

**Índice**

La guía de los triggers de Plastic SCM

Versión 3.0

Plastic SCM **Triggers**

© 2006-2010 Codice Software

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Codice Software cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

[1. Introducción 2](#_Toc262202670)

[1.1. Propósito 2](#_Toc262202671)

[2. Tipos de triggers 3](#_Toc262202672)

[2.1. Triggers de servidor 3](#_Toc262202673)

[2.2. Triggers de cliente 3](#_Toc262202674)

[3. Uso de los triggers 5](#_Toc262202675)

[3.1. Creación del primer trigger 5](#_Toc262202676)

[3.2. Listar, editar y eliminar triggers 6](#_Toc262202677)

[4. Referencias de triggers 7](#_Toc262202678)

[4.1. Listado de triggers 7](#_Toc262202679)

[4.2. Operaciones de triggers 8](#_Toc262202680)

[4.2.1. Crear un trigger 8](#_Toc262202681)

[4.2.2. Mostrar tipos de triggers 9](#_Toc262202682)

[4.2.3. Lista de triggers 10](#_Toc262202683)

[4.2.4. Cambiar un trigger 10](#_Toc262202684)

[4.2.5. Eliminar trigger 11](#_Toc262202685)

[4.3. Comunicación de triggers 12](#_Toc262202686)

[4.3.1. Entrada 12](#_Toc262202687)

[4.3.2. Salida 12](#_Toc262202688)

[4.3.3. Variables de entorno comunes 12](#_Toc262202689)

[4.3.4. Variables Server.conf 12](#_Toc262202690)

[4.4. Referencia detallada del trigger 13](#_Toc262202691)

[4.4.1. Add (añadir) 14](#_Toc262202692)

[4.4.2. Check in (Proteger) 15](#_Toc262202693)

[4.4.3. Check out (Proteger) 18](#_Toc262202694)

[4.4.4. Crear una rama 20](#_Toc262202695)

[4.4.5. Crear una etiqueta 21](#_Toc262202696)

[4.4.6. Crear un atributo 23](#_Toc262202697)

[4.4.7. Crear un repositorio 24](#_Toc262202698)

[4.4.8. Crear un espacio de trabajo 25](#_Toc262202699)

[4.4.9. Configurar el selector 26](#_Toc262202700)

[4.4.10. Obtener (update) 27](#_Toc262202701)

[4.4.11. Checkout en cliente 28](#_Toc262202702)

[4.4.12. Checkin en cliente 29](#_Toc262202703)

[5. Ejemplos 30](#_Toc262202704)

[5.1. Proteger 31](#_Toc262202705)

[5.1.1. Aplicar embellecedor de código a ficheros .java 31](#_Toc262202706)

[5.1.2. Aplicar acción de modificación a ítems en bloque 32](#_Toc262202707)

[5.1.3. Comprobar que se han incluido comentarios en la operación de checkin 33](#_Toc262202708)

[5.1.4. Generación de rss con el contenido del changeset 34](#_Toc262202709)

[5.2. Crear una etiqueta 35](#_Toc262202710)

[5.2.1. Validar que el nombre de la etiqueta comience con ‘release’ 35](#_Toc262202711)

[5.3. Client checkout 35](#_Toc262202712)

[5.3.1. Actualizar ficheros antes de hacer checkout 35](#_Toc262202713)

**Acerca de este manual**

Este manual descibe el mecanismo de triggers de Plastic SCM.

**A quién está dirigido**

Este manual está dirigido 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 aplicación cliente.

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)

# Introducción

## 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.

# Tipos de triggers

## 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.

Los triggers de servidor son los siguientes:

* Añadir items al control de código fuente.
* Check-in.
* Crear atributo.
* Crear rama.
* Crear marcador.
* Crear repositorio.

## Triggers de cliente

Algunos eventos que tienen lugar en el cliente pueden tener scripts o programas asociados. Actualmente, los triggers de cliente son:

* Obtener (antes y después)
* Checkin (antes y después)
* Checkout (antes y después)
* Crear un espacio de trabajo.
* Configurar el selector, incluyendo las operaciones “Cambiar a ”.

