

Versión 3.0

La guía para establecer la seguridad de su sistema Plastic SCM

Plastic SCM **Seguridad**

© 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.

**Contenido**

[1. Objetivos 2](#_Toc262202489)

[2. Objetos seguros 3](#_Toc262202490)

[2.1. Estructura de una ACL 5](#_Toc262202491)

[2.2. Permisos disponibles 5](#_Toc262202492)

[2.3. Cálculo de permisos 7](#_Toc262202493)

[3. Usuarios y grupos 9](#_Toc262202494)

[3.1. Usuarios especiales 11](#_Toc262202495)

[3.1.1. ALL USERS 11](#_Toc262202496)

[3.1.2. OWNER 11](#_Toc262202497)

[4. Fijar permisos desde la herramienta gráfica 12](#_Toc262202498)

[4.1. Fijando permisos básicos 12](#_Toc262202499)

[4.2. Información avanzada 13](#_Toc262202500)

[4.3. Romper herencia 14](#_Toc262202501)

[4.4. Reaplicar la herencia 15](#_Toc262202502)

[4.5. Extender herencia 15](#_Toc262202503)

[5. Configuración de la seguridad desde línea de comandos 17](#_Toc262202504)

[5.1. Mostrar permisos 17](#_Toc262202505)

[5.2. Fijar permisos 19](#_Toc262202506)

[5.3. Manipular la herencia 20](#_Toc262202507)

[6. Configurar la seguridad en Plastic SCM 22](#_Toc262202508)

[7. Casos de estudio 23](#_Toc262202509)

[7.1. Caso 1: Restringir ramas de integración 23](#_Toc262202510)

[7.2. Caso 2: Acceso restringido a directorios de código fuente 24](#_Toc262202511)

[7.3. Caso 3: Acceso restringido a ramas 24](#_Toc262202512)

[7.4. Caso 4: Consultores invitados 24](#_Toc262202513)

[7.5. Caso 5: Repositorio de sólo lectura para un grupo 25](#_Toc262202514)

[7.6. Caso 6: Acceso a integradores en la rama principal 25](#_Toc262202515)

[7.7. Caso 7: Acceso a integradores en main y mantenimiento 25](#_Toc262202516)

[7.8. Caso 8: Sólo un propietario de un check out puede deshacerlo 25](#_Toc262202517)

[7.9. Caso 9: Abrir un repositorio a un determinado grupo 26](#_Toc262202518)

**Tablas**

[Tabla 1. Descripción detallada de permisos 7](#_Toc262202519)

**Figuras**

[Figura 1. Jerarquía de permisos en un repositorio 4](#_Toc262202520)

[Figura 2. Jerarquía de permisos en un espacio de trabajo 5](#_Toc262202521)

[Figura 3. Lista de permisos disponibles 5](#_Toc262202522)

[Figura 4. Ejemplo de cálculo de permisos 8](#_Toc262202523)

[Figura 5. Ejemplo: usuario en dos grupos. Permisos en la rama br:/main. 10](#_Toc262202524)

[Figura 6. Ejemplo de seguridad: permisos denegados en grupos 10](#_Toc262202525)

[Figura 7. Diálogo de permisos 13](#_Toc262202526)

[Figura 8. Diálogo avanzado de permisos 14](#_Toc262202527)

[Figura 9. Estructura de directorios de ejemplo antes de extender herencia 15](#_Toc262202528)

[Figura 10. Estructura de directorios de ejemplo tras extender herencia 15](#_Toc262202529)

[Figura 11. Ejemplo: eliminación de una ACL al extender la herencia 16](#_Toc262202530)

[Figura 12. Comando para mostrar los permisos de un ítem 18](#_Toc262202531)

[Figura 13. Información de ACL extendida 18](#_Toc262202532)

[Figura 14. Herramienta CLI mostrando permisos de una revisión 18](#_Toc262202533)

[Figura 15. Explicación detallada de los permisos de una revisión 19](#_Toc262202534)

[Figura 16. Explicación detallada de la sintaxis del comando *acl* 19](#_Toc262202535)

[Figura 17. Denegando el permiso de *check out* a la rama principal 20](#_Toc262202536)

[Figura 18. Información extendida tras realizar cambios 20](#_Toc262202537)

[Figura 19. Ejemplo de cortar herencia copiando 21](#_Toc262202538)

[Figura 20. Ejemplo de añadir herencia 21](#_Toc262202539)

# Objetivos

Los principales objetivos del sistema de seguridad de Plastic SCM son:

* Proporcionar un mecanismo de control de acceso a los repositorios y ser capaz de restringir ciertas operaciones. Muchas empresas requieren poder fijar diferentes permisos a diferentes proyectos, usuarios y grupos, siendo capaces de limitar el acceso efectivo y de evitar que ciertos datos sensibles puedan ser accedidos o modificados por personal no autorizado.
* Promover políticas de desarrollo y de puesta en producción (deployment). Incluso en empresas con políticas de acceso abiertas puede ser necesario restringir el acceso a ciertas partes de los repositorios, no por razones de seguridad sino para prevenir errores. De ese modo en muchos grupos de desarrollo se restringe el acceso a la línea principal de desarrollo (la rama br:/main en Plastic SCM) de modo que únicamente los *integradores* puedan modificarla. La razón no es una falta de confianza en el resto del equipo sino prevenir errores innecesarios. Un gran número de escenarios similares se pueden configurar mediante Plastic SCM.

