MKS

Integrity

MKS Integrity 2009 CLI Reference Guide

Configuration Management

MKS Integrity 2009 CLI Reference Guide for Configuration Management

Corporate Headquarters

410 Albert Street
Waterloo, ON N2L 3V3
Canada

tel: 519 884 2251
fax: 519 884 8861
sales (toll free): 800 265 2797

www.mks.com

Worldwide Offices:

1815 South Meyers Rd.
Suite 220
Oakbrook Terrace, IL USA
60181
tel: 630 827 4900
fax: 630 629 9167
sales (toll free): 800 633 1235

12701 Fair Lakes Circle
Suite 350
Fairfax, VA USA
22033
tel: 1 703 803 3343
fax: 1 703 803 3344
sales (toll free): 1 800 637 8034

Martinstraße 42-44
73728 Esslingen
Germany
tel: +49 711 351775 0
fax: +49 711 351775 7555

Third Floor, Duke's Court
Duke Street, Woking
Surrey
GU21 5BH
United Kingdom
tel: +44 (0)1483 733900
fax: +44 (0)1483 733901
sales: +44 (0)1483 733919

3 Killiney Road
#07-05 Winsland House 1
Singapore 239519
tel: +65-6732-8768
fax: +65-6732-0768

5th Floor, Unosawa Tokyu Building
1-19-15, Ebisu, Shibuya-ku,
Tokyo 150-0013, Japan
tel: +81-3-5422-9503
fax: +81-3-5422-9509

Rev. DRT2009SP1

MKS Inegrity provides a complete command line interface (CLI) to manage projects and archives. The CLI and graphical user interfaces (GUI) provide the exact same functionality. In fact, in most cases, these two interfaces are interchangeable -- you can easily alternate between these interfaces and they are quickly synchronized to facilitate your user experience.

MKS Integrity configuration management commands follow the `si` prefix. For example, `si about` displays the product information for MKS Integrity, while `si co -l samplefile.c` checks out a locked copy of a Sandbox member file.

Each command allows a limited set of options. Single letter options must always be preceded by a single dash (`-`), while longer option strings must be preceded by a double dash (`--`). The long strings are not case sensitive, but are shown in mixed case to facilitate readability.

To see a list of the available MKS Integrity configuration management commands and brief descriptions of each, type `si`.

To see a list of options available to a particular command, simply append `-?` or `--usage` to the command, for example,

```
si add --usage
```

In options, square brackets indicate optional strings, for example, the `no` is an optional prefix in `--[no]batch`. The two ways to use this option would be `--nobatch` or `--batch`.

For information on Authorization Administrator commands and permissions, see the *MKS Integrity Server Installation and Configuration Guide*.

Note: The terms item and issue refer to the same Integrity object and are indistinguishable. Issue is a term embedded in legacy command and option names; therefore, item and issue are used interchangeably in the CLI documentation.

Introduction

MKS Integrity Configuration Management Commands

- si createdevpath
- si createmetricinfo
- si createproject
- si createsandbox
- si createsubproject
- si deletelabel
- si deleteprojectlabel
- si deleterevision
- si demote
- si demoteproject
- si diff
- si difffiles
- si discardcp
- si disconnect
- si drop
- si dropdevpath
- si dropmemberattr
- si dropproject
- si dropprojectattr
- si dropsandbox
- si dropsandboxattr
- si edit
- si editcp
- si exit
- si freeze
- si gui
- si import
- si importproject
- si importsandbox
- si integrations
- si loadrc
- si locate
- si lock
- si locks
- si makewritable
- si memberinfo
- si merge
- si mergebranch
- si mods
- si move
- si movesubproject
- si opencp
- si print
- si projectadd
- si projectci
- si projectco
- si projectinfo
- si projects
- si promote
- si promoteproject
- si rejectcp
- si rename
- si report
- si restoreproject
- si resync
- si resynccp
- si retargetsandbox
- si revert
- si revisioninfo
- si rlog

si sandboxes
- si sandboxinfo
- si serveralerts
- si servers
- si setmemberrule
- si setprefs
- si setprojectdescription
- si sharesubproject
- si snapshot
- si submit
- si submitcp
- si thaw
- si unlock
- si unlockarchive
- si updatearchive
- si updateclient
- si updaterevision
- si viewcp
- si viewcps
- si viewhistory
- si viewlabels
- si viewlocks
- si viewmetricsinfo
- si viewnonmembers
- si viewprefs
- si viewproject
- si viewprojecthistory
- si viewprojectmetrics
- si viewrevision
- si viewsandbox
- si viewserveralert

Miscellaneous Information

- ACL
- cc
- diagnostics
- envvar
- makefile
- options
- preferences
- rcsedit

MKS Toolkit Utilities

MKS Integrity provides MKS Toolkit command line utilities on the client. For information on using a command, consult its man page from the console. The following commands are available:

- a2p
- alias
- ar
- break
- cc
- cd
- cmp
- colon
- command
- continue
- cp
- date
- diff
- diff3

- dlg
- dot
- echo
- envvar
- eucm
- eval
- exec
- exit
- export
- fc
- functions
- getopts
- hash
- help
- history
- integer
- jobs
- ld
- let
- ln
- make
- makefile
- man
- manstrip
- mksinfo
- mks_env
- more
- mv
- nm
- print
- r
- read
- readonly
- return
- rm
- set
- sh
- shift
- shopt
- times
- touch
- trap
- type
- typeset
- umask
- unicode
- unset
- wait
- whence
- which

si intro

introduction to reference pages

DESCRIPTION

A description of an individual topic (for example, a command) is loosely called the ***reference page*** for that topic, even if it is actually several pages long.

There are three alternatives for accessing the reference pages to each MKS Integrity configuration management command through the CLI **man** command.

First, you may simply type the `si` prefix and the command together as one word. Second, you may type the `si` prefix and the command with an underscore between them. Third, you may quote the `si` prefix and the command, with a space in the middle. For example:

```
man siabout
man si_about
man "si about" (Windows client only)
```

See the reference page for the man command itself, by typing `man man`, to find out more details.

This reference page describes the parts of a reference page with examples taken from real MKS Integrity reference pages.

The following sections discuss the various elements of a reference page.

Name

The *NAME* section provides the name of the command and a brief functional description.

Synopsis

In the reference page for a command, the *SYNOPSIS* section provides a quick summary of the command's ***format***. For example, here is the synopsis of the **si add** command.

si add [**--archive=***filename*] [**--author=***name*] [(**--binaryFormat**|**--textFormat**)] [(**--cpid=***ID*|**--changePackageId=***ID*)]
[**--[no]defer**] [**--deploytype=***value*] [**--[no|confirm]closeCP**] [**--issueId=***ID*] [**--description=***desc*]
[**--descriptionFile=***file*] [**--issueId=***ID*] [(**-l**|**--lock**)]
[**--onExistingArchive=[confirm|sharearchive|newarchive|cancel]**] [**--[no]createSubprojects**]
[**--[no]retainWorkingFile**] [**--[no]saveTimestamp**] [**--[no]unexpand**] [(**-r** *rev*|**--revision=***rev*)]
[(**-R** |**--[no|confirm]recurse**)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*]
[**--password=***password*] [**--user=***name*] [**--cwd=***directory*] [(**-F** *file*|**--selectionFile=***file*)] [**--forceConfirm=***[yes|no]*]
[(**-g**|**--gui**)] [(**-N**|**--no**)] [**--[no]batch**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [(**-?**|**--usage**)]
[(**-Y**|**--yes**)] [**-R**|**--[no|confirm]recurse**] [**--[no|confirm]includeFormers**] [**--exclude=***file:pattern,dir:pattern...*]
[**--include=***file:pattern,dir:pattern...*] *nonmember...*

The synopsis takes the form of a command line as you might type it into the system; it shows what you can type in and the order in which you should do it. The parts that are enclosed in square brackets are ***optional***; you may omit them if you choose. Parts that are not enclosed in square brackets must be present for the command to be correct.

The synopsis begins with the name of the command itself. The MKS Integrity configuration management commands all include the ***si*** prefix. In MKS Integrity documentation, command names are always written in **bold Courier** font.

After the command name comes a list of options. A typical MKS Integrity command option consists of either a single dash (`-`) followed by a single character, usually an uppercase or lowercase letter, or it may consist of a double dash (`--`) followed by a multi-character option name. Often there are single-character and multi-character options that do the same thing. The multi-character strings are not case sensitive, but are shown in mixed case to facilitate readability. For example, you might have **-L** or **--label**.

The synopsis line shows options in **bold Courier** font. Note that the case of single-character options is important; for example, in the synopsis of **si add**, **-r** and **-R** are different options, with different effects.

Furthermore, in the synopsis all options are shown in one long string. Other common option forms are:

```
-r rev
--revision=rev
```

where *rev* or, in some cases, *value* provides extra information for using that option. For example, the **si lock** command locks Sandbox

members; here's the command's synopsis:

si lock [**--[no|confirm]branch**] [(**--cpid=***ID*|**--changePackageId=***ID*)] [**--issueId=***ID*] [(**-r** *rev*|**--revision=***rev*)] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--[no]recordAsWorkInProgress**] [**--[no|confirm]revisionMismatchIsError**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

In this example, there is the option

-r *rev*

This option tells the **si lock** command which revision to lock. In a command synopsis, anything appearing in *italics* is a ***placeholder*** for information that you are expected to supply. Sometime after the synopsis, the reference page explains what kind of information is expected in place of the placeholder.

The end of the **si lock** synopsis is

member...

Since there are no square brackets around the list, in this example, it is mandatory.

This means one or more member names; the ellipsis (`...`) stands for repetitions of whatever immediately precedes it. Most MKS Integrity commands allow you to specify lists of multiple items using spaces between them.

See the **options** reference page for more details on selecting members, Sandboxes and projects.

The order of items on the command line is important. When you type in a command line, you should specify the parts of the command line in the order they appear in the command synopsis. The exceptions to this are options marked with a `-` or a `--`; they do not have to be given in the exact order shown in the synopsis. However, all the `-` or `--` options must appear in the correct area of the command line. For example, you can specify

```
si lock -r 1.2 --gui member1
si lock --gui -r 1.2 member1
```

but you will not get correct results if you specify

```
si lock member1 -r 1.2 --gui
si lock -r 1.2 member1 --gui
```

and so on.

Description

The *DESCRIPTION* section simply describes what the command does and how each option works.

Inside the *DESCRIPTION* section, the names of files and directories are written in `normal Courier` font. The names of environment variables are written in ***italic Courier*** font.

Diagnostics

The *DIAGNOSTICS* division contains information about the exit status returned by the command. You can test this status to determine the result of the operation the command was asked to perform.

See Also

The *SEE ALSO* section refers to other reference pages that may contain information relevant to the reference page you have just read.

NAME

man — display online reference pages

SYNOPSIS

man [**-wx**] [**-M** *path*] [*type*] *entry* ...

man [**-wx**] [**-T** *txt_indexes*] [*type*] *entry* ...

man -h [**-wx**] [**-C** *chm_indexes*] [*type*] *entry* ...

man -k [**-M** *path*] *keyword* ...

DESCRIPTION

The **man** command either displays online reference pages or searches for reference pages that have specified keywords associated with them.

Normally, **man** displays the reference page for each specified *entry*. To display only a reference page of a given type, specify *type* on the command line. *type* is a number representing which type of reference pages to search. Reference pages come in the following types:

1	Commands and Utilities
3	Functions
4	File Formats
5	Miscellaneous

To indicate an operating system specific version of the entry (if one exists) or to indicate an command specific to a given set of commands and/or functions, append one of the following letters to the specified *type*:

n	for Windows NT/2000/XP/2003
w	for Windows Me
t	for Tcl

When output is sent to the terminal, **man** invokes a pager command to filter and display the reference pages. If ***MANPAGER*** is defined, it is used. If not, and if ***PAGER*** is defined, it is used. If neither is defined, **man** defaults to using the command **more -A -s**.

Options

-C *filelist*
: specifies a list of `.idx` files (corresponding to `.chm` files) to search before searching the files listed in ***MAN_CHM_INDEX***.

-h
: launches the HTML Help viewer and displays the HTML Help version of the reference page. The reference page is found by searching each `.idx` listed in the ***MAN_CHM_INDEX*** file (or indicated by the **-C** option) for an entry matching *entry* and *type* that indicates which page in the corresponding `.chm` file to display.

-k
: searches a precomputed database of synopsis lines for information on *keyword*s.

-M *path*
: searches the directories indicated by *path* for reference pages. If **-M** is not specified, **man** uses the path specified in the ***MANPATH*** environment variable if it is set; otherwise **man** searches ***ROOTDIR***`/etc`. All reference pages are found by searching similarly structured file trees rooted at one or more places. See the *FILES* section for a description of the files and directories **man** should find in each directory that it searches.

-T *filelist*
: specifies a list of `.idx` files to search before searching the files listed in ***MAN_TXT_INDEX*** when looking for a text version of a reference page.

-w
: displays only the file name of the file containing the specified entry.

-x
: displays the files that **man** is searching as it tries to find the entry.

Search Rules

To find a given entry, **man** follows a set of search rules. When you specify a *type*, **man** searches for the appropriate page amongst pages of that type; otherwise, **man** looks for the first page named *entry* regardless of the type.

When the **-h** option is specified, **man** searches the `.idx` files listed in the ***MAN_CHM_INDEX*** environment variable for an entry matching the specified *entry* which indicates the HTML Help page in corresponding `.chm` to display. The HTML Help viewer is launched, displaying the page. Once you exit, the view, the **man** command exits.

When **-h** is not specified, **man** takes the following steps to find the entry. Once a step results in finding the entry, **man** displays the reference page and exits.

- **man** searches the `.idx` files listed in the ***MAN_TXT_INDEX*** environment variable for an entry matching the request *entry* which indicates the text (`.txt`) reference page to display.
- **man** checks each directory in ***MANPATH*** for a file named `man.dbz`. If it exists, **man** looks for the requested *entry* in its index (see *man.dbz File Format*).
- For each possible type (that is, *type* if you specified it, or all types in order from 1 through 9, then 0 if you did not):
    - **man** checks each directory in ***MANPATH*** for a file named `cat`*n*`/`*entry*`.`*n*`[`*l*`]` where *n* is the type number, and *l* is the optional letter code. If it exists, **man** checks to see if it was compressed with **pack**, **compress** or **mkszip**, and uncompresses it (calling **pcat** if the file was **pack**ed).
    - **man** checks each directory in ***MANPATH*** for a file named `man`*n*`/`*entry*`.`*n*`[`*l*`]`.

man.dbz File Format

Sometimes, the reference pages are kept in a single large file, called `man.dbz`. The file starts with a magic text string:

```
!<man database compressed>\n
```

and continues with the index:

```
14 bytes formatted reference page name
9 bytes seek pointer
9 bytes length
```

The name is simply the page name, followed by a dot and the type number. For example, this reference page would be named `man.1`. When **man** finds a matching entry, it seeks to the point in the file specified by the given seek pointer, and uncompresses for length bytes. Each reference page is compressed separately.

EXAMPLES

To find the utilities that do comparisons, type:

```
man -k compar
```

ENVIRONMENT VARIABLES

MAN_CHM_INDEX
: contains a semicolon separated list of `.idx` files to search for *entry* when the **-h** is specified.

MAN_TXT_INDEX
: contains a semicolon separated list of `.idx` files to search for *entry* when the **-h** is not specified.

MANPATH
: contains a semicolon separated list of paths to search for reference pages.

MANPAGER, ***PAGER***
: contains an output filtering command for use when displaying reference pages on a terminal.

TMPDIR
: identifies the directory where temporary files reside.

FILES

ROOTDIR`/etc`
: is the default directory for the online reference pages. The rest of the files listed here reside in this directory.

`cat[0-9]/*.[0-9]`
: pre-formatted reference pages in normal, compressed, or packed form.

`man[0-9]/*.[0-9]`
: unformatted reference pages.

`whatis`

is a database used by **-k** option.

`*.chm`
HTML Help files containing collections of reference pages complete with index, table of contents, and full text search.

`*.idx`
index files that **man** how to find HTML Help and text versions of individual reference files. The `.idx` files to search are indicated by the ***MAN_CHM_INDEX*** and ***MAN_TXT_INDEX*** environment variables.

`man.dbz`
is a master file containing all reference pages.

The `etc` directory is found using the ***ROOTDIR*** environment variable.

DIAGNOSTICS

Possible exit status values are:

`0`
Successful completion.

`1`
Failure due to any of the following:

— unknown command line option
— missing *path* after an **-M** option
— no information available on the desired subject
— unable to create a child process to format reference page
— child process returned with non-zero exit status

PORTABILITY

POSIX.2. *x*/OPEN Portability Guide 4.0. All UNIX systems. Windows Me. Windows NT 4.0. Windows 2000. Windows XP. Windows Server 2003.

The **-C**, **-h**, **-M**, **-T**, **-w**, and **-x** options, the ***MANPAGER***, ***MAN_CHM_INDEX***, and ***MAN_TXT_INDEX*** environment variables, the default pager, the ability to specify *type* on the command line, and the ability to display reference pages in HTML Help format are all extensions to the POSIX and XPG standards.

AVAILABILITY

MKS Toolkit for Power Users
MKS Toolkit for System Administrators
MKS Toolkit for Developers
MKS Toolkit for Interoperability
MKS Toolkit for Professional Developers
MKS Toolkit for Enterprise Developers
MKS Toolkit for Enterprise Developers 64-Bit Edition
MKS AlertCentre
MKS Integrity

SEE ALSO

Commands:
help, **manstrip**, **more**

si about

displays product information

SYNOPSIS

si about [**--[no]batch**] [**--cwd=***directory*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [(**-?**|**--usage**)]

DESCRIPTION

si about displays information about this copy of MKS Integrity.

Options

si about takes a subset of the universal options available to ***si*** commands. For example,

```
si about --gui
```

displays the product information in a GUI window. See the **options** reference page for descriptions.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Miscellaneous:

diagnostics, **options**, **preferences**

si acceptcp

accepts change package

SYNOPSIS

si acceptcp [**--comments=***value*] [**--commentsFile=***value*] [**--reviewer=***[user|group:<Group-Name>|super|all]*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F***file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *issue|issue:change package id...*

DESCRIPTION

si acceptcp casts an accept vote on a change package under review and in the `submitted state`.

After a change package is submitted, each individual reviewer and one member of each reviewer group (if specified) in the reviewer list must accept the change package before it can be committed to the repository and then closed. For example:

```
si acceptcp --reviewer=user 3433:1
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--comments=*value*
: specifies comments recorded in the review log with the vote.

--commentsFile=*value*
: specifies a text file that contains the comments to be recorded in the review log with the vote.

--reviewer=*[user|group:<Group-Name>|super|all]*
: specifies the capacity in which you are casting an accept vote.

  user
  : casts a vote as an individual reviewer in the reviewer list.

  group:<Group-Name>
  : casts a vote on behalf of the entire group (only one user from a group is necessary to vote on behalf of the entire group).

  super
  : an overriding accept vote that is sufficient for accepting the change package even if there are additional reviewers. A super reviewer does is not required to be a listed reviewer for the change package.

  all
  : casts a vote as a specific user and any group to which the reviewer belongs.

issue...
issue:change package id...
: *issue* identifies a specific issue that contains all submitted change packages that you want to cast accept votes for; use spaces to specify more than one issue.

  issue:change package id identifies a specific change package to cast an accept vote for; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, **si ci**, **si co**, **si cpissues** , **si createcp**, **si drop**, **si lock**, **si viewcps**, **si rejectcp**, **si opencp**, **si discardcp**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si add

adds a new member to a project

SYNOPSIS

si add [**--archive=***filename*] [**--author=***name*] [(**--binaryFormat**|**--textFormat**)] [(**--cpid=***ID*|**--changePackageId=***ID*)] [**--[no]defer**] [**--deploytype=***value*] [**--[no|confirm]closeCP**] [**--issueId=***ID*] [(**-d***desc*|**--description=***desc*)] [**--descriptionFile=***file*] [(**-l**|**--lock**)] [**--[no]createSubprojects**] [**--onExistingArchive=[confirm|cancel|sharearchive|newarchive]**] [**--[no]retainWorkingFile**] [**--[no]failOnAmbiguousProject**] [**--[no]saveTimestamp**] [**--[no]unexpand**] [(**-r** *rev*|**--revision=***rev*)] [(**-R**|**--[no|confirm]recurse**)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [**--cwd=***directory*] [(**-F** *file*|**--selectionFile=***file*)] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [(**-N**|**--no**)] [**--[no]batch**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [(**-?**|**--usage**)] [(**-Y**|**--yes**)] [**-R**|**--[no|confirm]recurse**] [**--[no|confirm]includeFormers**] [**--exclude=***file:pattern,dir:pattern...*] [**--include=***file:pattern,dir:pattern...*] *nonmember...*

DESCRIPTION

si add adds one or more *nonmember* files located in a Sandbox directory to a project.

For example, you can add a member from the `c:\apps\prototype\` directory by specifying:

```
si add --description="Prototype application header"
 -S prototype.pj header.c
```

adds the `header.c` file to the project as a new member with the description "Prototype application header". Since the member is added to the project on the Integrity Server, all other users who have Sandboxes pointing to the project can see the same new member. MKS Integrity creates a history on the server for each newly added member that does not already have a history.

If you intend to add a dropped member (see **si drop**) from an archive located on the server, or to share the history of another member in a different project, then use the **si addmemberfromarchive** command.

If you are re-adding a dropped member (see **si drop**), an archive already exists for the member, and you must specify whether you want MKS Integrity to associate the member with the existing archive or generate a new one.

Tip: If you are using an RCS type repository, you can add existing members to a new project more quickly by placing the files on the server and using the **si import** command. MKS recommends using the **si import** command for batch imports of a directory tree, or of history files from older versions of MKS Integrity (formerly MKS Source).

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--archive=*filename*
: sets the archive file name, *filename*, containing the server-side path and name by which the newly added member will be archived. This must be an existing archive file - this option does not create a new archive. This is an advanced option that effectively specifies an archive path to override the default mapping, allowing a basic form of archive sharing between projects; its use is not recommended except by advanced MKS Integrity administrators.

-r=*value*
--revision=*value*
: specifies the revision number for the new member. If you specify an existing archive using the **--archive** option, MKS Integrity creates a branch, unless you specify **:head** or a revision that does not yet exist. If you specify **:head** and the current head revision is exclusively locked by another user, MKS Integrity creates a branch. If you have an exclusive lock on the revision being branched, your lock is removed.

--author=*name*
: specifies an author name.

--binaryFormat
--textFormat
: forces the storage of the member file to occur in binary format or text format. This option overrides the preferences settings that control default behavior in the application for storage format. See the **si setprefs** and **si viewprefs** commands for more details on preferences.

--[no]defer

controls whether to delay the add operation in the project until the deferred operation is submitted. The operation in the Sandbox still takes place immediately.

If the change package reviews are mandatory, specify the `--deferred` option to create a pending entry for this operation at the time of change package submission. If the `--deferred` option is not enabled, MKS Integrity creates the pending entry at the completion of this procedure. When a deferred add member operation is submitted as part of a review, a pending member is created For more information, see the *MKS Integrity User Guide*.

--deploytype=*value*
specifies the deploy type, if the project is in a staging system. For more information, see the *MKS Deploy Administration Guide*.

--[no|confirm]closeCP
controls whether to close the associated change package.

--nocloseCP means do not close the change package.

--confirmcloseCP means ask before closing the change package.

--closeCP always closes the change package.

-l
--lock
controls whether to lock the newly created revision. The lock type used is based on your locks policy. For information on your locks policy, contact your administrator.

--onExistingArchive=[confirm|cancel|sharearchive|newarchive]
controls whether to allow sharing of this member's history between projects, or to create a new archive if there is already an existing archive for the member.

--onExistingArchive=confirm means always ask whether to share the archive, create a new archive, or cancel the operation.

--onExistingArchive=cancel means cancel the operation.

--onExistingArchive=sharearchive means share the archive.

--onExistingArchive=newarchive means create a new archive.

--description=*desc*
specifies a description for the new archive; this setting and the **--descriptionFile** option are mutually exclusive. If specifying a *nonmember* that has an existing history, this description does not overwrite an existing description and is ignored.

Note: Descriptions that include spaces must be enclosed by quotes.

-d *desc*
--descriptionFile=*file*
specifies a file for obtaining a description to apply to the new archive; this setting and the **--description** option are mutually exclusive. If specifying a *nonmember* that has an existing history, this description does not overwrite an existing description and is ignored.

--[no]createSubprojects
controls whether to create subprojects for each subdirectory encountered when adding members. This option is commonly used where you anticipate working with a large directory structure, because multiple subprojects are easier to manage than many subdirectories within one project. For example, specifying:

```
si add --createSubprojects --description="Button Application" \buttons\activebutton.c
```

creates a subproject for the `\buttons` directory and adds the file `activebutton.c`. Without the **--createSubprojects** option, the file and its path simply would be added to the project in the current directory.

--[no]retainWorkingFile
controls whether to retain a working file in the Sandbox after adding the new member.

--[no]saveTimestamp
controls whether to set the revision timestamp to the modification date and time of the working file rather than the current date and time.

--[no|confirm]includeFormers

controls whether to include former members. Former members are members that have been dropped but whose working files still reside in the Sandbox directory.

--[no]unexpand
controls whether to unexpand keywords in the member file before checking it into the history. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
Revision 1.6.1.62 2009/05/27 10:33:14EDT Siobhan McRae (smcrae)
Updates for Docs item 140028.
Revision 1.6.1.61 2009/05/26 13:52:24EDT David Flett (dflett)
99518
$Revision$
$Name$
$ProjectLabel$
$ProjectName$
$ProjectSetting$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
$State$
```

-R
--[no|confirm]recurse
controls whether to select non-members recursively. Non-members are files that exist in the Sandbox directory but have not previously been added to the server repository.

--exclude=*file:pattern,dir:pattern...*
specifies a file that contains a glob pattern for excluding members.

--include=*file:pattern,dir:pattern...*
specifies a file that contains a glob pattern for including members.

nonmember...
identifies a specific file to add to your Sandbox; use spaces to specify more than one.

Note: All files must be added "in tree", that is, the nonmember must exist in the Sandbox directory or a subdirectory. Shared "out of tree" histories are supported by using the **--archive** option to point to the "out of tree" history that you want to associate with the member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addmemberfromarchive, **si import**, **si ci**, **si closecp**, **si co**, **si cpissues**, **si createcp**, **si drop**, **si projects**, **si sandboxes**, **si viewcps**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si addlabel

assigns a label to project members

SYNOPSIS

si addlabel [**--[no|confirm]moveLabel**] [(**-r** *rev*|**--revision=***rev*)] [(**-L** *label*|**--label=***label*)] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

siaddlabel assigns a label to a revision of one or more *members* of an MKS Integrity configuration management project. For example:

```
si addlabel -L Prototype innactivebutton.c
```

If you do not specify any members, the **si addlabel** command applies to all project members.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]moveLabel
controls whether the application should move a label from one revision to another.

--nomoveLabel disables the moving of a label.

--confirmmoveLabel displays a confirmation message.

--moveLabel moves the label.

-L *label*
--label=*label*
identifies a label. To add the label "October 15th Prototype" to a member file `activebutton.c`, you would enter

```
si addlabel -L "October 15th Prototype" activebutton.c
```

Note the following about using labels:

- Labels cannot contain colons(:), square brackets ([ ]), or leading spaces.
- Labels cannot have the same format as a valid revision number.
- Labels that include spaces must be enclosed by quotes.
- MKS recommends not using hyphens (-) in labels. Using hyphens may cause some ***si*** commands to fail.
- Labels cannot contain numbers without any spaces, for example, 14325. Numbers that contain spaces are acceptable, for example, 2432 1234.

member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si addprojectlabel, **si ci**, **si deletelabel**, **si deleteprojectlabel**, **si rlog**, **si viewlabels**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si addmemberattr

adds attributes to a member

SYNOPSIS

si addmemberattr [(**--attr=***key*[=*value*]|**--attribute=***key*[=*value*])] [(**-R**|**--[no]confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si addmemberattr adds attributes to a *member*, replacing any previous attribute of that same name specified with the member. If no members are specified, the command applies to all members of the project.

Attributes are key-value pairs of strings that are associated with a member. They are used for automated operations. For example, you could have an attribute indicating the target operating system type:

```
si addmemberattr --attr OS=WIN32 activebutton.c
```

and then you could perform other operations, such as **si resync**, on just those files through the **--filter** universal option:

```
si resync --filter=attribute:OS=WIN32
```

Note: Attribute keys cannot contain a hyphen (-) as the first character. The first character must either be a underscore (_) or an alpha character.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--attr=*key*[=*value*]
--attribute=*key*[=*value*]
: sets the attribute key and, optionally, the value. Note: you cannot have commas (,) in the *key* or *value*, nor an underscore (_) at the beginning of an attribute.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addprojectattr, **si dropmemberattr**, **si dropprojectattr**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si addmemberfromarchive

adds a member from a server archive

SYNOPSIS

si addmemberfromarchive [**--archive=***value*] [**--[no]createSubprojects**] [**--[no]defer**] [**--[no|confirm]includeFormers**] [**--include=***file:pattern,dir:pattern...*] [**--exclude=***file:pattern,dir:pattern...*] [**--cpid=***ID*|**--changePackageId=***ID*] [**--[no|confirm]closeCP**] [**--issueId=***value*] [**--[no]failOnAmbiguousProject**] [**--deployType=***value*] [**--devpath=***value*] [**-P**|**--project=***value*] [**-r**|**--revision=***value*] [**-S**|**--sandbox=***value*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F***file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *nonmember...*

DESCRIPTION

si addmemberfromarchive adds members from server archives.

The `si addmemberfromarchive` command can be used for the following:

- Add previously dropped members without requiring a Sandbox working file, while preserving the member history.

  From the Sandbox directory, for example, and assuming the three specified members have just been mistakenly dropped, the following command will re-add them:

  ```
  si addmemberfromarchive label.c button.c panel.c
  ```

- Add a member that shares the history of another existing member, where the source member can be located in another project from the added member.

  From the Sandbox directory, for example, the following command will add a member named largeLabel.c that shares the history with all members already linked to the specified archive location.

  ```
  si addmemberfromarchive --archive=/anotherproject/rcs/label.c largeLabel.c
  ```

Note the following about the command:

- can be deferred when performed from a Sandbox
- may be part of change package review (displayed as pending members in the project)
- supports multiple selection of archives using the wizard

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--archive=*value*
: a non-default archive location (the specified archive must exist on the server). To add several members with non-default archive locations in one operation, use the GUI wizard (`si addmemberfromarchive -g`).

-r=*value*
--revision=*value*
: specifies the revision at which the member is to be added. This must be an existing revision in the archive. By default, the member is added at the latest revision on the main branch.

--[no]createSubprojects
: controls whether to create subprojects for each subdirectory encountered when adding members. This option is commonly used where you anticipate working with a large directory structure, because multiple subprojects are easier to manage than many subdirectories within one project. For example, specifying:

  ```
  si addmemberfromarchive --createSubprojects --description="Button
  Application"  \buttons\activebutton.c
  ```

  creates a subproject for the `\buttons` directory and adds the file `activebutton.c`. Without the **--createSubprojects** option, the file and its path simply would be added to the project in the current directory.

--[no]defer
: controls whether to delay the add member from archive operation in the project until the deferred operation is submitted. The operation in

the Sandbox still takes place immediately.

If the change package reviews are mandatory, specify the `--defer` option to create a pending entry for this operation at the time of change package submission. If the `--defer` option is not specified, MKS Integrity creates the pending entry at the completion of this operation. When a deferred import member operation is submitted as part of a review, a pending member is created For more information, see the *MKS Integrity User Guide*.

--[no|confirm]includeFormers
controls whether to include former members. Former members are members that have been dropped but whose working files still reside in the Sandbox directory.

--exclude=*file:pattern,dir:pattern...*
specifies a file that contains a glob pattern for excluding members.

--include=*file:pattern,dir:pattern...*
specifies a file that contains a glob pattern for including members.

--[no|confirm]closeCP
closes the change package after command completion.

--deployType=*value*
specifies the deploy type, if the project is in a staging system. For more information, see the *MKS Deploy Administration Guide*.

nonmember...
identifies non-members to add; use spaces to specify more than one non-member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, **si ci**, **si co**, **si cpissues**, **si createcp**, **si drop**, **si lock**, **si viewcps**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si addproject

adds an existing project to the list of top-level projects

SYNOPSIS

si addproject [**--[no]openView**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--[no]add**] *project location...*

DESCRIPTION

si addproject adds an existing project to the list of top-level projects. For example,

```
si addproject C:/phonespec/aurora.pj
```

The `si addproject` command can be used for the following:

- Expose an existing subproject as a top-level project.
- Expose a project imported with the --noadd flag.
- Re-expose a dropped top-level project.

If you want to restore a project from a backup or a dropped project from an older version of MKS Integrity (formerly MKS Source or Source Integrity), use **si importproject**.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]openView
: controls whether to open the project view after this project is added. If **--noopenView** is used, the project view does not open.

project location...
: identifies a location on the Integrity Server for the existing project to be added. Regardless of your operating system, all paths are specified here using forward slashes (/) and should include the name of the project file (typically `project.pj`). Use spaces to specify more than one project location.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si checkpoint, **si createproject**, **si createsandbox**, **si createsubproject**, **si dropproject**, **si addsandbox**, **si projects**, **si importproject**, **si viewproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si addprojectattr

adds attributes to a project

SYNOPSIS

si addprojectattr [(**--attr=***key*[=*value*]|**--attribute=***key*[=*value*])] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si addprojectattr adds attributes to a project, replacing any previous attribute of the same name specified with the project.

