modos, el selector ya ha sido modificado.

##### Variables de entorno

Además de las variables definidas en las secciones "Variables de entorno comunes" y "Variables Server.conf" la siguiente variable está disponible:

PLASTIC_WORKSPACE_NAME	El nombre dado al espacio de trabajo en el cual se está modificando el selector.
PLASTIC_WORKSPACE_PATH	La ruta del espacio de trabajo en la máquina del cliente en el cual se está modificando el selector.

#### 4.4.9. Obtener (update)

##### Nombre de los triggers

```
before-update
after-update
```

##### Descripción

Ejecuta los scripts de usuario al hacer un update.

##### Comentarios

Este trigger se ejecuta en el cliente por lo que las localizaciones del script se refieren a los sistemas de ficheros en la máquina del cliente. Esta es una diferencia importante que se debe de tener en cuenta al crear este tipo de triggers.

##### Funcionamiento en

Cliente

##### Entrada estándar

No hay entrada estándar para estos triggers.

##### Resultado de salida

El código resultante del script del trigger o el ejecutable determina el éxito o error de la operación:

0	El trigger concluyó satisfactoriamente por lo que se realizará la operación de update.
No cero	El trigger da un error. Si el trigger es before-update, la operación parará y no se obtendrá el código. Si el trigger es after-setselector lanzará un error pero la operación ya ha sido realizada

##### Variables de entorno

PLASTIC_USER	El usuario que comenzó la operación de update.
PLASTIC_CLIENTMACHINE	La máquina de cliente que comenzó la operación.
PLASTIC_UPDATE_PATH	La ruta de cliente del espacio de trabajo que se obtiene.

## 5. Ejemplos

Todos los ejemplos de esta sección se pueden encontrar en la carpeta 'triggers' de la carpeta de instalación del servidor.

## 5.1. Proteger

### 5.1.1. Aplicar embellecedor de código a ficheros .java

Este ejemplo procesa todos los ficheros java a través de un embellecedor de código (aquí se utiliza jindent, que se puede sustituir por otra herramienta). El script está escrito en Ruby.

```
#!/usr/bin/env ruby

# temp file that will be used for jindent
tmpfile = "c:\\tmp\\triggers\\trigger-validate.java"

# Process each line of stdin
STDIN.readlines.each_with_index do |line, index|

  # split into item, revspec and wkspec
  splitted = line.split(';')

  # pick item name from item spec
  filename = splitted[0].split('#')[0]

  # if it is a .java file, apply jindent
  if (filename =~ /\.java$/) then

    # revspec is after the first ;
    revspec = splitted[1];

    # extract revision content from repository to temp file
    res = system("cm cat #{revspec} --file=\"#{tmpfile}\"")

    # execute jindent on temp file (jindent should be on path)
    if (res) then res = system("jindent \"#{tmpfile}\"") end

    # if jindent failed, signal the trigger failed too
    if (!res || $? != 0) then exit(1) end

    # store the re-formatted file on Plastic repository
    if (res) then system("cm shelve #{revspec} --
file=\"#{tmpfile}\"") end

    # delete the temp file
    if (res) then system("del \"#{tmpfile}\"") end

  end #if
end #each
```

Ejemplo de commando de creación de un trigger (en Windows)

```
cm maketrigger before-checkin "apply jindent" "ruby
c:\triggers\jindent.rb"
```

### 5.1.2. Aplicar acción de modificación a ítems en bloque

Este sería el mismo ejemplo que el anterior, pero en este caso todos los ficheros involucrados son 'cat' y 'shelved' en bloque, logrando un mayor rendimiento.

```
#!/usr/bin/ruby
tmpdir = 'c:\\tmp\\triggers\\'

$files = []
$cat_shelve_specs = []

# Apply command sending revision info
def commandOnSpecs(cmd)
  IO.popen(cmd, "w") do |io|
    $cat_shelve_specs.each do |spec|
      puts 'catting ' + spec
      io.puts spec
    end
  end
end

# Process stdin
STDIN.readlines.each do |line|
  itemspec, revspec, wkspec = line.split(';')
  filename, branchspec, revno = itemspec.split('#')

  # this may have problems with long paths
  filename.gsub!(/\//, '_') # replace / with _ in filenames
  filename = tmpdir + filename # add tmpdir

  $files << filename
  $cat_shelve_specs << "#{revspec};#{filename}"
end

# cat files on temp directory
commandOnSpecs("cm cat -")

# Apply action on files
$files.each { |file| system("jindent \"#{file}\") }

# shelve files
commandOnSpecs("cm shelve -")

# remove temp files
$files.each { |file| File.delete file }
```

Ejemplo de comando de creación de un trigger (en Windows)

```
cm maketrigger before-checkin "apply block jindent" "ruby
c:\triggers\jindent.rb"
```

### 5.1.3. Comprobar que se han incluido comentarios en la operación de checkin

Ejemplo de script ruby que comprueba la variable de entorno PLASTIC\_COMMENT:

```
c = ENV['PLASTIC_COMMENT']  
if (c == nil || c == '') then exit(1) end
```

Ejemplo de commando de creación del trigger:

```
cm maketrigger before-checkin "comment required" "ruby  
c:\triggers\check-comments.rb"
```

### 5.1.4. Generación de rss con el contenido del changeset

Este trigger en ruby combina todas las técnicas vistas en los apartados anteriores para ofrecer un script capaz de actualizar un fichero rss que puedan utilizar los agregadores rss para notificar los nuevos cambios del repositorio.

```
require 'rss/2.0'
require 'open-uri'
require 'rss/maker'

targetfile = "Z:\\cm\\tts\\plastic-changesets.rss"

# Read content if available
if (File.exists?(targetfile)) then
  content = ""
  open(targetfile) do |s| content = s.read end
  rss = RSS::Parser.parse(content, false)
else
  rss = RSS::Rss.new("2.0")
  channel = RSS::Rss::Channel.new
  channel.title = "Plastic updates feed"
  channel.link = http://www.plasticscm.com
  channel.description = ""
  channel.language = "en"
  rss.channel = channel
end

# Parse checkin item names from plastic

files = ''
STDIN.readlines.each do |line|
  itemspec, revspec, wkspec = line.split(';')
  filename, branchspec, revno = itemspec.split('#')
  files << filename << "<br/>"
end

# Add new rss item

item = RSS::Rss::Channel::Item.new
item.title = "#{ENV['PLASTIC_CHANGESET']} -
#{ENV['PLASTIC_COMMENT']} by #{ENV['PLASTIC_USER']}"
item.link = http://www.plasticscm.com
item.date = Time.now
item.description = files
rss.items << item

# Write the resulting file

File.open(targetfile, "w") do |f|
  f.write(rss)
end
```

Este es un trigger "after-checkin":

```
cm maketrigger after-checkin "generate rss" "ruby c:\triggers\rss-
gen.rb"
```

## 5.2. Crear una etiqueta

---

5.2.1. Validar que el nombre de la etiqueta comience con 'release'

```
if (ENV['PLASTIC_LABEL_NAME'] !~ /^release/) then exit(1) end
```

Comando de creación del trigger:

```
cm maketrigger before-mklabel "check label name" "ruby  
c:\plastic\triggers\validate-label.rb"
```