Siguiendo con la filosofía de diseño del resto del sistema, la seguridad permite un mecanismo muy flexible y potente para fijar permisos y configurar escenarios de uso diferentes. Plastic SCM facilita el manejo de la seguridad sin necesidad de escribir complejos *triggers*

# Objetos seguros

Los siguientes objetos tienen información de seguridad en Plastic SCM y tienen una lista de control de acceso (ACL: Access Control List) asociada. La Figura 1. Jerarquía de permisos en un repositorio y la Figura 2 proporcionan información más detallada.

* **Servidor**: cada servidor tiene una ACL asociada. Por defecto todos los objetos dentro de un servidor heredan sus permisos de él.
* **Repositorio**: un sistema puede tener múltiples repositorios. Cada repositorio hereda sus permisos del *servidor de repositorios*. Cada repositorio tiene una ACL asociada. Por defecto todos los objetos heredan sus permisos de su repositorio.
* **Ítem**: los *ítems* heredan sus permisos de un modo similar a un sistema de archivos. El ítem raíz del repositorio heredará sus permisos del repositorio, y el resto de ítems heredarán de él de manera directa o indirecta. Cada fichero hereda sus permisos de su directorio padre. La herencia de permisos no se recalcula cuando un ítem se mueve a otro directorio, para evitar posibles errores debido al desarrollo paralelo (un ítem podría estar en más de una ubicación al mismo tiempo, en ramas diferentes). En cualquier momento un usuario con privilegios puede *extender* los permisos de un directorio a todos los ítems que contiene, de manera recursiva, reestableciendo una jerarquía de permisos similar a la de un sistema de archivos.
* **Rama**: una rama hereda sus permisos de su repositorio. Por defecto los permisos de una rama afectarán a todas las revisiones que contiene.
* **Etiqueta**: las etiquetas heredan sus permisos de su repositorio.
* **Atributo**: los atributos heredan sus permisos de su repositorio.
* **Links** o enlaces: los *links* heredan sus permisos de su repositorio.
* **Todas las revisiones de una rama**: todas las revisiones en una rama de un ítem determinado compartirán, por defecto, los mismos permisos. Es posible fijar permisos a todas las revisiones de un ítem en una rama a la vez. El objeto *todas las revisiones en la rama* hereda sus permisos del ítem y de la rama. Por defecto es el único objeto que utiliza herencia múltiple de permisos. Los permisos calculados mediante *herencia múltiple* se calculan de la siguiente forma: para denegar un permiso valdrá con que esté denegado en uno de los padres, para concederlo tendrá que estar concedido en ambos.
* **Revisión**: una revisión heredará sus permisos del objeto especial *todas las revisiones en la rama*. Por defecto una revisión no tiene una ACL propia asociada, sino que usa la de su objeto padre. Cuando se cambian los permisos específicos de una revisión se crea una ACL específica para ella.

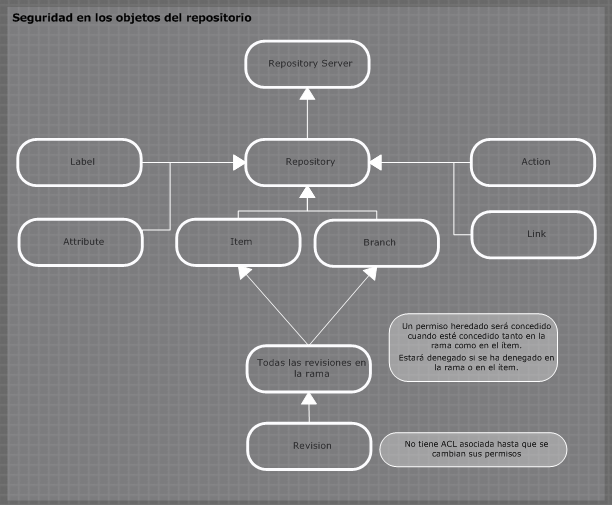


Figura 1. Jerarquía de permisos en un repositorio

## Estructura de una ACL

Una ACL consiste en lo siguiente:

* Entradas: cada usuario (SEID o SEcured IDentifier según la terminología de Plastic SCM) o grupo al que se hayan dado o denegado permisos tendrá una entrada en una ACL. Cada entrada tendrá permisos *permitidos* y *denegados*.
* Entradas de herencia: son *punteros* a las ACLs heredadas por el objeto ACL actual.

## Permisos disponibles

La Figura 3 muestra una lista de los permisos disponibles en Plastic SCM y cómo se aplican a los diferentes objetos.