# Uso de los triggers

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

## Creación del primer trigger

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

**cm showtriggertypes**

Para poder crear un trigger que valide los nombres de las etiquetas, estos deben crearse según un estándar determinado, el usuario puede lanzar un comando 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.

## 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, utilice el comando remove, indicando el tipo de trigger y su posición:

**cm removetrigger before-mklabel 1**

# Referencias de triggers

## 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 comando ‘add’ o “añadir”. Proporciona un listado de los ítems añadidos al script del trigger. |
| 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. |
| before-clientcheckin  after-clientcheckin | Son triggers de cliente. El script recibe las rutas de los elementos sobre los que se ha lanzado la operación de checkin. |
| before-clientcheckout  after-clientcheckout | Son triggers de cliente. El script recibe las rutas de los elementos sobre los que se ha lanzado la operación de checkout. |

**Tabla 1. Lista de triggers**

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

**cm showtriggertypes**

## Operaciones de triggers

### Crear un trigger

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

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

En la cual:

|  |  |  |
| --- | --- | --- |
| Parámetro | Descripción | Obligatorio |
| Type | Tipo de trigger, como viene listado en la tabla de arriba, como por ejemplo before-mkbranch, or mkbranch-before. | YES |
| Name | 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). | YES |
| Script | Ruta completa para el script o programa que sera ejecutado. La ruta a un fichero en el servidor de Plastic, por lo que debe ser una especificación de fichero válida que el servidor (sea Windows o Unix) comprenda. | YES |
| 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. | NO |
| Server | 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 sintaxis especifica un nombre de servidor ‘:’, y un puerto, por defecto 8084. | NO |

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.

### Mostrar tipos de triggers

Para conseguir un listado de los tipos de triggers disponibles se utilice el comando ‘showtriggertypes’. Esta es la sintaxis completa:

**cm showtriggertypes**

Este comando 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.

### 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 comandos 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 comando 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

Si no se proporciona un tipo de trigger, el commando **listtriggers** listará todos los triggers existentes en el servidor.

### Cambiar un trigger

Una vez que se ha creado un trigger se pueden cambiar sus opciones sin tener que volver a crearlo a través del comando changetrigger. 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 con 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 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

### 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*]

Donde:

* 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

## Comunicación de triggers

### 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 referentes 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.

### 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.

### 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. |

### 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 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 se 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

## 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 para las acciones más comunes.

Add (añadir)

#### Nombres de los triggers

before-add

after-add

#### Descripción

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

#### Comentarios

No hay nada que añadir.

#### Funcionamiento en

Servidor.

#### Entrada estándar

La entrada estándar recibe todos los ítems involucrados en la operación de add, uno por línea. Cada línea consta del path del ítem en la máquina cliente.

#### 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”. |

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 revision en el repositorio, el cliente que realize 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.

#### 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 **formato servidor**, 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: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 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 estándar y las redirecciona al fichero ‘c:\tmp\triggers\checkinout.txt’. Aquí lo importante es el uso del comando 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

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.

#### Entrada estándar

La especificación de la rama que será creada. Esta entrada la toma del comando make branch.

#### 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. | |

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.

#### Entrada estándar

La especificación de la etiqueta que será creada. Esta entrada la toma del comando make label.

#### 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 linea 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

Crear un atributo

#### Nombres de los triggers

before-mkatt

after-mkatt

#### Descripción

Ejecuta los scripts de usuario cuando se crea un atributo.

#### Comentarios

Estos triggers se lanzan al crearse un atributo.

#### Funcionamiento en

Servidor.

#### Entrada estándar

La especificación del atributo que será creado. Esta entrada la toma del comando make attribute.