Attributes are key-value pairs of strings that are associated with the project. They are used for automated operations. For example, you could have an attribute indicating the target operating system type:

```
si addprojectattr --project=C:/phonespec/aurora.pj --attr OS=WIN32
```

Note: Attribute keys cannot contain a hyphen (-) as the first character. The first character must either be an underscore (_) or an alpha character.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--attr=*key*[=*value*]
--attribute=*key*[=*value*]
: sets the attribute key and, optionally, the value. Note: you cannot have commas (,) in the *key* or *value*, nor an underscore (_) at the beginning of an attribute.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addmemberattr, **si dropmemberattr**, **si dropprojectattr**, **si dropsandboxattr**, **si addsandboxattr**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si addprojectlabel

assigns a label to a project

SYNOPSIS

si addprojectlabel [**--[no|confirm]moveLabel**] [(**-L** *label*|**--label=***label*)] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F***file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si addprojectlabel assigns a label to a checkpointed revision of a project. For example,

```
si addprojectlabel --project=c:/Aurora_Program/bin/Libra/project.pj --label="Release Candidate
002"
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]moveLabel
controls whether the application should move a label from one revision to another.

--nomoveLabel disables the moving of a label.

--confirmmoveLabel displays a confirmation message.

--moveLabel moves the label.

-L *label*
--label=*label*
identifies a label. Note the following about using labels:

- Labels cannot contain colons(:), square brackets ([ ]), or leading spaces.
- Labels cannot have the same format as a valid revision number.
- Labels that include spaces must be enclosed by quotes.
- MKS recommends not using hyphens (-) in labels. Using hyphens may cause some ***si*** commands to fail.
- Labels cannot contain numbers without any spaces, for example, 14325. Numbers that contain spaces are acceptable, for example, 2432 1234.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si addlabel, **si ci**, **si deletelabel**, **si deleteprojectlabel**, **si rlog**, **si viewlabels**

Miscellaneous:
diagnostics, **options**, **preferences**

si addprojectmetric

adds metrics to a project checkpoint

SYNOPSIS

si addprojectmetric [**--metric=***value*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si addprojectmetric adds the value for one or more metrics to the specific project checkpoint.For example,

```
si addprojectmetric --project=c:/Aurora_Program/bin/Libra/project.pj --projectRevision=1.2 --
metric=functions=50,10
```

You can use a third party tool to calculate metrics for C or C++ in the context of a sandbox. You add the calculated value of the metric to a project checkpoint using this command.

The **--projectRevision** option is mandatory. Metrics can only be added to a build project.

Note: Metrics are only supported for database type repositories.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--metric=*value*

specifies the metric value to add. The value is in the following format: *metricname=value[,count]*. For example:

--metric=functions=50,10

This example sets the count of the number of functions to 50, as computed over 10 files (this allows the metric to be displayed as an average of 5 functions per file).

The *metricname* must be a known metric. Metrics are created using the **si createmetricinfo** command.

This option can be repeated to add multiple metrics at the same time.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si viewprojectmetrics, **si createmetricinfo**, **si viewmetricsinfo**, **si calculateprojectmetrics**

Miscellaneous:

diagnostics, **options**, **preferences**

si addsandboxattr

adds Sandbox attributes

SYNOPSIS

si addsandboxattr [**--attr**|**attribute=***key*=*value*] [**-S**|**--sandbox=***value*] [**--[no]failOnAmbiguousProject**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si addsandboxattr adds Sandbox attributes. For example,

```
si addsandboxattr --sandbox=--project=c:/Aurora_Program/bin/Libra/project.pj --
attribute=OS=unix
```

Note: Attribute keys cannot contain a hyphen (-) as the first character. The first character must either be a underscore (_) or an alpha character.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--attr=*key*=*value*
--attribute=*key*=*value*
specifies the attibute to set.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si dropsandboxattr, **si configuresandbox**, **si addprojectattr**, **si dropprojectattr**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si addsubproject

adds an existing subproject to a project

SYNOPSIS

si addsubproject [**-r** *subproject revision*|**--subprojectRevision=***subproject revision*]
[**--subprojectDevelopmentPath=***subproject development path name*|**--variant=***subproject development path name*]
[**--type=[normal**|**variant**|**build**|**default]**] [(**-P***project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)]
[**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*]
[**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*]
[**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]
[**--cpid=***ID*|**--changePackageId=***ID*] [**--[no**|**confirm]closeCP**] [**--issueId=***value*] *subproject location...*

DESCRIPTION

si addsubproject allows you to re-add a dropped subproject.For example,

```
si addsubproject c:/Aurora_Program/bin/Libra/project.pj
```

Note: If you want to add a normal subproject to a variant project, the normal subproject does not require a development path.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

-r=*subproject revision*
--subprojectRevision=*subproject revision*
specifies the checkpoint number (for build subprojects). For example, `1.5`. This option is used with **--type=***build*.

--subprojectDevelopmentPath=*subproject development path name*
--variant=*subproject development path name*
specifies the development path name (for variant subprojects). For example, `Service_Pack3`. This option is used with **--type=***variant*

--type=[normal|**variant**|**build**|**default]**
specifies the new configuration type for the subproject.

--type=normal configures the subproject based upon the current state of the subproject.

--type=variant configures the subproject based upon a specific development path. The option is used with **--variant=***subproject development path name* or **--subprojectDevelopmentPath=***subproject development path name*.

--type=build configures the subproject as a static subproject based upon a specific checkpoint of the master project that is used for building or testing the project, but not for further development. This option is used with **-r** *subproject revision* or **--subprojectRevision=***subproject revision*.

--type=default configures the subproject based on the type that is consistent with the parent project that you are adding the subproject to. For example, if you add a subproject to a normal project, the subproject is added as a normal type. For information on what the default type is, see your administrator.

--cpid=*ID*
--changePackageId=*ID*
identifies a change package that is notified of this action, for example, *1452:1*. Note the following about using this option:

- This option can only be specified if change packages are enabled.
- If the integration is enabled, but it is not mandatory to specify a change package, or if no change package is applicable, you can specify **--changePackageId=***:none*.
- The next time you are prompted for a change package ID, the last used change package ID is displayed by default, if *:none* was specified or at least one change package was successfully updated from the last operation.

--[no|**confirm]closeCP**
closes the change package after command completion.

--issueId=*value*
specifies the issue ID that corresponds to the change package that records the changes.

subproject location...

specifies the path and name of the subproject you want to add, for example, `c:/Aurora_Program/bin/Libra/project.pj`

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si checkpoint, **si configuresubproject**, **si createproject**, **si createsubproject**, **si addsubproject**, **si movesubproject**, **si createsandbox**, **si drop**, **si dropproject**, **si importproject**, **si projectinfo**, **si projects**, **si sharesubproject**, **si viewproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si annotate

displays an annotated revision.

SYNOPSIS

si annotate [**--[no]defaultFormat**] [**--[no|un]expand**] [**--fields**=*field1[:width1],field2[:width2]...*] [**--guiCharacterEncoding**=*value*] [**--height**=*value*] [**--width**= *value*] [**-x**=*value*] [**-y**=*value*] [**--[no]failOnAmbiguousProject**] [**-r|--revision**=*value*] [**--devpath**=*value*] [**-P|--project**=*value*] [**-S|--sandbox**=*value*] [**--projectRevision**=*value*] [**--hostname**=*server*] [**--port**=*number*] [**--password**=*password*] [**--user**=*name*] [(**-?|--usage**)] [(**-N|--no**)] [(**-Y|--yes**)] [**--[no]batch**] [**--cwd**=*directory*] [**--forceConfirm**=*[yes|no]*] [(**-g|--gui**)] [**--quiet**] [**--settingsUI**=*[gui|default]*] [**--status**=*[none|gui|default]*] *member*

DESCRIPTION

si annotate displays an annotated revision. For example,

```
si annotate --project=c:/Aurora_Program/bin/Libra/project.pj cell.c
```

Use the command when you want to know the reason and circumstances a line was introduced or changed. Rather than viewing the content of each revision in the history one revision at a time, you can see the line by line history that includes information on a per revision basis.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]defaultFormat
specifies if the default format is used to display the output.

--fields=*field1[:width1],field2[:width2]...*

The fields available for printing can be one or more of the following:

author
displays the author of the revision.

cpid
displays the line's associated change package ID.

date
displays the date each line in the history was created.

labels
displays revision labels.

linenum
displays the line number for each line of text in the revision..

revision
displays the line's revision number.

text
displays the text contained in the line.

--[no|un]expand
specifies if the keywords are expanded in the output.

--guiCharacterEncoding=*value*
specifies the character encoding to use for the revision contents in the GUI, and can only be specified with the -g option. MKS Integrity automatically decodes UTF-8 revision contents based on the presence of a byte order mark (BOM) in the file. You can set character encoding preferences for each view and command through your client preferences. The preference setting is used if the character set cannot be determined through the file's BOM, or if no character set is specified in the command line. By default, the character set in the preferences matches the default character set provided by the client's operating system locale. The option does not apply to pure CLI output, or third party merge and/or differencing tools. Possible values are:

```
UTF-8
US-ASCII
windows-1252
```

```
ISO-8859 -1
ISO-8859 -15
IBM437
IBM850
IBM863
EUC-JP
Shift_JIS
x-euc-jp-linux
x-eucJP-Open
x-windows-iso2022jp
IBM862
ISO-8859-8
ISO-8859-9
```

member

specifies the name of the member to view.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, **si memberinfo**, **si mods**, **si print**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si viewhistory**, **si viewlabels**, **si viewprojecthistory**, **si viewsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si appendcheckpointdesc

appends additional text to a checkpoint description

SYNOPSIS

si appendcheckpointdesc [(**-d** *desc*|**--description=***desc*)] [**--descriptionFile=***file*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *project...*

DESCRIPTION

si appendcheckpointdesc appends additional text to a checkpoint description. For example,

```
si appendcheckpointdesc --projectRevision=1.23 --description="Release Canadidate"
c:/Aurora_Program/bin/Libra/project.pj
```

A checkpoint description is created when you checkpoint a project. When you review the description for a specific checkpoint, through the **si viewprojecthistory** or **si projectinfo** command, you see the appended information in a manner similar to the following:

```
--- Added comments --- sholmes [Oct 23, 2009 2:39:31 PM EDT]
Appended Description
```

Note:

Appending a checkpoint description recursively appends the checkpoint description for all subprojects.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions. You must specify the **--project** and **--projectRevision** options.

-d *desc*
--description=*desc*

specifies the new description text to be appended to the existing checkpoint description. These options and the **--descriptionFile** option are mutually exclusive.

Note:

Descriptions that include spaces must be enclosed by quotes.

--descriptionFile=*file*

specifies a file name, *file*, containing the new description text to be appended to the existing checkpoint description. This option and the **-d** or **--description** options are mutually exclusive.

project...

identifies a specific project; use spaces to specify more than one project.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addprojectlabel, **si projectinfo**, **si checkpoint**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si appendrevdesc

appends additional text to a revision description

SYNOPSIS

si appendrevdesc [(**-d** *desc*|**--description=***desc*)] [**--descriptionFile=***file*] [(**-r** *rev*|**--revision=***rev*)] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si appendrevdesc appends additional text to a revision description.For example,

```
si appendrevdesc --revision=1.38 --description="Release Canadidate"
c:/Aurora_Program/bin/Libra/cell.c
```

A revision description is created when you check in a new revision. When you review the description for a specific revision, through the **si viewhistory** command, you see the appended information in a manner similar to the following:

```
--- Added comments ---  sholmes [2002/07/20 20:23:55Z]
Appended Description
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

-d *desc*
--description=*desc*
specifies the new description text to be appended to the existing revision description. These options and the **--descriptionFile** option are mutually exclusive.

Note:
Descriptions that include spaces must be enclosed by quotes.

--descriptionFile=*file*
specifies a file name, *file*, containing the new description text to be appended to the existing revision description. This option and the **-d** or **--description** options are mutually exclusive.

member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si addlabel, **si archiveinfo**, **si ci**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si applycp

applies one or more change packages to a MKS Integrity configuration management project

SYNOPSIS

si applycp [--[no]alreadyInProjectIsError] [--backFill=[cp|revision|error|skip|ask]] [--[no]useMaster] [--[no]failOnAmbiguousProject] [--[no|confirm]closeCP] [--[no]confirm] [--[no|confirm]createVariants] [--notify=*[never|onCompletion|onError]*] [--[no]ignoreBranches] [--[no]ignoreServer] [--[no]otherProjectIsError] [--[no]spanProjects] [--[no]verbose] [(-P *project*|--project=*project*)] [(-S *sandbox*|--sandbox=*sandbox*)] [--devpath=*path*] [--hostname =*server*] [--port=*number*] [--password=*password*] [--user=*name*] [(-?|--usage)] [(-F *file*|--selectionFile=*file*)] [(-N |--no)] [(-Y|--yes)] [--[no]batch] [--cwd =*directory*] [--forceConfirm=*[yes|no]*] [(-g|--gui)] [--quiet] [--settingsUI=*[gui|default]*] [--status=*[none|gui|default]*] [--cpid=*ID*|--changePackageID=*ID*] [--[no]ignoreUpdateRevision] [--issueID=*value*] [--subprojectPropagation=[explicit|implicit]] *issue|issue:change package id...*

DESCRIPTION

si applycp propagates changes recorded in change packages from one project or development path to another. This enables you to propagate only the changes that you want to. The process updates member revisions, and may also involve adding, dropping, renaming, and moving files, or creating, adding, dropping or moving subprojects. The **si applycp** command is most useful for building software.

For more detailed information and examples for **si applycp**, see the *MKS Integrity User Guide*.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--cpid=*ID*

--changePackageId=*ID*

identifies a propagation change package to record the changes made as a result of the **si applycp** operation. Only open change packages can be used. This is not the change package that you want to apply to the project; it is the change package that you want to use to record the results of the operation.

--[no]alreadyInProjectIsError

Causes MKS Integrity to terminate the operation if the change being applied has already been applied to the project. If this setting is negated **--noalreadyInProjectIsError**, then the information is displayed as a warning.

--backFill=[cp|revision|error|skip|ask]

Specifies the way MKS Integrity treats historic revisions required by the specified change package.

--backFill=cp

recursively chooses all historic revisions required by the specified change packages and applies them by updating member revisions, adding files, or dropping files.

--backFill=revision

processes only the specified change package(s) and chooses only directly associated revisions. It does not process any change packages that are associated with intermediate revisions.

--backFill=error

terminates the operation if other change packages are required but are not specified.

--backFill=ask

allows you to specify the change packages you want to include.

--[no]confirm

specifies whether to confirm the actions before starting them.

--[no|confirm]createVariants

specifies whether to create new variant subprojects within the target variant project as required. This allows the subproject structure of the target variant to mirror the structure of the source project.

--notify=[never|onCompletion|onError]

specifies whether to display a report when the command is complete. The report details the operations that were performed and any errors that were encountered.

--notify=never

never displays the report.

--notify=onCompletion
always displays the report.

--notify=onError
displays the report if any errors were encountered.

--[no]ignoreBranches
specifies whether to use the most recent revision when MKS Integrity encounters two revisions of the same member on different branches in the change package being applied.

--[no]useMaster
operates on the top-level project if the target is a subproject. Changes are applied throughout the top-level project's hierarchy.

--[no]ignoreServer
specifies whether to perform the Apply CP operation even if the change package members reside on different servers.

Note: The **--ignoreServer** option is useful when a server is the same but is now identified differently, for example, the server name has changed. This option is required because projects are defined by their server and path. Use this option with caution.

--[no]ignoreUpdateRevision
ignores update revision change package entries. Use this option when the change package contains backward revision updates, for example, from 1.4 to 1.2. The default is to include update revision entries for backwards compatibility with older version of MKS Integrity (formerly MKS Source).

--[no]otherProjectIsError
specifies whether to terminate the command if the change is for a project other than the one you initially targeted. If this setting is negated **--nootherProjectIsError**, then the information is displayed as a warning.

--[no]spanProjects
specifies whether to apply all changes in the change package, even if this involves a different project than the one you initially targeted. If you are applying a change package that contains move member or move subproject entries between two project, the **--[no]spanProjects** option must be used.

Warning: This is the only operation that has the potential to affect other projects.

--[no]verbose
specifies whether to include additional information to track the current state of the command.

--[no|confirm]closeCP
closes the change package after command completion.

--subprojectPropagation=*[explicit|implicit]*
specifies how to apply subproject changes required by the specified change packages.

--subprojectPropagation=*explicit* creates, adds, drops, or moves a subproject only if there is a explicit command to do so in the change package.

--subprojectPropagation=*implicit* creates, adds, drops or moves a subproject if the operation is implicitly required based on the change package entries. For example, if you are adding a member that is part of a subproject that does not exist in the target project being updated, the subproject is added.

--issueID=*value*
specifies the issue ID that corresponds to the change package that records the changes.

issue...
issue:change package id...
issue identifies a specific issue that contains all change packages that you want to include; use spaces to specify more than one issue.

issue:change package id identifies a specific change package to include; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si resync, **si resynccp**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si archiveinfo

displays information about an archive

SYNOPSIS

si archiveinfo [**--[no]labels**] [**--[no]locks**] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--lockRecordFormat=***value*] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si archiveinfo displays global information about an archive. For example,

```
si archiveinfo --locks --lockrecordformat={locker}
c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

The command does not contain any per-revision information. For example:

```
Member Name: c:\Documentation\Man_Pages\xml_man\si_add.1.xml
Sandbox Name: c:\Documentation\Man_Pages\project.pj
Archive Name: \rd\doc\Man_Pages\xml_man\rcs\si_add.1.xml
Archive Type: Text
Shared
Exclusive Lock Mandatory
Archive Description:
Labels:
    TechReview1 1.7
    PeerReview_1 1.6
```

Note: `Shared` displays only if other members share the archive and you are using the database repository.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]labels
: controls whether to show labels and associated revision IDs.

--[no]locks
: controls whether to show locks and associated revision IDs.

--lockRecordFormat=*value*
: defines the format for displaying lock information for the archive. Specify a format string using keywords to represent the information you want to display. You can specify any of the following keywords:

  {revision}
  : displays the revision that is locked.

  {locker}
  : displays the user who locked the revision.

  {locktype}
  : displays the type of lock on the revision (exclusive or non-exclusive).

  {locktimestamp}
  : displays the time when the revision was locked.

  {lockcpid}
  : displays the change package associated with the lock on the revision.

  {project}
  : displays the name and path of the project where the member revision was locked from. If the member revision was locked from a shared subproject, it is the subproject name and path that are displayed.

  {devpath}
  : displays the name of the development path where the lock on the revision was made from.

  {sandbox}
  : displays the name of the Sandbox where the lock on the revision was made. This is relevant when viewing the information from

the locker host.

{hostname}
displays the hostname of the computer that locked the the revision.

{hascpid}
displays 1 if the lock has a change package associated with it, 0 if there is no associated change package.

{hassandbox}
displays 1 if there is Sandbox information available for the lock, 0 if no Sandbox information is available.

{hasdevpath}
displays 1 if the lock was made from a development path, 0 if it wasn't.

{member}
displays the name of the locked revision.

member...
identifies a specific member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si memberinfo, **si mods**, **si revisioninfo**, **si rlog**, **si viewhistory**, **si viewlabels**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si calculateprojectmetrics

calculates the metrics for the specified project checkpoint

SYNOPSIS

si calculateprojectmetrics [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]recomputeall**] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si calculateprojectmetrics calculates the metrics for the specified project checkpoint.For example,

```
si calculateprojectmetrics --project=c:/Aurora_Program/bin/Libra/project.pj --
projectRevision=1.34 --recomputeall
```

If you decide to track metrics for a project that has already been checkpointed, you can calculate metrics for existing checkpoints using this command. All project level statistics are recomputed, but each member's statistics are only computed if none currently exist. You can use the **--recomputeall**option to recompute all member metrics.

The **--projectRevision** option is mandatory. Metrics can only be calculated for a build project..

There may be a performance impact the first time you calculate project metrics. The first time you calculate metrics for a project with a large number of members that existed before implementing metrics, the metric calculation may take a significant amount of time. The increase in time is due to the need for the metrics to be initially computed for the pre-existing members. Subsequent metric calculations are aggregated for members of the project.

Note: Metrics are only supported for database type repositories.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]recomputeall
: specifies whether you want to recompute metrics for all members. This is useful when new metrics have been added.

  This recomputes metrics for all members recursively, summing up all their individual metrics, and putting the rolled up values on the specified project.

  If a given member has no metrics, then it is assumed to not have had metrics computed for it. This command computes metrics for them by invoking the metrics trigger. If a member has even one metric, it is assumed to have had metrics computed for it, and this command doesn't compute them. So if you changed the trigger to generate new metrics, they will not automatically be recomputed unless you use the **--recomputeall** option, which forces the command to ignore the existing metrics and rerun metrics on each member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si viewprojectmetrics, **si createmetricinfo**, **si createmetricinfo**, **si viewmetricsinfo**

Miscellaneous:

diagnostics, **options**, **preferences**

si checkpoint

checkpoint archives in a project

SYNOPSIS

si checkpoint [**--author=***name*] [(**-d** *desc*|**--description=***desc*)] [**--descriptionFile=***file*] [**--[no]notify**] [(**-L** *label*| **--label=***label*)] [**--[no]failOnAmbiguousProject**] [(**-s** *state*|**--state =***state*)] [**--[no]stateMembers**] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

You can track the evolution of an entire project by preserving the changes made to it from one revisions to another. In MKS Integrity, this process is called *checkpointing*. For example,

```
si checkpoint --project=c:/Aurora_Program/bin/Libra/project.pj --label="Release Candidate 002"
--description="Ready for Review" --notify
```

Checkpointing a project creates a new revision of the project and adds it to the project history. When you checkpoint a project, you save all the information needed to recreate the project completely at any time in the future. The saved information includes the project structure and the list of members with their revision numbers.

If you are using the database repository, all checkpoints are *transactional*. This means that a checkpoint records the structure and contents of the project at the time the checkpoint starts. It does not include anything added to the project or its subprojects after the checkpoint starts, such as check ins or submitted change packages. To ensure that only complete change packages are included in your project checkpoints, your administrator must enable change package reviews. For more information on change package reviews, see the *MKS Integrity User Guide*.

While the checkpoint is in progress, you can perform member and subproject operations on the project, but you cannot perform any operations that affect project checkpoints, development paths, or labels. You cannot perform the following operations:

- Checkpoint the project on another development path
- Restore the project to a previous checkpoint
- Delete the project from the database
- Add or delete project labels
- Create or remove development paths

Note:
Checkpointing a project checkpoints all its subprojects. If you do not want to create subproject checkpoints, configure them as build subprojects first. See the **si configuresubproject** command for more details.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--author=*name*
specifies an author name to identify the author of the new revision.

-d *desc*
--description=*desc*
specifies a description for the checkpointed revisions. These options and the **--descriptionFile** option are mutually exclusive.

Note:
Descriptions that include spaces must be enclosed by quotes.

--descriptionFile=*file*
specifies a file name, *file*, containing the description text for the checkpointed revisions. This option and the **-d** or **--description** options are mutually exclusive.

--[no]notify
controls whether to notify when the checkpoint is complete.

-L *label*
--label=*label*

identifies a label. The label is applied to the checkpointed revision of the project. Note the following about using labels:

- Labels cannot contain colons(:), square brackets ([ ]), or leading spaces.
- Labels cannot have the same format as a valid revision number.
- Labels that include spaces must be enclosed by quotes.
- MKS recommends not using hyphens (-) in labels. Using hyphens may cause some ***si*** commands to fail.

-s *state*
--state=*state*
specifies a state, *state*, for the checkpointed revisions. The state is applied to the checkpointed revision of the project, and to all members if **--stateMembers** is specified.

--[no]stateMembers
controls whether to set the state for members at member revision, in addition to the project.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addprojectattr, **si addprojectlabel**, **si ci**, **si createdevpath**, **si createproject**, **si createsubproject**, **si importproject**, **si memberinfo**, **si projectinfo**, **si projects**, **si promoteproject**, **si setprojectdescription**, **si viewproject**, **si viewprojecthistory**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si ci

checks in members of a Sandbox

SYNOPSIS

si ci [**--[no|confirm]ignoreUnresolvedMerge**] [(**-L** *label*|**--label=***label*)] [**--author=***name*] [**--[no]branch**] [**--[no|confirm]branchVariant**] [**--[no|confirm]branchUpdate**] [**--[no|confirm]updateIfNotCurrent**] [(**--cpid=***ID*|**--changePackageId=***ID*)] [**--[no]failOnAmbiguousProject**] [**--issueId=***ID*] [**--[no|confirm]checkinUnchanged**] [**--[no|confirm]closeCP**] [(**-d** *desc*|**--description=***desc*)] [**--[no]defer**] [**--descriptionFile=***file*] [(**-l**|**--[no]lock**)] [**--[no|confirm]moveLabel**] [(**-r** *rev*|**--revision=***rev*)] [**--[no]retainWorkingFile**] [**--revision=***value*] [**--[no]saveTimestamp**] [**--onNewerRevision=***{confirm|cancel|resync|resyncByCp}*] [**--[no|confirm]skipUpdateOnLockConflict**] [(**-u**|**--unlock**)] [**--[no]unexpand**] [**--[no]update**] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [**-F** *value*] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***value*] [**--forceConfirm=***[yes|no]*] [**--selectionFile=***value*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *sandbox member...*

DESCRIPTION

si ci checks in and saves changes to Sandbox members. For example,

```
si ci --label="Pre-Release Candidate" --description="Ready for Review"
c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

If you do not specify any members, **si ci** applies to all members of an associated Sandbox.

Check in is an operation that adds a new revision of a file to an archive. When a file is checked in to a revision other than the head revision or a branch tip revision, a new branch is created.

Assigning Revision Numbers

By default, when you check in a member, MKS Integrity automatically assigns a unique revision number to the new revision. It does this by incrementing the current revision number by one. For example, if the previous revision is 1.3, the new revision is assigned number 1.4.

You can choose the revision number of the changes you are checking in, so long as your revision number:

- is greater than the last revision number (you cannot use previously "skipped" revision numbers)
- has no leading zeros (zeros as complete revision numbers are acceptable)
- starts a new branch based on an existing revision

If you check in a revision using an already existing revision number, MKS Integrity attempts to add one to the revision number and check it in as that revision. If that revision already exists, MKS Integrity then chooses the next available branch number and creates a new branch.

For example, if you are checking in a new revision to an archive where the head revision is 1.7, the following numbers are valid:

- 1.8 (greater than head revision)--if you check in a revision as 1.7, which already exists, MKS Integrity assigns it 1.8
- 1.10 (greater than head revision)
- 1.72 (none of the numbers between 7 and 72 may be used afterwards)
- 2.0
- 1.7.1.1 (if it starts a new branch)
- 1.7.0.1 (leading zero as the branch number)

The following numbers are invalid:

- 1.3 even if there was no revision 1.3 previously
- 1.08 (leading 0 in last portion)
- 02.1 is considered the same as 2.1 (leading zero in branch number)

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]ignoreUnresolvedMerge
controls whether to ignore a previously unresolved merge. For more information on merging revisions, see **si merge** and **si mergebranch**.

-L *label*
--label=*label*
identifies a label that is applied to the new revision. Note the following about using labels:

- Labels cannot contain colons(:), square brackets ([ ]), or leading spaces.
- Labels cannot have the same format as a valid revision number.
- Labels that include spaces must be enclosed by quotes.
- MKS recommends not using hyphens (-) in labels. Using hyphens may cause some ***si*** commands to fail.

--author=*name*
specifies an author name.

--[no]branch
controls whether to force the creation of a branch revision.

Note:
If required, a branch revision is created even if this option is set to "no", for example, if you are checking in a member to a revision other than the head revision.

A ***branch*** is a revision path that diverges from the main line of development (also known as the ***trunk***) in a member or project history. A branch is typically created by checking in a file to a revision other than the head revision. The most recent revision of a branch is called the ***tip revision***.

MKS Integrity usually places new revisions at the top of the trunk, assigning them two-part revision numbers, such as 1.15. There are times, however, when you do not want your work to be checked into the trunk. You may be pursuing a line of development that will not be included in the finished product, for instance, or you may be doing post-release maintenance while development for the next release continues on the trunk.

Divergent lines of development in the same archive are managed through the use of branches. A branch is an independent revision line that uses an existing revision as its starting point. Members of a branch revision are identified by their revision numbers. Whereas revisions on the trunk are characterized by two-part revision numbers (for example, 1.2 or 3.5), branch revision numbers are prefixed with the number of the revision they start from. For example, if a branch revision is started from revision number 1.2, the members of that branch are numbered

```
1.2.1.1
1.2.1.2
1.2.1.3
```

and so on. The first two digits of the number identify the revision where the branch diverges from the trunk, and the last two represent a position on the branch.

--[no|confirm]updateIfNotCurrent
controls whether to update the member revision, even if the revision being checked in is not the member revision. This option only applies if **--update** is specified.

--noupdateIfNotCurrent means do not update the member revision.

--confirmupdateIfNotCurrent means ask before updating the member revision.

--updateIfNotCurrent always updates the member revision.

--[no|confirm]checkinUnchanged
controls whether to force the checkin so that the new revision is checked in even if it is not different from the preceding one.

--nocheckinUnchanged means do not force the checkin.

--confirmcheckinUnchanged means ask before forcing the checkin.

--checkinUnchanged always forces the checkin.

--[no|confirm]closeCP
controls whether to close the associated change package.

--nocloseCP means do not close the change package.

--confirmcloseCP means ask before closing the change package.

--closeCP always closes the change package.

-d *desc*
--description=*desc*
specifies a description for the new revision. This option and the **--descriptionFile** option are mutually exclusive.

Note:
Descriptions that include spaces must be enclosed by quotes.

--[no]defer
controls whether to delay the checkin operation in the project. The operation in the Sandbox still takes place immediately.

If reviews are mandatory, specify this option to submit the changes for review as a change package. If this option is not specified, a pending revision is created for the operation, that corresponds to a pending entry in the change package.

--descriptionFile=*file*
specifies a file name, *file*, containing the description text for the new revision. This option and the **-d** or **--description** options are mutually exclusive.

-l
--[no]lock
controls whether MKS Integrity locks the new revision. Locking the new revision allows you to update the archive while retaining control of the revision. The type of lock used is the same as the lock type used when the file was checked out.

--[no|confirm]moveLabel
controls whether the application should move a label from one revision to another.

--nomoveLabel disables the moving of a label.

--confirmmoveLabel displays a confirmation message.

--moveLabel moves the label.

--[no]retainWorkingFile
controls whether to keep the working file in the Sandbox after checkin. By default, a sparse Sandbox does not retain the working file after checkin.

--[no]saveTimestamp
controls whether to set the revision timestamp to the modification date and time of the working file rather than the current date and time.

-u
--unlock
unlocks the newly checked-in revision. This is equivalent to specifying **--nolock**.

--[no]unexpand
controls whether to unexpand or ignore keywords in the member file prior to the check in. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
$Revision$
$Name$
$ProjectLabel$
$ProjectName$
$ProjectSetting attribute$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
```

`$State$`

--[no]update
controls whether to set the checked in revision as the project's member revision.

--onNewerRevision=[confirm|cancel|resync|resyncByCp|]
controls what happens when the revision being checked in is not the member revision in the development path.

--onNewerRevision=confirm means ask for confirmation of the action to be taken.

--onNewerRevision=cancel means cancel the operation.

--onNewergRevision=resync means resynchronize the member revision into the working file and move the lock (if any) to the member revision. The check in operation is not completed. You should perform additional testing on the merged file before checking it in.

--onNewerRevision=resyncbycp means resynchronize the member revision into the working file by change package, and move the lock (if any) to the member revision. The check in operation is not completed. You should perform additional testing on the merged file before checking it in.

Note:
If you specify a revision using the **--revision** option or force the creation of a branch using the **--branch** option, no action is required when checking in a member that is not the member revision.

--[no|confirm]branchVariant
controls whether MKS Integrity creates a branch off the revision you are checking in, if you are working in variant Sandbox and this is the first time the member is checked in. This is useful for keeping changes in the variant separate from changes made in the mainline project.

--[no|confirm]skipUpdateOnLockConflict
instructs MKS Integrity not to update the member revision if the member revision is exclusively locked by another user. This option is development path specific. If this option is not specified, the default is to confirm. This option is not available in the GUI, but can be set for the GUI from the **si setprefs** command

sandbox member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si closecp, **si co**, **si cpissues**, **si createcp**, **si diff**, **si edit**, **si lock**, **si mergebranch**, **si rlog**, **si unlock**, **si viewcps**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si closecp

closes a change package.

SYNOPSIS

si closecp [**--[no|confirm]allowOrphanedDeferred**] [**--[no]confirm**] [**--[no|confirm]releaseLocks**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *issue|issue:change package id...*

DESCRIPTION

si closecp closes a change package. For example,

```
si closecp --confirmallowOrphanedDeferred --confirm --confirmreleaseLocks 3434:2
```

Note: If reviews are mandatory, you cannot close the change package using this command. The change package can only be closed by MKS Integrity once it has successfully completed the review process. For more information, see the *MKS Integrity User Guide*.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]allowOrphanedDeferred
: controls if orphaned deferred operations are permitted when the change package is closed.

--[no]confirm
: controls whether to confirm closing individual change packages.

--[no|confirm]releaseLocks
: controls whether or not outstanding locks are released when the change package is closed.

issue...
issue:change package id...
: *issue* identifies a specific issue that contains all open change packages that you want to close; use spaces to specify more than one issue.

  issue:change package id identifies a specific change package to close; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

Si add, **si ci**, **si co**, **si cpissues**, **si createcp**, **si drop**, **si lock**, **si viewcps**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si co

checks out members into working files in a Sandbox

SYNOPSIS

`si co` [`--mergeType=`*[confirm|cancel|automatic|manual]*] [`--lockType=`*[exclusive|nonexclusive|auto]*] [`--[no|confirm]lockOnFreeze`] [`--onMergeConflict=`*[confirm|cancel|mark|launchtool|highlight|error]*] [`--[no]failOnAmbiguousProject`] [(`--cpid=`*ID*|`--changePackageId=`*ID*)] [`--issueId=`*ID*] [(`-l`|`--[no|auto]lock`)] [`--[no|confirm]merge`] [`--[no|confirm]overwriteChanged`] [`--[no|confirm]downgradeOnLockConflict`] [`----[no|confirm]moveLock`] [(`-r` *rev*|`--revision=`*rev*)] [(`-u`|`--unlock`)] [`--[no]update`] [`--[no|un]expand`] [`-f`] [`--[no|confirm]overwriteDeferred`] [`--[no]restoreTimestamp`] [(`-R`|`--[no|confirm]recurse`)] [`--filter=`*filteroptions*] [(`-S` *sandbox*|`--sandbox=`*sandbox*)] [`--hostname=`*server*] [`--port=`*number*] [`--password=`*password*] [`--user=`*name*] [(`-?`|`--usage`)] [(`-F` *file*|`--selectionFile=`*file*)] [(`-N`|`--no`)] [(`-Y`|`--yes`)] [`--[no]batch`] [`--cwd=`*directory*] [`--forceConfirm=`*[yes|no]*] [(`-g`|`--gui`)] [`--quiet`] [`--settingsUI=`*[gui|default]*] [`--status=`*[none|gui|default]*] *sandbox member...*

DESCRIPTION

`si co` checks out members, typically for modification. For example,

```
si co c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

If no members are specified, **`si co`** applies to all Sandbox members.

Check out is an operation that extracts the contents of a revision in an archive and copies it to a working file. You can check out any revision by specifying either its revision number or label. By default, the member revision is used. If an existing revision other than the head revision is specified, a branch from that revision is created if you specify to do so.

Use the **`si resync`** command to get a read-only copy of a member in your project.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **`options`** reference page for descriptions.

`--mergeType=[confirm|cancel|automatic|manual]`
specifies how you want merge changes from the working file in your Sandbox into the working file that will be created upon check out.

`--mergeType=confirm` prompts you to confirm a merge type.

`--mergeType=cancel` cancels the selected merge.

`--mergeType=automatic` completes the merge process without launching the MKS Visual Merge tool.

Warning: While MKS Integrity and MKS Visual Merge are capable of performing automatic merging, MKS cannot guarantee that the merged results are “correct”. MKS recommends that you examine and test the merged results before checking them into the repository.

`--mergeType=manual` completes the merge operation through the MKS Visual Merge tool or a third party merge tool, depending on your Difference and Merge Tool preferences. The merge tool launches, displaying the revisions you want to merge. For more information on the MKS Visual Merge tool, refer to the *MKS Integrity User Guide*.

`--onMergeConflict=[confirm|cancel|mark|launchtool|highlight|error]`
specifies what to do when conflicts occur during a merge.

`--onMergeConflict=confirm` prompts you to confirm a merge conflict option.

`--onMergeConflict=cancel` cancels the merge.

`--onMergeConflict=mark` marks the revisions indicating that merging is required without completing all of the merge related tasks. This provides time you may need to investigate conflicts or consult on editing or difference blocks before finishing the merge.

`--onMergeConflict=launchtool` launches the MKS Visual Merge tool to resolve the conflicts, all non-conflicting blocks will already have been applied.

`--onMergeConflict=highlight` indicates conflicts in the file with the following characters: "<<<<<<" and ">>>>>>".

`--onMergeConflict=error` indicates merge conflicts with an error message prompt.

Note: **`--onMergeConflict=launchtool`** does not require the **`-g`** or **`--gui`** options.

-l

--[no|auto]lock
controls whether to create locks on members.

--lock means always lock the member.

--nolock means never lock the member.

--autolock means lock the member based on the locks policy. For information on the locks policy, contact your administrator.

--[no|confirm]lockOnFreeze
controls whether to lock the specified member even if it is frozen.

--lockOnFreeze means always lock the specified member.

--nolockOnFreeze means never lock the specified member.

--confirmlockOnFreeze means always ask before locking the specifies member.

--[no|confirm]merge
controls whether to merge changes from the working file in your Sandbox into the working file that will be created upon check out.

--merge means always merge.

--nomerge means never merge.

--confirmmerge means always ask before merging. When asked to confirm the merge, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the differences between the member's working file and the revision that is being checked out.

Note:
Specifying the **--yes** or **--no** options automatically answers **y** or **n** to the confirmation question, and also automatically answers **y** or **n** to all questions asked.

Note:
By default, MKS Integrity prompts you for the type of merge (manual or automatic) to perform and what action to take if a conflict is encountered.

--lockType=[exclusive|nonexclusive|auto]
specifies the type of lock obtained on checkout.

--lockType=exclusive obtains an exclusive lock on the member. Exclusive locks enable only one member at a time to lock a specific revision.

--lockType=nonexclusive obtains a non-exclusive lock on the member. Non-exclusive locks enable multiple users to check out the same revision for editing.

--lockType=auto obtains a lock on the member based on the locks policy. For information on the locks policy, contact your administrator. If the locks policy does noe require a lock, but the **--lock** option is set, a non-exclusive lock is obtained.

--[no|confirm]downgradeOnLockConflict
controls whether to downgrade your lock to non-exclusive if another user has an exclusive lock on the revision.

--downgradeOnLockConflict means always downgrade.

--nodowngradeOnLockConflict means never downgrade.

--confirmdowngradeOnLockConflict means always ask before downgrading.

--[no|confirm]moveLock
moves any lock you have on a revision in the same development path to the member revision, including the change package associated with the lock operation. Since you can only have one lock per member per development path, if you already have another revision locked, you need to move that lock to the member revision in order for the check in to succeed. The **downgradeOnLockConflict** option determines what happends if the member revision already has an exclusive lock. You also need to move your lock if it is associated with a different change package than the one you are using for the check out operation.

--moveLock means always move the lock.

--nomoveLock means never move the lock.

--confirmmoveLock means always ask whether to move the lock.

-u

--unlock

keeps the member unlocked, and is equivalent to **--nolock**.

--[no]update

controls whether to update the project's member revision to the checked out revision.

Note: In a variant Sandbox, specifying **si co --noupdate --branchVariant** *member* updates the member revision.

--[no|un]expand

controls whether to expand, unexpand, or ignore keywords in the member file. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
$Revision$
$Name$
$ProjectLabel$
$ProjectName$
$ProjectSetting attribute$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
$State$
```

-f

overwrites the working file if it has changed since it was last checked in. This is equivalent to specifying **--overwriteChanged**.

--[no|confirm]overwriteChanged

controls whether to overwrite the working file if it has changed since it was last checked in.

--overwriteChanged means always do it

--nooverwriteChanged means never do it

--confirmoverwriteChanged means always ask before doing it. When asked to confirm overwriting the working file, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the differences between the member's working file and the revision that is being checked out.

Note:

Specifying the **--yes** or **--no** options automatically answers **y** or **n** to the confirmation question, and also automatically answers **y** or **n** to all questions asked.

--[no|confirm]overwriteDeferred

controls whether to overwrite the working file if there is a deferred drop or update revision operation on the member.

--overwriteDeferred means always do it

--nooverwriteDeferred means never do it

--confirmoverwriteDeferred means always ask before doing it. When asked to confirm overwriting the working file, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the differences between the member's working file and the revision that is being checked out.

Note:

Specifying the **--yes** or **--no** options automatically answers **y** or **n** to the confirmation question, and also automatically answers **y** or **n** to all questions asked.

--[no]restoreTimestamp

controls whether to restore the timestamp of the working file to the timestamp of the revision in the member history. If this is not specified, the timestamp is set to the current date and time.

sandbox member...

identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si ci, **si closecp**, **si cpissues**, **si createcp**, **si diff**, **si edit**, **si lock**, **si resync**, **si rlog**, **si unlock**, **si viewcps**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si configuresandbox

configures Sandbox properties

SYNOPSIS

si configuresandbox [**--lineTerminator=***[lf|cr|crlf|native]*] [**--[no]shared**] [**--[no]sparse**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *sandbox location...*

DESCRIPTION

MKS Integrity provides a way to configure the following Sandbox properties: Sandbox sharing, line terminator and sparse settings. For example,

```
si configuresandbox --lineTerminator=cr --shared --sparse
c:/Aurora_Program/bin/Libra/project.pj
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

---lineTerminator=*[lf|cr|crlf|native]*

explicitly identifies the characters to use as the line terminator for this Sandbox. *lf* is a line feed character, *crlf* is the combination of the carriage return and line feed characters, and *native* uses the default line terminator in the client operating system: *crlf* in Windows, *cr* in Mac OS, and *lf* in UNIX.

Note:

When you create a Sandbox MKS Integrity remembers the setting of the line terminator, and all checkouts (using **si co**) and resyncs (using **si resync**) in that Sandbox use the same line terminator setting. If you want to override the line terminator on an individual member, you must use some utility such as the MKS Toolkit **flip** command.

--[no]shared

controls whether or not to make the Sandbox shared.

Configuring the Sandbox to be shared makes it possible for other users to share its working files. Other users must first import the Sandbox to register it with their Integrity clients (see **si importsandbox**). When a top-level Sandbox is configured to be shared, all subsandboxes are shared as well. Subsandboxes cannot be individually configured to be shared or not shared.

When Sandbox sharing is removed, users sharing the Sandbox each receive an error message stating the change in sharing configuration the next time they attempt to access the Sandbox or perform an operation on its contents.

--[no]sparse

controls whether or not to make the Sandbox sparse.

Making the Sandbox sparse with **--sparse** affects the way the Sandbox behaves with the **si ci** check in command and the **si co** check out command, as well as **si resync**. The intent of a sparse Sandbox is that it contains only those working files that are actively being worked on by you at the time. Using **si ci** to check in a member to a sparse Sandbox removes the working file from the Sandbox (unless you did a check in locked, using the **-l** option). Similarly, using **si resync** on the Sandbox will by default remove the working files for those members which are not locked by you, although you can override this with **si resync** **--populate** in a sparse Sandbox, which in effect causes the resync to behave like it does in a "normal", non-sparse Sandbox.

sandbox location...

specifies the path of the Sandbox directory including the project file. For example, `C:\sourcecode\framework\scripts\project.pj`.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si createsandbox, **si importsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si configuresubproject

configures a subproject's type

SYNOPSIS

si configuresubproject [**-r** *subproject revision*|**--subprojectRevision=***subproject revision*]
[**--subprojectDevelopmentPath=***subproject development path name*|**--variant=***subproject development path name*]
[**--[no]failOnAmbiguousProject**] [**--type=[normal**|**variant**|**build**|**default]**] [(**-P** *project*|**--project=***project*)]
[(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*]
[**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*]
[**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]
[**--cpid=***ID*|**--changePackageId=***ID*] [**--[no**|**confirm]closeCP**] [**--issueId=***value*] *subproject* or *subsandbox*

DESCRIPTION

Once you have created or added a subproject, you can modify the type to suit your needs. For example, you can change a normal subproject to a build subproject to suspend development, or you can change a variant subproject to a normal subproject to continue development on the main trunk.