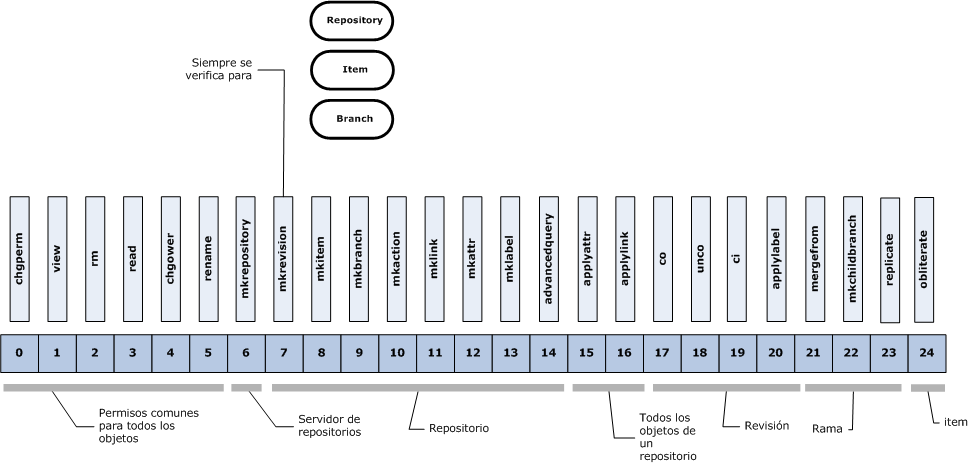


Figura 3. Lista de permisos disponibles

La Tabla 1 explica todos los permisos en detalle.

|  |  |  |
| --- | --- | --- |
| Se aplica a | Nombre | Descripción |
| Todos | chgperm | Permite o deniega cambiar los permisos a un objeto. Por defecto, activo. |
| view | Permiso para ver un objeto. Por ejemplo si un usuario tiene permiso para *ver* una rama, podrá visualizarla en la lista de ramas, pero el permiso de *read* será el que controle el acceso a los datos de la rama. Para etiquetas *view* es la capacidad de ser listadas y *read* se comprueba cuando se va a aplicar la etiqueta. Por defecto, activo. |
| rm | Eliminar un determinado objeto. Por defecto, activo. |
| read | Leer datos de un objeto. Por defecto, activo. |
| chgowner | Cambiar el propietario de un objeto. Por defecto, activo. |
| rename | Cambiar el nombre de un objeto.  Para *workspaces* es necesario para cambiar la ubicación de un workspace. Por defecto, activo. |
| Servidor de repositorios | mkrepository | Permite o deniega la creación de repositorios en un determinado servidor. Por defecto, activo. |
| Repositorio | mkrevision | *mkrevision* es un permiso especial que posibilita denegar todo tipo de creación de revisiones en un repositorio.  Se comprueba *siempre* que el repositorio, la rama y el ítem tengan ese permiso durante la creación de revisiones.  El permiso *mkrevision* se superpone sobre el permiso de *check out*. Por defecto, activo. |
| mkitem | Permite crear ítems en un repositorio. Se comprueba siempre durante operaciones de *add*. Por defecto, activo. |
| mkbranch | Controla la creación de ramas en un repositorio. Por defecto, activo. |
| mkaction | Creación de acciones (triggers) en un repositorio. Por defecto, activo. |
| mklink | Creación de links. Por defecto, activo. |
| mkattr | Creación de atributos. Por defecto, activo. |
| mklabel | Creación de etiquetas. Por defecto, activo. |
| advancedquery | Posibilidad de ejecutar consultas avanzadas en un repositorio. Por defecto, inactivo. |
| Todos los objetos dentro de un repositorio | applyattr | Permiso para aplicar un atributo. Por defecto, activo. |
| applylink | Permiso que permite que se *linke* un objeto. Por defecto, activo. |
| Revisión / Todas las revisiones de un ítem en una rama | co | Check out. Permiso para hacer check out (desproteger) una determinada revisión, creando una nueva. Por defecto, activo. |
| unco | Permiso para deshacer un checkout. Por defecto, activo. |
| ci | Permiso para hacer check in (proteger) una determinada revisión. Por defecto, activo. |
| applylabel | Permiso para aplicar etiquetas. Por defecto, activo. |
| Branch | mergefrom | Permiso para hacer merge desde una determinada rama. Por defecto, activo. |
| mkchildbranch | Permiso para crear ramas hijas de una rama. Por defecto, activo. |
| replicate | Permiso para replicar ramas. Por defecto, establecido. Da permiso únicamente para obtener los datos en el origen. Si en el destino no hay permisos suficientes, la operación de replicación fallará. Además, para que la replicación tenga éxito son necesarios los permisos de *mkbranch*, *mkchildbranch*, *mklabel*, *applylabel*, *mkrevision*. |
| Ítem | obliterate | Permiso para eliminar elementos del sistema permantemente. Estos elementos no podrán ser recuperados de ningún modo, pues se borra el ítem de la base de datos y todas sus revisiones. Por defecto, desactivado. |

Tabla 1. Descripción detallada de permisos

## Cálculo de permisos

Para calcular los permisos de un objeto se considera toda su jerarquía de permisos. Los permisos de sus *antecesores* se combinan con las entradas de ACL del propio objeto. *Combinar* significa que tanto los permisos permitidos como denegados para cada usuario y grupo se *mezclan* mediante una operación lógica OR. Por ejemplo si un repositorio no tiene el permiso *co* (check out) permitido ni denegado pero el *ítem* lo tiene permitido, la revisión tendrá el permiso.