#### 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-mkatt, la operación parará y el atributo no será creado.  Si el trigger es after-mkatt 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. | |

Crear un repositorio

#### Nombres de los triggers

before-mkrep

after-mkrep

#### Descripción

Ejecuta los scripts de usuario al crear un repositorio.

#### Comentarios

Estos triggers se lanzan al crear un repositorio

#### Funcionamiento en

Servidor.

#### Entrada estándar

El nombre del repositorio que será creado. Esta entrada es obtenida del comando make repository.

#### 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-mkrep, la operación parará y no se creará el repositorio.  Si el trigger es after-mkrep 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. |

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

Cliente.

#### Entrada estándar

El nombre del espacio de trabajo que será creado. Esta entrada es obtenida del comando make workspace.

#### 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. | |

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

Cliente.

#### 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. | |

Obtener (update)

#### Nombre de los triggers

before-update

after-update

#### Descripción

Ejecuta los scripts de usuario al hacer un update.

#### Comentarios

Este trigger se ejectua 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

La ruta a obtener, dada por el comando update.

#### 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-update 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. |

Checkout en cliente

#### Nombre de los triggers

before-clientcheckout

after-clientcheckout

#### Descripción

Ejecuta los scripts de usuario cuando se lanza una operación de checkout en el cliente.

#### Comentarios

Este trigger se ejectua 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 tener en cuenta al crear este tipo de triggers.

#### Funcionamiento en

Cliente.

#### Entrada estándar

La entrada estándar de estos triggers recibe la lista de elementos sobre la que se ejecuta la operación.

#### 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 checkout. |
| No cero | El trigger da un error.  Si el trigger es before-clientcheckout, la operación parará y no se obtendrá el código.  Si el trigger es after-clientcheckout lanzará un error pero la operación ya ha sido realizada. |

#### Variables de entorno

|  |  |
| --- | --- |
| PLASTIC\_USER | El usuario que comenzó la operación de checkout. |
| PLASTIC\_CLIENTMACHINE | La máquina de cliente que comenzó la operación. |
| PLASTIC\_COMMENT | Comentario introducido en la operación de checkout |

Checkin en cliente

#### Nombre de los triggers

before-clientcheckin

after-clientcheckin

#### Descripción

Ejecuta los scripts de usuario cuando se lanza una operación de checkin en el cliente.

#### 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 tener en cuenta al crear este tipo de triggers.

#### Funcionamiento en

Cliente.

#### Entrada estándar

La entrada estándar de estos triggers recibe la lista de elementos sobre la que se ejecuta la operación.

#### 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 checkin. |
| No cero | El trigger da un error.  Si el trigger es before-clientcheckin, la operación parará y no se obtendrá el código.  Si el trigger es after-clientcheckin lanzará un error pero la operación ya ha sido realizada. |

#### Variables de entorno

|  |  |
| --- | --- |
| PLASTIC\_USER | El usuario que comenzó la operación de checkin. |
| PLASTIC\_CLIENTMACHINE | La máquina de cliente que comenzó la operación. |
| PLASTIC\_COMMENT | Comentario introducido en la operación de checkin. |

# Ejemplos

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

## Proteger

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

**cm maketrigger before-checkin “apply jindent” “ruby c:\triggers\jindent.rb”**

### 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 comandonSpecs(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  
comandonSpecs("cm cat -")  
  
# Apply action on files  
$files.each { |file| system("jindent \"#{file}\"") }  
  
# shelve files  
comandonSpecs("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”**

### 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 comando de creación del trigger:

**cm maketrigger before-checkin “comment required” “ruby c:\triggers\check-comments.rb”**

### 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”**

## Crear una etiqueta

### 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”**

## Client checkout

### Actualizar ficheros antes de hacer checkout

El script en ruby sería:

#!/usr/bin/ruby

files = ''

STDIN.readlines.each do |line|

files << " " << line

end

system("cm update #{files}")

Y el comando para crear el trigger:

**cm maketrigger before-clientcheckout “actualizar antes de checkout” “ruby c:\plastic\triggers\update-antes-co.rb”**