Important: Any changes you make when reconfiguring a subproject affects the project as a whole and any shared subprojects. More specifically, changing the type, revision, or development path of a subproject can radically change the list of members of that subproject. After configuring a subproject, existing sandboxes whose contents were in sync with the subproject before it was configured will display deltas when you use **si viewproject** and **si viewsandbox**, reflecting the following possible differences:

- Members that exist in both the old and new version of the subproject display a delta if the working revision in the Sandbox is different from the new member revision.
- Members that are in the new version of the subproject but were not in the old version of the subproject display a "no working file" delta, indicating that the Sandbox does not have a copy of the new member yet.
- Members that were in the old version of the subproject but are not in the new version display as former members.
- Subprojects that were in the old version of the subproject but are not in the new version will display as former subprojects.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

-r=*subproject revision*
--subprojectRevision=*subproject revision*
specifies the checkpoint number (for build subprojects). For example, `1.5`. This option is used with **--type=***build*.

--subprojectDevelopmentPath=*subproject development path name*
--variant=*subproject development path name*
specifies the development path name (for variant subprojects). For example, `Service_Pack3`. This option is used with **--type=***variant*

--type=[normal|**variant**|**build**|**default]**
specifies the new configuration type for the subproject.

--type=normal configures the subproject to the master (non-variant) version of the subproject.

--type=variant configures the subproject based upon a specific development path. The option is used with **--variant=***subproject development path name* or **--subprojectDevelopmentPath=***subproject development path name*.

--type=build configures the subproject as a static subproject based upon a specific checkpoint of the master project that is used for building or testing the project, but not for further development. This option is used with **-r** *subproject revision* or **--subprojectRevision=***subproject revision*.

--type=default configures the subproject based on the type that is consistent with the parent project that you are adding the subproject to. For example, if you add a subproject to a normal project, the subproject is added as a normal type. For information on what the default type is, see your administrator.

--cpid=*ID*
--changePackageId=*ID*
identifies a change package that is notified of this action, for example, *1452:1*. Note the following about using this option:

This option can only be specified if change packages are enabled.

- If the integration is enabled, but it is not mandatory to specify a change package, or if no change package is applicable, you can specify **--changePackageId=***:none*.
- The next time you are prompted for a change package ID, the last used change package ID is displayed by default, if *:none* was specified or at least one change package was successfully updated from the last operation.

--[no|confirm]closeCP
closes the change package after command completion.

--issueId=*value*
specifies the issue ID that corresponds to the change package that records the changes.

subproject
subsandbox
specifies the subproject or sub Sandbox name to which the new configuration applies, for example, `tools.pj`

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addsubproject, **si checkpoint**, **si createproject**, **si createsandbox**, **si createsubproject**, **si drop**, **si dropproject**, **si importproject**, **si projectinfo**, **si projects**, **si sharesubproject**, **si movesubproject**, **si viewproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si connect

establishes a connection to an Integrity Server

SYNOPSIS

si connect [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si connect establishes a connection to an Integrity Server host. Most commands implicitly connect to the host, this does so explicitly. For example

```
si connect --hostname=abcFinancial --port=7001 --user=jriley --password=jriley
```

connects to the *abcFinancial* server, logged in as *jriley*.

In fact, all the other commands call **si connect** to establish the connection. The client supports multiple connections to multiple servers.

TIP: You can use **si disconnect** to disconnect from an Integrity Server host.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--hostname=*server*
: identifies the name of the host server where the Integrity Server is located.

--password=*password*
: identifies the password to use for connecting to the Integrity Server.

--port=*number*
: identifies the port on the host server where the Integrity Server is located.

--user=*name*
: identifies the user to use for connecting to the Integrity Server. This typically defaults to the name you have used to log into your client machine.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si disconnect, **si exit**, **si servers**

Miscellaneous:

diagnostics, **options**, **preferences**

si cpissues

displays all MKS Integrity issues that are in a valid state to accept new change packages

SYNOPSIS

si cpissues [**--fields=***field1[:width1],field2[:width2]...*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [**--[no]persist**] [**--quiet**]

DESCRIPTION

si cpissues displays all MKS Integrity issues that are in a valid state to accept new change packages. For example,

```
si cpissues --fields=id,summary --persist
```

Only issues assigned to the user are displayed.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--fields=*field1[:width1],field2[:width2]...*

allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a `space`.

The fields available for printing can be one or more of the following:

id

displays issue IDs.

summary

displays issue summaries.

--[no]persist

controls the persistence of CLI views.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

Si add, **si ci**, **si co**, **si createcp**, **si createcp**, **si drop**, **si lock**, **si viewcps**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si createcp

creates a new change package

SYNOPSIS

si createcp [**--description=***value*] [**--issueId=***value*] [**--summary=***value*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si createcp creates a new change package. For example,

```
si createcp --description="Second attempt at patch fix." --issueId=34332 --summary="Patch Fix
For Customer Ajax"
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--description=*value*
specifies a description of the change package being created.

--issueId=*value*
specifies the ID of the issue you are creating a change package for. This option can only be specified if MKS Integrity configuration management is enabled with workflows and documents.

Important: Your ability to create a change package is based on the change package creation policy for the type that you want to create a change package for.

--summary=*value*
specifies a brief summary of the change package being created.

Note: A change package summary is mandatory.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, **si ci**, **si co**, **si cpissues**, **si drop**, **si lock**, **si viewcps**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si createdevpath

creates a development path

SYNOPSIS

si createdevpath [**--projectRevision=***rev*] [**--devpath=***path*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si createdevpath creates a development path from a specific project checkpoint.

A development path is an identifier given to a new branch of software development. Changes made through the new development path are kept separate from the main development trunk unless you choose to merge them later.

Once you have created a development path, you can open and work in a project on the development path by opening a variant Sandbox using the **si createsandbox** command.

MKS Integrity allows multiple developers to point to the same development path, each using their own variant Sandbox. In the variant Sandbox, you can see the current state of the project along the development path and the changes made by other developers using it.

When a development path is created for a project, it is also created for all subprojects, reserving the assigned name as a unique identifier and ensuring no two paths can share the same name. The project is also automatically checkpointed when a development path is created

Branching projects is useful for:

- extracting and building from previous versions of a project
- building customized versions of a product
- performing branch development work
- performing post-release maintenance
- fixing defects in previous versions of the product
- testing new features outside of the main development path
- experimenting with research that does not affect regular development

There are main models to follow when branching projects:

- release based branching
- project based branching

For more information on these models, see the *MKS Integrity User Guide*

Note:

You can branch members instead of projects by specifying **--branch** when checking files in. You still may have to merge changes, but the development path remains the same. For details on branching members, see the **si ci** reference page.

The following is an example for creating a development path:

```
si createdevpath --project=c:/Aurora_Program/bin/Libra/project.pj --projectRevision=1.3 --
devpath=DevStream2
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--devpath=*path*

identifies the name of the new development path.

Note: The following characters may not be used in a development path: \n, \r, \t, :, [', '], **#**.

--projectRevision=*rev*

identifies the checkpointed revision number to create the variant against.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, si createsandbox, si dropdevpath, si memberinfo, si projectinfo, si revisioninfo, si rlog

Miscellaneous:

ACL, diagnostics, options, preferences

si createmetricinfo

creates a new metric to be measured

SYNOPSIS

si createmetricinfo [**--description=***value*] [**--name=***value*][**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si createmetricinfo creates a new metric to be measured.

You can use a third party tool to track metrics in addition to the standard metrics tracked by MKS Integrity. Metrics tracked by a third party tool must be defined in MKS Integrity and recorded for a specific project checkpoint. You define a metric by specifying a name and description for it using this command. Both **--name** and **--description** are mandatory.

You can run this command against an existing metric name to replace the description.

To define a metric you need the `Metrics` permission.

Note: Metrics are only supported for database type repositories.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--name=*value*
: specifies a name for the metric being created

--description=*value*
: specifies a description for the metric being created

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si viewmetricsinfo, **si calculateprojectmetrics**, **si viewprojectmetrics**, **si addprojectmetric**

Miscellaneous:

diagnostics, **options**, **preferences**

si createproject

creates a new empty project

SYNOPSIS

si createproject [**--[no]openView**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *project location*

DESCRIPTION

si createproject creates a new, empty project on the Integrity Server in a specified directory. For example,

```
si createproject c:/Aurora_Program/bin/Libra/project.pj
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]openView
: controls whether to open the project view after this new project is registered on the server. If **--noopenView** is used, the project view does not open.

project location
: identifies a location on the Integrity Server for the new project. Regardless of your operating system, all paths are specified here using forward slashes (/), and are server absolute paths. You must also specify the name of the project file as part of the location; typically this is `project.pj`.

Note:
: By convention, you should name your new project file with a suffix of `.pj`.

Intermediate directories on the Integrity Server are created for you. Remember, however, that server-side restrictions may exist controlling where you are permitted to locate any projects. Contact your system administrator for permitted locations.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si checkpoint, **si createsandbox**, **si createsubproject**, **si dropproject**, **si importproject**, **si projects**, **si viewproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si createsandbox

creates a new Sandbox on your local machine

SYNOPSIS

si createsandbox [(**-R**|**--**[**no**|**confirm**]**recurse**)] [**--lineTerminator=***[lf|cr|crlf|native]*] [**--**[**no**]**failOnAmbiguousProject**] [**--**[**no**]**populate**] [**--**[**no**]**sparse**] [**--**[**no**]**openView**] [**--**[**no**]**shared**] [(**-P***project*|**--project=***project*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--**[**no**]**batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--xmlapi**] *directory*

DESCRIPTION

si createsandbox creates a new Sandbox on the client machine in a specified directory. When you want to work with a project, you create a Sandbox. A Sandbox resides on your local drive and is a mirror image of the project. Working in a Sandbox allows you to operate in your own workspace without interfering with the work of others. Once a Sandbox is created, you can add files or members to that Sandbox. The project is then updated to reflect the addition of new project members.

Different types of sandboxes are available for different types of development. ***Normal*** sandboxes are useful for the sequential development of a project over the long or short term. ***Variant*** sandboxes are useful for branching off the main development path. ***Build*** sandboxes are useful for testing a specific checkpoint of the project.

Using Variant Sandboxes

A variant Sandbox is based on a specific checkpoint of your project that has been branched off the main development path. When you create a variant Sandbox, you choose the development path to use. For example:

```
si createsandbox --devpath=MyVariant
```

In the variant Sandbox, you can see the current state of the project along that development path and the changes made by other developers using it. When a variant Sandbox is created for the first time, it is also created for all subprojects, reserving the assigned name as a unique identifier and ensuring no two paths can share the same name. For more information on creating development paths, see **si createdevpath**

Using Build Sandboxes

After major milestones, such as product releases, you might want to recreate a static version of an entire project as it existed at some point in the past. You create a build Sandbox, to build or test the project, not to begin further work along a new development path.

A build Sandbox is associated with a particular project checkpoint, and has no development path (since it is static and not meant for further development). For example:

```
si createsandbox --projectRevision=1.34
```

creates a ***build*** Sandbox based on the project checkpoint 1.34, and it cannot be modified.

With a build Sandbox, you cannot:

- check out, lock, or check in members
- remove or add members
- freeze or thaw members
- checkpoint the master project
- modify project or member attributes
- revert members
- set the member revision

However, with a build Sandbox, you can:

- change labels and states (see **si addlabel**)
- resynchronize your Sandbox (see **si resync**)
- compare a member revision in the build Sandbox to another revision (see **si diff**)
- merge a member revision (**si merge**) in the build Sandbox with another revision (of course, you cannot check that merged file back into the build Sandbox)
- check for differences between project checkpoints, (see **si mods**)

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--lineTerminator=*[lf|cr|crlf|native]*

explicitly identifies the characters to use as the line terminator for this Sandbox. *lf* is a line feed character, *crlf* is the combination of the carriage return and line feed characters, and *native* uses the default line terminator in the client operating system: *crlf* in Windows, *cr* in Mac OS, and *lf* in UNIX.

Note:

When you create a Sandbox, MKS Integrity remembers the setting of the line terminator, and all checkouts (using **si co**) and resyncs (using **si resync**) in that Sandbox use the same line terminator setting. If you want to override the line terminator on an individual member, you must use some utility such as the MKS Toolkit **flip** command.

--[no]populate

controls whether to populate the Sandbox with read-only working files for all members.

--[no]sparse

controls whether to create a sparse Sandbox.

Creating a sparse Sandbox with **--sparse** affects the way the Sandbox behaves with the **si ci** check in command and the **si co** check out command, as well as **si resync**. This option creates a Sandbox with no working files and defers the creation of Sandbox directories and subsandboxes until you need to work with them. A sparse Sandbox does not retain working files when a member is checked in and continues to function this way throughout its use; however, once created, Sandbox directories and subsandboxes remain in the Sandbox. Using **si ci** to check in a member to a sparse Sandbox removes the working file from the Sandbox (unless you did a check in locked, using the **-l** option). Similarly, using **si resync** on the Sandbox by default removes the working files for those members that are not locked by you, although you can override this with **si resync --populate** in a sparse Sandbox, which in effect causes the resync to behave like it does in a normal, non-sparse Sandbox.

--[no]openView

controls whether to open the Sandbox view after this new Sandbox is registered on the client. If **--noopenView** is used, the Sandbox view does not open.

--[no]shared

controls whether or not to make the Sandbox shared. The default is not shared.

Configuring the Sandbox to be shared makes it possible for other users to share its working files. Other users must first import the Sandbox to register it with their MKS Integrity clients (see **si importsandbox**). When a top-level Sandbox is configured to be shared, all subsandboxes are shared as well. Subsandboxes cannot be individually configured to be shared or not shared.

directory

identifies a location on the MKS Integrity client system for the new Sandbox. Use spaces to identify more than one directory.

Note:

Do NOT append the name of the project file (typically `project.pj`) to the directory. The name will be calculated automatically.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, **si checkpoint**, **si co**, **si createdevpath**, **si createproject**, **si dropsandbox**, **si importsandbox**, **si merge**, **si resync**, **si sandboxes**, **si viewsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si createsubproject

creates a subproject or adds an existing project as a subproject

SYNOPSIS

si createsubproject [**--[no|confirm]reuseDroppedSubproject**] [**--[no]createSubprojects**] [(**-P** *project*| **--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*| **--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port** =*number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--cpid=***ID*|**--changePackageId=***ID*] [**--[no|confirm]closeCP**] [**--issueId=***value*] *subproject location...*

DESCRIPTION

si createsubproject creates a subproject on the server in a specified directory, which must be under an existing project specified by the **-P** *project* or **--project=***project* options. For example,

```
si createsubproject --project=c:/Aurora_Program/project.pj
c:/Aurora_Program/bin/Libra/project.pj
```

This command works with Regular and Variant sandboxes, but not with Build sandboxes.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]reuseDroppedSubproject
: controls whether to add an existing subproject to the named project. This can be useful if the existing subproject had been dropped from a project.

  Note: While **si addsubproject** is the recommended command for adding an existing subproject to a project, this option is available for backwards compatibility and any scripts you may use.

--[no]createSubprojects
: controls whether to create one subproject for each directory encountered when creating subprojects.

--cpid=*ID*
--changePackageId=*ID*
: identifies a change package that is notified of this action, for example, *1452:1*. Note the following about using this option:

  - This option can only be specified if change packages are enabled.
  - If the integration is enabled, but it is not mandatory to specify a change package, or if no change package is applicable, you can specify **--changePackageId=***:none*.
  - The next time you are prompted for a change package ID, the last used change package ID is displayed by default, if *:none* was specified or at least one change package was successfully updated from the last operation.

--[no|confirm]closeCP
: closes the change package after command completion.

--issueId=*value*
: specifies the issue ID that corresponds to the change package that records the changes.

subproject location...
: identifies a location on the Integrity Server for the new subproject. Regardless of your operating system, all paths are specified here using forward slashes (/), and may be server relative or absolute paths. You must also specify the name of the project file as part of the location; typically this is `project.pj`.

  Note:
  Subprojects must be added "in tree", that is, in the project directory or a subdirectory.

  As with the **si createproject** command, remember that server-side restrictions may exist controlling where you are permitted to locate any subprojects on the Integrity Server. Contact your system administrator for permitted locations.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si checkpoint, **si createproject**, **si createsandbox**, **si drop**, **si dropproject**, **si importproject**, **si projectinfo**, **si projects**, **si viewproject**, **si configuresubproject**, **si addsubproject**, **si movesubproject**, **si sharesubproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si deletelabel

removes a label from a member

SYNOPSIS

si deletelabel [(**-L** *label*|**--label=***label*)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [**-R**|**--[no|confirm]recurse**] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si deletelabel removes a label from one or more members. For example,

```
si deletelabel --label="GA Release" c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

If you do not specify any members, the **si deletelabel** command applies to all project members with associated archives, deleting the label from whatever revision the label happens to exist on in the archive. Please note the **--projectRevision** option is not used for identifying the revision to delete labels from.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

-L *label*
--label=*label*
: identifies a label. Labels cannot contain colons(:), square brackets ([ ]), or leading spaces. Additionally, they cannot have the same format as a valid revision number.

Note:
: Labels that include spaces must be enclosed by quotes.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
: **si addlabel**, **si addprojectlabel**, **si deleteprojectlabel**, **si viewlabels**

Miscellaneous:
: **ACL**, **diagnostics**, **options**, **preferences**

si deleteprojectlabel

removes a label from a project

SYNOPSIS

si deleteprojectlabel [(**-L** *label*|**--label=***label*)] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--[no]failOnAmbiguousProject**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si deleteprojectlabel removes a label from an MKS Integrity configuration management project, deleting the label from whatever project checkpoint the label happens to exist on. For example,

```
si deleteprojectlabel --label="GA Release" --project=c:/Aurora_Program/bin/Libra/project.pj
```

Note that the **--projectRevision** option is not used for identifying the project checkpoint to delete labels from; it is the generic option available on all project-based commands that allows you to specify the checkpoint of the build project you want to work with.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

-L *label*
--label=*label*
identifies a label. Labels cannot contain colons(:), square brackets ([ ]), or leading spaces. Additionally, they cannot have the same format as a valid revision number.

Note:
Labels that include spaces must be enclosed by quotes.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si addlabel, **si addprojectlabel**, **si deletelabel**, **si viewlabels**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si deleterevision

permanently deletes a revision from a member

SYNOPSIS

si deleterevision [**--[no]confirm**] [**--[no]confirminuse**] [**-f**] [**--range=***value*] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si deleterevision deletes an archived revision of a project member. For example,

```
si deleterevision --range=1.2-1.5 --confirm --confirminuse
c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

deletes revisions 1.2, 1.3, 1.4, and 1.5 from the archive `xml_man` without prompting to confirm to the user if the revisions are in use in other projects.

IMPORTANT:

It is recommended that you never delete a revision. Since historical versions of projects may still point at the deleted revision, it may become impossible to recreate old projects.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]confirm

controls the display of confirmation messages.

--[no]confirminuse

controls whether MKS Integrity warns you about deleting the revision if it is used in other projects. If the revision is used in other projects, MKS Integrity warns you that deleting the selected revision will break the listed items. This option is valid only with the database repository.

-f

forces the revision to be deleted, without any confirmation message.

--range=*value*

specifies a range of revision numbers. The format of *value* for a single revision is simply the revision number itself. Separate multiple non sequential revisions with a comma, and use a dash to indicate a sequential range, for example, `1.2-1.5`.

You may also prefix or append a dash to a single revision number, meaning you want to delete from 1.1 to the specified revision, or from the specified revision to the tip. For example, you would specify `-1.6` to delete revisions 1.1 to 1.6. Or you could specify `1.2-` to delete revisions 1.2 to the tip revision, whatever it may be.

The revisions are deleted in order from the highest revision to the lowest.

member...

identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si diff, **si merge**, **si revisioninfo**, **si rlog**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si demote

demotes the state of a member

SYNOPSIS

si demote [(**-r** *rev*|**--revision=***rev*)] [(**-s** *state*|**--state=***state*)] [(**-R**|**--[no|confirm]recurse**)] [**--devpath=***path*] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--projectRevision=***rev*] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [**--cwd=***directory*] [(**-F** *file*|**--selectionFile=***file*)] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [(**-N**|**--no**)] [**--[no]batch**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [(**-?**|**--usage**)] [(**-Y**|**--yes**)] *member...*

DESCRIPTION

si demote demotes a project member to a lower state of development. For example,

```
si demote --state=Development c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

A state is a textual description defined by the administrator to classify the condition of a revision in a history. If no state is assigned to a revision at check-in, a default value of `Exp` (for Experimental) is used. See the *Integrity Server Administration Guide* for details on setting up states.

Project members can also be demoted to a predefined lower state of development using this command.

Note: Demoting members is for historical purposes only. To more effectively control project workflow, MKS recommends using the MKS Integrity integration.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

-s *state*
--state=*state*
: specifies the target state to be set for the demoted member. If a state is not specified, the member is demoted to the next state.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si demoteproject, **si promote**, **si promoteproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si demoteproject

demotes the state of a project

SYNOPSIS

si demoteproject [(**-s** *state*|**--state=***state*)] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si demoteproject demotes a project to a lower state of development. For example,

```
si demoteproject --state=Development --project=c:/Aurora_Program/bin/Libra/project.pj
```

A state is a textual description, defined by the administrator, used to classify the condition of a revision in a history. If no state is assigned to a revision at checkpoint, a default value of `Exp` (for Experimental) is used. See the *Integrity Server Administration Guide* for details on setting up states.

Just as you can with a project member, you can demote the project itself (change its state, in other words). Only the project is demoted, not the members of the project.

Note: Demoting projects is for historical purposes only. To more effectively control project workflow, MKS recommends using MKS Integrity.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

-s *state*
--state=*state*
: specifies the target state to be set for the demoted project. If no state is specified, the project is demoted to the next state.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si demote, **si promote**, **si promoteproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si diff

displays differences between two revisions of a member

SYNOPSIS

si diff [**--context=***value*] [**--[no]ignoreBlanks**] [**--[no]ignoreCase**] [**--[no]ignoreWhitespace**] [**-r** *rev1*[**-r** *rev2*]] [**--[no]failOnAmbiguousProject**] [(**-R**|**--[no|confirm]recurse**)] [**--guiCharacterEncoding=***value*] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si diff displays the differences between two revisions of the specified member, or one revision and the member's working file. For example,

```
si diff -r 1.3 -r 1.4 --project=c:/Documentation/Man_Pages/xml_man/project.pj si_add.1.xml
```

If no members are specified, **si diff** operates on all project members.

By default, in a Sandbox, this compares the member revision against the working file.

In graphical mode where the **-g** or **--gui** option is used, this invokes the visual difference utility. Otherwise, the **diff** or **diffb** utilities are used depending on whether it is evaluating differences in a text or binary file.

If used with **-g** or **--gui**, only one member can be differenced at a time, and the **--context** option is ignored.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--context=*value*
shows *value* number of lines of context, before and after each difference between two files. **si diff** marks lines removed from the "old" revision with -, marks lines added to the "new" revision with +, and marks lines changed in both files with !

--[no]ignoreBlanks
controls whether to ignore blanks when comparing textual revisions. When **--ignoreBlanks** is used, all differences involving blanks are ignored except blanks at the start of a line. These are treated differently than blanks between other characters. The following pair would be considered different, for example, because of the leading blank:

```
 a b c def
a b c def
```

--[no]ignoreCase
controls whether to ignore case when comparing revisions.

--[no]ignoreWhitespace
controls whether to ignore white space at the end of each line (except the newline) and treat all consecutive strings of white space elsewhere in a line as equivalent (effectively, reducing all strings of white space to a single space for the purpose of comparing lines).

-r *rev1*[**-r** *rev2*]
uses a specified revision. Specify this option twice to identify the two revisions to be compared; the first instance is the "old" revision, the second is the "new". If the second revision is not specified, the working file is assumed. Both revisions must be specified if operating against a project. *rev* can be a valid revision number or a label. If neither is specified, it defaults to member revision.

--guiCharacterEncoding=*value*
specifies the character encoding to use for the revision contents in the GUI, and can only be specified with the -g option. MKS Integrity automatically decodes UTF-8 revision contents based on the presence of a byte order mark (BOM) in the file. You can set character encoding preferences for each view and command through your client preferences. The preference setting is used if the character set cannot be determined through the file's BOM, or if no character set is specified in the command line. By default, the character set in the preferences matches the default character set provided by the client's operating system locale. The option does not apply to pure CLI output, or third party merge and/or differencing tools. Possible values are:

```
UTF-8
US-ASCII
windows-1252
ISO-8859 -1
```

```
ISO-8859 -15
IBM437
IBM850
IBM863
EUC-JP
Shift_JIS
x-euc-jp-linux
x-eucJP-Open
x-windows-iso2022jp
IBM862
ISO-8859-8
ISO-8859-9
```

member...

identifies a specific project or Sandbox member, depending on whether the **-S** *sandbox* option or the **-P** *project* option is specified. To specify more than one member, use spaces between them.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

diff, **si ci**, **si merge**, **si mods**, **si revisioninfo**, **si rlog**, **si updaterevision**, **si viewrevision**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si difffiles

compares the differences between two text files

SYNOPSIS

si difffiles [**--context=***value*] [**--[no]ignoreBlanks**] [**--[no]ignoreCase**] [**--[no]ignoreWhitespace**] [**--guiCharacterEncoding=***value*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *file1 file2*

DESCRIPTION

si difffiles compares the differences between two text files. This is useful when you want to compare the differences between two arbitrary text files (not members or revisions).

In graphical mode where the **-g** or **--gui** option is used, this invokes a dialog box that allows you to browse for each file. The results appear in MKS Visual Difference.

Note: The dialog box keeps track of the last 10 files in both fields.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--context=*value*
: shows *value* number of lines of context, before and after each difference between two files. **si difffiles** marks lines removed from the first specified file with -, marks lines added to the second specified file with +, and marks lines changed in both files with !

--[no]ignoreBlanks
: controls whether to ignore blanks when comparing text files. When **--ignoreBlanks** is used, all differences involving blanks are ignored except blanks at the start of a line. These are treated differently than blanks between other characters. The following pair would be considered different, for example, because of the leading blank:

```
 a b c def
a b c def
```

--[no]ignoreCase
: controls whether to ignore case when comparing files.

--[no]ignoreWhitespace
: controls whether to ignore white space at the end of each line (except the newline) and treat all consecutive strings of white space elsewhere in a line as equivalent (effectively, reducing all strings of white space to a single space for the purpose of comparing lines).

--guiCharacterEncoding=*value*
: specifies the character encoding to use for the revision contents in the GUI, and can only be specified with the -g option. MKS Integrity automatically decodes UTF-8 revision contents based on the presence of a byte order mark (BOM) in the file. You can set character encoding preferences for each view and command through your client preferences. The preference setting is used if the character set cannot be determined through the file's BOM, or if no character set is specified in the command line. By default, the character set in the preferences matches the default character set provided by the client's operating system locale. The option does not apply to pure CLI output, or third party merge and/or differencing tools. Possible values are:

```
UTF-8
US-ASCII
windows-1252
ISO-8859 -1
ISO-8859 -15
IBM437
IBM850
IBM863
EUC-JP
Shift_JIS
x-euc-jp-linux
x-eucJP-Open
x-windows-iso2022jp
IBM862
```

```
ISO-8859-8
ISO-8859-9
```

file1 file2

identifies the two files you want to compare.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

diff, **si diff**, **si ci**, **si merge**, **si mods**, **si revisioninfo**, **si rlog**, **si updaterevision**, **si viewrevision**,

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si discardcp

discards a change package

SYNOPSIS

si discardcp [**--[no]confirm**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F***file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *issue|issue:change package id...*

DESCRIPTION

si discardcp discards a change package. For example,

```
si discardcp --confirm 34343:2
```

MKS Integrity provides a way to remove change packages from active use when they are not needed by discarding them.

Change packages can only be discarded if they are in the Open or Rejected states. In order to discard a change package, that change package must be both created by you and not contain any committed entries. However, a change package administrator may discard a change package created by another user.

When a change package is discarded, the following happens where applicable:

- all uncommitted entries are removed from the change package
- all deferred operations corresponding to deferred entries are reverted
- all locks on members corresponding to lock entries are released
- all work in progress entries are reverted
- all pending operations corresponding to pending entries are reverted, and appear as discarded entries in the review log (to preserve the review history)
- all pending revisions that correspond to pending entries are deleted
- any archives created for pending members associated with pending entries in the change package are deleted from the server (pre-existing archives that were shared to create a pending member are not deleted)

Note the following about discarded change packages:

- The change package ID for the discarded change package can never be used for any future change packages that are created.
- Discarded change packages have a state of Discarded, and can still be viewed using the Change Package filter.
- If the change package has undergone a review, the review log persists.
- Discarded change packages can be opened and used again.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]confirm
: specifies if to confirm discarding the change package.

issue...
issue:change package id...
: *issue* identifies a specific issue that contains all change packages that you want to discard; use spaces to specify more than one issue.
issue:change package id identifies a specific change package to discard; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, **si ci**, **si co**, **si cpissues**, **si createcp**, **si drop**, **si lock**, **si viewcps**, **si acceptcp**, **si rejectcp**, **si opencp**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si disconnect

disconnects from the Integrity Server

SYNOPSIS

si disconnect [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--[no]confirm**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si disconnect disconnects the client connection to the host Integrity Server. For example

```
si disconnect --hostname=abcFinancial --port=7001 --user=jriley
```

Note: When disconnecting a connection that is the current connection, all open client views close. All new views use the connection specified in the MKS Integrity preferences, or an existing connection, as the new current connection.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]confirm
: controls whether to implement the Integrity Server disconnection confirmation policy.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si connect, **si servers**

Miscellaneous:

diagnostics, **options**, **preferences**

si drop

drops a member or subproject from a project

SYNOPSIS

si drop [**--cpid=***ID*|**--changePackageId=***ID*] [**--[no|confirm]closeCP**] [**--issueId=***ID*] [(**-f**|**--[no]confirm**)] [**--[no]defer**] [**--[no]delete**] [**--filter=***filteroptions*] [**--[no]failOnAmbiguousProject**] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member... or subproject...*

DESCRIPTION

si drop drops specified members and subprojects from a project. For example,

```
si drop --confirm --nodelete c:/Aurora_Program/bin/Libra/project.pj
```

Files dropped in a Sandbox are not actually dropped from the Sandbox until you use the **si resync** command, or if you specify **--delete** with this command. This allows Sandbox users to be protected from a file deletion until an appropriate point, and therefore be in control.

Note:
This command does not operate in a Build Sandbox.

If a member has outlived its usefulness or just does not belong in a project any more, you can remove it at any time. After you remove a member from a project, the member is no longer listed as part of the Sandbox or master project, but the member history remains, in case you need to recreate an earlier version of the project. The build project and variants may still have the file as a member.

Adding a member (see **si add**) which has previously been dropped will result in the history of the dropped member becoming the history of the newly added member.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]closeCP
controls whether to close the associated change package.

--nocloseCP means do not close the change package.

--confirmcloseCP means ask before closing the change package.

--closeCP always closes the change package.

-f
--[no]confirm
controls the display of confirmation messages. **-f** forces an action without any prompt.

--[no]defer
controls whether to delay the drop operation in the project until the deferred operation is submitted. The operation in the Sandbox still takes place immediately.

--[no]delete
control whether the working file should be deleted immediately from your Sandbox, when using **-S** or **--sandbox**.

member... or subproject...
identifies a specific member or subproject; use spaces to specify more than one.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si add, **si closecp**, **si cpissues**, **si createcp**, **si viewcps**

Miscellaneous:

ACL, diagnostics, options, preferences

si dropdevpath

drops a variant project from the master project

SYNOPSIS

si dropdevpath [**--devpath=***path*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [**--[no]failOnAmbiguousProject**] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si dropdevpath drops a variant project. For example,

```
si dropdevpath --forceconfirm=yes --project=c:/Aurora_Program/bin/Libra/project.pj --
devpath=ReleaseCycle2
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si createdevpath, **si createproject**, **si dropproject**, **si projectinfo**, **si projects**, **si viewproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si dropmemberattr

drops an attribute from a member

SYNOPSIS

si dropmemberattr [(**--attr=***key*[=*value*]|**--attribute=***key*[=*value*])] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si dropmemberattr drops an attribute from a member. For example,

```
si dropmemberattr --attribute=OS=nt c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

If no members are specified, the command applies to all members of the project.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--attr=*key*=*value*
--attribute=*key*=*value*
: identifies the attribute key and, optionally, the value to be dropped. If only the key is given, this command deletes any attribute with that key. If a *value* is given for the key, then only a key with that value is dropped.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addmemberattr, **si addprojectattr**, **si dropprojectattr**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si dropproject

drops (unregisters) a project from an Integrity Server

SYNOPSIS

si dropproject [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *project...*

DESCRIPTION

si dropproject unregisters a project from the Integrity Server. For example,

```
si dropproject --forceConfirm=yes c:/Aurora_Program/project.pj
```

Once a project is dropped, project commands do not operate on the project, and any sandboxes pointing to that project do not operate.

Note:

This works only on "top level" projects; to drop a subproject from its master project, use **si drop**.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

project...

identifies a specific project location with a fully qualified server-relative path; use spaces to specify more than one.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si createproject, **si createsubproject**, **si drop**, **si importproject**, **si projects**, **si viewproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si dropprojectattr

drops an attribute from a project

SYNOPSIS

si dropprojectattr [(**--attr=***key*[=*value*]|**--attribute =***key*[=*value*])] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si dropprojectattr drops an attribute from a project.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--attr=*key*=*value*
--attribute=*key*=*value*
: identifies the attribute key and, optionally, the value to be dropped. If only the key is given, this command deletes any attribute with that key. If a *value* is given for the key, then only a key with that value is dropped.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addmemberattr, **si addprojectattr**, **si dropmemberattr**, **si dropsandboxattr**, **si addsandboxattr**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si dropsandbox

drops a Sandbox

SYNOPSIS

si dropsandbox [**--[no]confirm**] [**--delete=***[none|members|all]*] [**-f**] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *sandbox location...*

DESCRIPTION

si dropsandbox unregisters a Sandbox from the Integrity Client. For example,

```
si dropsandbox --confirm --delete=none c:/Aurora_Program/project.pj
```

Once a Sandbox is dropped, Sandbox commands do not operate on the Sandbox.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

-f
--[no]confirm
controls the display of confirmation messages. **-f** forces an action without any prompt.

--delete=*[none|members|all]*
deletes files from the client Sandbox directory according to the value specified. If *none* is specified, then no deletion occurs after the Sandbox is dropped. If *members* is specified, then the specified Sandbox, sub Sandboxes, and members are deleted. If *all* is specified, then everything in the current directory and subdirectories is deleted. For example:

```
--delete=none sandbox.pj
```

causes sandbox.pj to be removed from the registry, but not deleted from the client directory.

```
--delete=members sandbox.pj
```

causes sandbox.pj to be removed from the registry and deleted from the client directory along with all its sub Sandboxes and members.

```
--delete=all sandbox.pj
```

causes sandbox.pj to be removed from the registry and all files in the client directory and subdirectories to be deleted.

Note:
Since **--delete=***all* is a destructive operation, it always requires confirmation. This is a preventative security measure. For example, if you create a Sandbox in the root of a file system, this operation would delete your entire drive.

sandbox location...
identifies the location of a specific Sandbox; use spaces to specify more than one Sandbox.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si createsandbox, **si importsandbox**, **si sandboxes**, **si viewsandbox**

Miscellaneous:
diagnostics, **options**, **preferences**

si dropsandboxattr

drops Sandbox attributes

SYNOPSIS

si dropsandboxattr [**--attr**|**attribute=***key=value*] [**-S**|**--sandbox=***value*] [**--[no]failOnAmbiguousProject**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F***file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si dropsandboxattr drops Sandbox attributes.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--attr=*key=value*
--attribute=*key=value*
: specifies the attribute to drop.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addsandboxattr, **si configuresandbox**, **add projectattr**, **dropprojectattr**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si edit

opens a revision or working file for editing

SYNOPSIS

si edit [**--editor=***value*] [(**-r** *rev*|**--revision=***rev*)] [**--[no]systemEditor**] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si edit opens a current, former, or deferred member revision for editing. For example,

```
si edit c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

In a Sandbox, by default not specifying the **-r** option edits the working file.

If this command is invoked in the CLI mode (that is, no **-g** or **--gui** option is used), then the ***EDITOR*** environment variable is used and runs with the name of the temporary file containing the requested revision. If the ***EDITOR*** variable is not set, or if the **-g** or **--gui** option is used, a system-specific method is used. In particular, on WIN32-based systems, the operating system is instructed to open the file and does so based on the extension association. Under UNIX, the **-g** or **--gui** option uses the ***EDITOR*** environment variable; if it is not set, then Vi is used.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--editor=*value*
: identifies the editor to be used, overriding any ***EDITOR*** environment variables or system setting.

--[no]systemEditor
: controls whether to use the system editor, set through **si setprefs**. If set, this overrides any editor setting if **-g** or **--gui** is given. This is used to temporarily override the preference system.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si co, **si ci**, **si lock**, **si unlock**, **si viewrevision**,

Miscellaneous:

diagnostics, **options**, **preferences**

si editcp

edits an existing change package

SYNOPSIS

si editcp [**--description=***value*] [**--descriptionFile=***value*]
[**--state=[open**|**closed**|**submitted**|**accepted**|**rejected**|**discarded**|**commitfailed]**] [**--summary=***value*] [**--summary=***value*]
[**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)]
[(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [**--quiet**] [(**-g**|**--gui**)]
[**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *issue|issue:change package id...*

DESCRIPTION

si editcp edits an existing change package. For example,

```
si editcp --state=open 3433:3
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--description=*value*
specifies a description of the change package being edited.

--descriptionFile=*value*
specifies the file that contains the description for the change package.

--state=[open|**closed**|**submitted**|**accepted**|**rejected**|**discarded**|**commitfailed]**
specifies the state of the change package. The change package progresses through states as part of the change package review workflow. If reviews are not mandatory, some states are not applicable. For more information, see the *MKS Integrity User Guide*

--state=open specifies that the change package is open.

--state=closed specifies that the change package is closed.

--state=submitted specifies that the change package has been submitted for review.

--state=accepted specifies that the change package has been accepted by all reviewers.

--state=rejected specifies that the change package has been rejected by at least one reviewer.

--state=discarded specifies that the change package has been discarded.

--state=commitfailed specifies that the change package has failed to commit to the repository.

-summary=*value*
specifies a brief summary of the change package being edited.

issue...
issue:change package id...
issue identifies a specific issue that contains all change packages that you want to edit; use spaces to specify more than one issue.

issue:change package id identifies a specific change package to edit; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si discardcp, **si opencp**, **si closecp**, **si createcp**, **si add**, **si ci**, **si co**, **si cpissues**, **si createcp**, **si drop**, **si lock**, **si viewcps**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si exit

exits the current MKS Integrity client session

SYNOPSIS

si exit [**--[no]abort**] [**--[no|confirm]shutdown**] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si exit exits the current MKS Integrity client session. When you run any MKS Integrity command from the CLI, or when you open the Integrity Client GUI, you start a client session. Only one client session is running at a time, regardless of how many GUI windows you have open or how many CLIs you are using. To close the GUI you use the appropriate menu commands, and to close the CLI you use the **si exit** command. In both cases you may have the further option of shutting down the Integrity client altogether, based on your preferences (see **si setprefs**); if you do not, the client is still running and available for additional interaction. If you do shut down the client completely, then running any MKS Integrity command from the CLI or opening the Integrity Client GUI starts a new client session.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]abort
: controls whether to shut down any other MKS Integrity commands that may be running. Some commands allow you to specify a **--persist** option which keeps those commands active during a client session. Using **--abort** with **si exit** is recommended for stopping all persistent views that have been specified with another command's **--persist** option.