Cuando una ACL en la cadena de herencia tiene dos antecesores se produce un caso especial de *combinación* de permisos. En lugar de simplemente mezclar ambos antecesores juntos lo que ocurre es que los permisos *concedidos* se combinan con un operador lógico *AND* mientras que los denegados se combinan con un OR. De este modo para que un permiso esté concedido *todos* los antecesores tendrán que tenerlo, sin embargo bastará con que uno tenga el permiso denegado para que finalmente el permiso efectivo esté denegado.

Una vez que los permisos se han calculado para un usuario se combinan con los permisos del grupo especial *todos los usuarios* (ALL USERS).

La Figura 4 muestra un ejemplo de cálculo de permisos. La primera fila muestra permisos concedidos y la segunda denegados. El ejemplo muestra cómo se combinan permisos de diferentes antecesores.

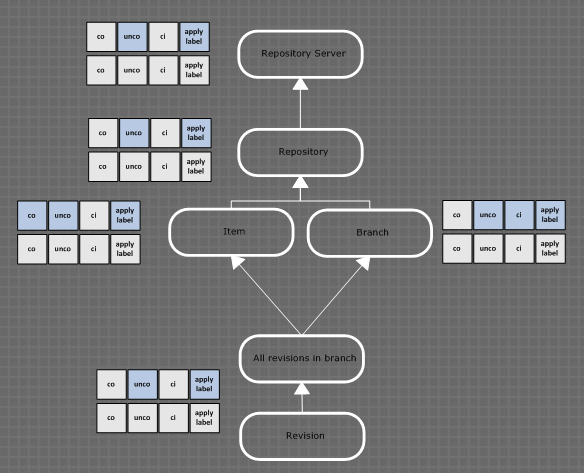


Figura 4. Ejemplo de cálculo de permisos

# Usuarios y grupos

Las ACLs especifican permisos concedidos o denegados a diferentes usuarios y grupos. Cuando los permisos de un usuario en un grupo concreto son calculados, Plastic primero resuelve la herencia de cada SEID (usuario o grupo) y depués los combina teniendo en cuenta las relaciones entre los usuarios y los grupos.

La Figura 5 ilustra un escenario en el que un usuario es a la vez desarrollador e integrador. Los permisos mostrados hacen referencia a la línea principal de desarrollo: la rama br:/main. Los integradores tienen permisos para hacer *check out*, *check in* y etiquetar en la rama principal, pero los desarrolladores no pueden hacer cambios en esa rama, así que no tienen los permisos *check in* ni *applylabel* activos.

Tenga en cuenta que en lugar de **denegar**permisos el ejemplo **no los asigna** a los desarrolladores: denegar es más restrictivo y siempre prevalecerá en un escenario con múltiples grupos.

Plastic combinará los permisos de desarrolladores e integradores para calcular los permisos del usuario que finalmente tendrá permisos para realizar las operaciones en la rama principal porque pertenece al grupo de integradores.

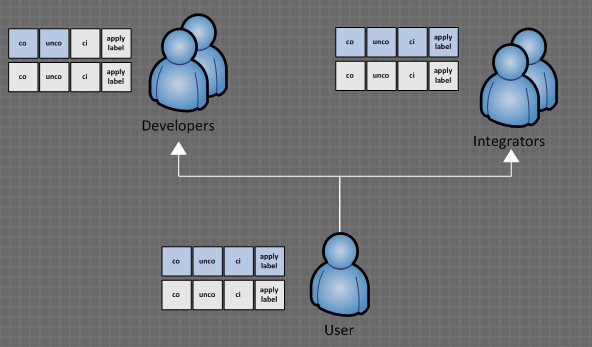


Figura 5. Ejemplo: usuario en dos grupos. Permisos en la rama br:/main.

La Figura 6 muestra cómo se aplican permisos denegados en un escenario con múltiples grupos. A diferencia de lo que ocurre con permisos *no habilitados* (no permitidos ni denegados), los denegados siempre prevalecerán en las operaciones de combinación.

El ejemplo muestra los permisos de un repositorio: el Group 1 tiene el permiso de *mkbranch* (crear rama) denegado: ningún usuario en el Group 1 podrá crear ramas en el repositoio.

El Group 2 tiene todos los permisos concedidos.

El usuario del ejemplo pertenece a ambos grupos así que plastic denegará el permiso *mkbranch*: la combinación de los permisos de ambos grupos hace que los denegados permanezcan.

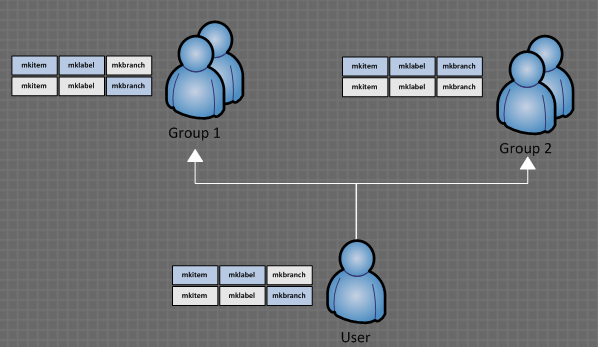


Figura 6. Ejemplo de seguridad: permisos denegados en grupos

## Usuarios especiales

El sistema de seguridad de Plastic SCM incluye un usuario y un group especiales que ayudan en la configuración de políticas de seguridad.

### ALL USERS

Hace referencia a todos los usuarios reconocidos por Plastic SCM. Fijando permisos a este grupo especial todos los usuarios se verán afectados. Por defecto, tras la instalación, el sistema está totalmente abierto así que el group *todos* dispone de todos los permisos.

Es importante destacar que denegar permisos al grupo *all users* afectará a *todos* los usuarios del sistema, prevalenciendo sobre cualquier otro permiso asignado.

### OWNER

El *owner* o propietario es un usuario especial que permite asignar o denegar permisos al propietario (puede cambiarse pero por defecto sera el usuario que ha creado el objeto).

Cambiando los permisos del usuario *propietario* el administrador puede configurar lo que cada propietario puede hacer.

Por ejemplo puede usarse para prevenir que nadie pueda deshacer un checkout excepto quien lo haya creado.

# Fijar permisos desde la herramienta gráfica

La configuración de seguridad puede hacerse desde la herramienta gráfica mediante el *diálogo de permisos*. El diálogo puede mostrar y editar la ACL de cualquier objeto en Plastic SCM.

## Fijando permisos básicos

La Figura 7 muestra los principales elementos del diálogo de permisos. El diálogo consiste en dos *pestañas*, la primera muestra los usuarios y grupos de la ACL y sus permisos, y el segundo muestra información y operaciones avanzadas. En la parte superior del diálogo se muestra una especificación completa del objeto cuya ACL asociada se está mostrando.

La *pestaña* de permisos muestra la siguiente información:

* Una lista de los usuarios y grupos presentes en la ACL. Cada uno de ellos tendrá una lista de permisos concedidos, denegados o no asignados.
* Una lista de los permisos y el estado de cada uno de ellos se muestra justo debajo de la lista de usuarios y grupos. Cada vez que el usuario selecciona un elemento (usuario o grupo) en la lista superior, el contenido de la lista de permisos se actualiza.
* La información sobre el propietario del objeto se muestra en la parte inferior.
* Los botones de *añadir* y *borrar* situados bajo la lista de usuarios y grupos permiten añadir y eliminar usuarios y grupos a la ACL. Cuando un usuario o un grupo es *heredado* desde un ACL padre, el sistema no permitirá eliminarlo.
* El botón *propietario* permite cambiar el propietario del objeto.

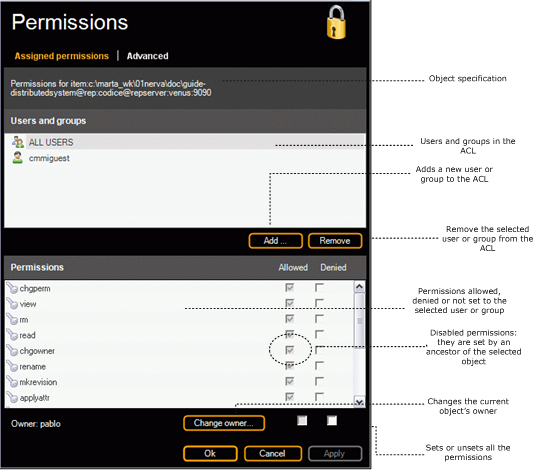


Figura . Diálogo de permisos

## Información avanzada

La *pestaña* de permisos avanzados muestra información adicional sobre la ACL. La Figura 8 muestra una ACL de ejemplo y toda su información avanzada.

La lista de *permisos heredados* detalla información sobre cada uno de los permisos presentes en la ACL. Muestra qué usuario tiene cada permiso, si está permitido o denegado y cuál es el objeto desde el que se hereda.

La especificación de objeto que se muestra en la columna *herencia* se calcula de la siguiente manera: considere que está mostrando permisos para un ítem determinado, como hace la Figura 8. Indica que tiene el permiso *read* permitido y que hereda desde el *servidor de repositorios*. ¿Qué significa? El ítem hereda sus permisos del repositorio que a su vez los hereda del servidor. Así que el objeto *más alto en la jerarquía* que define los permisos es el que se muestra.

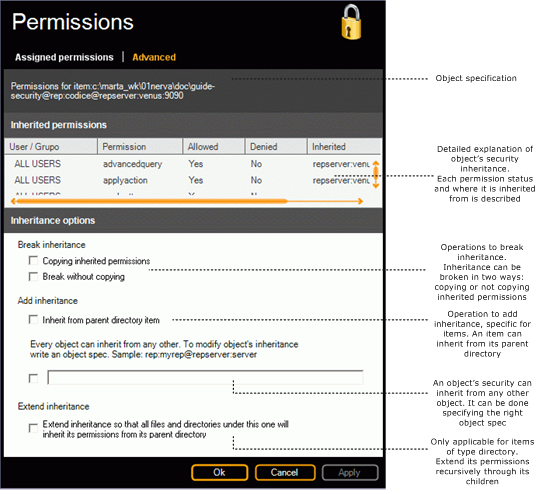


Figura 8. Diálogo avanzado de permisos

## Romper herencia

La parte inferior de la pestaña de opciones avanzadas del diálogo de permisos se centra en operaciones de herencia.