--[no|confirm]shutdown
: controls the shutting down of the MKS Integrity client without getting a prompt.

Note:
: Specifying **--noshutdown** with **si exit** does nothing.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si about, **si gui**, **si setprefs**

Miscellaneous:

diagnostics, **options**, **preferences**

si freeze

freezes a project member

SYNOPSIS

si freeze [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si freeze freezes one or more project members. For example,

```
si freeze c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

If no members are specified, **si freeze** assumes all members of the project. Members can be unfrozen with the **si thaw** command. When a member is frozen, all MKS Integrity operations are run on the frozen member revision. The member revision and attributes of the frozen member cannot be changed until it is thawed. This ensures that a particular revision is always picked up, even if development on it continues.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

member...

identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si checkpoint, **si thaw**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si gui

starts the Integrity Client graphical user interface

SYNOPSIS

si gui [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si gui starts the Integrity Client graphical user interface.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Miscellaneous:

diagnostics, **options**, **preferences**

si import

imports a member from a server file or from an old MKS Integrity archive into an existing MKS Integrity configuration management project

SYNOPSIS

si import [**--archive=***filename*] [**--author=***name*] [(**--binaryFormat**|**--textFormat**)] [(**-d** *desc*|**--description=***desc*)] [**--descriptionFile=***file*] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--[no]createSubprojects**] [**--[no]unexpand**] [(**-r** *rev*|**--revision=***rev*)] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [**--cwd=***directory*] [(**-F** *file*|**--selectionFile=***file*)] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [(**-N**|**--no**)] [**--[no]batch**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [(**-?**|**--usage**)] [(**-Y**|**--yes**)] [**--cpid=***ID*|**--changePackageId=***ID*] [**--[no|confirm]closeCP**] [**--[no]defer**] [**--issueId=***value*] [**-R**|**--[no|confirm]recurse**] [**--[no|confirm]includeFormers**] [**--exclude=***file:pattern,dir:pattern...*] [**--include=***file:pattern,dir:pattern...*] *nonmember...*

DESCRIPTION

si import imports an external file or archive into the MKS Integrity configuration management repository and adds it as a member of a MKS Integrity configuration management project.For example,

```
si import c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

Note: This command is not supported for database repositories.

The `si import` command can be used to do the following:

- Import archives from Source Integrity Standard (an older version of MKS Integrity or MKS Source) into an existing project on the Integrity Server.

  Note: Once a member has been imported, you may not use an older version of MKS Integrity to manipulate its archives.

- Import server resident files

Note: To retrieve a dropped member history, use the **si addmemberfromarchive** command.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--archive=*filename*
: sets the archive file name, *filename*, containing the server-side path and name by which the newly added member will be archived. This must be an existing archive file - this option does not create a new archive. This is an advanced option that effectively specifies an archive path to override the default mapping, allowing a basic form of archive sharing between projects; its use is not recommended except by advanced MKS Integrity administrators.

-r=*value*
--revision=*value*
: specifies the revision at which the member is to be added. This must be an existing revision in the archive. By default, the member is added at the latest revision on the main branch.

--author=*name*
: specifies an author name.

--binaryFormat
--textFormat
: forces the storage of the member file to occur in binary format or text format. This option overrides the preferences settings that control default behavior in the application for storage format. See the **si setprefs** and **si viewprefs** commands for more details on preferences.

--devpath=*path*
: identifies the name of the development path.

-d *desc*
--description=*desc*
: specifies a description for the new member history; this setting and the **--descriptionFile** option are mutually exclusive. If specifying a *nonmember* that has an existing history, this description does not overwrite an existing description and is ignored.

--descriptionFile=*file*
specifies a file for obtaining a description to apply to the new member history; this setting and the **--description** option are mutually exclusive. If specifying a *nonmember* that has an existing history, this description does not overwrite an existing description and is ignored.

--[no]createSubprojects
controls whether to create subprojects for each subdirectory encountered when importing members. This option is commonly used where you anticipate working with a large directory structure, because multiple subprojects are easier to manage than many subdirectories within one project. For example, specifying:

```
si import --createSubprojects D:/Aurora_Project/source_code/buttons/activebutton.c
```

creates a subproject for the `/buttons` directory and adds the file `activebutton.c`. Without the **--createSubprojects** option, the file and its path simply would be added to the project in the project directory.

--[no]unexpand
controls whether to unexpand keywords in the member file before checking it into the history. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
$Revision$
$Name$
$ProjectName$
$ProjectSetting attribute$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
$State$
```

--[no|confirm]closeCP
closes the change package after command completion.

--[no]defer
controls whether to delay the operation in the project until the deferred operation is submitted. The operation in the Sandbox still takes place immediately.

--[no|confirm]includeFormers
controls whether to include former members when importing.

-R
--[no|confirm]recurse
controls whether to select non-members recursively. Non-members are files that exist in the server directory but have not previously been imported to the server repository.

--exclude=*file:pattern,dir:pattern...*
specifies a file that contains a glob pattern for excluding members.

--include=*file:pattern,dir:pattern...*
specifies a file that contains a glob pattern for including members.

nonmember...
identifies a specific file to import to your project; use spaces to specify more than one. The file pathnames are server-side, relative to the project.

Note: All files must be added "in tree", that is, the nonmember must exist in the project directory or a subdirectory. Shared "out of tree" histories are supported by using the --archive option to point to the "out of tree" history that you want to associate with the member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using si setprefs or si viewprefs, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, si addmemberfromarchive

Miscellaneous:

ACL, diagnostics, options, preferences

si importproject

imports an external project and registers it on an Integrity Server

SYNOPSIS

si importproject [**--[no]openView**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--[no]add**][**--onExistingProject=***[confirm|cancel|error|add|overwrite]*] *project location...*

DESCRIPTION

si importproject imports an external project into the Integrity Server and registers it with the Integrity Server. For example,

```
si importproject --onExistingProject=cancel c:/Aurora_Program/bin/Libra/project.pj
```

Once a project or member is imported, MKS Integrity commands can be performed upon it.

Note:

This command is not supported for database repositories. For more information on importing projects for database repositories, see the *Database Repository Migrator*.

You import projects if you need to migrate them from Source Integrity Standard (an earlier version of MKS Integrity or MKS Source) or if you need to restore projects that were dropped in earlier versions of MKS Integrity. If you only want to expose a project already in the MKS Integrity configuration management repository, use the **si addproject** command.

Note:

MKS Integrity does not support encrypted archives. If your existing project has encrypted archives, you must first decrypt the archives before importing and registering the project with the MKS Integrity Server.

When importing projects, the import process automatically creates histories for any non-archived members and checkpoints any non-archived projects. Before you import a MKS Integrity project, the project files must already reside in the server repository.

If you are importing a dropped or corrupted project to its former repository location, the project must first be reclaimed. For more information, see the *MKS Integrity Server Adminstration Guide*.

The processing of this command may take some time, since it traverses all subprojects in the imported project, and may perform various operations on them, such as checkpointing.

The administrator may restrict the path names of projects being imported on the server. This is done through the *si.serverRoot* property in the `si.properties` file on the MKS Integrity Server. If this is set, then any project location given must be under that directory. For details, see the *MKS Integrity Server Installation and Configuration Guide*.

Note:

Once a project has been imported, you cannot perform commands on it using an older version of MKS Integrity (formerly MKS Source or Source Integrity). If for some reason you need to use an older version of MKS Integrity on a project, you must first drop it using **si dropproject**. The project must be re-imported before it can be used again with this version of MKS Integrity.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]openView
controls whether to open the project view after this project is imported and registered on the server. If **--noopenView** is used, the project view does not open.

--[no]add
specifies if to add the project to the list of registered projects.

--onExistingProject=[confirm|cancel|error|add|overwrite]
controls whether to allow the project to be imported if a project with the same name is already registered and whether the imported project is added to or overwrites the existing project.

--onExistingProject=confirm means always ask whether to add to the existing project, overwrite the existing project, or cancel the operation.

--onExistingProject=cancel means cancel the operation.

--onExistingProject=error means do not allow the project to be imported if it already exists.

--onExistingProject=add means add the contents of the imported project to the existing project.

--onExistingProject=overwrite means overwrite the contents of the existing project with the imported project.

project location...

identifies a location on the Integrity Server for the existing project to be imported. Regardless of your operating system, all paths are specified here using forward slashes (/) and should include the name of the project file (typically `project.pj`). Use spaces to specify more than one project location.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addproject, **si checkpoint**, **si createproject**, **si createsandbox**, **si createsubproject**, **si dropproject**, **si importsandbox**, **si projects**, **si viewproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si importsandbox

imports an existing sandbox

SYNOPSIS

si importsandbox [**--[no]openView**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [**--[no]failOnAmbiguousProject**] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *file...*

DESCRIPTION

si importsandbox imports an existing Sandbox and registers it as a new Sandbox on the Integrity Client. For example,

```
si importsandbox c:/Aurora_Program/bin/Libra/project.pj
```

You must correlate this Sandbox with a project that already exists on the Integrity Server, and it is assumed that the project you identify is the project the Sandbox was first created from. A Sandbox must first be registered before further commands can operate on the Sandbox and its members.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]openView
: controls whether to open the Sandbox view after this Sandbox is imported and registered on the client. If **--noopenView** is used, the Sandbox view does not open.

file...
: identifies an existing Sandbox file (`project.pj`) on the MKS Integrity client system to be imported; use spaces to specify more than one.

Note:
: Unlike the **si createsandbox** command, for **si importsandbox** you must include the name of the Sandbox file, which is typically `project.pj`.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, **si createproject**, **si dropsandbox**, **si importproject**, **si sandboxes**, **si viewsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si integrations

manages integrations.

SYNOPSIS

si integrations [**--action=***[list|enable|enableAll|disable|disableAll]*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *string...*

DESCRIPTION

si integrations enables or disables MKS Integrity integrations. You can select from a list of available IDE integrations and enable or disable them as required to work with your preferred IDE. There is no requirement to re-install the Integrity Client, but if prompted, you may need to reboot your computer or restart the IDE for the integration to take effect.

An Integrated Development Environment, or IDE, is a supported development application, such as Sybase PowerBuilder, that allows you to access MKS Integrity functionality by installing the appropriate extension.

For complete information on using third party integrations with MKS Integrity, see the Integrations User Guide.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--action=[*list*|*enable*|*enableAll*|*disable*|*disableAll*]
: specifies the action to perform on the integration. An action must be specified. There is no default specification for this option, so to list all of the available integrations, use the *list* flag.

  Note: MKS Integrity asks you to restart your computer to complete the integration changes. If you are disabling[enabling] a number of integrations, you do not need to restart between each **si integrations** command. When you have disabled[enabled] all the required integrations, you can restart your machine.

string...
: specifies the integration to perform an action upon.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si loadrc

loads the Integrity Client preferences file

SYNOPSIS

si loadrc [**--[no]merge**] [**--rc=***value*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [**-F**|**--selectionFile=***value*]

DESCRIPTION

si loadrc loads the user's `IntegrityClient.rc` file, which contains your personal preferences for configuring the Integrity Client. The `IntegrityClient.rc` file is loaded when you start the client; however, this command reloads the file without restarting the client. If for some reason your personal `IntegrityClient.rc` file has changed, this command will reload your preferences. Your preferences file shouldn't change, unless you happen to copy someone else's file or if you happen to restore a backup that you made.

Your administrator can lock certain preferences from the Integrity Server, preventing you from configuring them using the **si setprefs** command. Preferences that are locked -- viewable with **si viewprefs** -- display (`locked`) at the end of the output line. The following preferences can be locked from the server by editing MKS Integrity policies in the Administration Client:

Tip: For information on editing MKS Integrity policies, see the *Integrity Server Installation and Configuration Guide*.

Preference	Affected Commands
`branchIfVariant`	**si co**, **si lock**
`breakLock`	**si unlock**
`changePackageID`	**si co**, **si lock**
`createBranch`	**si lock**
`onLock`	**si co**
`restoreTimestamp`	**si resync**, **si resynccp**, **si revert**
`retainWorkingFile`	**si add**, **si ci**
`saveTimestamp`	**si add**, **si ci**
`sparse`	**si importsandbox**
`updateMemberRev`	**si ci**

Warning: Do not edit the `IntegrityClient.rc` file manually, because preferences that appear more than once in the `IntegrityClient.rc` file can cause the Integrity Client to behave unpredictably.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]merge
: controls whether settings from the loaded file should be merged into existing preferences.

--rc=*value*
: identifies the file containing settings for running the Integrity Client. The default is the `IntegrityClient.rc` file in your home directory.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
: **si setprefs**, **si viewprefs**

Miscellaneous:
: **diagnostics**, **options**, **preferences**

si locate

displays where a project, subproject, member, or revision is used

SYNOPSIS

si locate [**--depth=**[**current**|**build**|**all**]] [**--devpathscope=**[**this**|**others**|**all**]] [**--**[**no**]**failOnAmbiguousProject**] [**--distinct=**[**project**|**devpath**|**registeredproject**]] [**--**[**no**]**exactmatch**] [**--**[**no**]**casesensitive**] [**--**[**no**]**limittoactivepaths**] [**--listfields=***field1[:width1],field2[:width2]...*] [**--mode=**[**distinct**|**list**]] [**--numberofresults=***value*] [**--memberbyname=***value*] [**--subprojectbyname=***value*] [**--projectscope=**[**this**|**others**|**all**]] [(**-r** *value*|**--revision=***value*)] [**--height=***value*] [**--width=***value*] [**-x=***value*] [**-y=***value*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--**[**no**]**batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member/subproject*

DESCRIPTION

As projects grow in complexity, it can be difficult to determine the relationships that MKS Integrity configuration management objects--projects, subprojects, and members--have with one another. When working with multiple development paths and shared subprojects or shared member archives, it becomes very important to understand what areas of a project might be affected by parallel development. Using the **si locate** command, you can locate where a specific MKS Integrity object is used in your source code repository and refine and sort the search results.

For example, you might need to:

- Determine what active projects (a project currently under development) a member, subproject, or registered top-level project is shared into.
- Determine what development paths a selected member or subproject is a part of.
- Determine if a selected member revision is part of a checkpoint.
- Confirm all existing checkpoints for a selected revision before you delete it.
- Display the development path that a corresponding member revision belongs to.
- Display which master projects a selected subproject belongs to.

MKS Integrity displays the search results in one of the following modes:

- Distinct Mode provides a high-level view of the information.
- List Mode provides a detailed view of the information.

Key Considerations:

- The **si locate** command can only be used in the database repository. Using the command in the RCS based repository returns an error message.
- You require the *OpenProject* permission for the project you are running the **si locate** command from. When the Locate view displays, MKS Integrity informs you if there are additional projects that you do not have permission to view.
- Archive locations are not displayed in the search results.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--depth=[**current**|**build**|**all**]

specifies the depth of the search. **--depth=current** searches current project configurations (normal and variant). **--depth=build** searches current project configurations and checkpoints. **--depth=all** searches current project configurations, checkpoints, and in between checkpoints.

--devpathscope=[**this**|**others**|**all**]

specifies which development paths to include in the search. **--devpathscope=this** searches the current development path. **--devpathscope=others** searches all development paths, except the current development path. **--devpathscope=all** searches all development paths.

--distinct=[**project**|**devpath**|**registeredproject**]

specifies the information you want displayed in Distinct Mode. This option must be used with **--mode=distinct**. **--distinct=project** lists only distinct project names. **--distinct=devpath** lists only development path names. **--distinct=registeredproject** lists only registered top-level project names.

--[**no**]**exactmatch**

when searching by name, specifies whether to match the name exactly. This option can only be used in conjunction with the **--**

memberbyname or **--subprojectbyname** options.

--[no]casesensitive
when searching by name, specifies whether to match the case of the name. This option can only be used in conjunction with the **--memberbyname** or **--subprojectbyname** options and with case insensitive databases.

--[no]limittoactivepaths
specifies whether to search projects referenced by current registered top level projects on the server.

--listfields=*field1[:width1],field2[:width2]...*
allows you to select fields to be displayed when you use the **--mode=list** option, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[: width]* (in pixels) for each field is optional (widths are only available in the GUI). In the CLI, fields are separated with a space. This option must be used with **--mode=list**.

You can display one or more of the following fields:

checkpoints
displays the checkpoints that the object belongs to.

configPath
displays the configuration path for the object.

dates
displays the date ranges in which the object exists.

devpath
displays the development paths that the object belongs to.

flatPath
displays the flat path for the object.

name
displays the object's path and name.

project
displays the projects that the object belongs to.

revisions
displays the member's revisions.

registeredproject
displays the top-level projects that the object belongs to.

--mode=[distinct|list]
specifies the level of detail to display in the search results. **--mode=distinct** displays only the projects and/or development paths that contain the object you are searching. This option must be used with **--distinct=[project|devpath|registeredproject]**. **--mode=list** displays all occurrences of the object in the projects and/or development paths that contain the object. If you do not specify a mode, **--mode=distinct** and **--distinct=devpath** are used.

--memberbyname=*value*
specifies the name of the member to search for. The name you enter can contain the * and ? wildcard characters. For UNIX, the name you enter can contain the \ escape character.

--subprojectbyname=*value*
specifies the name of the subproject to search for. The name you enter can contain the * and ? wildcard characters. For UNIX, the name you enter can contain the \ escape character.

--numberofresults=*value*
when searching by name, specifies the maximum number of results to return. This option can only be used in conjunction with the **--memberbyname** or **--subprojectbyname** options.

--projectscope=[this|others|all]
specifies which projects to search. **--projectscope=this** searches the specified project. **--projectscope=others** searches all projects and subprojects, except the current project. **--projectscope=all** searches all projects and subprojects.

DIAGNOSTICS

See the diagnostics reference page for possible exit status values.

PREFERENCES

Using si setprefs or si viewprefs, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si deleterevision, si viewproject, si viewprojecthistory, si viewhistory

Miscellaneous:

ACL, diagnostics, options, preferences

si lock

locks project members

SYNOPSIS

si lock [**--cpid=***ID*|**--changePackageId=***ID*] [**--lockType=***[exclusive|nonexclusive|auto]*]
[**--[no|confirm]downgradeOnLockConflict** [**--issueId =***ID*] [(**-r** *rev*|**--revision=***rev*)] [(**-R**|**--[no|confirm]recurse**)]
[**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)]
[**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)]
[(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*]
[**--[no|confirm]revisionMismatchIsError**] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*]
[**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si lock locks one or more project members. For example,

```
si lock --lockType=nonexclusive c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

If no members are specified, **si lock** applies to all project members. By default, the member revision is locked and not necessarily the working revision, if in a Sandbox.

This is purely a project operation -- if performed in a Sandbox, no memory of what revision was locked is maintained in the Sandbox. However, you can create a work in progress indicator for the the associated working file in the Sandbox.

Depending on the locking policy configured by your administrator, you may hold a lock on only one revision of a particular member at a time. It is not an "error" to relock an already locked member.

If the target revision is already locked by a user other than yourself, then specifying the **--branch** option creates a branch revision and locks it. Otherwise, you cannot lock the revision at all.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--lockType=[exclusive|nonexclusive|auto]
: specifies the type of lock obtained on checkout.

  --lockType=exclusive obtains an exclusive lock on the member. Exclusive locks enable only one member at a time to lock a specific revision.

  --lockType=nonexclusive obtains a non-exclusive lock on the member. Non-exclusive locks enable multiple users to check out the same revision for editing.

  --lockType=auto obtains a lock on the member based on the locks policy. For information on the locks policy, contact your administrator. If the locks policy does noe require a lock, but the **--lock** option is set, a non-exclusive lock is obtained.

--[no|confirm]downgradeOnLockConflict
: controls whether to downgrade your lock to non-exclusive if another user has an exclusive lock on the revision.

  --downgradeOnLockConflict means always downgrade.

  --nodowngradeOnLockConflict means never downgrade.

  --confirmdowngradeOnLockConflict means always ask before downgrading.

--[no|confirm]revisionMismatchIsError
: controls whether to display an error message if the working revision does not match the member revision. This only applies if you are performing the lock operation in a Sandbox.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

[si ci](), [si co](), [si edit](), [si rlog](), [si unlock]()

Miscellaneous:

[ACL](), [diagnostics](), [options](), [preferences]()

si locks

displays a user's locks.

SYNOPSIS

si locks [**--locker=***value*] [**--fields=***field1[:width1],field2[:width2]...*] [**--height=***value*] [**--width=***value*] [**-x=***value*] [**-y=***value*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--[no]persist**] [**--status=***[none|gui|default]*]

DESCRIPTION

si locks displays all of a user's locks on the target server.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--locker=*value*
: specifies the user whose locks are retrieved.

--fields=*field1[:width1],field2[:width2]...*

: The fields available for printing can be one or more of the following:

  archive
  : displays the name of the locked archive.

  cpid
  : displays the change package ID number if it is a lock entry in a change package.

  devpath
  : displays the name of the development path where the lock on the revision was made from. This information is relevant when the locking policy is set to devpath, allowing a single user to have a single lock on an archive per development path.

  host
  : displays the hostname of the computer which the lock was performed on.

  member
  : displays the name of the locked member.

  project
  : displays the name and path of the project where the member revision was locked from. If the member revision was locked from a shared subproject, it is the subproject name and path that are displayed.

  revision
  : displays the locked revision number.

  locktype
  : displays the type of lock on the revision: exclusive or non-exclusive.

  sandbox
  : displays the name of the sandbox where the lock on the revision was made, and is relevant when viewing the information from the machine that locked it.

  timestamp
  : displays the time the archive was locked.

  user
  : displays the user holding the lock.

--[no]persist
: controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using si setprefs or si viewprefs, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, si memberinfo, si mods, si print, si revisioninfo, si rlog, si sandboxinfo, si viewhistory, si viewlabels, si viewprojecthistory, si viewsandbox

Miscellaneous:

ACL, diagnostics, options, preferences

si makewritable

makes Sandbox members writable

SYNOPSIS

si makewritable [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *sandbox member...*

DESCRIPTION

si makewritable makes Sandbox members writable. For example,

```
si makewritable c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

If no members are specified, **si makewritable** applies to all Sandbox members.

This command is useful when you want to edit a file locked by someone else, and you have no intention of checking the files back in.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

sandbox member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si ci, **si co**, **si diff**, **si edit**, **si lock**, **si resync**, **si rlog**, **si unlock**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si memberinfo

lists member information

SYNOPSIS

si memberinfo [**--[no]acl**] [**--[no]attributes**] [**--[no]changePackage**] [**--[no]labels**] [**--[no]rule**] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--lockRecordFormat=***value*] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--[no]locate**] *member*

DESCRIPTION

si memberinfo lists current information about a project member. This includes general information about the member, and specific information on the member revision. For example:

```
si memberinfo c:\Documentation\Man_Pages\xml_man\si_add.1.xml
```

displays

```
Member Name: c:\Documentation\Man_Pages\xml_man\si_add.1.xml
Sandbox Name: c:\Documentation\Man_Pages\project.pj
The member archive is shared
Member Revision: 1.7
    Created By: sueq on July 11, 2009 - 11:36 AM
    State: state_a
    Revision Description:
       Updated for Technical Review
    Labels:
       TechReview1
    Change Package:
        ID: 1:1
    Member Revision:
       In project C:\Aurora_Program\Documentation\project.pj, this revision
       is member revision on:
          development path:Service Pack 1
Attributes: none
This member does not have a revision rule.
```

Note: `The member archive is shared` displays only if the member shares another member's archive and you are using the database repository.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]acl
shows ACL information for the member.

--[no]attributes
controls whether to display member attributes. If no attributes exist, `Attributes:none` is displayed.

--[no]changePackage
controls whether to display change package information (applies only change packages are enabled).

--[no]labels
controls whether to display member labels.

--[no]rule
controls whether to display the member revision rule.

--[no]locate
controls whether to display the project and development path that the member is a member revision in.

Note: If a revision is member revision in a shared subproject, only the original (canonical) project path displays. Any projects where the subproject was added as shared do not display.

Only projects that you have the `OpenProject` permission for display.

Note: This option is not valid for RCS style database repositories.

--lockRecordFormat=*value*

defines the format for displaying lock information for the member. Specify a format string using keywords to represent the information you want to display. You can specify any of the following keywords:

{revision}
displays the revision that is locked.

{locker}
displays the user who locked the revision.

{locktype}
displays the type of lock on the revision (exclusive or non-exclusive).

{locktimestamp}
displays the time when the revision was locked.

{lockcpid}
displays the change package associated with the lock on the revision.

{project}
displays the name and path of the project where the member revision was locked from. If the member revision was locked from a shared subproject, it is the subproject name and path that are displayed.

{devpath}
displays the name of the development path where the lock on the revision was made from.

{sandbox}
displays the name of the Sandbox where the lock on the revision was made. This is relevant when viewing the information from the locker host.

{hostname}
displays the hostname of the computer that locked the the revision.

{hascpid}
displays 1 if the lock has a change package associated with it, 0 if there is no associated change package.

{hassandbox}
displays 1 if there is Sandbox information available for the lock, 0 if no Sandbox information is available.

{hasdevpath}
displays 1 if the lock was made from a development path, 0 if it wasn't.

{member}
displays the name of the locked revision.

member
identifies one specific member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, **si mods**, **si revisioninfo**, **si rlog**, **si setmemberrule**, **si updatemember**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si merge

merges data from two revisions of a project member into a working file

SYNOPSIS

si merge [**--mergeType=***[confirm|cancel|automatic|manual]*] [**--onMergeConflict=***[confirm|cancel|mark|launchtool|highlight|error]*] [**--[no]failOnAmbiguousProject**] [**--outputFile=***value*] [**-r** *rev*] [**--resolve**] [**--guiCharacterEncoding=***value*] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *sandbox member...*

DESCRIPTION

si merge merges data from two revisions of a project member against a root revision. By default, the result overwrites the working file. This operates in a Sandbox. You cannot perform a merge on a project-only basis, since the result is left in your Sandbox.

The second **-r** *rev* specified is merged with the revision in the current working file. The first **-r** *rev* specified is the *root* revision. Both the working file revision and the other revision should have been derived from this *root* revision. For example, suppose development has branched from revision 1.3, with maintenance occurring on the 1.3.1 branch. Further suppose the tip of that branch is revision 1.3.1.3, the headrev member revision is 1.5, and that the revision in the Sandbox is 1.5.

```
1.5 (headrev)
1.4
        1.3.1.3 (maintenance branch tip)
        1.3.1.2
        1.3.1.1
1.3 (root)
```

The command **si merge -r** *1.3* **-r** *1.3.1.3* merges the changes in 1.3.1.3 that have occurred since 1.3, into revision 1.5 in the working file in the Sandbox.

si merge is useful for combining separate changes into a checked out revision. Suppose *root* is the original, and both *headrev* and *branch tip* are modifications of *root*. Then, **si merge** combines both changes.

An overlap occurs if both branches of development (*headrev* and *branch tip*) have changes in a common segment of lines. Depending on the value used in **--onMergeConflict**, **si merge** displays how many overlaps occurred, and includes both alternatives in the result, delimiting them as follows:

```
<<<<<<< file1
lines in file1
=======
lines in file2
>>>>>>> file2
```

If there are overlaps, by default MKS Integrity asks you how you want to resolve the conflict. If you specify **--mergeType=automatic** and **--onMergeConflict=highlight**, it is up to you to determine how to resolve the conflict by editing the result.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--mergeType=*[confirm|cancel|automatic|manual]*

specifies how you want to complete the merge.

--mergeType=*confirm* prompts you to confirm a merge type.

--mergeType=*cancel* cancels the selected merge.

--mergeType=*automatic* completes the merge process without launching the MKS Visual Merge tool.

Warning: While MKS Integrity and MKS Visual Merge are capable of performing automatic merging, MKS cannot guarantee that the merged results are “correct”. MKS recommends that you examine and test the merged results before checking them into the repository.

--mergeType=*manual* completes the merge operation through the MKS Visual Merge tool or a third party merge tool, depending on your Difference and Merge Tool preferences. The merge tool launches, displaying the revisions you want to merge. For more information on the MKS Visual Merge tool, refer to the *MKS Integrity User Guide*.

--onMergeConflict =*[confirm|cancel|mark|launchtool|highlight|error]*

specifies what to do when conflicts occur during the merge.

--onMergeConflict=*confirm* prompts you to confirm a merge conflict option.

--onMergeConflict=*cancel* cancels the merge.

--onMergeConflict=*mark* marks the working file in your Sandbox indicating that merging is required without completing all of the merge related tasks. This provides the time to investigate conflicts, edit, or difference blocks before finishing the merge.

Tip: To display the affected member revisions, use the **si print --filter=unresolvedmerges** command. After you resolve the merge conflicts, merge the revisions using the **si merge --resolve** command.

--onMergeConflict=*launchtool* launches the MKS Visual Merge tool to resolve the conflicts, all non-conflicting blocks will already have been applied.

--onMergeConflict=*highlight* indicates conflicts in the working file with the following characters: "<<<<<<" and ">>>>>>". You are responsible for manually resolving conflicts in the working file.

Note: **--onMergeConflict=***launchtool* does not require the **-g** or **--gui** options.

--onMergeConflict=*error* displays an error when a merge conflict is encountered.

--outputFile=*value*
identifies the location and name of a file that is to contain the result of merging data from two member revisions. If you do not specify a file, the working file is used.

-r *rev*
uses a specified revision. *value* can be a valid revision number or a label.

Specify this option twice to identify the two revisions to be merged; the first instance is the "old" revision, the second is the "new". The changes needed to make the "old" into the "new" are then incorporated into the working file or into the **--outputFile**. If you specify the **-r** *rev* option only once, **si merge** uses the latest revision on the default branch as the other revision.

--resolve
merges a previously unresolved merge.

--guiCharacterEncoding=*value*
specifies the character encoding to use for the revision contents in the GUI, and can only be specified with the -g option. MKS Integrity automatically decodes UTF-8 revision contents based on the presence of a byte order mark (BOM) in the file. You can set character encoding preferences for each view and command through your client preferences. The preference setting is used if the character set cannot be determined through the file's BOM, or if no character set is specified in the command line. By default, the character set in the preferences matches the default character set provided by the client's operating system locale. The option does not apply to pure CLI output, or third party merge and/or differencing tools. Possible values are:

```
UTF-8
US-ASCII
windows-1252
ISO-8859 -1
ISO-8859 -15
IBM437
IBM850
IBM863
EUC-JP
Shift_JIS
x-euc-jp-linux
x-eucJP-Open
x-windows-iso2022jp
IBM862
ISO-8859-8
ISO-8859-9
```

sandbox member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using si setprefs or si viewprefs, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si ci, si co, si diff, si mergebranch, si revisioninfo

Miscellaneous:

diagnostics, options, preferences

si mergebranch

merges changes made on a branch into a single revision

SYNOPSIS

si mergebranch [**--[no|confirm]branch**] [**--[no|confirm]branchVariant**] [**--mergeType=***[confirm|cancel|automatic|manual]*] [**--onMergeConflict=***[confirm|cancel|mark|launchtool|highlight|error]*] [**--[no]failOnAmbiguousProject**] [**--branchRevision=***value*] [(**--cpid=***ID*|**--changePackageId=***ID*)] [**--issueId=***ID*] [(**--[no]lock**)] [**--targetRevision=***value*] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *sandbox member...*

DESCRIPTION

si mergebranch allows you to combine the development occurring on different branches by merging the branched revisions into a single revision.

MKS Integrity recognizes cases where a previous merge has occurred, (whether by merging the branch, automatic merging, or using the MKS Visual Merge or a third party merge tool), and merges only changes since the last merge operation. Thus, the first merge branch operation occurring on a given branch merges all changes throughout the branch. Any subsequent merge branch operations will merge only changes since the previous merge operation.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]branch
specifies whether to create a branch off of the target revision (the member revision that the working revision is merged into), if it is locked by another user.

--[no|confirm]branchVariant
specifies whether to create a new branch for the merged revision, if the specified variant does not already have a branch.

--mergeType=*[confirm|cancel|automatic|manual]*
specifies how you want to complete the merge.

--mergeType=*confirm* prompts you to confirm a merge type.

--mergeType=*cancel* cancels the selected merge.

--mergeType=*automatic* completes the merge process without launching the MKS Visual Merge tool.

Warning: While MKS Integrity and MKS Visual Merge are capable of performing automatic merging, MKS cannot guarantee that the merged results are “correct”. MKS recommends that you examine and test the merged results before checking them into the repository.

--mergeType=*manual* completes the merge operation through the MKS Visual Merge tool or a third party merge tool, depending on your Difference and Merge Tool preferences. The merge tool launches, displaying the revisions you want to merge. For more information on the MKS Visual Merge tool, refer to the *MKS Integrity User Guide*.

--onMergeConflict=*[confirm|cancel|mark|launchtool|highlight|error]*
specifies what to do when conflicts occur during the merge.

--onMergeConflict=*confirm* prompts you to confirm a merge conflict option.

--onMergeConflict=*cancel* cancels the merge.

--onMergeConflict=*mark* marks the working file indicating that merging is required without completing all of the merge related tasks. This provides the time you may need to investigate conflicts, edit, or difference blocks before finishing the merge.

--onMergeConflict=*launchtool* launches the MKS Visual Merge tool to resolve the conflicts, all non-conflicting blocks will already have been applied.

--onMergeConflict=*highlight* indicates conflicts in the file with the following characters: "<<<<<<" and ">>>>>>". You are responsible for manually resolving conflicts in the working file.

Note: **--onMergeConflict=***launchtool* does not require the **-g** or **--gui** options.

--onMergeConflict=*error* displays an error when a merge conflict is encountered.

--branchRevision=*revision*
specifies the branch revision that you want to merge into the head revision. If no revision is specified, the working revision is used.

--[no]lock
controls whether to lock the target revision, so that the results of the merge can be checked in. The lock type used is based on your locks policy. For information on your locks policy, contact your administrator.

--targetRevision=*revision*|*:symbolic*
specifies the symbolic name (i.e. member, working, branch, etc.), revision number, or label of the revision that you want to merge the branch revision into; typically the head revision. If no revision is specified, the member revision is used.

sandbox member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si ci, **si diff**, **si edit**, **si lock**, **si merge**, **si resync**, **si rlog**, **si unlock**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si mods

lists changes made to a project since a checkpoint

SYNOPSIS

si mods [**--fields=***field1[:width1],field2[:width2]...*] [(**-r** *rev*)] [**--[no]showAppliedCPList**] [**--[no]showAttrChanges**] [**--[no]showChangePackages**] [**--[no]failOnAmbiguousProject**] [**--[no]showMemberChanges**] [**--[no]showRevDescription**] [**--devpath=***value*] [**--projectRevision=***value*] [(**-R**|**--[no|confirm]recurse**)] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [**--height=***value*] [**--width=***value*] [**-x=***value*] [**-y=***value*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si mods displays modifications to a project between checkpoints or between a checkpoint and the working project.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--fields=*field1[:width1],field2[:width2]...*
: allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a `space`.

  The fields available can be one or more of the following:

  change
  : displays details of changes made to project members or subprojects.

  name
  : displays the member or subproject name.

-r *rev*
: specifies one or more project checkpoints to compare. *rev* can be a valid checkpoint number or label. To compare two specified revisions of the project, use the **-r** option twice. If you only use the **-r** option once, the working project (or last checkpoint on a build project's branch) is compared to the specified project checkpoint. If you do not use this option at all, the working project (or build project checkpoint) is compared to the last checkpoint (or last checkpoint on the build project's branch).

--[no]showAppliedCPList
: controls whether to show change packages applied to a project through **si applycp** or **si resynccp**.

  For more information on applying change packages, see **si applycp**.

--[no]showAttrChanges
: controls whether to show the member attribute changes.

--[no]showMemberChanges
: controls whether to show member changes.

--[no]showRevDescription
: controls whether to show the revision description of new revisions that have been added.

--[no]showChangePackages
: controls whether to show the change packages that correspond to member and subproject operations. Change package information displayed is drawn from the modifications calculated from the checkpoint comparison.

  The By Change Package panel displays the following information:

  - ID displays the change package ID for change packages that contain entries with member or subproject changes between project checkpoints
  - Summary displays the change package summary for change packages that contain entries with member or subproject changes between project checkpoints.

  Modifications that are not associated with a change package (unresolved) are displayed in the bottom frame of the By Change Package panel. Where possible, the member or project name is displayed with full path information. The following are reasons why modifications

can appear in the bottom frame:

- The change package entry is missing; either because one was not recorded for the member or subproject operation, or the entry was discarded.
- The operation that made the modification cannot use a change package, for example, freezing or thawing a member.
- The project state captures compared are from different development paths.

Pending change package entries do not appear in the By Change Package panel (GUI).

-R

--[no|confirm]recurse

controls whether to recurse into subprojects that are determined to be in common between the two projects.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si appendrevdesc, **si archiveinfo**, **si checkpoint**, **si ci**, **si co**, **si diff**, **si memberinfo**, **si merge**, **si projectinfo**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si updatearchive**, **si updaterevision**, **si viewhistory**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si move

moves one or more project members between projects, or between directories in a single project

SYNOPSIS

si move [(**--cpid=***ID*|**--changePackageId=***ID*)] [**--[no|confirm]closeCP**] [**--[no]confirm**] [**--[no]defer**] [**-f**] [**--issueId=***ID*] [**--[no]moveWorkingFile**] [**--[no]failOnAmbiguousProject**] [**--[no|confirm]overwrite**] [**--[no|confirm]branchVariant**] [**--[no]createSubprojects**] [**--targetDevpath=***value*] [**--targetDir=***value*] [**--targetProject=***value*] [**--targetSandbox=***value*] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

As software designs and project structures change, it may become necessary to move one or more members between projects, variants of the same project, or directories in a project.

Moving a member performs a drop in the source location and an add in the target location, creating a new revision in the existing archive. The member's archive remains in its original location in the repository so that change packages, member histories, and project histories continue to work. If the move is performed with a change package, the member is added as a single *Move* entry in the change package. If you are moving multiple members, any common directory prefix shared by the members is automatically removed during the move.

Key Considerations

- Moving members between projects on different servers is not supported.
- You can perform deferred member moves only if both the source and target locations are sandboxes.
- **si move** does not work recursively on subprojects. To move one or more subprojects, use the **si movesubproject** command.
- MKS Integrity detects whether any ACLs exist in the tree defined by the source project, the target project, and the common root between them (or the global ACL, if there is no common root). If any ACLs are found, MKS Integrity warns you so you can manually make any required adjustments to the ACLs after the move is complete.

The following is an example of the command syntax:

```
si move --confirm --targetProject=c:/Aurora_Program/bin/project.pj
c:/Aurora_Program/bin/Libra/code.c
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]closeCP
: controls whether to close the associated change package.

  --nocloseCP means do not close the change package.

  --confirmcloseCP means ask before closing the change package.

  --closeCP always closes the change package.

--[no]confirm
: controls whether to confirm the move before proceeding.

--[no]defer
: controls whether to delay the move operation in the project until the deferred operation is submitted. The move operation in the sandbox still takes place immediately. This option is available only if the source and target are sandboxes.

-f
: forces the move without confirmation.

--[no|confirm]overwrite
: controls whether to overwrite the working file if it exists. This option is valid only if you are moving one or more members from a source sandbox to a target sandbox.