La ACL de un objeto puede heredar de las ACLs de otros objetos, pero esa herencia también se puede romper. La herencia se puede *cortar* de dos maneras: copiando o no copiando los permisos heredados.

* Romper herencia copiando: cuando se fija esta opción las entradas de herencia de la ACL del objeto se eliminarán, pero todos los permisos heredados se copiarán sobre la ACL. Por ejemplo si el permiso de *read* está heredado del servidor y asignado a *ALL USERS*, tras cortar la herencia copiando esa entrada estará presente en la ACL del propio objeto.
* Romper herencia sin copiar: si se fija esta opción las entradas de herencia se eliminarán de la ACL pero los permisos no se copiarán. Si la ACL tenía sus propias entradas se mantendrán, pero si no tenía entradas la ACL quedará totalmente vacía.

## Reaplicar la herencia

La herencia también se puede *reaplicar* o añadir usando las opciones de *agregar herencia* de la pestaña de opciones avanzadas.

Si la ACL que se está editando corresponde a un ítem, entonces la opción de *reheredar del directorio padre* estará habilitada, y el usuario podrá restaurar la seguridad de un fichero o directorio haciendo que herede de su directorio contenedor.

La segunda opción de *agregar herencia* permite introducir una especificación completa de objeto. El sistema añadirá a la herencia la ACL del objeto especificado.

## Extender herencia

La opción *extender herencia* está disponible para ítems de tipo directorio. El usuario podrá extender los permisos de un ítem tipo directorio a todos los ficheros y subdirectorios que contenga, recursivamente, reemplazando sus permisos actuales por aquellos fijados en su directorio. Los permisos no se copian sino que la ACL de cada objeto se *limpia* y hereda de su directorio padre.

El ejemplo que se muestra en la Figura 9 muestra una estructura de directorios en la que dos ítems tienen ACLs propios que no heredan del directorio padre.



Figura 9. Estructura de directorios de ejemplo antes de extender herencia

Después de expandir la herencia desde el directorio raíz del ejemplo, las ACLs de los ficheros y directorios heredan directamente de las ACLs de sus directorios padre. Sus permisos serán los que tengan asignados sus directorios padre, y la jerarquía de seguridad se reestablece. La Figura 10 muestra la situación tras expandir la herencia en el directorio raíz del ejemplo.



Figura 10. Estructura de directorios de ejemplo tras extender herencia

Si un fichero o un directorio dentro del árbol de directorios sobre el que se está aplicando la *expansión* tiene su propia ACL que a su vez tiene definidas entradas de herencia, sus entradas serán eliminadas y la herencia fijada al ítem correspondiente a su directorio padre. La Figura 11 muestra un ejemplo que ilustra la situación descrita.



Figura 11. Ejemplo: eliminación de una ACL al extender la herencia

# Configuración de la seguridad desde línea de comandos

La seguridad puede ser totalmente configurada desde la línea de comandos usando los comandos *cm acl* y *cm sa*. Para listar todos los permisos es posible usar *cm sp* o *cm showpermissions*.

Para obtener ayuda acerca de los comandos de seguridad teclee: *cm help acl*, *cm help sa* o *cm help showpermissions*.

## Mostrar permisos

El comando *cm showacl* o símplemente *cm sa*, muestra los permisos de un objeto a partir de una *especificación de objetos*. La Figura 12 es un ejemplo en el que se muestran los permisos de un ítem de tipo directorio denominado *01nerva*. El comando *cm sa* le indica al usuario qué permisos están disponibles, a qué usuarios o grupos han sido asignados y de dónde vienen considerando la jerarquía de seguridad.



Figura 12. Comando para mostrar los permisos de un ítem

El modificador *– -extended* permite al usuario obtener información más detallada acerca de cómo se asignan realmente los permisos. La Figura 13 muestra un ejemplo completo de una ejecución con la opción *- - extended*..



Figura 13. Información de ACL extendida

Los dos ejemplos anteriores se centran en permisos de ítems, pero la línea de comandos puede trabajar con cualquier tipo de objeto. La Figura 14 muestra los permisos de una revisión concreta de un directorio. Para elegir una revisión concreta se utiliza una especificación de revisión.



Figura 14. Herramienta CLI mostrando permisos de una revisión

La Figura 15 incluye una explicación detallada de cómo se fijan los permisos en la revisión considerando toda la jerarquía de herencia múltiple de seguridad.



Figura 15. Explicación detallada de los permisos de una revisión

## Fijar permisos

Para modificar permisos desde la línea de comandos debe usarse el comando *cm acl*. La Figura 16 ilustra todas las opciones del comando *cm acl*. Usando el comando *acl* será posible añadir o eliminar usuarios y grupos de una ACL y conceder, denegar y anular concesiones y denegaciones. La herencia también puede manipularse mediante los modificadores *–cut*, *-cutncpy* y *–inherit*.



Figura 16. Explicación detallada de la sintaxis del comando *acl*

Usando el mismo ejemplo introducido en el punto anterior, la Figura 17 deniega el permiso de *co* a la rama *main* y verifica cómo los permisos de la revisión 1 del directorio *01nerva* se ven alterados.



Figura 17. Denegando el permiso de *check out* a la rama principal

La Figura 18 ilustra el ejemplo en mayor detalle usando la opción *- -extended*.

Ahora la ACL asociada a la rama, que antes sólo tenía una entrada de herencia, introduce su propia entrada de ACL para el grupo especial *ALL USERS*, indicando que se deniegue el permiso *co*.

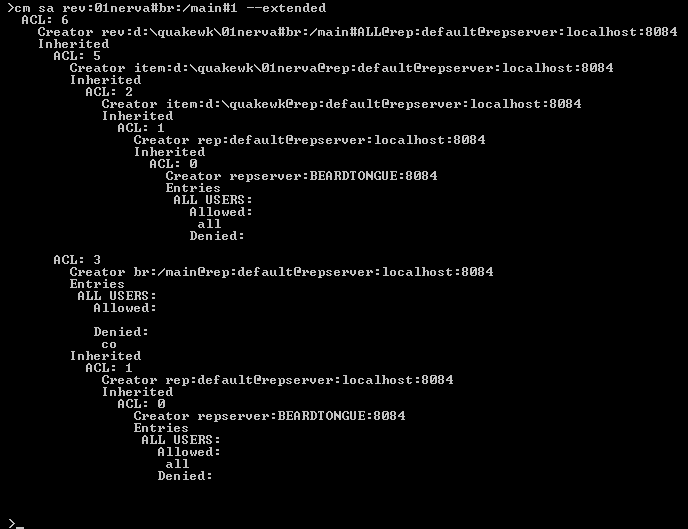


Figura 18. Información extendida tras realizar cambios

## Manipular la herencia

La herencia puede manipularse usando el comando *cm acl* y los modificadores *–cut*, *-cutncpy* y *–inherit*.

La Figura 19 muestra un ejemplo en el que la rama *br:/main* hereda sus permisos del servidor de repositorios y también tiene un permiso en su propia ACL denegado para todos los usuarios. Tras cortar la herencia copiando ambas entradas aparecen como *no heredadas* y se muestran con el comando *cm sa*.



Figura 19. Ejemplo de cortar herencia copiando

Es posible también añadir herencia de seguridad a un objeto, haciendo que sus permisos hereden de la ACL de cualquier objeto del sistema (teniendo en cuenta las restricciones de jerarquía: ningún objeto puede heredar de otro que esté por debajo en la jerarquía, por ejemplo un repositorio no puede heredar de una revisión).

La Figura 20 muestra cómo la herencia puede añadirse de nuevo a la rama *br:/main* del ejemplo anterior.



Figura 20. Ejemplo de añadir herencia

# Configurar la seguridad en Plastic SCM

Cuando Plastic SCM se instala por primera vez, la configuración de seguridad está totalmente abierta. Esto significa que cualquier usuario que pueda ser autentificado por el sistema tendrá acceso total.

Si en su empresa es necesario establecer restricciones de seguridad, ya sean para prevenir accesos no deseados o para reforzar la política de desarrollo, considere lo siguiente:

* Cambie los permisos del servidor de repositorios como primer paso para establecer una política de seguridad. Cambiando los permisos en el elemento de mayor nivel hará que todos los objetos queden asegurados.
* Defina qué usuarios y grupos tendrán acceso al sistema Plastic SCM, y asigne los permisos adecuados en el servidor de repositorios. Posteriormente podrá configurar privilegios específicos en repositorios, ramas e incluso ítems si es necesario.
* Asegúrese de que comprende correctamente el significado de ada uno de los permisos y su impacto en el sistema, leyendo con atención este manual.

# Casos de estudio

## Caso 1: Restringir ramas de integración

Los desarrolladores usan varias ramas para trabajar, siguiendo un patrón de *rama por tarea*. Existen dos grupos diferentes: desarrolladores e integradores. Algunos desarrolladores pueden actuar como integradores.

La política de seguridad a implementar será la siguiente: los desarrolladores no podrán hacer *check out*, *check in* o etiquetar en las ramas de integración, sin embargo los integradores sí que podrán.

Es importante destacar que un usuario que sea desarrollador e integrador deberá tener los privilegios de integrador.

Para configurar la seguridad el administrador hará lo siguiente:

* En las ramas de integración rompa la herencia copiando. De este modo las ramas de integración continuarán teniendo los mismos permisos pero ahora no será posible alterarlos mediante la herencia.
* Deshabilite los permisos de *check out*, *check in* y *applylabel* para los desarrolladores en las ramas de integración. Tenga en cuenta que es necesario *deshabilitarlos*, no denegarlos. De ese modo los integradores tendrán esos permisos pero los desarrolladores no dispondrán de la capacidad de hacer *check out, check in* o aplicar etiquetas. Un usuario que pertenezca a ambos grupos tendrá los permisos concedidos como resultado de la combinación de ambos grupos.

## Caso 2: Acceso restringido a directorios de código fuente

Exiten dos grupos diferentes en la empresa: desarrolladores y analistas. Los desarrolladores tendrán acceso total al árbol de directorios de código fuente pero los analistas sólo acceso de lectura.