--[no]moveWorkingFile

controls whether to move the working file into the sandbox immediately. This option is valid only if you are moving one or more members from a source sandbox to a target sandbox.

--[no]createSubprojects
controls whether to create subprojects in existing directories that do not contain subprojects.

--targetDevpath=*value*
specifies the target development path (for variant projects). This is a label that was associated with a branch of the project by **si createdevpath**. Paths that include spaces must be enclosed by quotes. The following characters may not be used in a development path: \n, \r, \t, :, ['], #.

Note: This option cannot be used if you specify a target project using a keyword string for the **--targetProject** option.

--targetDir=*value*
specifies the subdirectory in the destination project or sandbox's directory that you want to move the member(s) to.

--targetSandbox=*value*
specifies the name of the target sandbox (can be used as a project redirector).

--targetProject=*value*
specifies the name of the target project. For information on how to specify a project, see the **options** reference page.

--[no|confirm]branchVariant
controls whether MKS Integrity create a branch for the moved member in the target location.

member...
identifies a member to move; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si co, **si rename**, **si movesubproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si movesubproject

moves a subproject between projects or between directories of a single project

SYNOPSIS

si movesubproject [**--[no]confirm**] [**--[no]createSubprojects**] [**-f**] [**--moveWorkingFiles=[none**|**members**|**all]**] [**--[no**|**confirm]overwrite**] [**--targetDevpath=***value*] [**--targetDir=***value*] [**--targetProject=***value*] [**--targetSandbox=***value*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--cpid=***ID*|**--changePackageId=***ID*] [**--[no**|**confirm]closeCP**] [**--issueId=***value*] *subproject or subsandbox*

DESCRIPTION

To meet the needs of changing project configurations, you can move one or more subprojects, and all of its members and sub-subprojects, between projects, and/or directories in a single project, or variants of the same project on the same Integrity Server.

When a subproject is moved, it behaves like a shared subproject. The subproject in the new location continues to be backed by the underlying subproject in the old location and the path and name of the subproject file in the repository remains the same. Any external references (ACL names, event triggers, policy statements) to the moved subproject continue to work because they are based on the subproject's original name; however, a subproject that is moved into a new project hierarchy continues to inherit ACLs from its original hierarchy and not from the new parent project. The moved subproject also retains its configuration type (normal, variant, build). If you are moving multiple subprojects, any common directory prefix shared by the subprojects is automatically removed during the move.

Before using **si movesubproject**, note the following:

- Moving subprojects between projects on different servers is not supported.
- The moved subproject inherits the project or directory ACLs from its original location. You cannot apply the ACLs from the new location to the subproject.
- The path and name of the subproject file in the repository is permanently reserved. If you attempt to create a new subproject using the moved subproject's path and name in the repository, you are prompted to add the existing subproject. If you answer no, the create subproject operation exits without providing you with the option of creating a subproject with a different path and name.
- Deferred subproject moves are not supported.
- You can move one or more subprojects across directories within a single project.
- Moved subprojects are not displayed as shared in a Sandbox or Project view unless the subproject was shared before the move.
- When you move one or more subprojects, you cannot co-locate subprojects in the same directory. If you want to co-locate an existing subproject with another subproject, use the **si sharesubproject** command on the subproject you want to move, followed by **si dropsubproject** in the original location.
- If you move a subproject that is associated with any MKS Integrity issues, the issues no longer display as associated issues for the project. You must edit the MKS Integrity issues to associate them with the subprojects in their new locations.

The following is an example of the command syntax:

```
si movesubproject --confirm --targetProject=c:/Alias_Program/Capricorn/project.pj
c:/Aurora_Program/bin/Libra/project.pj
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]confirm
controls whether to confirm the move before proceeding.

--[no]createSubprojects
controls whether to create subprojects in existing directories that do not contain subprojects.

-f
force the move without confirmation.

--moveWorkingFiles=[none|**members**|**all]**

controls whether to move existing working files in the subproject. This option is valid only if you are moving one or more subprojects from a source Sandbox to a target Sandbox.

--[no|confirm]overwrite
controls whether to confirm before overwriting any existing working files that exist in the new location. This option is valid only if you are moving one or more subprojects from a source Sandbox to a target Sandbox.

--targetDevpath=*value*
specifies the target development path (for variant projects). This is a label that was associated with a branch of the project by **si createdevpath**. Paths that include spaces must be enclosed by quotes. The following characters may not be used in a development path: \n, \r, \t, :, ['], #.

Note: This option cannot be used if you specify a target project using a keyword string for the **--targetProject** option.

--targetDir=*value*
specifies the subdirectory in the destination project or Sandbox's directory that you want to move the subproject(s) to.

--targetProject=*value*
specifies the name of the target project. For information on how to specify a project, see the **options** reference page.

--targetSandbox=*value*
specifies the name of the target Sandbox (can be used as a project redirector).

--cpid=*ID*
--changePackageId=*ID*
identifies a change package that is notified of this action, for example, *1452:1*. Note the following about using this option:

- This option can only be specified if change packages are enabled.
- If the integration is enabled, but it is not mandatory to specify a change package, or if no change package is applicable, you can specify **--changePackageId=***:none*.
- The next time you are prompted for a change package ID, the last used change package ID is displayed by default, if *:none* was specified or at least one change package was successfully updated from the last operation.

--[no|confirm]closeCP
closes the change package after command completion.

--issueId=*value*
specifies the issue ID that corresponds to the change package that records the changes.

subproject or subsandbox
identifies a subproject or sub Sandbox to move; use spaces to specify more than one subproject

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si move, **si sharesubproject**, **si createsubproject**, **si configuresubproject**, **si addsubproject**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si opencp

reopens a change package

SYNOPSIS

si opencp [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F***file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *issue|issue:change package id...*

DESCRIPTION

si opencp reopens a change package. For example,

```
si opencp 3433:5
```

A change package is in the `Open` state when it is first created. This command is useful for reopening a change package after it has been submitted.

The open state is the only state where you can add new entries to a change package. As part of the review workflow, change packages in the following states can be reopened: `CommitFailed`, `Rejected`, and `Discarded`.

Caution:

- Only a Change Package Administrator can reopen a change package in the `Closed` state. Modifying closed change package contents can compromise the integrity of the repository.
- A change package cannot be reopened if it has been propagated to another development path. For more information on propagation change packages, see the *MKS Integrity User Guide*.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

issue...
issue:change package id...

issue identifies a specific issue that contains all change packages that you want to open; use spaces to specify more than one issue.

issue:change package id identifies a specific change package to open; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si cpissues , **si createcp**, **si viewcps**, **si acceptcp**, **si rejectcp**, **si discardcp**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si print

displays member information

SYNOPSIS

si print [**--format=***value*] [**--headerFormat=***value*] [**--noFormat**] [**--noHeaderFormat**] [**--noTrailerFormat**] [**--lockRecordFormat=***value*] [**--lockRecordDetailFormat=***value*] [**--lockseparator=***value*] [**--maxTrunkRevs=***value*] [**--rfilter=***filteroptions*] [(**-r** *rev*|**--revision=***rev*)] [**--range=***value*] [**--trailerFormat=***value*] [**--fields=***field1[:width1],field2[:width2]...*] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y***value*] [**--[no]failOnAmbiguousProject**] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [**--[no]persist**] [**--quiet**] *member...*

DESCRIPTION

si print displays member information on the standard output. For example,

```
si print c:/Documentation/Man_Pages/xml_man/si_add.1.xml
```

If you do not specify any members, **si print** displays all project members. This command is the same as **si rlog**; however, it has a different set of default options. In particular, **--noFormat** and **--noTrailerFormat** are specified, and **--headerFormat=***value* produces one line per member.

Options

See the **si rlog** reference page for descriptions of all options applicable to **si print**. This command takes the universal options available to all ***si*** commands, as well as some general options - these are described on the **options** reference page.

Note: The **--headerFormat=***value* and **--trailerFormat=***value* options only accept global fields. See **si rlog** for the list of global fields. Revision specific fields, such as State, cannot be used with these options.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, **si diff**, **si memberinfo**, **si mods**, **si projectinfo**, **si report**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si viewhistory**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si projectadd

adds projects

SYNOPSIS

si projectadd [(**-P** *value*|**--project=***value*)] [**--archive=***value*] [**--author=***value*] [**--devpath=***value*] [**--onExistingArchive=[confirm**|**cancel**|**sharearchive**|**newarchive]**] [(**--binaryFormat**|**--textFormat**)] [**--[no**|**confirm]includeFormers**] [**--include=***file:pattern,dir:pattern...*] [**--exclude=***file:pattern,dir:pattern...*] [(**--cpid=***ID*|**--changePackageId=***ID*)] [**--issueId=***value*] [**--[no**|**confirm]closeCP**] [(**-d** *value*|**--description=***value*)] [**--descriptionFile=***value*] [(**-l**|**--lock**)] [(**-r** *value*|**--revision=***value*)] [**--[no]retainSourceFile**] [**--sourceFile=***value*] [**--[no]saveTimestamp**] [**--[no]unexpand**] [**--hostname=***value*] [**--port=***value*] [**--password=***value*] [**--user=***value*] [(**-?**|**--usage**)] [**-F***value*] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--forceConfirm=***[yes|no]*] [(**-F** *file*|**--selectionFile=***file*)] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *nonmember...*

DESCRIPTION

si projectadd adds members to MKS Integrity configuration management through a project.

Note:

This command can be only used with the MKS API. It is supported in the CLI only for the purpose of prototyping MKS API implementations. It is used to add members from third party applications to MKS Integrity. Use the **si add** command to add members through a Sandbox.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--author=*value*

specifies an author name.

--onExistingArchive=[confirm|**cancel**|**sharearchive**|**newarchive]**

controls whether to allow sharing of this member's history between projects, or to create a new archive if there is already an existing archive for the member.

--onExistingArchive=confirm means always ask whether to share the archive, create a new archive, or cancel the operation.

--onExistingArchive=cancel means cancel the operation.

--onExistingArchive=sharearchive means share the archive.

--onExistingArchive=newarchive means create a new archive.

--binaryFormat
--textFormat

forces the storage of the member file to occur in binary format or text format.

--[no|**confirm]closeCP**

controls whether to close the associated change package.

--nocloseCP means do not close the change package.

--confirmcloseCP means ask before closing the change package.

--closeCP always closes the change package.

-d *desc*
--description=*desc*

specifies a description for the new archive. This option and the **--descriptionFile** option are mutually exclusive.

Note:

Descriptions that include spaces must be enclosed by quotes.

--descriptionFile=*file*

specifies a file name, *file*, containing the description text for the new archive. This option and the **-d** or **--description** options are mutually exclusive.

--exclude=*file:pattern,dir:pattern...*

specifies a file that contains a glob pattern for excluding members.

--include=*file:pattern,dir:pattern...*
specifies a file that contains a glob pattern for including members.

-l
--lock
keeps locks on members.

--[no]retainSourceFile
controls whether to keep the source file after the project has been added.

--sourceFile=*value*
identifies the source file being added.

--[no]saveTimestamp
controls whether to set the revision timestamp to the modification date and time of the working file rather than the current date and time.

--[no]unexpand
controls whether to expand, unexpand, or ignore keywords in the member file. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
$Revision$
$Name$
$ProjectName$
$ProjectSetting attribute$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
$State$
```

nonmember...
identifies a specific project member; use spaces to specify more than one project member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si projectci, **si projectco**,

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si projectci

checks in members of a project through a project

SYNOPSIS

si projectci [(**-L** *value*|**--label=***value*)] [(**-P** *value*|**--project=***value*)] [**--author=***value*] [**--devpath=***value*] [**--[no]branch**] [**--[no|confirm]branchVariant**] [**--[no|confirm]branchUpdate**] [(**--cpid=***ID*|**--changePackageId=***ID*)] [**--[no|confirm]differentNames**] [**--issueId=***value*] [**--[no|confirm]checkinUnchanged**] [**--[no|confirm]closeCP**] [(**-d** *value*|**--description=***value*)] [**--descriptionFile=***value*] [(**-l**|**--[no]lock**)] [**--[no|confirm]moveLabel**] [(**-r** *value*|**--revision=***value*)] [**--[no]retainSourceFile**] [**--sourceFile=***value*] [**--[no]saveTimestamp**] [(**-u**|**--unlock**)] [**--onExistingRevision=***{resync|resyncbycp|confirm|cancel}*] [**--[no]unexpand**] [**--[no]update**] [**--hostname=***value*] [**--port=***value*] [**--password=***value*] [**--user=***value*] [(**-?**|**--usage**)] [**-F***value*] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--forceConfirm=***[yes|no]*] [(**-F** *file*|**--selectionFile=***file*)] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si projectci checks in and saves changes to project members through a project.

Check in is an operation that adds a new revision of a file to an archive. When a file is checked in to a revision other than the head revision or a branch tip revision, a new branch is created.

Note:

This command can be only used with the MKS API. It is supported in the CLI only for the purpose of prototyping MKS API implementaions. It is used to check in source information from third party applications to MKS Integrity configuration management projects. Use the **si ci** command to check in through a Sandbox.

Assigning Revision Numbers

By default, when you check in a member, MKS Integrity automatically assigns a unique revision number to the new revision. It does this by incrementing the current revision number by one. For example, if the previous revision is 1.3, the new revision is assigned number 1.4.

You can choose the revision number of the changes you are checking in, so long as your revision number:

- is greater than the last revision number (you cannot use previously "skipped" revision numbers)
- has no leading zeros (zeros as complete revision numbers are acceptable)
- starts a new branch based on an existing revision

If you check in a revision using an already existing revision number, MKS Integrity attempts to add one to the revision number and check it in as that revision. If that revision already exists, MKS Integrity then chooses the next available branch number and creates a new branch.

For example, if you are checking in a new revision to an archive where the head revision is 1.7, the following numbers are valid:

- 1.8 (greater than head revision)--if you check in a revision as 1.7, which already exists, MKS Integrity assigns it 1.8
- 1.10 (greater than head revision)
- 1.72 (none of the numbers between 7 and 72 may be used afterwards)
- 2.0
- 1.7.1.1 (if it starts a new branch)
- 1.7.0.1 (leading zero as the branch number)

The following numbers are invalid:

- 1.3 even if there was no revision 1.3 previously
- 1.08 (leading 0 in last portion)
- 02.1 is considered the same as 2.1 (leading zero in branch number)

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]differentNames
controls whether to allow different source file and member names.

-L *value*
--label=*value*
identifies a label that is applied to the new revision. Note the following about using labels:

- Labels cannot contain colons(:), square brackets ([ ]), or leading spaces.
- Labels cannot have the same format as a valid revision number.
- Labels that include spaces must be enclosed by quotes.
- MKS recommends not using hyphens (-) in labels. Using hyphens may cause some ***si*** commands to fail.

--author=*value*
specifies an author name.

--[no]branch
controls whether to force the creation of a branch revision.

Note:
If required, a branch revision is created even if this option is set to "no", for example, if you are checking in a member to a revision other than the head revision.

A ***branch*** is a revision path that diverges from the main line of development (also known as the ***trunk***) in a member or project history. A branch is typically created by checking in a file to a revision other than the head revision. The most recent revision of a branch is called the ***tip revision***.

MKS Integrity usually places new revisions at the top of the trunk, assigning them two-part revision numbers, such as 1.15. There are times, however, when you do not want your work to be checked into the trunk. You may be pursuing a line of development that will not be included in the finished product, for instance, or you may be doing post-release maintenance while development for the next release continues on the trunk.

Divergent lines of development in the same archive are managed through the use of branches. A branch is an independent revision line that uses an existing revision as its starting point. Members of a branch revision are identified by their revision numbers. Whereas revisions on the trunk are characterized by two-part revision numbers (for example, 1.2 or 3.5), branch revision numbers are prefixed with the number of the revision they start from. For example, if a branch revision is started from revision number 1.2, the members of that branch are numbered

```
1.2.1.1
1.2.1.2
1.2.1.3
```

and so on. The first two digits of the number identify the revision where the branch diverges from the trunk, and the last two represent a position on the branch.

--[no|confirm]branchUpdate
controls whether to update the member revision, even if it is on a branch. This option stops the member revision from accidentally moving onto a branch if a branch was created during check out. This option only applies if **--update** is specified, **--revision** was not specified, and the member revision is on a different branch from the working revision.

--nobranchUpdate means do not update the member revision.

--confirmbranchUpdate means ask before updating the member revision.

--branchUpdate always updates the member revision.

--[no|confirm]checkinUnchanged
controls whether to force the checkin so that the new revision is checked in even if it is not different from the preceding one.

--nocheckinUnchanged means do not force the checkin.

--confirmcheckinUnchanged means ask before forcing the checkin.

--checkinUnchanged always forces the checkin.

--[no|confirm]closeCP
controls whether to close the associated change package.

--nocloseCP means do not close the change package.

--confirmcloseCP means ask before closing the change package.

--closeCP always closes the change package.

-d *desc*
--description=*desc*
specifies a description for the new revision. This option and the **--descriptionFile** option are mutually exclusive.

Note:
Descriptions that include spaces must be enclosed by quotes.

--descriptionFile=*file*
specifies a file name, *file*, containing the description text for the new revision. This option and the **-d** or **--description** options are mutually exclusive.

-l
--[no]lock
controls whether to keep locks on members.

--[no|confirm]moveLabel
controls whether the application should move a label from one revision to another.

--nomoveLabel disables the moving of a label.

--confirmmoveLabel displays a confirmation message.

--moveLabel moves the label.

--[no]retainSourceFile
controls whether to keep the source file after checkin.

--sourceFile=*value*
identifies the source file being checked in.

--[no]saveTimestamp
controls whether to set the revision timestamp to the modification date and time of the working file rather than the current date and time.

-u
--unlock
unlocks the newly checked-in revision. This is equivalent to specifying **--nolock**.

--[no]unexpand
controls whether to unexpand or ignore keywords in the member file prior to checking it into the archive. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
$Revision$
$Name$
$ProjectName$
$ProjectSetting attribute$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
$State$
```

--[no]update
controls whether to set the checked in revision as the project's member revision.

--onExistingRevision=[resync|resyncbycp|confirm|cancel]
controls what happens when the revision being checked in is not the member revision in the development path.

--onExistingRevision=resync means resynchronize the member revision into the working file and move the lock (if any) to the member revision. The check in operation is not completed. You should perform additional testing on the merged file before checking it in.

--onExistingRevision=resyncbycp means resynchronize the member revision into the working file by change package, and move the lock (if any) to the member revision. The check in operation is not completed. You should perform additional testing on the merged file before checking it in.

--onExistingRevision=confirm means ask for confirmation of the action to be taken.

--onExistingRevision=cancel means cancel the operation.

--[no|confirm]branchVariant
controls whether MKS Integrity creates a branch off the revision you are checking in, if you are working in variant Sandbox and this is the first time the member is checked in. This is useful for keeping changes in the variant separate from changes made in the mainline project.

member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si projectco, **si projectadd**,

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si projectco

checks out members of a project into working files

SYNOPSIS

si projectco [**--lockType=***[exclusive|nonexclusive|auto]*] [**--lineTerminator=***[lf|cr|crlf|native]*] [(**-P** *value*|**--project=***value*)] [**--devpath=***value*] [(**--cpid =***ID*|**--changePackageId=***ID*)] [**--issueId=***value*] [(**-l**|**--[no|auto]lock**)] [**--[no|confirm]lockOnFreeze**] [**--[no|confirm]downgradeOnLockConflict**] [**--targetFile=***value*] [(**-r** *value*|**--revision=***value*)] [(**-u**|**--unlock**)] [**--[no]update**] [**--[no|un]expand**] [**--[no|confirm]overwriteExisting**] [**--[no]restoreTimestamp**] [**--hostname=***value*] [**--port=***value*] [**--password=***value*] [**--user=***value*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [(**-F** *file*|**--selectionFile=***file*)] [**--quiet**] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si projectco checks out project members of a project into working files.

Check out is an operation that extracts the contents of a revision in an archive and copies it to a working file. You can check out any revision by specifying either its revision number or label. By default, the member revision is used. If an existing revision other than the head revision is specified, a branch from that revision is created.

Note:

This command can be only used with the MKS API. It is supported in the CLI only for the purpose of prototyping MKS API implementations. It is used to check out MKS Integrity configuration management project members for use in a third party application. To check out project members in a Sandbox, use the **si co** command.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--lineTerminator=*[lf|cr|crlf|native]*

specifies the line terminator used for the working file. Different operating systems use different characters to indicate the end of a line.

--lf for UNIX, Linux or other Posix systems.

--cr for Mac OS systems.

--crlf for Windows systems.

--native for the line terminator that is native to the client operating system (for example, **crlf** on Windows, **lf** on UNIX)

--[no]branch

controls whether to force the creation of a branch revision. Specifying **--branch** always creates a branch.

For details on branch revisions, see the **si ci** reference page.

--[no|confirm]branchVariant

controls whether MKS Integrity should create a branch off of the revision you are checking out if this is the first time you have attempted to check it out in a particular variant. Specifying **--confirmbranchVariant** causes a prompt to be displayed so you can confirm the branching.

--targetFile=*value*

identifies where to check the member out to

-l

--[no|auto]lock

controls whether to create locks on members.

--lock means always lock the member.

--nolock means never lock the member.

--autolock means lock the member based on the locks policy. For information on the locks policy, contact your administrator.

--lockType=[exclusive|nonexclusive|auto]

specifies the type of lock obtained on checkout.

--lockType=exclusive obtains an exclusive lock on the member. Exclusive locks enable only one member at a time to lock a specific revision.

--lockType=nonexclusive obtains a non-exclusive lock on the member. Non-exclusive locks enable multiple users to check out the same revision for editing.

--lockType=auto obtains a lock on the member based on the locks policy. For information on the locks policy, contact your administrator. If the locks policy does noe require a lock, but the **--lock** option is set, a non-exclusive lock is obtained.

--[no|confirm]lockOnFreeze
controls whether to lock the specified member even if it is frozen.

--lockOnFreeze means always do it.

--nolockOnFreeze means never do it.

--confirmlockOnFreeze means always ask before doing it.

--[no|confirm]downgradeOnLockConflict
controls whether to downgrade your lock to non-exclusive if another user has an exclusive lock on the revision.

--downgradeOnLockConflict means always do it.

--nodowngradeOnLockConflict means never do it.

--confirmdowngradeOnLockConflict means always ask before doing it.

-u
--unlock
keeps the member unlocked, and is equivalent to **--nolock**.

--[no]update
controls whether to update the project's member revision to the checked out revision.

--[no|un]expand
controls whether to expand, unexpand, or ignore keywords in the member file. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
$Revision$
$Name$
$ProjectName$
$ProjectSetting attribute$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
$State$
```

--[no|confirm]overwriteExisting
controls whether to overwrite an existing target file.

--overwriteExisting means always do it

--nooverwriteExisting means never do it

--confirmoverwriteExisting means always ask before doing it. When asked to confirm overwriting the working file, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the differences between the member's working file and the revision that is being checked out.

Note:
Specifying the **--yes** or **--no** options automatically answers **y** or **n** to the confirmation question, and also automatically answers **y** or **n** to all questions asked.

--[no]restoreTimestamp
: controls whether to restore the timestamp of the working file to the timestamp of the revision in the member history. If this is not specified, the timestamp is set to the current date and time.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si projectci, **si projectadd**,

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si projectinfo

lists project information

SYNOPSIS

si projectinfo [**--[no]acl**] [**--[no]attributes**] [**--[no]devpaths**] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--[no]showCheckpointDescription**] [**--[no]associatedIssues**] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si projectinfo displays current project information, for example:

```
si projectinfo /rd/doc/MKS_Integrity/Man_Pages/project.pj
```

displays

```
Project Name: /rd/doc/MKS_Integrity/Man_Pages/project.pj
Server: server.mks.com:7001
Revision: 1.1
Last Checkpointed: Jul 18, 2009 - 4:14 PM
Members: 83
Subprojects: 0
Description: Documentation Set
Checkpoint Description:
    Service Pack 1
Attributes:
    rd=/rd
Development Paths:
    Variant Project for Docs Man Pages (1.1)
Associated Issues:
    78484: Feature Request - Add calculator function
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]acl
controls whether to display ACL information for the project.

--[no]attributes
controls whether to display project attributes.

--[no]devpaths
controls whether to display project development paths.

--[no]showCheckpointDescription
controls whether to display the checkpoint description.

--[no]associatedIssues
controls whether to display information for any MKS Integrity items that are associated with the project. Only item types that you have permission to view are displayed.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si viewproject, **si viewprojecthistory**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si projects

lists projects on the Integrity Server

SYNOPSIS

si projects [**--[no]displaySubs**] [**--height=***value*] [**--manage**] [**--width=***value*] [**-x** *value*] [**-y** *value*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--[no]persist**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si projects displays a list of projects registered to the currently connected MKS Integrity Server. A registered project is currently visible on the MKS Integrity Server. A project dropped with the **si dropproject** command still exists on the MKS Integrity Server, but does not display in the list of registered projects.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]displaySubs
: controls whether to display subprojects. By default, only top level registered projects are displayed.

--manage
: enables the project manager view for those using the **-g** or **--gui** options.

--[no]persist
: controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si createproject, **si dropproject**, **si importproject**, **si projectinfo**, **si viewproject**, **si viewprojecthistory**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si promote

promotes the state of a member

SYNOPSIS

si promote [(**-r** *rev*|**--revision=***rev*)] [(**-s** *state*|**--state=***state*)] [(**-R**|**--[no]confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si promote promotes a project member to a higher state of development. A state is a textual description, defined by the administrator, used to classify the condition of a revision in a history. If no state is assigned to a revision at check-in, a default value of `Exp` (for Experimental) is used. See the *Integrity Server Administration Guide* for details on setting up states.

At particular milestones, project members are ready to move to the next state of their development cycle (for example, from Development to Testing). This is done through states and their promotion.

Note: Promoting members is for historical purposes only. To more effectively control project workflow, MKS recommends using the MKS Integrity integration.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

-s *state*
--state=*state*
: specifies the target state to be set for the promoted member. If no state is specified, the member is promoted to the next state.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si demote, **si demoteproject**, **si promoteproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si promoteproject

promotes the state of a project

SYNOPSIS

si promoteproject [**--[no|confirm]force**] [(**-s** *state*|**--state=***state*)] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si promoteproject promotes a project to a higher state of development. A state is a textual description, defined by the administrator, used to classify the condition of a revision in a history. If no state is assigned to a revision at check-in, a default value of `Exp` (for Experimental) is used. See the *Integrity Server Administration Guide* for details on setting up states.

Just as you can with a project member, you can promote the project itself (change its state, in other words). This can assist in management of an entire project through a development cycle, especially if the promotion functions are restricted to key project personnel. Normally, you promote a project (set its state) when you **si checkpoint** it, but you can do so at any time. Only the project is promoted, not the members of the project.

Note: Promoting projects is for historical purposes only. To more effectively control project workflow, MKS recommends using the MKS Integrity integration.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]force
: controls whether to force promotion of a project even if members have been changed.

-s *state*
--state=*state*
: specifies the target state to be set for the promoted project. If no state is specified, the project is promoted to the next state.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si demote, **si demoteproject**, **si promote**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si rejectcp

rejects a change package

SYNOPSIS

si rejectcp [**--comments=***value*] [**--commentsFile=***value*] [**--reviewer=***[user|group:<Group-Name>|super]*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F***file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *issue|issue:change package id...*

DESCRIPTION

si rejectcp rejects a change package under review. For example,

```
si rejectcp --reviewer=super 32434:3
```

If a reviewer of a change package determines that additional development must be completed to resolve an issue, the reviewer can reject the change package. A single reviewer rejection is enough to reject a change package, even if there are multiple reviewers. The creator of the change package can reject it without being listed as a reviewer.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--comments=*value*
: specifies comments recorded in the review log with the vote.

--commentsFile=*value*
: specifies a text file that contains the comments to be recorded in the review log with the vote.

--reviewer=*[user|group:<Group-Name>|super]*
: specifies the capacity in which you are rejecting the change package.

  user
  : casts a vote as an individual reviewer in the reviewer list.

  group:<Group-Name>
  : casts a vote on behalf of the entire group (only one user from a group is necessary to vote on behalf of the entire group).

  super
  : an overriding reject vote. A super reviewer is not required to be a listed reviewer for the change package. You must have the SuperReviewer permission to use this option.

issue...
issue:change package id...
: *issue* identifies a specific issue that contains all submitted change packages that you want to reject; use spaces to specify more than one issue.

  issue:change package id identifies a specific change package to reject; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si co, **si cpissues**, **si createcp**, **si viewcps** **si acceptcp**, **si opencp**, **si discardcp**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si rename

renames an existing project member

SYNOPSIS

si rename [(**--cpid=***ID*|**--changePackageId=***ID*)] [**--[no|confirm]closeCP**] [**--[no]confirm**] [**--[no]defer**] [**-f**] [**--issueId=***ID*] [(**--newName=***value*)] [**--[no|confirm]overwrite**] [**--[no]renameWorkingFile**] [**--[no|confirm]branchVariant**] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si rename renames an existing project member. For example,

```
si rename --newName=si_addmember.xml c:/Documentation/Man_Pages/xml_man/si_add.xml
```

This command only changes the basename of the member and cannot be used to move the member into a different directory or project. To move members, use the **si move** command.

Before using **si rename**, note the following:

- You can only rename members within a directory. You cannot rename a member and move it to a different directory or project.
- Renaming a locked revision in a variant project moves the lock to the duplicate revision, even if the lock exists in the master project.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]closeCP
: controls whether to close the associated change package.

  --nocloseCP means do not close the change package.

  --confirmcloseCP means ask before closing the change package.

  --closeCP always closes the change package.

--[no]confirm
: controls whether to confirm the rename before proceeding.

--[no]defer
: controls whether to delay the rename operation in the project until the deferred operation is submitted. The rename operation in the Sandbox still takes place immediately. This option is valid only if you are performing the command from a Sandbox.

-f
: force the rename without confirmation.

--newName=*value*
: specifies the new name of the renamed member.

--[no|confirm]overwrite
: controls whether to overwrite the target (new name) working file if it exists. Specifying **--overwrite** means always do it, **--nooverwrite** means never do it, and **--confirmoverwrite** means always ask before doing it. This option is valid only if you are performing the command from a Sandbox.

--[no]renameWorkingFile
: controls whether to rename the working file in the Sandbox immediately.

--[no|confirm]branchVariant
: controls whether MKS Integrity creates a branch for the renamed member.

member...
: identifies a specific member. You can only specify one member at a time.

DIAGNOSTICS

See the diagnostics reference page for possible exit status values.

PREFERENCES

Using si setprefs or si viewprefs, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, si closecp, si co, si cpissues, si createcp, si diff, si drop si edit, si lock, si move si mergebranch, si rlog, si unlock, si viewcps

Miscellaneous:

ACL, diagnostics, options, preferences

si report

generates a project report

SYNOPSIS

si report [**--basename=***value*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision [--[no]failOnAmbiguousProject] =***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**-F**|**--selectionFile=***value*] [**--status=***[none|gui|default]*]

DESCRIPTION

si report generates a report for a project. Use of this command requires your MKS Integrity client to be running Microsoft Access (TM) on the Windows operating system.

Reporter calculates a summary of the changes to a project and all its archives, then displays or prints the summary as text or, for some reports, as a graph. Customized reports, created using Microsoft Access, can also be produced.

Reporter's data files (containing the results of its project or archive analysis) can be saved as text files that can be read by most database applications. Reporter generates:

- changes introduced by individual authors
- changes between the member revision and a revision with a particular label
- changes between the member revision and any other revision
- a list of locked members and the names of users who locked them
- a list of revisions with a particular label or state
- project member history (including revision descriptions) by file

For more details on report types, see the *MKS Integrity User Guide*.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--basename=*value*
: Specifies the name of generated tables. If you do not specify **--basename**=*value*, **si report** launches MKS Integrity Reporter (if installed).

  si report uses the following suffixes to indicate the contents of the table:

.txp	project information
.txm	member information
.txa	archive information
.txr	revision information
.txs	symbols
.txx	access information

Data Fields

In the generated data files, the first line of each data file contains field names.

The fields for the project information file (.txp) are:

ProjId
: (integer) A unique identifier of a project (Primary Key)

ProjName
: (string) The project path/name

The fields for the member information file (.txm) are:

ProjId
(integer) A unique identifier for the project, can be used as a key. Taken from project info (Foreign Key - 1:M).

MemberName
(string) Member file path/name.

MemberType
(string) Valid values are:

- `a` archive
- `f` other (only valid for projects imported from older versions of MKS Integrity (formly MKS Source or Source Integrity Standard)
- `i` sub-project

Revision
(string) Revision number of member (can be NULL).

ArchId
(integer) A unique identifier of the member file, may be an archive id or another project id (can be NULL). Can be used as a primary key.

The fields for the project information file (`.txa`) are:

ArchId
(integer) A unique identifier of the member file, may be an archive id or another project id (can be NULL). Can be used as a primary key.

ArchiveName
(string) File path/name, only for archives.

Head
(string) Revision number of the head revision.

Branch
(string) Branch number of a default branch if one has been set by **`si archiveinfo`** –b. (can be NULL).

NumLocks
(integer) The number of locked revisions.

Format
(string) One of text, binary or auto-detect.

Encrypted
(integer) Set to 1 if the archive is encrypted. This option is not supported in MKS Integrity.

Compressed
(integer) Set to 1 if the archive is compressed.

TotalRevs
(integer) Total number of revisions in the archive.

Branches
(integer) Total number of branches in the archive.

BranchRevs
(integer) Total number of revisions on branches.

Comment
(string) Comment delimiter if set (can be NULL). This field is historical.

Description
(string) Text describing an archive. This text is entered when the archive is first created.

The fields for the revision information file (`.txr`) are:

ArchId
(integer) A unique identifier of the member file, may be an archive id or another project id (can be NULL). Can be used as a primary key.

RevNum
(string) The revision number of this revision (unique per ArchId).

Date
(string) The date this revision was entered, in the format *yyyy/mm/dd hh:mm:ss*.

Author
(string) The author's name (usually the author's user ID).

State
(string) The state assigned to this revision; this is taken from state list items).

LinesAdded
(integer) The number of lines added by this revision; derived from delta information.

LinesDeleted
(integer) The number of lines deleted by this revision.

Locker
(string) The name of the user holding a lock on this revision (usually the user's user ID) and the time when the revision was locked.

LogMsg
(string) Text description of the changes made by this revision as entered at check in.

The fields for the symbol information file (`.txs`) are:

ArchId
(integer) A unique identifier of the member file, may be an archive id or another project id (can be NULL). Can be used as a primary key.

RevNum
(string) The revision number of a revision that has an associated symbolic label. The revision number may not be unique in a list.

Symbol
(string) The symbolic label.

The fields for the access information file (`.txx`) are:

ArchId
(integer) A unique identifier of the member file, may be an archive id or another project id (can be NULL). Can be used as a primary key.

UserId
(string) Name (user ID) of user who is included in an access list (Source Integrity Standard only) for an archive. The archive and project ID numbers (*ArchId* and *ProjId*) are unique for any set of reports (though they may change from one use of report to another). They may be used as keys.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si projectinfo, **si projects**, **si viewproject**, **si viewprojecthistory**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si restoreproject

restores a project to a particular revision

SYNOPSIS

si restoreproject [(**-r** *rev*|**--revision=***rev*)] [(**--reason=***reason*|**--reasonFile=***file*)] [**--devpath=***path*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**-F**|**--selectionFile=***value*] [**--status=***[none|gui|default]*]

DESCRIPTION

si restoreproject restores a project to a particular checkpoint.For example,

```
si restoreproject --project=c:/Aurora_Program/bin/Libra/project.pj --revision=1.34 --
reason="Restore due to Exception Error"
```

When you apply the **si restoreproject** command, MKS Integrity recursively checkpoints the selected project, then modifies it to reflect the member list of the target checkpoint. Any further development then proceeds from the restored checkpoint.

Restoring a project is useful when development must revert back to an earlier version and there are no plans to proceed from the current version of the project. Restoring a project is not an option when the goal is to test a particular version. The project is always checkpointed before and after the restore.

Note the following:

- You cannot restore a build project using the **si restoreproject** command.
- You cannot restore a project that is associated with a deploy stage using the **si restoreproject** command. For more information on deploy stages, see the *MKS Deploy Administration Guide*.
- The **si restoreproject** command can potentially restore and checkpoint dropped subprojects that existed at the target revision, even if they are not currently a member of the project.
- Do not use the **si restoreproject** command to create a new development branch from a project checkpoint. For new development paths, you should instead create a new development path.
- To restore a variant project to a specific project revision, the development path must exist in all subprojects referenced by the project checkpoint.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--reason=*reason*
--reasonFile=*file*
specifies the reason text to be associated with restoring the project, or specifies a file where text can be retrieved.

Note:
If the reason text includes spaces, enclose the text in quotes.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si checkpoint

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si resync

updates a Sandbox with the member revision

SYNOPSIS

si resync [**--mergeType=***[confirm|cancel|automatic|manual]*] [**--onMergeConflict=***[confirm|cancel|mark|launchtool|highlight|error]*] [**--[no]byCP**] [**--[no]confirm**] [**--[no]confirmPopulateSparse**] [**--[no]includeDropped**] [**--[no|confirm]merge**] [**--[no|un]expand**] [**-f**] [**--[no|confirm]overwriteChanged**] [**--[no|confirm]overwriteIfPending**] [**--[no|confirm]overwriteDeferred**] [**--[no]overwriteUnchanged**] [**--[no]populate**] [**--[no]restoreTimestamp**] [(**-R**|**--[no|confirm]recurse**)] [**--[no]failOnAmbiguousProject**] [**--[no|confirm]downgradeOnLockConflict** [**--filter=***filteroptions*] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]awaitServer**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *current or dropped member/subproject...*

DESCRIPTION

si resync compares the contents of project members with their corresponding working files in the Sandbox and replaces the working file with an exact copy of the member revision, if differences are detected or if no working file exists. If the working file is writable, you are optionally prompted for confirmation before it is replaced. This command has no effect on working files that are identical to the member revision.

The following is an example of the command syntax:

```
si resync c:/Aurora_Program/bin/Libra/project.pj
```

Caution:

This command overwrites files, even ones that you have locked and that have changed since you last checked them out. Be certain of your response to any prompts indicating that the files will be overwritten - once replaced, you cannot get back your changes. If files have been dropped from a project, resynchronizing deletes them.

If the revision that your working file is based on is locked by you, your lock is automatically moved to the member revision before the resync proceeds. If another user already has an exclusive lock on the member revision, you are prompted to downgrade your lock to non-exclusive.

If you are working in a sparse Sandbox, resynching deletes your working file, unless the revision that it is based on is locked by you.

When working in your Sandbox, you can use the resynchronize by change package option.

The Resync By Change Package Option

The Resync by CP (**--[no]byCP**) option is primarily a tool for developers. When you want to resynchronize files in your Sandbox, you usually do so by choosing individual files and then using **si resync** command. However, if the files you are resynchronizing have changes that are linked to other files, the standard resync operation would not include those related files. To resynchronize all related files, you would have to manually search for all the changes associated with the change package on the member you are resynchronizing.

The Resync by CP option automates this process by searching the change package specified on the member revision you are resynchronizing and then bringing the changes from the project to your Sandbox.

While the Resync CP option searches all files related to a selected change package, and all the change packages that may be associated with the related files, the Resync By CP option only processes the change packages associated with the member you are resynchronizing.

How Does the Resync By CP Option Work?

The functioning of Resync by CP is affected by the options you choose for **si resynccp**. For more information, see **si resynccp**.

When Should I Use the Resync By CP Option?

Developers should use Resync By CP when they want to ensure that they have all dependent changes associated with a member revision, even if these changes are contained in other files. For example, a developer needs to check out (locked) a file and modify it. The developer finds that other revisions have been checked in since the member was resynchronized in the Sandbox. Because the Sandbox is quite large and contains many unrelated changes, the developer does not want to perform a standard resynchronization. The Resync By CP option can be used in this situation.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

Note:

If the working file of the member you are resyncing has been modified, MKS Integrity asks you to confirm that you want to merge your

modifications into the working file.

--mergeType=*[confirm|cancel|automatic|manual]*
specifies how you want to merge changes from the working file in your Sandbox into the member revision.

--mergeType=*confirm* prompts you to confirm a merge type.

--mergeType=*cancel* cancels the selected merge type.

--mergeType=*automatic* completes the merge process without launching the MKS Visual Merge tool.

Caution: While MKS Integrity and MKS Visual Merge are capable of performing automatic merging, MKS cannot guarantee that the merged results are “correct”. MKS recommends that you examine and test the merged results before checking them into the repository.

--mergeType=*manual* completes the merge operation through the MKS Visual Merge tool or a third party merge tool, depending on your Difference and Merge Tool preferences. The merge tool launches, displaying the revisions you want to merge. For more information on the MKS Visual Merge tool, refer to the *MKS Integrity User Guide*.

--onMergeConflict=*[confirm|cancel|mark|launchtool|highlight|error]*
specifies what to do when conflicts occur during a merge.

--onMergeConflict=*confirm* prompts you to confirm a merge conflict option.

--onMergeConflict=*cancel* cancels the merge.

--onMergeConflict=*mark* marks the working file indicating that merging is required without completing all of the merge related tasks. This provides the time you may need to investigate conflicts or edit difference blocks before finishing the merge.

--onMergeConflict=*launchtool* launches the MKS Visual Merge tool to resolve the conflicts; all non-conflicting blocks will already have been applied.

--onMergeConflict=*highlight* indicates conflicts in the file with the following characters: "<<<<<<" and ">>>>>>". You are responsible for resolving merge conflicts before checking in the changes.

Note: **--onMergeConflict=***launchtool* does not require the **-g** or **--gui** options.

--onMergeConflict=*error* indicates merge conflicts with an error message prompt.

--[no]byCP
controls whether to cause MKS Integrity to operate in change package mode. MKS Integrity automatically searches the change package associated with the member and resynchronizes all member files contained in that change package.

Note:
If there are no revisions in the change package to resynchronize, a normal resync is performed.

--[no]confirm
controls whether to proceed without confirmation messages when working with a change package (applies only when change packages are enabled).

--[no]confirmPopulateSparse
controls whether to proceed without confirmation messages when populating a sparse Sandbox.

--[no]includeDropped
controls whether to delete dropped members from your Sandbox.

--[no|confirm]merge
controls whether to merge changes from the working file in your Sandbox into the member revision.

--merge means always merge.

--nomerge means never merge.

--confirmmerge means always ask before merging. When asked to confirm the merge, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the differences between the member's working file and the revision it is being resynchronized with.

Note:

Specifying the **--yes** or **--no** options automatically answers **y** or **n** to the confirmation question, and also automatically answers **y** or **n** to all questions asked.

Note:

By default, MKS Integrity prompts you for the type of merge (manual or automatic) to perform and what action to take if a conflict is encountered.

--[no|un]expand

controls whether to expand, unexpand, or ignore keywords in the member file. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
$Revision$
$Name$
$ProjectLabel$
$ProjectName$
$ProjectSetting attribute$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
$State$
```

-f

overwrites the working file if it has changed since it was last checked in. This is equivalent to specifying **--overwriteChanged**.

--[no|confirm]overwriteChanged

controls whether to overwrite the working file if it has changed since it was last checked in.

--overwriteChanged means always overwrite.

--nooverwriteChanged means never overwrite.

--confirmoverwriteChanged means always ask before overwriting. When asked to confirm overwriting the working file, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the differences between the member's working file and the revision it is being resynchronized with.

Note:

Specifying the **--yes** or **--no** options automatically answers **y** or **n** to the confirmation question, and also automatically answers **y** or **n** to all questions asked.

--[no|confirm]overwriteIfPending

controls whether to overwrite the working file if there is a pending operation on the revision corresponding to the working file.

--overwriteIfPending means always overwrite

--nooverwriteIfPending means never overwrite

--confirmoverwriteIfPending means always ask before overwriting. When asked to confirm overwriting the working file, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the differences between the member's working file and the revision it is being resynchronized with.

Note:

Specifying the **--yes** or **--no** options automatically answers **y** or **n** to the confirmation question, and also automatically answers **y** or **n** to all questions asked.

--[no|confirm]overwriteDeferred

controls whether to overwrite the working file if there is a deferred operation on the revision corresponding to the working file.

--overwriteDeferred means always overwrite

--nooverwriteDeferred means never overwrite

--confirmoverwriteDeferred means always ask before overwriting. When asked to confirm overwriting the working file, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the differences between the member's working file and the revision it is being resynchronized with.

Note:

Specifying the **--yes** or **--no** options automatically answers **y** or **n** to the confirmation question, and also automatically answers **y** or **n** to all questions asked.

--[no]overwriteUnchanged

controls whether to overwrite files that have not been modified in the Sandbox. This may be useful for forcing files to pick up new keyword expansions or timestamps even if their content has not changed.

--[no]populate

controls whether to populate the Sandbox with a working file for the specified member. If you are working with a sparse Sandbox (see **si createsandbox**), specifying **--populate** with this command overrides the default sparse Sandbox behavior of removing working files.

--[no]restoreTimestamp

controls whether to restore the timestamp of the working file to the timestamp of the revision in the member history.

--[no|confirm]downgradeOnLockConflict

If the revision that your working file is based on is locked by you, your lock is automatically moved to the member revision before the resync proceeds. This option controls whether to downgrade your lock to non-exclusive if another user has an exclusive lock on the member revision.

--downgradeOnLockConflict means always downgrade

--nodowngradeOnLockConflict means never downgrade

--confirmdowngradeOnLockConflict means always ask before downgrading

--[no]awaitServer

instructs the client to repeatedly attempt the operation if the MKS Integrity Server does not respond.

current or dropped member/subproject...

identifies a specific member or subproject that currently exists in the project, or one that has been dropped from the project but still exists in the Sandbox. Use spaces to specify more than one. If you specify a dropped member or subproject, and if you specify **--includeDropped**, this command deletes the corresponding working file.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si applycp, **si ci**, **si co**, **si resynccp**, **si revert**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si resynccp

merges change package members to an MKS Integrity Sandbox

SYNOPSIS

si resynccp [**--allowOpenCP**] [**--[no]alreadyInProjectIsError**] [**--backFill=[cp|revision|error|skip|ask]**]
[**--mergeType=[confirm|cancel|automatic|manual]**]
[**--onMergeConflict=[confirm|cancel|mark|launchtool|highlight|error]**] [**--[no]confirm**]
[**--notify=***[never|onCompletion|onError]*] [**--[no]ignoreBranches**] [**--[no]ignoreServer**] [**--[no]continueOnErrors**]
[**--[no|confirm]merge**] [**--[no|confirm]mergeOnBranch**] [**--[no]failOnAmbiguousProject**]
[**--[no|confirm]createVariants**] [**--[no]otherProjectIsError**] [**--changePackageID=***value*] [**--cpid=***ID*]
[(**--issueID=***value*)] [**--[no]spanProjects**] [**--[no]useMaster**] [**--[no]verbose**] [(**-S** *sandbox*|**--sandbox=***sandbox*)]
[**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)]
[(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**]
[**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--[no]ignoreUpdateRevision**]
[**--subprojectPropagation=[explicit|implicit]**] *issue|issue:change package id...*

DESCRIPTION

si resynccp allows you to preview the changes listed in change packages in the context of a Sandbox before you propagate them to the project. The Resync CP command searches all files related to a selected change package, and all the change packages that may be associated with the related files. Resync CP is most useful for incorporating new software features or bug fixes during the software development process.

For more information on how the Resync CP command works and how you can use it in your development environment, see the *MKS Integrity User Guide*.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--allowOpenCP
: allows MKS Integrity to resynchronize open change packages. Open change packages are not allowed if a propagation change package is specified using the **--cpid** option.

--[no]alreadyInProjectIsError
: Specifies whether MKS Integrity terminates the operation if the change being applied has already been applied to the project. If this setting is negated **--noalreadyInProjectIsError**, then the information is displayed as a warning.

--backFill=[cp|revision|error|skip|ask]
: Specifies the way MKS Integrity treats historic revisions required by the specified change package. For example, **--backfill=ask** means MKS Integrity asks you to specify the change packages you want to exclude from the operation.

  The available values for the **--backFill** option include:

  --backFill=cp
  : recursively chooses all historic revisions required by the specified change packages and applies them by updating member revisions, adding files, or dropping files.

  --backFill=revision
  : processes only the specified change package(s) and chooses only directly associated revisions. It does not process any change packages that are associated with intermediate revisions.

  --backFill=error
  : terminates the operation if other change packages are required but are not specified.

  --backFill=skip
  : causes MKS Integrity to merge around specified backfill revisions. Because the Apply CP command does not perform merging, this is treated as an error in Apply CP, allowed in Resync CP.

  --backFill=ask
  : allows you to specify the change packages you want to include.

--[no]ignoreUpdateRevision
: ignores update revision change package entries when performing the command. The default is to include update revision entries.

--mergeType=*[confirm|cancel|automatic|manual]*

specifies how you want to merge changes from the specified change package revisions into the working file in your Sandbox.

--mergeType=*confirm* prompts you to confirm a merge type.

--mergeType=*cancel* cancels the merge.

--mergeType=*automatic* completes the merge process without launching the MKS Visual Merge tool.

Caution: Even if a no conflict is reported as a result of a merge operation, there can still be inconsistencies in the merge results that can only be detected by someone with an understanding of the context of the change. Therefore, anytime a merge is performed, MKS recommends that you examine and test the merged results before checking them into the repository.

--mergeType=*manual* completes the merge operation through the MKS Visual Merge tool or a third party merge tool, depending on your Difference and Merge Tool preferences. The merge tool launches, displaying the revisions you want to merge. For more information on the MKS Visual Merge tool, refer to the *MKS Integrity User Guide*.

--onMergeConflict=*[confirm|cancel|mark|launchtool|highlight|error]*
specifies what to do when conflicts occur during a merge.

--onMergeConflict=*confirm* prompts you to confirm a merge conflict option.

--onMergeConflict=*cancel* cancels the merge.

--onMergeConflict=*mark* marks the working file indicating that merging is required without completing all of the merge related tasks. This provides the time you may need to investigate conflicts or edit difference blocks before finishing the merge.

--onMergeConflict=*launchtool* launches the MKS Visual Merge tool to resolve the conflicts, all non-conflicting blocks will already have been applied.

--onMergeConflict=*highlight* indicates conflicts in the file with the following characters: "<<<<<<" and ">>>>>>". You are responsible for resolving conflicts in the working file.

Note: **--onMergeConflict=***launchtool* does not require the **-g** or **--gui** options.

--onMergeConflict=*error* indicates merge conflicts with an error message prompt.

--[no]confirm
Confirm the actions before starting them.

--[no|confirm]createVariants
controls whether to confirm the creation of variant projects as needed.

--notify=[never|onCompletion|onError]
specifies whether to display a report when the command is complete. The report details the operations that were performed and any errors that were encountered.

--notify=never
never displays the report.

--notify=onCompletion
always displays the report.

--notify=onError
displays the report if any errors were encountered.

--[no]ignoreBranches
Causes MKS Integrity to use the most recent revision when it encounters two revisions of the same member on two development paths in the change package being applied.

--[no]ignoreServer
Causes MKS Integrity to perform the Apply CP operation even if the change package members reside on different servers.

--[no]continueOnErrors
Causes MKS Integrity to continue to process the change package if errors occur while resychronizing. If the **--notify** option is specified as **onCompletion** or **onError**, any errors are reported when the command is complete.

--[no|confirm]merge
controls whether to confirm all merge operations before completing them.

--[no|confirm]mergeOnBranch
controls whether to cause MKS Integrity to perform a merge if the target revision is on a branch. MKS Integrity differences the two file revisions and merges any changes into the working file without modifying its revision number. You must then check in the working file to advance the revision to the next available revision number.

--[no]otherProjectIsError
Causes MKS Integrity to terminate the command if the member or subproject is in another project, for example, if the change package contains multiple entries in different top level projects. Change package entries referring to other projects are therefore treated as an error.

--cpid=*ID*
--changePackageID=*value*
specifies whether to create a propagation change package that is detached from the set of change packages it propagates (where *value* is the change package ID number). All the required operations are automatically represented as deferred entries (or pending entries for subproject operations). You can then discard entries that you do not want represented in the final application of changes to the project. You can also add or change entries in the propagation change package as required.

--issueID=*value*
the container ID used to find the propagation change package. If the MKS Integrity integration is enabled, the container ID is the same as the issue ID.

--[no]spanProjects
Causes MKS Integrity to apply the command to any member or subproject specified in the change package, even if this involves multiple projects. This option also applies across all Sandboxes and results in a search of local Sandboxes for all entries in the change package.

--[no]useMaster
causes MKS Integrity to operate on the top-level Sandbox.

--[no]verbose
Means MKS Integrity includes additional information to track the current state of the command.

--subprojectPropagation=*[explicit|implicit]*
specifies how to apply subproject changes required by the specified change packages.

--subprojectPropagation=*explicit* creates, adds, drops, or moves a subproject only if there is a explicit command to do so in the change package.

--subprojectPropagation=*implicit* creates, adds, drops or moves a subproject if the operation is implicitly required based on the change package entries. For example, if you are adding a member that is part of a subproject that does not exist in the targer project being updated, the subproject is added.

issue...
issue:change package id...
specifies the identification numbers for the change package you want to apply. For example, *11804:2*.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si applycp, **si resync**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si retargetsandbox

points a Sandbox at a different project configuration

SYNOPSIS

si retargetsandbox [**--[no]openView**] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--[no]failOnAmbiguousProject**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si retargetsandbox allows you to point your Sandbox at a different project configuration. For example, a developer can change a variant Sandbox to point to a different variant project, or a buildmaster can change a build Sandbox to point to a different build. You can also change the type of project that the Sandbox points to, for example, you can change a normal Sandbox to point to a variant project.

Note:

You cannot retarget a Sandbox containing deferred member operations or locks.
You cannot retarget a shared Sandbox.
You can only retarget top level Sandboxes.
You can only retarget to another type of the same project that the Sandbox is currently pointing to.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]openView
controls whether to open the Sandbox view after this Sandbox is retargeted. If **--noopenView** is used, the Sandbox view does not open.

--[no]failOnAmbiguousProject
Specifies if to abort command operation when multiple projects correspond to a flat project string.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si checkpoint, **si createdevpath**, **si createsandbox**, **si dropsandbox**, **si importsandbox**, **si sandboxes**, **si viewsandbox**

Miscellaneous:

diagnostics, **options**, **preferences**

si revert

overwrites a sandbox file with a fresh copy of the working file, discarding changes

SYNOPSIS

si revert [**--**[**no**|**un**]**expand**] [**-f**] [**--**[**no**|**confirm**]**overwriteChanged**] [**--**[**no**|**confirm**]**overwriteDeferred**] [**--**[**no**]**overwriteUnchanged**] [**--**[**no**]**restoreTimestamp**] [(**-R**|**--**[**no**|**confirm**]**recurse**)] [**--filter=***filteroptions*] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--**[**no**]**failOnAmbiguousProject**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--**[**no**]**batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *sandbox member...*

DESCRIPTION

si revert restores a working file to a state before changes were made, by overwriting the Sandbox file with a fresh copy of the working file at the same revision you currently have in your Sandbox, not the member revision. If you had the member locked, it is unlocked.

Note:

If the revision you have locked is not the working revision, the lock is retained.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[**no**|**un**]**expand**

controls whether to expand, unexpand, or ignore keywords in the member file. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
$Revision$
$Name$
$ProjectLabel$
$ProjectName$
$ProjectSetting attribute$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
$State$
```

-f

overwrites the working file if it has changed since it was last checked in. This is equivalent to specifying **--overwriteChanged**.

--[**no**|**confirm**]**overwriteChanged**

controls whether to overwrite the working file if it has changed since it was last checked in.

--overwriteChanged means always overwrite

--nooverwriteChanged means never overwrite

--confirmoverwriteChanged means always ask before overwriting. When asked to confirm overwriting the working file, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the differences between the member's working file and the revision it is being reverted to.

Note:

Specifying the **--yes** or **--no** options automatically answers **y** or **n** to the confirmation question, and also automatically answers **y** or **n** to all questions asked.

--[**no**|**confirm**]**overwriteDeferred**

controls whether to overwrite the working file if there is a deferred operation on the member.

--overwriteDeferred means always overwrite

--nooverwriteDeferred means never overwrite

--confirmoverwriteDeferred means always ask before overwriting. When asked to confirm overwriting the working file, you have the option of specifying **y**, **n** or **d**. Specifying **d** displays the

differences between the member's working file and the revision it is being reverted to.

Note:

Specifying the `--yes` or `--no` options automatically answers `y` or `n` to the confirmation question, and also automatically answers `y` or `n` to all questions asked.

`--[no]overwriteUnchanged`

controls whether to overwrite files that have not been modified in the sandbox. This may be useful for forcing files to pick up new keyword expansions or timestamps even if their content has not changed.

`--[no]restoreTimestamp`

controls whether to restore the timestamp of the working file to the timestamp of the revision in the member history.

sandbox member...

identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the `diagnostics` reference page for possible exit status values.

PREFERENCES

Using `si setprefs` or `si viewprefs`, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

`si ci`, `si co`, `si resync`,

Miscellaneous:

`diagnostics`, `options`, `preferences`

si revisioninfo

displays revision information

SYNOPSIS

si revisioninfo [**--[no]changePackage**] [**--[no]labels**] [(**-r** *rev*|**--revision=***rev*)] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--lockRecordFormat=***value*] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--[no]locate**] *member...*

DESCRIPTION

si revisioninfo displays revision information for a member, for example:

```
si revisioninfo c:\Documentation\Man_Pages\xml_man\si_add.1.xml
```

displays

```
Member Name: c:\Documentation\Man_Pages\xml_man\si_add.1.xml
Sandbox Name: c:\Documentation\Man_Pages\project.pj
Revision: 1.7
   Created By: pmolinas on Jul 11, 2009 - 11:36 AM
   State: state_a
   Revision Description:
      Updated for Technical Review
   Labels:
      TechReview1
   Change Package:
      ID: 1:1
   Member Revision:
      In project C:\Aurora_Program\Documentation\project.pj, this revision
      is member revision on:
         development path:Service Pack 1
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]changePackage
controls whether to display change package information (applies only when change packages are enabled).

--[no]labels
controls whether to display member labels.

--[no]locate
controls whether to display the project and development path that the member is a member revision in.

Note: If a revision is member revision in a shared subproject, only the original (canonical) project path displays. Any projects where the subproject was added as shared do not display.

Only projects that you have the `OpenProject` permission for display.

Note: This option is not valid for RCS style database repositories.

--lockRecordFormat=*value*
defines the format for displaying lock information for the revision. Specify a format string using keywords to represent the information you want to display. You can specify any of the following keywords:

{revision}
displays the revision that is locked.

{locker}
displays the user who locked the revision.

{locktype}
displays the type of lock on the revision (exclusive or non-exclusive).

{locktimestamp}
displays the time when the revision was locked.

{lockcpid}
displays the change package associated with the lock on the revision.

{project}
displays the name and path of the project where the member revision was locked from. If the member revision was locked from a shared subproject, it is the subproject name and path that are displayed.

{devpath}
displays the name of the development path where the lock on the revision was made from.

{sandbox}
displays the name of the Sandbox where the lock on the revision was made. This is relevant when viewing the information from the locker host.

{hostname}
displays the hostname of the computer that locked the the revision.

{hascpid}
displays 1 if the lock has a change package associated with it, 0 if there is no associated change package.

{hassandbox}
displays 1 if there is Sandbox information available for the lock, 0 if no Sandbox information is available.

{hasdevpath}
displays 1 if the lock was made from a development path, 0 if it wasn't.

{member}
displays the name of the locked revision.

member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si archiveinfo, **si memberinfo**, **si mods**, **si rlog**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si rlog

displays the revision log

SYNOPSIS

si rlog [**--format=***value*] [**--headerFormat=***value*] [**--noFormat**] [**--noHeaderFormat**] [**--noTrailerFormat**]
[(**-r** *rev*|**--revision=***rev*)] [**--range=***value*] [**--trailerFormat=***value*]
[**--rfilter=***filteroptions*][**--fields=***field1[:width1],field2[:width2]...*] [**--[no]failOnAmbiguousProject**] [**--height=***value*]
[**--width=***value*] [**--lockRecordFormat=***value*] [**--lockRecordDetailFormat=***value*] [**--lockseparator=***value*]
[**--maxTrunkRevs=***value*] [**-x***value* [**-y***value* [(**-R** |**--no[confirm]recurse**] [**--filter=***filteroptions*]
[(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*]
[**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [**--[no]persist**] [**--cwd=***directory*]
[(**-F** *file*|**--selectionFile=***file*)] [**--forceConfirm=***[yes|no]*] [(**-N**|**--no**)] [**--[no]batch**] [**--quiet**] [(**-?**|**--usage**)] [(**-Y**|**--yes**)]
member...

DESCRIPTION

si rlog displays the revision log and general archive information for every revision in a member history. The formatting is, by default, suitable for user viewing but it is also programmable, suitable for use in scripts. See the **--format=***value* option description below.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

For a member, first the **--headerFormat** is processed. Then each revision is processed and formatted using the **--fields** or **--format** option. Finally, the **--trailerFormat** is processed. All three sections have available the global archive information. For each revision, the fields designated as per-revision are shown.

--format=*value*
: defines an output format for user-formatted text. The default formatting is suitable for interpretation by most users; the various formatting options are provided for programmatic control.

  This option and the **--fields** option are mutually exclusive.

  --format options use the same values as **--fields**, but similar to a JAVA MessageFormat string (that is, it requires { } to surround each field). The **--fields** option automatically adds a newline on the end, but you must supply the newline for formats with a \n, for example:

  ```
  si rlog --format="{membername},{revision}\n"
  ```

--headerFormat=*value*
: defines the header to be used.

--noFormat
: disables formatting for user-formatted text. If **--noFormat** is specified, the per-revision iteration is not performed.

--noHeaderFormat
: disables the header formatting.

--noTrailerFormat
: disables the trailer formatting.

--range=*value*
: specifies a range of revision numbers. The format of *value* for a single revision is simply the revision number itself. Separate multiple non sequential revisions with a comma, and use a dash to indicate a sequential range, for example, `1.2-1.5`.

  You may also prefix or append a dash to a single revision number, meaning you want to show from 1.1 to the specified revision, or from the specified revision to the tip. For example, you would specify `-1.6` to show revisions 1.1 to 1.6. Or you could specify `1.2-` to show revisions 1.2 to the tip revision, whatever it may be.

  Another alternative is to use *?locker[:user]* to match revisions that are locked, or locked by a certain user. For example,

  ```
  si rlog --noHeaderFormat --noTrailerFormat --range=?locker --
  format="{membername},{locker},{revision}\n"
  ```

  displays one line per locked revision, containing the membername, locker, and the revision locked.

The revisions are printed in order from the highest revision to the lowest.

--trailerFormat=*value*
defines the trailer to be used.

--fields=*field1[:width1],field2[:width2]...*
allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a `space`.

This option and the **--format** option are mutually exclusive.

The fields available for printing can be one or more of the following:

added
per revision: the number of lines/bytes added for the current revision.

alllabels
global: all labels on any revision of the member, in the format *label:revision[,label:revision]...*

archivedescription
global: the archive description.

archivename
global: the name of the archive to which the member refers.

archiveshared
global: indicators for members that share another member's archive; 1 if shared, 0 if not shared.

attributes
global: member attributes in the form *attr=value[,attr=value]*.

author
per revision: author of the revision.

branchcount
global: number of branches.

branchid
global: default branch.

branchtiprev
global: tip of the branch on which the member revision is located.

compressed
global: 1 if archive is stored compressed, otherwise 0.

cpid
per revision: the associated change package identifier (applies only when change packages are enabled).

cpsummary
per revision: the associated change package summary (applies only when change packages are enabled).

date
per revision: date on the revision.

deleted
per revision: the number of lines/bytes deleted on the current revision.

description
per revision: the revision description.

format
exclusivelockmandatory
global: if exclusive lock mandatory; 1 if exclusive lock is mandatory, 0 if exclusive lock is not mandatory.

format

global: the storage format; 1 if stored as text, 0 if stored in binary format.

formattedlabels
global: all labels formatted into 80 column lines.

frozen
global: indicators for frozen members; 1 if frozen, or 0 if not frozen.

hasexclusivelocker
global: indicators for exclusively locked members, 1 if has exclusive locker, 0 if not.

haslabels
global: 1 if alllabels is non-null, 0 if null.

haslocker
global: 1 if has one or more lockers, 0 if null.

per revision: 1 if labels is non-null, 0 if null.

hasnewrevdelta
global: displays 1 if the member revision is not on a tip.

haspackage
per revision: displays 1 if *cpid* is set, indicating the presence of a change package.

headrev
global: displays the head revision number.

isreference
global: displays 1 if the member belongs to a shared or moved subproject, 0 if the member belongs to a subproject that is not shared or moved.

issandbox
global: displays 1 if it is a Sandbox, 0 if a project.

labels
per revision: a comma separated list of labels on this revision.

lockrecord
per revision: lock information for each lock. By default, the locker and lock type display in a comma separated list.You can customize the lock information that displays by using the **--lockRecordFormat** option. You can customize the separator between locks by using the **--lockseparator** option.

memberdevbranch
global: displays the member's development branch.

membername
global: displays the name of the member.

memberrev
global: displays the member revision number.

memberrule
global: displays the member rule on one line (in the header, if it exists).

projectname
global: from a project, displays the name of the current project or subproject that the member belongs to. From a Sandbox, displays the name of the Sandbox or sub Sandbox that the member belongs to.

projecttype
global: displays 1 if the project is a regular project, 2 if the project is a build project, 3 if the project is a variant project.

rawmemberrule
global: displays the member rule using the CLI syntax (in the header section, if it exists).

referenceproject

global: project references.

relativemembername
global: displays the member name relative to the top level project.

revision
per revision: revision number.

revisioncount
global: displays the count of the revisions.

revisionsonbranchcount
global: the number of revisions on branches.

state
per revision: displays the state.

storageformat
global: displays 1 if stored by reference, 0 if stored by delta.

trunktip
global: displays the trunk tip revision number.

type
global: always 0 to indicate a normal file member.

workingarchive
global: displays the name of the archive from which the working file is derived .

NOTE: In the case where the working file was derived from an archive that was based on a server-side system setup (i.e. based on some default RCSPath or WorkToArch setting, as opposed to an explicit archive= reference), the *workingarchive* field displays the value "(`default:<server directory>`)". The actual name of the archive from which the working file was derived can then be determined from the value for the *archivename* field.

workingfile
global: displays the working file name; *null* in a project.

workingfileexists
global: displays 1 if a Sandbox and working file exists, 0 if operating on a project.

workingfilerevunknown
global: displays 1 in a Sandbox if the working revision is unknown, or 0 if known. Always displays 0 in a project.

workingrevdelta
global: displays 1 in a Sandbox if the working revision is the same as the member revision, or displays 0 in a project.

workingfilewritable
global: displays 1 if a Sandbox and working file are writable.

workingrev
global: displays the working revision.

--rfilter=*filteroptions*
allows you to display any revisions of the current member that meet the selection criteria specified in the *filteroptions*. Comma separated expressions are combined using an OR operator. Multiple occurrences of an option are combined using an AND operator. A leading ! negates any simple filter that follows. For example,

--rfilter=labeled:Release 3.0,branchboundaries --rfilter=!label:obsolete
displays all revisions with a label of Release 3.0 or a revision at a branch boundary and whose label is not Obsolete.

The *filteroptions* can be one or more of the following:

range*:low-high*
selects revisions that are in the specified revision number range. If required, you can specify just one end of the range. The range includes the low and high values. To specify multiple ranges, the range keyword must be repeated, for example

--rfilter=range:1.100-1.200,range:1:400-1:500

branchrange*:low-high*
selects branched revisions that are in the specified revision number range. If required, you can specify just one end of the range. The range includes the low and high values. To specify multiple ranges, the branchrange keyword must be repeated, for example **--rfilter=branchrange:1.2.1.1-1.2.1.5,branchrange:1.4.1.1-1.4.1.5**

daterange*:<date>-<date>*|:**past**:*<nb>*:**years**|**months**|**days**|**hours**
selects revisions that were created during a specified time range or during a specified amount of time in the past. *<date>* specifies a date in any of the local date/time formats. For more information on date/time formats, see the **options** reference page. *<nb>* is used to specify the number of units to include in the range for a specified amount of time in the past.

Note: If the date/time format contains either "-" or "," you must quote each date. For example:
--rfilter=daterange:"June 21, 2009 - 6:07:28 PM"-"June 25, 2009 - 6:07:28 PM"
When in a shell, you must use two kinds of quotes so that the shell does not remove them. For example:
--rfilter=daterange:'"June 21, 2009 - 6:07:28 PM"'-'"June 25, 2009 - 6:07:28 PM"'
or
--rfilter=daterange:"'June 21, 2009 - 6:07:28 PM'"-"'June 25, 2009 - 6:07:28 PM'"

branch*[::current|:name]*
selects revisions that are on the specified branch.

labeled*:name*
selects revisions with labels. If a label name is specified, only revisions with that label are displayed. If no label name is specified, all labeled revisions are displayed.

labellike*:pattern*
selects revisions that match the specified pattern. Your administrator determines whether **glob** or **regex** patterns are used for matching.

locked*[::me|::anyone|userName]*
selects revisions that are locked. If *me* or a user name is specified, only revisions locked by that user are displayed. If no user or *anyone* is specified, all locked revisions are displayed.

locktype*[:exclusive|:nonexclusive|:any]*
selects revisions that are locked with the specified lock type. If no lock type or *any* is specified, all locked revisions are displayed.

state*:name*
selects revisions that are at the specified state.

author*[::me|:userName]*
selects revisions created by the specified author.

pending
selects revisions that are pending.

anyspecial
selects working revisions.

branchboundaries
selects revisions that are branched, or at the base or tip of a branch.

--lockRecordFormat=*value*
defines the format for displaying lock information in the **--lockrecord** field. Specify a format string using keywords to represent the information you want to display. You can specify any of the following keywords:

{revision}
displays the revision that is locked.

{locker}
displays the user who locked the revision.

{locktype}
displays the type of lock on the revision (exclusive or non-exclusive).

{locktimestamp}
displays the time when the revision was locked.

{lockcpid}
displays the change package associated with the lock on the revision.

{project}
displays the name and path of the project where the member revision was locked from. If the member revision was locked from a shared subproject, it is the subproject name and path that are displayed.

{devpath}
displays the name of the development path where the lock on the revision was made from.

{sandbox}
displays the name of the Sandbox where the lock on the revision was made. This is relevant when viewing the information from the locker host.

{hostname}
displays the hostname of the computer that locked the the revision.

{hascpid}
displays 1 if the lock has a change package associated with it, 0 if there is no associated change package.

{hassandbox}
displays 1 if there is Sandbox information available for the lock, 0 if no Sandbox information is available.

{hasdevpath}
displays 1 if the lock was made from a development path, 0 if it wasn't.

{member}
displays the name of the locked revision.

--lockRecordDetailFormat=*value*
defines the format for displaying lock information in the details panel of the **Member History** view. Specify a format string using keywords to represent the information you want to display. See the **--lockRecordFormat** option for a list of keywords.

--lockseparator=*value*
defines the separator used between each lock displayed in the **--lockrecord** field.

--maxTrunkRevs=*value*
specifies the maximum number of revisions along the mainline project to display. If you limit the number of revisions to be displayed, and the maximum is reached, a trailing message displays at the bottom of the list. The maximum number of revisions is based on the number of visible mainline revisions; branched revisions are not included in the count.

--[no]persist
controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si archiveinfo, **si diff**, **si memberinfo**, **si mods**, **si print**, **si projectinfo**, **si report**, **si revisioninfo**, **si sandboxinfo**, **si viewhistory**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si sandboxes

lists Sandboxes on the client

SYNOPSIS

si sandboxes [**--[no]displaySubs**] [**--height=***value*] [**--manage**] [**--width=***value*] [**-x** *value*] [**-y** *value*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--[no]persist**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**-F**|**--selectionFile=***value*] [**--status=***[none|gui|default]*]

DESCRIPTION

si sandboxes displays Sandboxes currently registered on the MKS Integrity client and the corresponding project, server name, and port number, for example:

```
c:\demoapp\project.pj -> /projects/demoapp/project.pj
(server.intra-wif:7001)
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]displaySubs
: controls whether to display sub Sandboxes.

--manage
: enables the Sandbox manager view for those using the **-g** or **--gui** options.

--[no]persist
: controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si createsandbox, **si dropsandbox**, **si importsandbox**, **si sandboxinfo**, **si viewsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si sandboxinfo

lists information about a Sandbox

SYNOPSIS

si sandboxinfo [**--[no]attributes**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--[no]showCheckpointDescription**] [**--[no]associatedIssues**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**-F**|**--selectionFile=***value*] [**--status=***[none|gui|default]*]

DESCRIPTION

si sandboxinfo displays current information about a Sandbox, for example:

```
Sandbox Name: c:\SourceCode\project.pj
Project Name: c:/master_projects/SourceCode/project.pj
Server: intra-wif:7001
Revision: 1.4
Last Checkpoint: Feb 15, 2009 - 3:28 PM
Members: 0
Subsandboxes: 3
Line Terminator: native
Project Description:
Checkpoint Description:
    Service Pack 1
Project Attributes: none
Sandbox Attributes: none
Associated Issues:
    78484: Feature Request - Add calculator function
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]attributes
controls whether to display project and Sandbox attributes.

--[no]showCheckpointDescription
controls whether to display the checkpoint description of the master project.

--[no]associatedIssues
controls whether to display information for any MKS Integrity items that are associated with the project. Only item types that you have permission to view are displayed.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si archiveinfo, **si ci**, **si memberinfo**, **si mods**, **si projectinfo**, **si rlog**, **si sandboxes**, **si viewsandbox**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si serveralerts

displays MKS Integrity Server alert messages for all currently connected servers

SYNOPSIS

si serveralerts [**--user=***name*] [**--hostname=***server*] [**--password=***password*] [**--port=***number*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [**-g** | **--gui**] [(**-F***file*|**--selectionFile=***file*)] [**--settingsUI=***[gui|default]*] [**--quiet**] [**--status=***[none|gui|default]*]

DESCRIPTION

si serveralerts displays MKS Integrity Server alert messages for all servers that you are currently connected to. The alert message displays who sent the message, the server it came from, when it was sent, and the message. If you are not connected to any servers, a message informs you that there are no alert messages. Alert messages are sent by your administrator and are useful for notifying users about important information, such as an impending server upgrade in which the server will be shut down.

Note the following:

- In the Web interface, the date displayed for an alert message is the server's date, time, and time zone. In the GUI and CLI, the date displayed for an alert message is the client's date, time, and time zone.
- To avoid manually checking alert messages from the command line, launch the alert messages dialog box from the command line by specifying `-g` or `--gui` and keep the dialog box open. The dialog box automatically refreshes to display new alert messages.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

SEE ALSO

Commands:

si viewserveralerts, **si admingui**, **si servers**,

Miscellaneous:

options

si servers

displays the current connections to an Integrity Server

SYNOPSIS

si servers [**--[no]showVersion**] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--[no]persist**] [**--quiet**] [**-F**|**--selectionFile=***value*] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si servers displays active server connections in the format *user@host_name:port*.

The default server connection is indicated by *user@host_name:port(default)*.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]showVersion
: controls whether to show build version information for the connected server. The presentation of this information is in the format *[Build: 2015]*.

--[no]persist
: controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si connect, **si disconnect**

Miscellaneous:

diagnostics, **options**, **preferences**

si setmemberrule

sets the member rule for one or more members

SYNOPSIS

si setmemberrule [**--clear**] [**--[no]confirm**] [**--[no]expand**] [**-f**] [**--[no|confirm]overrideRule**] [(**-r** *rev*|**--revision=***rev*)] [(**-R**|**--[no|confirm]recurse**)] [**--[no]failOnAmbiguousProject**] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

A member rule is a revision--typically a symbolic revision--attached to a member that you can use regularly to update the member's revision or use it with any command that allows you to specify the member rule as a pre-defined revision. In the graphical user interface, only the Check Out and Update Member Revision commands allow you to specify a member rule. In the command line interface, any command that accepts the **-r**|**--revision** option allows you to specify a member rule. For example, you can check out a revision corresponding to the member rule in the graphical user interface or add a label to a revision corresponding to the member rule in the command line interface. After you create a rule for a member, you can also view the rule using the **si memberinfo** command.

Important: By design, applying a rule as a member revision to a member does not dynamically update the member revision of the corresponding member in an external project configuration (normal, variant, build). For example, if member 1's member revision is updated in project A, the corresponding member in a variant of project A with the rule configured to link to project A is not updated with the same member revision. To update the corresponding member in the variant according to the member rule, you must use the **si updaterevision** command with the member rule, or your administrator can configure the `ClientLink.js` sample event trigger script that enables dynamic updating of linked member revisions under certain conditions. For more information, contact your administrator.

Example:

1. A project administrator defines a rule based on a label and implements it:

   `si setmemberrule -r`*ReadyForUse demoapp.c demoapp.h*

   `si updaterevision -r:rule` *demoapp.c demoapp.h*

   `si freeze` *demoapp.c demoapp.h*

2. Developers work as usual, using `:member`.

   Note: In this scenario, it is not expected that members with the rule are modified locally.

3. From time to time, the project administrator re-applies the rule, based on new revisions marked *ReadyForUse*:

   `si thaw` *demoapp.c demoapp.h*

   `si updaterevision -r:rule` *demoapp.c demoapp.h*

   `si freeze` *demoapp.c demoapp.h*

   Several rule filters are available so you can easily work with members that contain rules. For more information, see the **--filter** option in the **options** reference page.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--clear
clears the member rule and removes it from the specified members.

--[no]confirm
causes MKS Integrity to confirm that the rule will be cleared.

--[no]expand
controls whether to expand the revision before setting the rule.

-f

forces removal of the rule without confirmation.

--[no|confirm]overrideRule
controls whether to override the existing member rule.

member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si ci, **si co**, **si edit**, **si resync**, **si revert**, **si updatearchive**, **si updaterevision**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si setprefs

sets preferences

SYNOPSIS

si setprefs [**--command=***value*] [**--[no]resetToDefault**] [**--[no]save**] [**--[no]ask**] [**--ui=***[unspecified|gui|cli|api]*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] *pref[=value]...*

DESCRIPTION

si setprefs sets preferences and configuration options. These settings are used to determine default behaviors for other commands - each option on each command has a preference key associated with it. The **si viewprefs** command lists the commands and preference keys. Changes to your preferences are either for the current client session (until **si exit** is used) or may be permanently saved in your system's *home* directory, into a file named `IntegrityClient.rc`, using the **--save** option.