Los pasos para configurar el escenario serán:

* Asegúrese de que los permisos están correctamente fijados en todo el árbol de directorios de código fuente *expandiendo* la seguridad del directorio raíz de dicho árbol (pestaña de opciones avanzadas del diálogo de permisos).
* Corte la herencia copiando en el directorio raíz del código fuente.
* Deshabilite los permisos de *ci*, *co* y *applylabel* el la raíz del directorio de código fuente para el grupo de analistas. Mantenga los permisos para los desarrolladores.

## Caso 3: Acceso restringido a ramas

Un equipo de desarrollo en el que existen los siguientes miembros:

* Administrador de sistemas
* Desarrolladores
* Grupo de integradores

Los desarrolladores sólo pueden trabajar en ramas de desarrollo.

Los integradores sólo pueden hacer cambios en las ramas de integración.

Los administradores tienen acceso total.

Si un usuario pertenece a varios grupos debería tener todos sus privilegios combinados (si un desarrollador es también administrador, debería tener acceso total).

Pasos para configurar el escenario

* Deshabilitar el permiso *mkbranch* a nivel del repositorio para integradores y desarrolladores.
* En ramas de desarrollo: los integradores tendrán los permisos *co*, *ci*, *mkrev* y *applylabel* deshabilitados. Los desarrolladores todos los permisos concedidos.
* En ramas de integración: los desarrolladores tendrán permisos de *co*, *ci*, *mkrev*y *applylabel* deshabilitados. Los integradores todos los permisos concedidos.

## Caso 4: Consultores invitados

Cuando un grupo de consultores externos trabajan junto al equipo de desarrollo durante una auditoría de calidad (por ejemplo una evaluación CMMi), será necesario que cuenten con acceso a los repositorios, pero asegurándose de que no puedan realizar cambios ni al código ni a la documentación.

Para configurar este escenario usando Plastic SCM siga los siguientes pasos:

* Cree un grupo *invitados* en su sistema operativo.
* Añada el grupo a los repositorios necesarios y deniegue los permisos *mkrevision*, *mkbranch*, *mkchildbranch*, *mkitem*, *co* y *ci.*

## Caso 5: Repositorio de sólo lectura para un grupo

Para restringir el acceso a un grupo determinado de usuarios a un repositorio concreto, de modo que puedan tener acceso de lectura pero no ser capaces de modificar ningún objeto, siga los siguientes pasos:

* Añada el grupo al repositorio.
* Deniegue (o deshabilite, dependiendo de si necesita que los miembros de ese grupo tengan los privilegios menos restrictivos en caso de pertencer a algún otro) los siguientes permisos: *chgpermission*, mkrevision, *mkbranch*, *mkchildbranch*, *mkattr*, *applylabel*, *mklabel*, *unco*, *ci*.

## Caso 6: Acceso a integradores en la rama principal

En este escenario sólo los integradores pueden hacer *check in* sobre la rama principal, aunque los desarrolladores podrían hacer check out de ficheros y directorios, pero nunca *check in*.

Para configurar la seguridad considere el siguiente paso:

* Deshabilite el permiso *ci* a los desarrolladores en la rama principal. De ese modo un usuario que sea desarrollador e integrador podrá hacer *check in*, pero los que sólo sean desarrolladores no podrán hacer check in.

## Caso 7: Acceso a integradores en main y mantenimiento

El propósito es restringir el acceso a las revisions almacenadas tanto en la rama principal como en la de mantenimiento. Tan solo los integradores podrán realizar cambios en estas ramas. Los desarrolladores solo podrán ver la información almacenada en estas ramas pero no podrán realizar cambios.

Para implementer el escenario necesario hay que realizar los siguientes pasos:

* Inutilizar para el grupo de desarrolladores todos los permisos menos el de ver y leer.

## Caso 8: Sólo un propietario de un check out puede deshacerlo

Este escenario es útil para prevenir que un usuario pueda *deshacer* un checkout de otro. En un sistema con seguridad habilitada solo el propietario y el administrador deberían tener permiso para deshacer checkouts (uncheckout).

Configurar esta política de seguridad es muy sencillo con Plastic SCM: deshabilite el permiso *unco* para el grupo especial ALL USERS a nivel de repositorio o de servidor de repositorios (según quiera hacer la política global o local para un repositorio concreto), y deje el permiso habilitado sólo para el usuario especial OWNER y para el grupo de administradores.

## Caso 9: Abrir un repositorio a un determinado grupo

Considere una empresa que desarrolle varios proyectos. Cada uno de ellos será desarrollado por un grupo concreto al que se asignará un determinado repositorio.

Cada repositorio puede considerarse un *componente* en el que se implementará una cierta funcionalidad. De ese modo, en un instante concreto, un segundo grupo puede necesitar *acceso de sólo lectura* a un repositorio distinto del que utilizan para su trabajo habitual. Podrán usar dicho repositorio como un *subproducto* para su proyecto.

Configurar un escenario como el descrito en Plastic SCM implicará las siguientes consideraciones:

* Los dos repositorios se denominarán, por ejemplo, *rep00* y *rep01.*
* Los dos grupos *grp00* y *grp01*.
* *grp00* tiene acceso total a *rep00*y *grp01* acceso a *rep01*. De este modo cada grupo puede autogestionar su repositorio.
* *grp01* necesitará acceso de lectura a *rep00*.
* El administrador dará permisos *read* y *view* a la rama principal de *rep00* para el grupo *grp01*.