To permanently enable keyword expansion when checking in members, for example, you would specify:

```
si setprefs --command=ci --save keywordExpand=expand
```

Your administrator can also lock certain preferences from the Integrity Server, preventing you from configuring them using the **si setprefs** command. Preferences that are locked -- viewable with **si viewprefs** -- display (`locked`) at the end of the output line. The following preferences can be locked from the server by editing MKS Integrity policies in the Administration Client:

Tip: For information on editing MKS Integrity policies, see the *Integrity Server Installation and Configuration Guide*.

Preference	Affected Commands
branchIfVariant	**si ci**
breakLock	**si unlock**
changePackageID	**si co**, **si lock**
moveLock	**si co**
onExistingArchive	**si add**
restoreTimestamp	**si resync**, **si revert**
retainWorkingFile	**si add**, **si ci**
saveTimestamp	**si add**, **si ci**
sparse	**si importsandbox**
updateMemberRev	**si ci**

Warning: Do not edit the `IntegrityClient.rc` file manually, because preferences that appear more than once in the `IntegrityClient.rc` file can cause MKS Integrity to behave unpredictably.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--command=*value*
: identifies the command to be set. For an easy way to see a list of commands and values that may be set, simply type the **si viewprefs** command, either piped through |**more** or redirected to a file, for example:

```
si viewprefs --global --showValidValues >prefs.txt
```

The commands and preference keys are also listed on the **preferences** reference page.

--[no]resetToDefault
: controls whether to revert specified settings to the default values as shipped with MKS Integrity Client. If specifying **--resetToDefault**, you must not specify =*value* for each preference.

--[no]save
: controls whether changes should be permanently saved.

--[no]ask
: controls prompts to the user for specific preferences. Each preference option may be set to either **--ask** or **--noask**. When the

command itself is run, any option set to **--ask** and that is not explicitly set with command line options will be queried. If this **--ask** option is set, then you do not specify a value for the preference at the same time, but instead the *pref=value* must supply one of the following four valid *ask* values:

once
asks the user the first time only, and then uses the provided value every time after for the duration of that command.

never
never asks the user for a response, but uses the current setting (which may be specified by a preference).

element-last
asks the user for each element of the selection, providing the most recently used value as the default.

element-pref
asks the user for each element of the selection, resetting the default to the value specified by the preference.

For example, to set the server host for **si connect** to a specific host name, you specify something like:

```
si setprefs --command=connect
server.hostname=specific.hostname.com
```

but to set the preference to ask for a host name when using **si connect**, you specify something like:

```
si setprefs --command=connect --ask
server.hostname=element-last
```

--ui=[unspecified|gui|cli|api]
controls whether to apply the preference to the graphical user interface, the command line interface, or when the interface is unspecified. By default, **--ui=***cli* is implied when issuing the **si setprefs**. To set preferences for GUI behavior, however, you should specify **--ui=***gui*. For example, to set the *manage* preference to be true in the GUI for the **si sandboxes** command, you would type:

```
si setprefs --command=sandboxes --ui=gui manage=true
```

These correlate to settings in the `IntegrityClient.rc` file, which can be seen as having the *gui.si.* or *cli.si.* prefix, or simply the *si.* prefix when it is unspecified.

pref[=value]...
identifies the preference string. If you specified the **--resetToDefault** option, then you only need to specify the preference name; otherwise specify a value for the preference. Use spaces to specify multiple preferences.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si loadrc, **si viewprefs**

Miscellaneous:
diagnostics, **options**, **preferences**

si setprojectdescription

sets the project description

SYNOPSIS

si setprojectdescription [**--description=***desc*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**-F**|**--selectionFile=***value*] [**--status=***[none|gui|default]*]

DESCRIPTION

si setprojectdescription sets the one line description of a project. For example:

```
si setprojectdescription --project=c:/Aurora_Program/bin/Libra/project.pj --
description="Requirements for product component HFD3-3433"
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--description=*desc*

specifies the new description text to be set as the project description.

Note:

Descriptions that include spaces must be enclosed by quotes.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addprojectlabel, **si appendrevdesc**, **si checkpoint**, **si projectinfo**, **si viewproject**, **si viewprojecthistory**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si sharesubproject

shares an existing subproject between multiple projects

SYNOPSIS

si sharesubproject [**--sharedProject=***project location*] [**-r** *subproject revision*|**--subprojectRevision=***subproject revision*] [**--subprojectDevelopmentPath=***subproject development path name*|**--variant=***subproject development path name*] [**--type=[normal**|**variant**|**build**|**default]**] [**--[no**|**confirm]closeCP**] [**--cpid**|**--changePackageID=***ID*] [**--issueID=***value*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***development path name*] [**--hostname=***server*] [**--port=***number*] [**--password** =*password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *subproject location*

DESCRIPTION

A *shared subproject* is a subproject that is a member of more than one project. MKS Integrity allows you to share a subproject between two or more projects by referencing the original subproject. A shared subproject allows you to access common members across many projects. Shared subprojects are not required to be located within the same directory structure or project hierarchy.

A shared subproject functions the same as an unshared subproject and is accessible by the same commands.

Shared subprojects that were created in a previous version of MKS Integrity (formly MKS Source or Source Integrity Standard Edition) are detected as they are accessed by MKS Integrity without disrupting the operation. The format of these subprojects is retained until you change or update it to the new format by re-configuring it.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--sharedProject=*project location*
: specifies the path and name of the subproject you want to share. For details on how to specify a project, see the **-P** option on the the **options** reference page

--[no|**confirm]closeCP**
: closes the change package after command completion.

-r *subproject revision*
--subprojectRevision=*subproject revision*
: specifies the checkpoint number (for build subprojects). For example, `1.5`. This option is used with **--type=***build*.

Note: This option cannot be used if you specify a project using a keyword string for the **--sharedProject** option. This option is also mututally exclusive with the **--subprojectDevelopmentPath** or **--variant** options.

--subprojectDevelopmentPath=*subproject development path name*
--variant=*subproject development path name*
: identifies the development path of a variant subproject. This is a label that was associated with a branch of the subproject by **si createdevpath**. Paths that include spaces must be enclosed by quotes. The following characters may not be used in a development path: \n, \r, \t, :, ['], #. This option is used with **--type=***variant*

Note: This option cannot be used if you specify a project using a keyword string for the **--sharedProject** option. It is also mutually exclusive with the **--subprojectRevision** option.

--type=*[normal|variant|build|default]*

specifies the new configuration type for the subproject. If you specify the **--sharedProject** option using a flat string, by default the subproject type is the same as the parent project type. If you specify the **--sharedProject** option using a keyword-based string, by default the subproject type is determined by the specified configuration. Use this option to specify a configuration type other than the default.

--type=normal configures the subproject to the master (non-variant) version of the subproject.

--type=variant configures the subproject based upon a specific development path. The option is used with **--variant=***subproject development path name* or **--subprojectDevelopmentPath=***subproject development path name*.

--type=build configures the subproject as a static subproject based upon a specific checkpoint of the master project that is used for building or testing the project, but not for further development. This option is used with **-r** *subproject revision* or **--**

subprojectRevision=*subproject revision*.

--type=default configures the subproject to be the same as the parent project that you are adding the subproject to. For example, if you add a subproject to a normal project, the subproject is added as a normal type. For information on what the default type is, see your administrator.

subproject location

specifies where in the project you want the shared subproject to appear, for example, `c:/Aurora_Program/bin/Libra/`.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si addsubproject, **si checkpoint**, **si configuresubproject**, **si createproject**, **si createsandbox**, **si createsubproject**, **si movesubproject**, **si drop**, **si dropproject**, **si importproject**, **si projectinfo**, **si projects**, **si viewproject**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si snapshot

records the state of a sandbox.

SYNOPSIS

si snapshot [**--targetDevpath=***value*] [**--author=***value*] [**-d** | **--description=***value*] [**--descriptionFile=***snapshot*] [**-L** | **--label=***value*] [**--[no]notify**] [**--[no]failOnAmbiguousProject**] [**-s** | **--state=***value*] [**--[no]stateMembers**] [**-S** | **--sandbox=***value*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si snapshot records a capture of the current state of a Sandbox, where each element in the Sandbox can be identified with a pre-existing entity in the repository on the Integrity Server. The Sandbox snapshot creates a project checkpoint that you can create a build Sandbox or a development path from. The revision number of a checkpoint created by a snapshot includes the revision number of the last checkpoint. For example, if the head revision of the project is 1.1, then the project checkpoint created by the snapshot will be 1.1.1.1. The Sandbox snapshot displays as a branched project checkpoint in the project history.

Note: **si snapshot** records the state of a sandbox, while **si checkpoint** records the state of a project. Checkpointing is recommended for recording milestones during the development of a project; however, there may be situations where you may need to take a snapshot of a sandbox to record the state for later use.

The state of a sandbox is defined by its set of elements, which include the following:

- sandbox members identified with an archive and working revision from which the working file was created
- former members that were dropped, but are still present in your sandbox
- sub Sandboxes, identified by project name and type
- former sub Sandboxes that were dropped but are still present in your Sandbox

Note the following about how **si snapshot** handles the set of Sandbox elements:

- To be included in the snapshot, all working files must be checked in. There must be no working file changes in the Sandbox.
- If the working file revision differs from the member revision, it is the working file revision that is included in the snapshot.
- Members with no working files are not included in the snapshot.
- Former members that still have working files in the Sandbox directory appear as members in the snapshot.
- Former subprojects that are still in the Sandbox directory appear as restored subprojects in the snapshot.

For more information on Sandbox snapshots, see the *MKS Integrity User Guide*.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--targetDevpath=*value*
specifies the development path to create the snapshot from. With this option you can only specify an existing development path.

--author=*value*
specifies the author of the snapshot revision.

-d
--description *value*
specifies the description for the snapshot revision.

--descriptionFile=*value*
specifies a file containing the description for the snapshot revision.

-L
--label=*value*
specifies a label for the snapshot revision. Note the following about using labels:

- Labels cannot contain colons(:), square brackets ([ ]), or leading spaces.
- Labels cannot have the same format as a valid revision number.
- Labels that include spaces must be enclosed by quotes.

- MKS recommends not using hyphens (-) in labels. Using hyphens may cause some ***si*** commands to fail.

--[no]notify
provides a notification when the snapshot has been created.

-s
--state=*value*
specifies the state for the project checkpoint created by the snapshot.

--[no]stateMembers
specifies whether to apply the state to all members or only the project.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, **si memberinfo**, **si mods**, **si print**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si viewhistory**, **si viewlabels**, **si viewproject**, **si viewprojecthistory**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si submit

submits a deferred operation

SYNOPSIS

si submit [(**--cpid**|**--changePackageId=***ID*)] [**--issueId=***ID*] [**--**[**no**|**confirm**]**closeCP**] [**-f**] [**--**[**no**]**overrideCPID**] [(**-R**|**--**[**no**|**confirm**]**recurse**)] [**--**[**no**]**failOnAmbiguousProject**] [**--filter=***filteroptions*] [**--**[**no**|**confirm**]**recurse**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [**-F** *value*] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--**[**no**]**batch**] [**--cwd=***value*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**-F**|**--selectionFile=***value*] *sandbox member...*

DESCRIPTION

si submit submits a deferred operation in your Sandbox. Submitting the operation completes the command and makes it visible in the associated project. If reviews are mandatory, submitting a deferred operation creates a pending entry in the associated change package.

Note:
If strict locking is enabled, you cannot submit a member with a deferred check in operation if you do not have a lock on the member.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--cpid=*ID*
--changePackageId=*ID*
identifies a change package that is notified of this action, for example, *145:2*. By default, the change package that was specified during the deferred operation is used. You can override this change package using the **--overrideCPID** option.

--[**no**|**confirm**]**closeCP**
controls whether to close the associated change package.

--nocloseCP means do not close the change package.

--confirmcloseCP means ask before closing the change package.

--closeCP always closes the change package.

-f
force all elements to be submitted with the specified change package ID override.

--issueId=*ID*
identifies an MKS Integrity issue that is notified of this action. This option is another way of redirecting to a change package. If you have an issue assigned to you that contains ***one*** open change package, you can specify the issue ID instead of the change package ID. This option can only be specified if you are using MKS Integrity.

--[**no**]**overrideCPID**
force all elements to be submitted with the specified change package ID override.

Note:
If you use the **--overrideCPID** option when submitting a change to a staging system, the stages or deploy targets will become out of sync with each other. You can use the **sd syncdeploytarget** command to resync your deploy targets. For more information, see the *MKS Deploy CLI Reference Guide*.

sandbox member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si add, **si ci**, **si drop**, **si move**, **si rename**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si submitcp

submits a change package.

SYNOPSIS

si submitcp [**--[no|confirm]closeCP**] [**--[no]showSubmitSuccessful**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?|--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N|--no**)] [(**-Y|--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g|--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--[no|confirm]commit**] [**--[no|confirm]deferredIsError**] [**--[no|confirm]ignoreNoDeferred**] *issue|issue:change package id...*

DESCRIPTION

si submitcp submits the uncommitted operations in a change package. For example:

```
si submitcp 2:34
```

Although you can submit operations individually using **si submit**, you can ensure that no operations are missed by submitting a change package that contains deferred operations. Deferred operations are visible only from the MKS Integrity Client, and are not committed to the project or repository until they are submitted.

Note:

If strict locking is enabled, you cannot submit a change package that contains deferred check in or work in progress entries that do not have corresponding locks.

When reviews are mandatory, submitting a change package begins the review process. For more information, see the *MKS Integrity User Guide*

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]closeCP

specifies whether to close the change package after it has been submitted.

--[no]showSubmitSuccessful

specifies to display a confirmation dialog box indicating the successful submission of a change package. This option is enabled by default.

--[no|confirm]commit

specifies whether to commit deferred or pending changes to the server repository, or submit them for review. The default is to commit without confirmation.

--[no|confirm]deferredIsError

specifies whether to proceed with the submit operation if there are deferred change package entries.

--[no|confirm]ignoreNoDeferred

specifies whether to proceed with the submit operation if there are no deferred, lock, or work in progress change package entries.

issue...
issue:change package id...

issue identifies a specific issue that contains all open change packages that you want to close; use spaces to specify more than one issue.

issue:change package id identifies a specific change package to close; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si acceptcp, **si rejectcp**, **si closecp**, **si add**, **si ci**, **si co**, **si cpissues**, **si createcp**, **si drop**, **si viewcps**, **si viewcp**.

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si thaw

thaws a project member

SYNOPSIS

si thaw [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si thaw thaws project members previously frozen with the **si freeze**command. Once thawed, all operations normally associated with a member can be performed again.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

member...

identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si checkpoint, **si freeze**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si unlock

unlocks a member

SYNOPSIS

si unlock [**--[no|confirm]breakLock**] [**--action=[downgrade|remove]**] [**--locker=***name*] [(**-r** *rev*|**--revision=***rev*)] [(**-R**|**--[no|confirm]recurse**)] [**--[no]failOnAmbiguousProject**] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si unlock removes or downgrades the lock on a member. If you do not specify any members, **si unlock** applies to all members. By default, **si unlock** downgrades your lock on the member revision of each archived member.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--action=[downgrade|remove]
: controls whether you want to remove or downgrade the lock on the member.

  --action=downgrade means downgrade an exclusive lock to a non-exclusive lock.

  --action=remove means remove the lock on the member.

--locker=*name*
: specifies the name of another user whose lock you want to remove or downgrade. If this option is not specified, the command removes or downgrades your lock.

--[no|confirm]breakLock
: controls whether to remove or downgrade someone else's lock. If someone else has the member locked and you have the ***DowngradeOtherUserLock*** or ***RemoveOtherUserLock*** permission (see **ACL**), you will normally be prompted to confirm whether you want to remove or downgrade their locks.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si ci, **si co**, **si edit**, **si lock**, **si rlog**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si unlockarchive

unlocks a revision in an archive.

SYNOPSIS

si unlockarchive [**--archive=***value*] [**-r** | **--revision=***value*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**-F**|**--selectionFile=***value*] [**--status=***[none|gui|default]*]

DESCRIPTION

si unlockarchive unlocks a revision in an archive. You can use this command to unlock members that have been dropped from a project, and members locked in a project for which you longer have permission to view.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--archive=*value*
: specifies the server side archive name.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, **si memberinfo**, **si mods**, **si print**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si viewhistory**, **si viewlabels**, **si viewprojecthistory**, **si viewsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si updatearchive

manipulates various member archive attributes

SYNOPSIS

si updatearchive [**--archiveDescription=***description*] [**--archiveType=***[binary]*] [**--[no]compress**] [**--defaultBranch=***value*] [**--deployType=***value*] [**--storageFormat=***[delta|reference]*] [**--[no]exclusiveLockMandatory**] [(**-R**|**--[no|]confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si updatearchive manipulates various member archive attributes.

Note:
This command affects the base archive, and, therefore, all projects and development paths referring to the archive.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--archiveDescription=*description*
sets the archive description for a member. *description* is stored in the member's meta data as an archive description. If *description* has spaces then it must be enclosed by quotes.

--archiveType=*[binary]*
converts a text type archive to a binary type archive.

Note: You must have the **changeArchiveType** permission to use this option. You can only perform this conversion on a database type repository.

Warning: Once you have converted an archive to binary format, you cannot convert it back to text format.

--[no]compress
controls whether to compress the member's archive. This is recommended for use by the system administrator only. On WIN32/NT systems, use the NTFS built-in compression instead.

--defaultBranch=*value*
sets the default branch MKS Integrity uses to check in files. If unspecified, files are checked in to the trunk.

--deployType=*value*
sets the deploy type, if the project is in a staging system. For more information, see the *MKS Deploy Administration Guide*.

--storageFormat=*[delta|reference]*
identifies what type of format to use in the archive. This is recommended for use by the system ADMIN only. *delta* is the traditional MKS Integrity reverse delta format, while *reference* means to store member history by reference -- each revision is stored as a separate file. Particularly for binary files, storing by reference can enhance performance; though the functionality is available for text files as well. This option can be used for archives stored in either a file system or database repository.

--[no]exclusiveLockMandatory
controls whether an exclusive locks policy is in effect for the archive. With an exclusive locks policy, you must have an exclusive lock before checking in any changes.

member...
identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si ci, **si co**, **si createdevpath**, **si edit**, **si resync**, **si revert**, **si updaterevision**

Miscellaneous:

ACL, diagnostics, options, preferences

si updateclient

updates the Integrity Client with a Service Pack

SYNOPSIS

si updateclient [**--[no|confirm]download**] [**--[no|confirm]shutdown**] [**--[no|confirm]rollback**] [**--[no|confirm]rollbackshutdown**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*]

DESCRIPTION

si updateclient updates the Integrity Client with a Service Pack from the server if one is available. A Service Pack may be designated as required to address a known issue, or may provide enhancements. Client side Service Pack numbers are designated with a "C", for example, C04030003.

Options

This command takes the universal options available to some ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]download
automatically downloads a Service Pack if one is available.

--[no|confirm]shutdown
automatically shutdowns the client if a Service Pack is downloaded.

--[no|confirm]rollback
automatically initiates a service pack rollback, if required to connect to the MKS Integrity Server.

--[no|confirm]rollbackshutdown
automatically shutdowns the client if a service pack rollback is initiated.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si connect, **si disconnect**

Miscellaneous:
ACL, **diagnostics**, **options**, **preferences**

si updaterevision

updates a project member revision to a specific revision

SYNOPSIS

si updaterevision [**--cpid**|**--changepackageid=***ID*] [**--[no|confirm]closeCP**] [**--[no]defer**] [**--issueId=***value*] [(**-r** *rev*|**--revision=***rev*)] [(**-R**|**--[no|confirm]recurse**)] [**--[no]failOnAmbiguousProject**] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--[no]save**] *member...*

DESCRIPTION

si updaterevision updates a member revision to a specific pre-existing revision. This is useful when you want the member revisions in a project to reflect revisions based on a symbolic location in the development path (working file, head revision, trunk tip, member branch tip), property (state, label, timestamp), current project configuration, or external project configuration.

Note: You cannot update a frozen member's revision. You must first thaw the member (**si thaw**) and then update it.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]save
: Used to save any revision specified by `-r` (see **options**). Although existing saved revisions can be referenced using the `:rule` symbolic revision, this option has been deprecated in favor of the member rule. MKS recommends using the member rule instead of the **--[no]save** option. For more information about the member rule, see **si setmemberrule**).

--[no|confirm]closeCP
: closes the change package after command completion.

--[no]defer
: controls whether to delay the operation in the project until the deferred operation is submitted. The operation in the Sandbox still takes place immediately.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si ci, **si co**, **si edit**, **si resync**, **si revert**, **si updatearchive**, **si setmemberrule**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewcp

displays details for an MKS Integrity configuration management change package

SYNOPSIS

si viewcp [**--fields=***field1[:width1],field2[:width2]...*] [**--[no]showCommitted**] [**--[no]showUncommitted**] [**--[no]showPending**] [**--[no]showReviewLog**] [**--[no]showPropagationInfo**] [**--format=***value*] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*] [**--headerFormat=***value*] [**--noFormat=***value*] [**--noHeaderFormat=***value*] [**--noTrailerFormat=***value*] [**--trailerFormat=***value*] [**--width=***value*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [**--quiet**] [(**-g**|**--gui**)] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *issue|issue:change package id...*

DESCRIPTION

si viewcp allows you to view details on any MKS Integrity configuration management change package you select. For example:

```
si viewcp 2:23
```

You can select the change package using an issue ID or change package ID, and the change package does not have to be assigned to you.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--fields=*field1[:width1],field2[:width2]...*

allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional. Fields are separated with a space.

The fields available for printing can be one or more of the following:

bytesadded
displays the number of bytes added by the operation. For text files, this field displays 0.

bytesdeleted
displays the number of bytes deleted by the operation. For text files, this field displays 0.

closeddate
displays the date the change package was closed on.

cptype
displays the change package type.

configpath
displays the configuration path of the change package entry (repository location of the member/subproject).

creationdate
displays date the change package was created..

deployrequestid
displays the ID of the deploy request for the change package. For more information, see the *MKS Deploy Administration Guide*.

deployrequeststate
displays the state of the deploy request for the change package. For more information, see the *MKS Deploy Administration Guide*.

deploytarget
displays the deploy target for the change package. For more information, see the *MKS Deploy Administration Guide*.

description
displays the description of the selected change package.

haspropagated
displays 1 if the change package has propagated other change packages, 0 if not.

haspropagatedby
displays 1 if the change package has been propagated by other change packages, 0 if not.

hasstaginginfo
displays 1 if the change package has staging information, 0 if not. For more information, see the *MKS Deploy Administration Guide*.

id
displays the ID of the issue associated with the selected change package.

isclosed
indicates if the change package is closed.

isdeploy
indicates if the change package is a deploy change package. For more information, see the *MKS Deploy Administration Guide*.

isstaging
indicates if the change package is a staging change package. For more information, see the *MKS Deploy Administration Guide*.

istext
indicates if the change package entry has a text archive.

linesadded
displays the number of lines added by the operation. For binary files, this field displays 0.

linesdeleted
displays the number of lines deleted by the operation. For binary files, this field displays 0.

location
displays the archive location for members or the location of the parent project for subprojects.

member
displays the name of the member or subproject affected by the operation.

membertype
displays the type of project element affected by the operation (member or subproject).

project
displays the name and path of the project where the operation was performed. If the operation occurred in a shared subproject, the subproject name and path are displayed. If the operation involved two different projects (for example, moving a member from one project to another), information for both projects is displayed.

propagated
displays a list of the change packages that have been propagated by the selected change package.

propagatedby
displays a list of the change packages that propagated the selected change package.

revision
displays the member revision number.

sandbox
displays the name of the sandbox where the deferred operation or check out took place.

server
displays the server the change package resides on.

showpropagationinfo
displays the propagation information for the change package.

siserver
displays the MKS Integrity server the change package resides on.

stage
displays the stage of the change package in the staging system. For more information, see the *MKS Deploy Administration Guide*

stagingsystem
displays the staging system that the change package is being deployed in. For more information, see the

MKS Deploy Administration Guide.

state
displays the status of the change package.

summary
displays the change package summary.

timestamp
displays the timestamp of the selected change package.

type
displays how the member was added to the change package.

user
displays the ID of the user who added the member to the change package.

variant
displays the names of variant projects associated with the member.

--[no]showPropagationInfo
specifies whether to display a list of the change packages that have been propagated by the selected change package and a list of the change packages that propagated the selected change package.

--[no]showCommitted
specifies whether to display committed operations. Committed operations appear as change package entries that contain changes committed to the server repository.

--[no]showUncommitted
specifies whether to display uncommitted operations. Uncommitted operations are work in progress, lock or deferred entries that can be submitted to the server repository.

--[no]showPending
specifies whether to display pending entries that have not yet committed to the repository.

--[no]showReviewLog
specifies whether to display review information. MKS Integrity provides review logs as part of a complete review audit process. A log consists of individual records for each reviewer vote cast. Each time the change package is submitted for review, a new log begins.

Each review record contains the following information:

Review States The possible review states are:

- **Review Pending** the change package is still in a state of Submitted and there are outstanding votes to be cast by reviewers.
- **Accepted** the change package is in a state of Accepted, and all of the individual reviewers and a user from each reviewer group have accepted the change package.
- **Rejected** the change package is in a state of Rejected, and at least one reviewer has rejected the change package.

Reviewer Type The possible reviewer types are:

- **User Reviewer** is a user voting in an individual capacity.
- **Group Reviewer** is a user voting in a group capacity on behalf of that group.
- **Super Reviewer** is a user casting an overriding vote in a super reviewer capacity, and is not required to be a user on the reviewer list or in a group on the group reviewer list.

Votes The possible vote values are:

- **Pending** signifies that the reviewer or group reviewer has not yet cast a vote.
- **Accepted** signifies that the user has cast an accept vote for the change package.
- **Rejected** signifies that the user has cast a reject vote for the change package.

Comments Displays information that reviewers optionally provide to clarify their votes.

Changes Displays in tabular form the changes that were made to upon submission of the change package.

si viewcp also includes options that deal with formatting for the view.

--format=*value*

defines an output format for user-formatted text. The default formatting is suitable for interpretation by most users; the various formatting options are provided for programmatic control.

This option and the **--fields** option are mutually exclusive.

--format options use the same values as **--fields**, but similar to a JAVA MessageFormat string (that is, it requires { } to surround each field). The **--fields** option automatically adds a newline on the end, but you must supply the newline for formats with a \n, for example:

```
si viewcp --format="{membername},{revision}\n"
```

--headerFormat=*value*

header for user-formatted text (see --format).

--noFormat=*value*

format for user-formatted text (see --format).

--noHeaderFormat=*value*

header for user-formatted text (see --format).

--noTrailerFormat=*value*

trailer for user-formatted text (see --format).

--trailerFormat=*value*

trailer for user-formatted text (see --format).

issue...
issue:change package id...

issue identifies a specific issue that contains all change packages that you want to view; use spaces to specify more than one issue.

issue:change package id identifies a specific change package to view; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, **si ci**, **si closecp**, **si co**, **si cpissues**, **si createcp**, **si drop**, **si lock**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewcps

displays all open change packages created by you

SYNOPSIS

si viewcps [**--fields=***field1[:width1],field2[:width2]...*] [**--filter=***value*] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*] [**--query=***value*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [**--[no]persist**] [**--quiet**] [**--myReviews**] [(**-g**|**--gui**)] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--[no]persist**] *issue|issue:change package id...*

DESCRIPTION

si viewcps displays all open change packages created by you.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--myReviews
displays change packages you are to review (which may include change packages not created by you). Change packages only appear in the list after they have been submitted for review. After you vote on the change package, it no longer appears in this list.

--query=*value*
the MKS Integrity query used to find change packages. For information on creating and using queries, see the *MKS Integrity Workflows and Documents CLI Reference Guide*.

--[no]persist
controls the persistence of CLI views.

--fields=*field1[:width1],field2[:width2]...*
allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a `space`.

The fields available for printing can be one or more of the following:

closeddate
displays the date the change package was closed.

cptype
displays the change package type.

creationdate
displays the date the change package was created.

deployrequestid
displays the ID of the deploy request for the change package. For more information, see the *MKS Deploy Administration Guide*.

deployrequeststate
displays the state of the deploy request for the change package. For more information, see the *MKS Deploy Administration Guide*.

deploytarget
displays the deploy target for the change package. For more information, see the

MKS Deploy Administration Guide.

description
displays change package descriptions.

id
displays change packages IDs.

summary
displays change package summaries.

issue

displays the issue ID if the MKS Integrity integration is enabled.

propagated
displays the list of change packages propagated by the change package (the list is displayed by default for propagation change packages).

propagatedby
displays the list of change packages that propagated the change package.

stage
displays the stage of the change package in the staging system. For more information, see the *MKS Deploy Administration Guide*

state
displays the current state of the change package.

siserver
displays the MKS Integrity server the change package resides on.

stagingsystem
displays the staging system that the change package is being deployed in. For more information, see the *MKS Deploy Administration Guide*.

state
displays the status of the change package.

user
displays the username who created the change package.

--filter=*value*
displays the change packages according to one of the following filters:

issueID:<issue>
displays change packages based on specified issue ID.

state[:closed|:open|:submitted|:accepted|:rejected|:discarded|:commitfailed]
displays change packages according to their state, which can be one of the following:

:closed specifies that the change package is in a closed state (all changes committed to repository).

:open specifies that the change package has in an open state (work in progress, available to add entries too).

:submitted specifies that the change package has been submitted for review.

:accepted specifies that the change package has been accepted by all reviewers.

:rejected specifies that the change package has been rejected by at least one reviewer.

:discarded specifies that the change package has been discarded.

:commitfailed specifies that the change package has failed to commit to the repository.

cptype[:development|:propagation|:deploy|:staging|:resolution]
displays change packages based on the type of change package.

stagingsystem:<expression>
displays change packages are being deployed in a staging system. For more information, see the *MKS Deploy Administration Guide*.

deploytarget:<expression>
displays change packages that have a deploy target. For more information, see the *MKS Deploy Administration Guide*.

state:<expression>
displays change packages that have a deploy target. For more information, see the *MKS Deploy Administration Guide*.

deployrequeststate[|:cancelled|:cleanedup|:cleaningup|:cleanupf
ailed|:created|:deployed|:executed|:executing|:packageactionsfailed|:packagecont

entfailed|:packagingactions|:packagingcontent|:prepared|:preparing|:queuedonsource|:queuedontarget|:readytodeploy|:readytotransfer|:rollbackfailed|:rolledback|:rollingback|:stopped|:transferfailed|:transferring]
displays change packages that are in a specified state. For more information, see the *MKS Deploy Administration Guide*.

closeddate:<date>
displays closed change packages based on a specified closed date. Valid date formats are the following: **between** *MM/dd/yyyy* **and** *MM/dd/yyyy*, **in the last|in the next** *number* **days|months|years yesterday|today|tomorrow**.

creationdate:<date>
displays change packages based on a specified creation date. Valid date formats are the following: **between** *MM/dd/yyyy* **and** *MM/dd/yyyy*, **in the last|in the next** *number* **days|months|years yesterday|today|tomorrow**.

membertype[:member|:subproject];
displays change packages based on the type of project element they contain.

member:<expression>
specifies a text string to filter change packages by member name.

project:<expression>
specifies a text string to filter change packages by project.

variant:<expression>
specifies a text string to filter change packages by variant.

description:<expression>
specifies a text string to filter change packages by their description.

summary:<expression>
specifies a text string to filter change packages by their summary.

type[:add|:addfromarchive|:drop|:import|:exclusivelock|:nonexclusivelock|:movememberto|:movememberfrom|:renamefrom|:renameto|:update|:updatearchive|:updaterevision|:createsubproject|:addsubproject|:addsharedsubproject|:configuresubprojectfrom|:configuresubprojectto|:movesubprojectfrom|:movesubprojectto|:dropsubproject]
displays change packages based on their entry type.

typemodifier[:committed|:pending]
displays change packages based on their entry category.

hasissue
displays change packages that have an associated issue.

pendingreviewby:name
displays change packages that have not yet been accepted or rejected by a specified reviewer.

user:name
displays change packages created by a specified username.

acceptedby:name[;date]
displays change packages accepted by a specified username on a specified date. Valid date formats are the following: **between** *MM/dd/yyyy* **and** *MM/dd/yyyy*, **in the last|in the next** *number* **days|months|years yesterday|today|tomorrow**.

rejectedby:name[;date]
displays change packages rejected by a specified username on a specified date. Valid date formats are the following: **between** *MM/dd/yyyy* **and** *MM/dd/yyyy*, **in the last|in the next** *number* **days|months|years yesterday|today|tomorrow**.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si add, si ci, si closecp, si co, si cpissues, si createcp, si drop, si lock

Miscellaneous:

ACL, diagnostics, options, preferences

si viewhistory

displays a member's history

SYNOPSIS

si viewhistory [**--fields=***field1[:width1],field2[:width2]...*] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*] [**--filter=***filteroptions*] [**--rfilter=***filteroptions*][(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--lockRecordFormat=***value*] [**--lockRecordDetailFormat=***value*] [**--maxTrunkRevs=***value*] [**--checkpointLabelFilter=***value*] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--maxTrunkRevs=***value*] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--[no]persist**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si viewhistory displays a member's history, one line per revision, per member. This is similar to **si print** and **si rlog**, but with different defaults.

While Sandboxes and projects allow you to manage and access project members and the contents of individual members, the history of changes are saved in member histories.

MKS Integrity lets you save and recreate every stage (or revision) in the development of each member you use. When you make changes to a project member and check it back in, your changes are automatically added to the member history.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--fields=*field1[:width1],field2[:width2]...*

allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a `space`.

The fields available for printing can be one or more of the following:

author

displays author names.

checkpointlabels

displays labels of checkpoints in which the member revision participated. This field is not displayed by default. Can be used with `--checkpointLabelFilter` to filter the checkpoint labels displayed.

cpid

displays the associated change package identifier (applies only if change packages are enabled).

date

displays the revision date.

description

displays revision descriptions.

labels

displays labels.

lockrecord

a comma separated list of locks on the revision. The locker and lock type are displayed by default for each lock. You can customize the lock information that displays in the List Member History view in the GUI by using the **--lockRecordFormat** option. You can customize the lock information that displays in the Details panel of the Graphical Member History view in the GUI by using **--lockRecordDetailFormat** option.

state

displays the state.

--[no]persist

controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

--rfilter=*filteroptions*

allows you to display any revisions of the current member that meet the selection criteria specified in the *filteroptions*. Comma separated expressions are combined using an OR operator. Multiple occurrences of an option are combined using an AND operator. A leading ! negates any simple filter that follows. For example,

--rfilter=labeled:Release 3.0,branchboundaries --rfilter=!labeled:obsolete

displays all revisions with a label of "Release 3.0" or at a branch boundary and whose label is not "Obsolete".

The *filteroptions* can be one or more of the following:

range*:low-high*

selects revisions that are in the specified revision number range. If required, you can specify just one end of the range. The range includes the low and high values. To specify multiple ranges, the range keyword must be repeated, for example **--rfilter=range:1.100-1.200,range:1:400-1:500**

branchrange*:low-high*

selects branched revisions that are in the specified revision number range. If required, you can specify just one end of the range. The range includes the low and high values. To specify multiple ranges, the branchrange keyword must be repeated, for example **--rfilter=branchrange:1.2.1.1-1.2.1.5,branchrange:1.4.1.1-1.4.1.5**

daterange*:<date>-<date>*|**:past:***<nb>***:years**|**months**|**days**|**hours**

selects revisions that were created during a specified time range or during a specified amount of time in the past. *<date>* specifies a date in any of the local date/time formats. For more information on date/time formats, see the **options** reference page. *<nb>* is used to specify the number of units to include in the range for a specified amount of time in the past.

Note: If the date/time format contains either "-" or "," you must quote each date. For example:
--rfilter=daterange:"June 21, 2009 - 6:07:28 PM"-"June 25, 2009 - 6:07:28 PM"
When in a shell, you must use two kinds of quotes so that the shell does not remove them. For example:
--rfilter=daterange:'"June 21, 2009 - 6:07:28 PM"'-'"June 27, 2005 - 6:07:28 PM"'
or
--rfilter=daterange:"'June 21, 2009 - 6:07:28 PM'"-"'June 25, 2009 - 6:07:28 PM'"

branch[::current|*:name*]

selects revisions that are on the specified branch.

labeled*:name*

selects revisions with labels. If a label name is specified, only revisions with that label are displayed. If no label name is specified, all labeled revisions are displayed.

labellike*:pattern*

selects revisions that match the specified pattern. Your administrator determines whether **glob** or **regex** patterns are used for matching.

locked[**::me**|**::anyone**|*userName*]

selects revisions that are locked. If *me* or a user name is specified, only revisions locked by that user are displayed. If no user or *anyone* is specified, all locked revisions are displayed.

locktype*[:exclusive*|*:nonexclusive*|*:any]*

selects revisions that are locked with the specified lock type. If no lock type or *any* is specified, all locked revisions are displayed.

state*:name*

selects revisions that are at the specified state.

author[**::me**|*:userName]*

selects revisions created by the specified author.

pending

selects revisions that are pending.

anyspecial

selects working and member revisions.

branchboundaries

selects revisions that are branched, or at the base or tip of a branch.

--lockRecordFormat=*value*

defines the format for displaying lock information in the **--lockrecord** field of the List Member History view in the GUI. Specify a format string using keywords to represent the information you want to display. You can specify any of the following keywords:

{revision}
displays the revision that is locked.

{locker}
displays the user who locked the revision.

{locktype}
displays the type of lock on the revision (exclusive or non-exclusive).

{locktimestamp}
displays the time when the revision was locked.

{lockcpid}
displays the change package associated with the lock on the revision.

{project}
displays the name and path of the project where the member revision was locked from. If the member revision was locked from a shared subproject, it is the subproject name and path that are displayed.

{devpath}
displays the name of the development path where the lock on the revision was made from.

{sandbox}
displays the name of the Sandbox where the lock on the revision was made. This is relevant when viewing the information from the locker host.

{hostname}
displays the hostname of the computer that locked the the revision.

{hascpid}
displays 1 if the lock has a change package associated with it, 0 if there is no associated change package.

{hassandbox}
displays 1 if there is Sandbox information available for the lock, 0 if no Sandbox information is available.

{hasdevpath}
displays 1 if the lock was made from a development path, 0 if it wasn't.

{member}
displays the name of the locked revision.

--lockRecordDetailFormat=*value*

defines the format for displaying lock information in the **--lockrecord** field of the Graphical Member History view. Specify a format string using keywords to represent the information you want to display. See the **--lockRecordFormat** option for a list of keywords.

--maxTrunkRevs=*value*

specifies the maximum number of revisions to display. If you limit the number of revisions to be displayed, and the maximum is reached, a trailing message displays at the bottom of the list. The maximum number of revisions is based on the number of visible mainline revisions; branched revisions are not included in the count.

--checkpointLabelFilter=*value*

specifies a regular expression used to filter the checkpoint labels displayed for the member revisions. Used with `--fields=checkpointlabels`, for example:

```
si viewhistory --checkpointLabelFilter=IS_.* --
fields=revision,author,date,checkpointlabels Console.java
```

member...

identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using si setprefs or si viewprefs, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, si memberinfo, si mods, si print, si revisioninfo, si rlog, si sandboxinfo, si viewlabels, si viewproject, si viewprojecthistory, si viewsandbox

Miscellaneous:

ACL, diagnostics, options, preferences

si viewlabels

displays a member's labels

SYNOPSIS

si viewlabels [**--fields=***field1[:width1],field2[:width2]...*] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*] [(**-R**|**--[no|confirm]recurse**)] [**--[no]failOnAmbiguousProject**] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--[no]persist**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si viewlabels displays a member's labels: one label per line, showing the label and its revision number.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--fields=*field1[:width1],field2[:width2]...*
: allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a space.

  The fields available for printing can be one or more of the following:

  cpid
  : displays the associated change package identifier (applies only if change packages are enabled).

  label
  : displays labels.

  revision
  : displays the revision number.

--[no]persist
: controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, **si memberinfo**, **si mods**, **si print**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si viewhistory**, **si viewproject**, **si viewprojecthistory**, **si viewsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewlocks

displays locks

SYNOPSIS

si viewlocks [**--format=***value*] [**--headerFormat=***value*] [**--noFormat**] [**--noHeaderFormat**] [**--noTrailerFormat**]
[(**-r** *rev*|**--revision=***rev*)] [**--range=***value*] [**--trailerFormat=***value*] [**--fields=***field1[:width1],field2[:width2]...*]
[**--[no]failOnAmbiguousProject**] [**--maxTrunkRevs=***value*] [**--lockRecordFormat=***value*] [**--lockRecordDetailFormat=***value*]
[**--lockseparator=***value*] [**--rfilter=***filteroptions*] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*]
[(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)]
[**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*]
[(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*]
[**--forceConfirm=***[yes|no]*] [**--[no]persist**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si viewlocks displays the locks held in a project and some information about the members, for example:

```
c:\app\connect.txt 1.4 mkern
Jan 9, 2002 - 4:06 PM
c:\app\source.txt 1.3 mkern
Jan 9, 2002 - 4:06 PM
c:\app\config.c 1.3 mkern
Jan 8, 2002 - 2:26 PM
```

Note: By default, **si viewlocks** displays the member names of locked members, not the working file names.

Options

This command takes some of the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

See the **si rlog** reference page for descriptions of all options applicable to **si viewlocks**.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, **si memberinfo**, **si mods**, **si print**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si viewhistory**, **si viewlabels**, **si viewprojecthistory**, **si viewsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewmetricsinfo

displays information about the metrics being tracked

SYNOPSIS

si viewmetricsinfo [**--fields=***field1[:width1],field2[:width2]...*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si viewmetricsinfo displays information about the metrics created using **si createmetricinfo.**

Note: Metrics are only supported for database type repositories.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--fields=*field1[:width1],field2[:width2]...*

allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a `space`.

The fields available for printing can be one or more of the following:

description

displays the description of the metric.

metric

displays the value of the metric.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si createmetricinfo, **si calculateprojectmetrics**, **si viewprojectmetrics**, **si addprojectmetric**

Miscellaneous:

diagnostics, **options**, **preferences**

si viewnonmembers

displays non-members in a Sandbox

SYNOPSIS

si viewnonmembers [**--[no|confirm]includeFormers**] [**--exclude=***file:pattern,dir:pattern...*] [**--include=***file:pattern,dir:pattern...*] [**--fields=***field1[:width1],field2[:width2]...*] [**-R** | **--[no|confirm]recurse**] [**-S** | **--sandbox=***value*] [**--[no]failOnAmbiguousProject**] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F***file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *nonmember...*

DESCRIPTION

si viewnonmembers displays non-members in a Sandbox.

From MKS Integrity, you can view non-member files existent in your Sandbox directory. The Non-Members view is useful when used recursively to identify all of the files that need to be placed under source control. By default, former members are not displayed in the Non-Members view.

The Non-Members view does not display files that are:

- deferred imported members
- deferred add members from archive
- pending members (add, add from archive, import)
- working files from members that have been renamed but not resynchronized

Non-members can only be viewed in a Sandbox context, and not from any project view operation.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|confirm]includeFormers
: specifies if to include members that have been dropped from the project, but where there still exists a working file in the Sandbox directory..

--exclude=*file:pattern,dir:pattern...*
: specifies a file that contains a glob pattern for excluding members.

--include=*file:pattern,dir:pattern...*
: specifies a file that contains a glob pattern for including members.

--fields=*field1[:width1],field2[:width2]...*
: allows you to select fields to be displayed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options.

  The fields available for printing can be one or more of the following:

  absolutepath
  : displays the absolute file path of the file.

  closestproject
  : displays the project associated with the Sandbox that is closest to the directory containing the file.

  closestsandbox
  : displays the Sandbox that is closest to the directory containing the file.

  lastmodified
  : displays the date that the file was last modified.

  memberid
  : displays the default member name for the file as it would appear if it was added to the nearest project. In the case where the nearest project is subproject, the relative path is displayed with the member name.

  size
  : displays the size of the file in bytes.

nonmember...

identifies nonmembers to display; use spaces to specify more than one change package.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si viewsandbox, **si viewsproject**, **si add**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewprefs

displays preferences

SYNOPSIS

si viewprefs [**--[no]global**] [**--command=***value*] [**--[no]showValidValues**] [**--[no]ask**] [**--ui=***[unspecified|gui|cli|api]*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [(**-F** *file*|**--selectionFile=***file*)] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si viewprefs displays preferences and configuration options. These settings are used to determine default behaviors for other commands.

Your administrator can lock certain preferences from the Integrity Server, preventing you from configuring them using the **si setprefs** command. Preferences that are locked display (`locked`) at the end of the output line. The following preferences can be locked from the server by editing MKS Integrity policies in the Administration Client:

Tip: For information on editing the MKS Integrity policies, see the *Integrity Server Installation and Configuration Guide*.

Preference	Affected Commands
`branchIfVariant`	**si ci**
`breakLock`	**si unlock**
`changePackageID`	**si co**, **si lock**
`moveLock`	**si co**
`onExistingArchive`	**si add**
`restoreTimestamp`	**si resync**, **si revert**
`retainWorkingFile`	**si add**, **si ci**
`saveTimestamp`	**si add**, **si ci**
`sparse`	**si importsandbox**
`updateMemberRev`	**si ci**

Warning: Do not edit the `IntegrityClient.rc` file manually, because preferences that appear more than once in the `IntegrityClient.rc` file can cause MKS Integrity to behave unpredictably.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions. For an easy way to see a list of commands and values that may be set, simply type the **si viewprefs** command, either piped through |**more** or redirected to a file, for example:

```
si viewprefs --global --showValidValues >prefs.txt
```

Alternatively, the **--gui** option presents a simple-to-use dialog box that lets you view and configure the preferences.

--[no]global
controls whether to view all preferences.

--command=*value*
identifies the command preference to be viewed.

--[no]showValidValues
controls whether to list valid values for the preferences.

--[no]ask
controls whether to view the ask control over the preference options. See the description for **--[no]ask** on the **si setprefs** command.

--ui=*[unspecified|gui|cli|api]*
controls whether to view the preference as it applies to the graphical user interface, the command line interface, application programming interface, or when the interface is unspecified. By default, **--ui=***cli* is implied when issuing the **si viewprefs**. To view preferences for GUI behavior, however, you should specify **--ui=***gui*.

The preferences you see correlate to settings in the `IntegrityClient.rc` file, which can be seen as having the *gui.si.* or *cli.si.* prefix, or simply the *si.* prefix when it is unspecified.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si loadrc, **si setprefs**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewproject

displays the contents of a project

SYNOPSIS

si viewproject [**--fields=***field1[:width1],field2[:width2]...*] [**--[no]filterSubs**] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*] [(**-R**|**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--lockRecordFormat=***value*] [(**-F** *file*|**--selectionFile=***file*)] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd** *=directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--[no]persist**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member/subproject...*

DESCRIPTION

si viewproject displays the contents of a project and some information about the members, for example:

```
si viewproject c:/Aurora_Program/bin/Libra/project.pj
```

displays

```
connect.txt 1.4 archived
source.txt 1.3 archived
config.c 1.3 archived
```

Note: Specifying `si viewproject -S sandbox` does not view the Sandbox; it redirects through the Sandbox to view the project. Use **si viewsandbox** to view a Sandbox.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--fields=*field1[:width1],field2[:width2]...*

allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional. Widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a space.

The fields available for printing can be one or more of the following:

archiveshared

displays indicators for members that share another member's archive. This column is valid only if you are using the database repository.

attributes

displays member attributes.

context

When used with the MKS API, displays the name of the project, and indicates if the project is a build or variant.

Note: This field is not valid for use with the CLI.

cpid

displays the change package associated with the operation that set the member revision.

creationcpid

displays the change package that created the revision that is currently the member revision. This revision may be different from the Member CPID if an import, add member from archive, or set member revision operation was used.

frozen

displays indicators for frozen members.

indent

indents a field. For example:

```
si viewproject --fields=ident, name, type, memberrev
```

indents the `name` field, followed by `type` and `memberrev` fields, indenting as you recurse into subprojects.

Note: This field is not valid for use with the MKS API.

labels
displays labels.

lockrecord
a comma separated list of locks on the member. The locker and lock type are displayed by default for each lock. You can customize the lock information that displays by using the **--lockRecordFormat** option.

memberarchive
displays the name and path of the member archive.

memberdescription
displays the member description.

memberrev
displays the member revision.

membertimestamp
displays the member timestamp.

name
displays the member name.

newrevdelta
displays indicators for new revisions.

pendingcpid
displays the change package associated with a pending operation.

state
displays the member state.

type
displays the type of each item in the project: project, subproject, shared-subproject, shared-variant-subproject, shared-build-subproject, or member.

--[no]filterSubs
controls whether to display subprojects and directories that do not contain members matching the current filter.

--[no]persist
controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

--lockRecordFormat=*value*
defines the format for displaying lock information in the **--lockrecord** field. Specify a format string using keywords to represent the information you want to display. You can specify any of the following keywords:

{revision}
displays the revision that is locked.

{locker}
displays the user who locked the revision.

{locktype}
displays the type of lock on the revision (exclusive or non-exclusive).

{locktimestamp}
displays the time when the revision was locked.

{lockcpid}
displays the change package associated with the lock on the revision.

{project}
displays the name and path of the project where the member revision was locked from. If the member revision was locked from a shared subproject, it is the subproject name and path that are displayed.

{devpath}
displays the name of the development path where the lock on the revision was made from.

{sandbox}
displays the name of the Sandbox where the lock on the revision was made. This is relevant when viewing the information from the locker host.

{hostname}
displays the hostname of the computer that locked the the revision.

{hascpid}
displays 1 if the lock has a change package associated with it, 0 if there is no associated change package.

{hassandbox}
displays 1 if there is Sandbox information available for the lock, 0 if no Sandbox information is available.

{hasdevpath}
displays 1 if the lock was made from a development path, 0 if it wasn't.

{member}
displays the name of the locked revision.

member/subproject...
identifies a specific member or subproject; use spaces to specify more than one.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, **si memberinfo**, **si mods**, **si print**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si viewhistory**, **si viewlabels**, **si viewprojecthistory**, **si viewsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewprojecthistory

displays a history of the project

SYNOPSIS

si viewprojecthistory [**--fields=***field1[:width1],field2[:width2]...*] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*] [(**-R**|**--[no|confirm]recurse**)] [(**-P** *project*|**--project=***project*)] [**--[no]failOnAmbiguousProject**] [**--rfilter=***filteroptions*] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd** *=directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--maxTrunkRevs=***value*] [**--[no]persist**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] [**--xmlapi**]

DESCRIPTION

si viewprojecthistory displays a history of the project, including information on each historical checkpoint. For example:

```
si viewprojecthistory --project=c:/Aurora_Program/bin/Libra/project.pj
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--fields=*field1[:width1],field2[:width2]...*

allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a `space`.

The fields available for printing can be one or more of the following:

associatedIssues

displays the ID and summary of any MKS Integrity items associated with the project. Only item types that you have permission to view are displayed.

author

displays author names.

date

displays the project checkpoint date.

description

displays checkpoint and project descriptions.

labels

displays labels.

revision

displays the checkpoint number.

state

displays the checkpoint state.

--[no]persist

controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

--rfilter=*filteroptions*

allows you to display any checkpoints of the current project that meet the selection criteria specified in the *filteroptions*. Comma separated expressions are combined using an OR operator. Multiple occurrences of an option are combined using an AND operator. A leading ! negates any simple filter that follows. For example,

--rfilter=labeled:Release 3.0,branchboundaries --rfilter=!labeled:obsolete

displays all checkpoints with a label of "Release 3.0" or at a branch boundary and whose label is not "obsolete".

The *filteroptions* can be one or more of the following:

range*:low-high*

selects checkpoints that are in the specified number range. If required, you can specify just one end of the range. The range

includes the low and high values. To specify multiple ranges, the range keyword must be repeated, for example
--revisionFilter=range:1.100-1.200,range:1:400-1:500

branchrange:*low-high*
selects branched checkpoints that are in the specified number range. If required, you can specify just one end of the range. The range includes the low and high values. To specify multiple ranges, the branchrange keyword must be repeated, for example
--rfilter=branchrange:1.2.1.1-1.2.1.5,branchrange:1.4.1.1-1.4.1.5

daterange:*<date>*-*<date>*|**:past:***<nb>***:years**|**months**|**days**|**hours**
selects checkpoints that were created during a specified time range or during a specified amount of time in the past. *<date>* specifies a date in any of the local date/time formats. For more information on date/time formats, see the **options** reference page. *<nb>* is used to specify the number of units to include in the range for a specified amount of time in the past.

Note: If the date/time format contains either "-" or "," you must quote each date. For example:
--rfilter=daterange:"June 21, 2005 - 6:07:28 PM"-"June 25, 2005 - 6:07:28 PM"
When in a shell, you must use two kinds of quotes so that the shell does not remove them. For example:
--rfilter=daterange:'"June 21, 2005 - 6:07:28 PM"'-'"June 25, 2005 - 6:07:28 PM"'
or
--rfilter=daterange:"'June 21, 2005 - 6:07:28 PM'"-"'June 25, 2005 - 6:07:28 PM'"

devpath [**::current** | *:name*]
selects checkpoints that are on the specified development path.

branch [**::current** | *:name*]
selects checkpoints that are on the specified branch.

labeled*:name*
selects checkpoints with labels. If a label name is specified, only checkpoints with that label are displayed. If no label name is specified, all labeled checkpoints are displayed.

labellike*:pattern*
selects checkpoints that match the specified pattern. Your administrator determines whether **glob** or **regex** patterns are used for matching.

state*:name*
selects checkpoints that are at the specified state.

author [**::me** | *:userName]*
selects checkpoints created by the specified author.

anyspecial
selects base project checkpoints.

branchboundaries
selects checkpoints that are branched, or at the base or tip of a branch.

--maxTrunkRevs=*value*
specifies the maximum number of checkpoints along the mainline project to display. If you limit the number of checkpoints to be displayed, and the maximum is reached, a trailing message displays at the bottom of the list. The maximum number of checkpoints is based on the number of visible mainline checkpoints; branched checkpoints are not included in the count.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si archiveinfo, **si memberinfo**, **si mods**, **si print**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si viewhistory**, **si viewlabels**, **si viewproject**, **si viewsandbox**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewprojectmetrics

displays all the metrics for a specified project checkpoint

SYNOPSIS

si viewprojectmetrics [**--fields=***field1[:width1],field2[:width2]...*] [**--metrics=***value*] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*]

DESCRIPTION

si viewprojectmetrics displays all the metrics for the specified project checkpoint. For example:

```
si viewprojectmetrics --project=c:/Aurora_Program/bin/Libra/project.pj --projectRevision=1.2
```

You can track metrics for top-level MKS Integrity projects and calculate them through event triggers. Calculated metrics can be viewed for an MKS Integrity configuration management project or through a computed field on an MKS Integrity workflows and documents project issue. Metrics provide information on a project as of a specific checkpoint.

MKS Integrity provides some standard physical metrics that you can use. You can also create your own metrics using a third party tool.

The **--projectRevision** option is mandatory. Metrics can only be viewed for a build project..

Note: Metrics are only supported for database type repositories.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--metrics=*value*

specifies the metrics to display information for. The *value* is a comma-delimited list of metric names, for example, **lines,functions,bytes**. If you do not specify a value, all metrics are displayed.

Each metric is displayed on a line by itself, displaying the fields specified by **--fields** If no fields are specified, all fields are displayed except for **description**.

--fields=*field1[:width1],field2[:width2]...*

allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[:width]* (in pixels) for each field is optional - widths are only available with the **-g** or **--gui** options. Under the CLI the fields are separated with a `space`.

The fields available for printing can be one or more of the following:

average

displays the average for the metric.

count

displays the number of issues that this metric is tracked for.

metric

displays the name of the metric.

description

displays the description of the metric.

value

displays the value of the metric.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si createmetricinfo, **si calculateprojectmetrics**, **si addprojectmetric**, **si viewmetricsinfo**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewrevision

opens a revision for viewing

SYNOPSIS

si viewrevision [**--[no|un]expand**] [(**-r***rev*|**--revision=***rev*)] [(**-P** *project*|**--project=***project*)] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--[no]failOnAmbiguousProject**] [**--devpath=***path*] [**--projectRevision=***rev*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *member...*

DESCRIPTION

si viewrevision opens a member revision for viewing.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no|un]expand
: controls whether to expand, unexpand, or ignore keywords in the member file. Keyword expansion is only available in text archives, not binary archives. For descriptions of the MKS Integrity keywords, see the *MKS Integrity User Guide*. Possible keywords are:

```
$Author$
$CompanyInfo$
$Date$
$Header$
$Id$
$Locker$
$Log$
$Revision$
$Name$
$ProjectName$
$ProjectSetting attribute$
$ProjectRevision$
$RCSfile$
$Revision$
$SandboxSetting attribute$
$Setting attribute$
$Source$
$State$
```

member...
: identifies a specific member; use spaces to specify more than one member.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:

si ci, **si co**, **si edit**, **si lock**, **si unlock**

Miscellaneous:

ACL, **diagnostics**, **options**, **preferences**

si viewsandbox

displays the contents of a Sandbox

SYNOPSIS

si viewsandbox [**--[no]includeDropped**] [**--fields=***field1[:width1],field2[:width2]...*] [**--[no]filterSubs**] [**--height=***value*] [**--width=***value*] [**-x** *value*] [**-y** *value*] [(**-R** |**--[no|confirm]recurse**)] [**--filter=***filteroptions*] [**--[no]failOnAmbiguousProject**] [(**-S** *sandbox*|**--sandbox=***sandbox*)] [**--lockRecordFormat=***value*] [**--hostname=***server*] [**--port=***number*] [**--password=***password*] [**--user=***name*] [(**-?**|**--usage**)] [(**-F** *file*|**--selectionFile=***file*)] [(**-N** |**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [(**-g**|**--gui**)] [**--[no]persist**] [**--quiet**] [**--settingsUI=***[gui|default]*] [**--status=***[none|gui|default]*] *current or dropped member/subproject...*

DESCRIPTION

si viewsandbox displays the contents of a Sandbox, for example:

```
 c:\Documentation\xml_man\si_about.1.xml archived 1.6
 c:\Documentation\Man_Pages\xml_man\si_acv.1.xml archived 1.7
 c:\Documentation\Man_Pages\xml_man\si_add.1.xml  archived 1.7 sueq Jun 21, 2009 - 10:14 AM
     Working file 2,739 bytes smaller, older (Jun 20, 2009 - 1:46:27 PM)
 c:\Documentation\Man_Pages\xml_man\si_addlabel.1.xml  archived 1.6 sueq Jun 21, 2009 - 10:14
AM
    Working file 7,344 bytes smaller, older  (Jun 20, 2009 - 2:49:28 PM)
 c:\Documentation\Man_Pages\xml_man\si_addmemberattr.1.xml  archived 1.4 sueq Jun 21, 2009 -
10:14 AM
    Working file 5,795 bytes smaller, older  (Jun 20, 2009 - 3:16:29 PM)
 c:\Documentation\Man_Pages\xml_man\si_addprojectattr.1.xml  archived 1.4
 c:\Documentation\Man_Pages\xml_man\si_addprojectlabel.1.xml  archived 1.4
 c:\Documentation\Man_Pages\xml_man\si_appendrevdesc.1.xml  archived 1.6
 c:\Documentation\Man_Pages\xml_man\si_archiveinfo.1.xml  archived 1.5
```

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--[no]includeDropped
: controls whether to include members dropped from the project that still have working files in your Sandbox. Dropped members will be marked as `dropped`.

--fields=*field1[:width1],field2[:width2]...*
: allows you to select fields to be printed, specified in the format *field1[:width1],field2[:width2]...*. Specifying the column *[: width]* (in pixels) for each field is optional - widths are only available with the options. Under the CLI the fields are separated with a space>.

  The fields available for printing can be one or more of the following:

  archiveshared
  : displays indicators for members that share another member's archive. This column is valid only if you are using the database repository.

  attributes
  : displays member and project attributes.

  context
  : When used with the MKS API, displays the name of the project, and indicates if the project is a build or variant.

    Note: This field is not valid for use with the CLI.

  cpid
  : displays the change package associated with the operation that set the member revision.

  creationcpid
  : displays the change package that created the revision that is currently the member revision. This revision may be different from the Member CPID if an import, add member from archive, or set member revision operation was used.

  deferred
  : displays all deferred operations.

  frozen
  : displays indicators for frozen members.

  indent

indents a field. For example:

```
si viewsandbox --fields=indent, name, type, memberrev
```

indents the `name` field, followed by `type` and `memberrev` fields, indenting as you recurse into subprojects.

Note: This field is not valid for use with the MKS API.

labels
displays labels.

lockrecord
a comma separated list of locks on the member. The locker and lock type are displayed by default for each lock. You can customize the lock information that displays by using the **--lockRecordFormat** option.

memberarchive
display the name of the archive to which the member refers.

memberdescription
displays the revision description assigned to the Sandbox member.

memberrev
displays the member revision.

membertimestamp
displays the date and time the member revision is set.

merge
displays any merge information between member revisions.

name
displays the member name.

newrevdelta
displays indicators for new revisions.

pendingcpid
displays the change package associated with a pending operation.

revsyncdelta
displays an indicator for out of sync members.

state
displays the member state.

type
displays the type of each item in the Sandbox: Sandbox, sub Sandbox, shared sub Sandbox, shared variant sub Sandbox, shared build sub Sandbox, or member.

wfdelta
displays an indicator when the working file is different from the member revision.

workingarchive
displays the name of the archive from which the working file is derived.

In the case where the working file was derived from an archive that was based on a server-side system setup (based on some default RCSPath or WorkToArch setting, as opposed to an explicit archive=reference), the Working Archive field displays the value `default:`*server directory*. The actual name of the archive the working file was derived from can then be determined from the value for the Archive Name field.

workingcpid
displays the change package associated with a deferred or a lock operation performed by the current user from the current Sandbox.

workingrev
displays the working file revision.

--[no]filterSubs
controls whether to display sub Sandboxes and directories that do not contain members matching the current filter.

--[no]persist
controls whether this presentation of information should continue to be updated as new information becomes available. **--nopersist** forces a static "snapshot" of information, while **--persist** gives real-time updates.

--lockRecordFormat=*value*
defines the format for displaying lock information in the **--lockrecord** field. Specify a format string using keywords to represent the information you want to display. You can specify any of the following keywords:

{revision}
displays the revision that is locked.

{locker}
displays the user who locked the revision.

{locktype}
displays the type of lock on the revision (exclusive or non-exclusive).

{locktimestamp}
displays the time when the revision was locked.

{lockcpid}
displays the change package associated with the lock on the revision.

{project}
displays the name and path of the project where the member revision was locked from. If the member revision was locked from a shared subproject, it is the subproject name and path that are displayed.

{devpath}
displays the name of the development path where the lock on the revision was made from.

{sandbox}
displays the name of the Sandbox where the lock on the revision was made. This is relevant when viewing the information from the locker host.

{hostname}
displays the hostname of the computer that locked the the revision.

{hascpid}
displays 1 if the lock has a change package associated with it, 0 if there is no associated change package.

{hassandbox}
displays 1 if there is Sandbox information available for the lock, 0 if no Sandbox information is available.

{hasdevpath}
displays 1 if the lock was made from a development path, 0 if it wasn't.

{member}
displays the name of the locked revision.

current or dropped member/subproject...
identifies a specific member or subproject that currently exists in the project, or one that has been dropped from the project. Use spaces to specify more than one.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

PREFERENCES

Using **si setprefs** or **si viewprefs**, you are able to set or view the preference keys for this command.

SEE ALSO

Commands:
si archiveinfo, **si memberinfo**, **si mods**, **si print**, **si revisioninfo**, **si rlog**, **si sandboxinfo**, **si viewhistory**, **si viewlabels**, **si viewproject**, **si viewprojecthistory**

Miscellaneous:

ACL, diagnostics, options, preferences

si viewserveralert

displays MKS Integrity Server alert messages for a target server and all related servers

SYNOPSIS

si viewserveralert [**--user=***name*] [**--hostname=***server*] [**--password=***password*] [**--port=***number*] [(**-?**|**--usage**)] [(**-N**|**--no**)] [(**-Y**|**--yes**)] [**--[no]batch**] [**--cwd=***directory*] [**--forceConfirm=***[yes|no]*] [**-g** | **--gui**] [(**-F***file*|**--selectionFile=***file*)] [**--settingsUI=***[gui|default]*] [**--quiet**] [**--status=***[none|gui|default]*]

DESCRIPTION

si viewserveralert displays MKS Integrity Server server alert messages for a target sever and all related servers, for example, a configuration management server and a proxy server. The alert message displays who sent the message, the server it came from, when it was sent, and the message. Alert messages are sent by your administrator and are useful for notifying users about important information, such as an impending server upgrade in which the server will be shut down.

- In the Web interface, the date displayed for an alert message is the server's date, time, and time zone. In the GUI and CLI, the date displayed for an alert message is the client's date, time, and time zone.
- To avoid manually checking alert messages from the command line, launch the alert messages dialog box from the command line by specifying `-g` or `--gui` and keep the dialog box open. The dialog box automatically refreshes to display new alert messages.
- A connection to the target server is required for the **si viewserveralert** command to display alert messages.

Options

This command takes the universal options available to all ***si*** commands, as well as some general options. See the **options** reference page for descriptions.

--hostname=*server*
: specifies the host name of the target MKS Integrity Server to retrieve alert messages from. If this option is not specified, the default server is used.

--port=*number*
: specifies the port of the target MKS Integrity Server to retrieve alert messages from.

SEE ALSO

Commands:

si serveralerts, **si admingui**, **si servers**,

Miscellaneous:

options

ACL (Access Control List)

permissions for MKS Integrity Server ACL interaction

DESCRIPTION

The ACL (Access Control List) permissions control user access to MKS Integrity Server functions by associating development objects and operations with specific permissions. For example, whenever a user initiates an operation such as checking a file in or out, MKS Integrity queries the ACL database to determine whether the user has permission to perform the operation. This reference page is provided as a guide to the ACL permissions.

There are five server-level ACLs shipped by default: **mks:aa** controls the Login access to the AA application for managing the ACLs, **mks:aa:mks** controls Read and Update access to the ACLs, **mks:im** controls access to MKS Integrity operations, `mks:patch` controls the Download permission required for service pack management, and **mks:si** controls access to MKS Integrity operations. For the most part, however, you will be working with project ACLs, that control the permissions for a particular directory. Working with member ACLs is also possible, which control permissions for specific files -- this would be for those rare circumstances where security on specific, individual files must be heavily controlled and where the administrative costs are known and accepted.

The ACL name itself follows a specific hierarchical format:

- The default server-level ACL is named `mks:si`. All project and member ACLs will inherit permissions from this one.
- Project-level ACL names include a specific prefix, taking the format **mks:si:project:id:<project directory>**. The project directory is relative to the root of the MKS Integrity Server.
- Subproject ACLs have the same format as projects, simply appending the subdirectories using colons (:) instead of slashes.
- Variant project ACLs have a slightly different prefix, taking the format **mks:si:project:devpath:<devpathname>:id**.
- Member ACLs simply specify the file name in the ACL name, such as **mks:si:project:id:<project directory>:<member file name>**.
- Archive ACLs simply specify the archive name in the ACL name, such as **mks:si:archive:<archive path>**.

ACL PERMISSIONS

You must have the appropriate ACL permissions before you can perform MKS Integrity operations. For details on configuring ACLs, see the *MKS Integrity Server Installation and Configuration Guide*.

Member Permissions

Possible MKS Integrity member-related permissions are:

ApplyLabel
Allows a user to add labels to revisions or move labels between revisions.

Prerequisites: **Login**, **OpenProject**.

CheckIn
Allows a user to check in working files as new revisions of members.

Prerequisites: **Login**, **OpenProject**, **ApplyLabel**, **Lock**, **ModifyAuthor**, **ModifyMemberRev**, **ModifyMemberAttribute**.

DeleteLabel
Allows users to delete a label from a revision.

Prerequisites: **Login**, **OpenProject**.

DeleteRevision
Allows a user to delete revisions from the member history.

Note:
This permission allows users to irrevocably delete revisions from the member history. Administrators should assign this permission carefully.

Prerequisites: **Login**, **OpenProject**.

Demote
This permission allows a user to change the promotion state of revisions from a higher setting to a lower one, when the **States=**

configuration option defines a sequence of promotion states. For details, see the *MKS Integrity Server Administration Guide*.

Prerequisites: **Login**, **OpenProject**.

DowngradeOtherUserLock

Allows a user to downgrade exclusive locks held by other users to non-exclusive locks.

Prerequisites: **Login**, **OpenProject**.

FetchRevision

Allows a user check out member revisions.

Prerequisites: **Login**, **OpenProject**, **Lock**, **ModifyMemberRev**.

Freeze

Allows a user to freeze members. When a member is frozen, all MKS Integrity operations are run on the frozen member revision.

Prerequisites: **Login**, **OpenProject**.

Lock

Allows a user to lock revisions.

Prerequisites: **Login**, **OpenProject**.

ModifyAuthor>

Allows a user to change the author name associated with a revision.

Prerequisites: **Login**, **OpenProject**.

ModifyMemberAttribute

Allows a user to set an attribute for a member that can be used later in a search.

Prerequisites: **Login**, **OpenProject**.

ModifyMemberRule

Allows a user to configure a member revision rule that can be applied to one or more members.

Prerequisites: **Login**, **OpenProject**.

MoveLabel

Allows a user to move a member label to another revision within the member history.

Prerequisites: **Login**, **OpenProject, ApplyLabel**.

Promote

This permission specifies that a user may promote revisions from the current promotion state to a higher state, when the **States=** configuration option defines a sequence of promotion states. For details, see the *MKS Integrity Server Administration Guide*.

Prerequisites: **Login**, **OpenProject**.

RemoveOtherUserLock

Allows a user to remove locks held by other users.

Note:

This permission should be restricted to administrators.

Prerequisites: **Login**, **OpenProject**.

ShareArchive

Allows sharing of member archives between two or more members.

Note:

Archive sharing is not recommended. Instead, creating variant sandboxes is considered a better practice.

Prerequisites: **Login**, **OpenProject**, **Checkpoint**, **CheckIn**, **Lock**.

Thaw

Allows a user to thaw frozen members.

Prerequisites: **Login**, **OpenProject**.

Change Package Permissions

Possible MKS Integrity configuration management change Package related permissions are:

BypassChangePackageMandatory
Allows the user to bypass the Change Packages Mandatory policy, permitting the user to perform MKS Integrity operations without change packages.

Prerequisites: none.

ChangePackageAdmin
Allows a user to edit, discard, close, and submit change packages; as well as move or discard change package unties, regardless of any documented user restrictions.

Prerequisites: none.

CreateChangePackage
If you are using MKS Integrity only, this permission allows a user to create change packages.

If MKS Integrity configuration management is enabled with workflows and documents, this permission allows a user to create change packages based on the Change Package Creation Policy for the type that they want to create change packages for.

Prerequisites: **Login**, **OpenProject**.

PromoteCP
To use this permission you must be licensed for MKS Deploy. For more information, see the *MKS Deploy Administration Guide*.

Prerequisites: none.

SelfReview
Allows user to accept change packages under review that were created by that user.

SuperReview
Allows a user to accept or reject a change package under review regardless of the reviewer rules. **Note:** This permission supersedes the SelfReview permission.

Project Permissions

Possible MKS Integrity configuration management project-related permissions are:

AddMember
Allows a user to add new members to projects through a sandbox.

Prerequisites: **Login**, **OpenProject**, **Lock**, **ShareArchive**, **ModifyAuthor**.

AddProject
Allows a user to re-add a dropped project.

Prerequisites: **Login**, **OpenProject**.

AddSubproject
Allows a user to re-add dropped subprojects to a project.

Prerequisites: **Login**, **OpenProject**.

ApplyProjectLabel
Allows a user to add labels to projects or move labels between revisions of the project.

Prerequisites: **Login**, **OpenProject**.

CheckPoint
Allows a user to check in a new revision of a project (that is, checkpoint the project).

Prerequisites: **Login**, **OpenProject**, **ApplyLabel**, **Promote**, **PromoteProject**, **ApplyProjectLabel**.

ConfigureSubproject

Allows a user to configure a subproject's type. A subproject can be configured as a Normal, Variant, or Build subproject.

Prerequisites: **Login**, **OpenProject.**

CreateDevPath

Allows a user to create new development paths for variants of a project.

Prerequisites: **Login**, **OpenProject**.

CreateProject

Allows a user to create new projects.

Prerequisites: **Login**, **OpenProject**.

CreateSubproject

Allows a user to create new subprojects below existing projects.

Prerequisites: **Login**, **OpenProject**.

DeleteProjectLabel

Allows a user to delete a label from a project checkpoint.

Prerequisites: **Login**, **OpenProject**.

DemoteProject

This permission specifies that a user may demote projects from a higher promotion state to a lower state, when the **States=** configuration option defines a sequence of promotion states. For details, see the *MKS Integrity Server Administration Guide*.

Prerequisites: **Login**, **OpenProject**.

Deploy

To use this permission you must be licensed for MKS Deploy. For more information, see the *MKS Deploy Administration Guide*.

Prerequisites: **Login**, **OpenProject**.

DropDevPath

Allows a user to drop a development path, also known as "dropping variants" from a project.

Prerequisites: **Login**, **OpenProject**.

DropMember

Allows a user to remove members from projects. The member archive remains, but the member is no longer treated as part of the project.

Prerequisites: **Login**, **OpenProject**.

DropProject

Allows a user to drop one or more top-level, registered projects from the server. The projects then become unregistered projects.

Prerequisites: **Login**, **OpenProject**.

DropSubProject

Allows a user to drop one or more subprojects from the server. The projects then become unregistered projects.

Prerequisites: **Login**, **OpenProject**.

ImportProject

Allows a user to import one or more projects from an earlier version of MKS Integrity (formerly MKS Source). The import operation registers the project on the MKS Integrity Server.

Prerequisites: **Login**, **OpenProject**.

Metrics

Allows metrics to be tracked for a project. Allows a user to define metrics to be tracked for projects.

Prerequisites: **Login**, **OpenProject**.

ModifyMemberRev

Allows a user to make changes to the member revision of members.

Prerequisites: **Login**, **OpenProject**.

ModifyProjectAttribute

Allows a user to set an attribute for a project, which can be used later in a filter or search.

Prerequisites: **Login**, **OpenProject**.

MoveProjectLabel

Allows a user to move a project label to another project checkpoint within the project history.

Prerequisites: **Login**, **OpenProject**, **ApplyProjectLabel**.

OpenProject

Allows a user to open existing registered projects. This is required for most actions.

Note:

When OpenProject is granted or denied on a project, clients accessing the project must disconnect and then reconnect in order to get the new permission set. If you do not disconnect and reconnect your client, you may see unexpected behavior due to out-of-date permissions.

Prerequisites: **Login**.

PromoteProject

This permission specifies that a user may promote projects from the current promotion state to a higher state, when the **States=** configuration option defines a sequence of promotion states. For details, see the *MKS Integrity Server Administration Guide*.

Prerequisites: **Login**, **OpenProject**.

RestoreProject

Allows a user to restore a project to a particular checkpointed version.

Prerequisites: **Login**, **OpenProject**, **Checkpoint**.

SnapshotSandbox

Snapshot creates and records the state of the user's sandbox as a project checkpoint that you can create a build Sandbox or a development path from.

Prerequisites: **Login**, **OpenProject**, **Checkpoint, AddMember, DropMember**.

StagingSystemAdmin

To use this permission you must be licensed for MKS Deploy. For more information, see the *MKS Deploy Administration Guide*.

Prerequisites: **Login**, **OpenProject**.

ViewDeployRequest

To use this permission you must be licensed for MKS Deploy. For more information, see the *MKS Deploy Administration Guide*.

Prerequisites: **Login**, **OpenProject**.

Available Solution Permissions

The following summarizes the permissions available under mks:sd to perform specific commands:

CreateStagingSystem

To use this permission you must be licensed for MKS Deploy. For more information, see the *MKS Deploy Administration Guide*.

Prerequisites: none.

SEE ALSO

Miscellaneous:

diagnostics

NAME

cc — cc configuration files.

INTRODUCTION

cc reads commands from the configuration file, and executes them to run the appropriate compiler and linker. The command line that was passed to **cc** is examined by the code in the configuration file. If you are familiar with MKS Toolkit, you may notice that the commands resemble AWK or MKS KornShell commands. However, these commands are sufficiently different that you should study them carefully before using them. Look at sample configuration files, and experiment with **cc** in its interactive mode as you program new configuration files. The program tries to load the script `$ROOTDIR/etc/compiler.ccg`. You can modify this behavior by setting the ***CCG*** environment variable. If ***CCG*** contains the name of a file, **cc** loads and runs that file; if ***CCG*** contains the name of a directory, the program loads and runs the script in that directory.

If you rename the **cc** executable, it attempts to find its default configuration in a `.ccg` file with the same base name as the renamed executable. For example, if you rename `cc.exe` to `c89.exe`, then **c89** looks for a default configuration file named `c89.ccg` in ***ROOTDIR***`/etc`.

In any case, if you have set the ***CCG*** environment variable, its value takes precedence. If ***CCG*** contains a file name, **cc** looks for the default configuration in that file. If ***CCG*** contains a directory, **cc** looks for the default configuration in a `.ccg` file in that directory. The base name of the `.ccg` file is the same as that of the executable. That is, if the executable is `cc.exe`, the default configuration is looked for in `$CCG/cc.ccg` and if the executable has been renamed `c89.exe`, the default configuration is looked for in `$CCG/c89.ccg`.

To enter interactive mode, set the environment variable ***CCG*** to the value `-` or `stdin`. **cc** then reads its configuration file from its standard input. This is useful for debugging new configuration files: you can interactively enter commands, dump **cc**'s symbol table, and follow the execution of your **cc** program.

COMMENTS

A comment starts with a `#` character and extends to the end of the line. This is the same convention as MKS Make, AWK, and the MKS KornShell.

LITERALS

cc can manipulate three types of objects: integers, strings, and arrays of strings or integers. Since arrays are limited to one dimension, they are also called *vectors*. Unlike AWK, **cc** does not support floating point numbers.

Integers are written in the usual way. Here are some sample integers:

```
123    -49    20000
```

Strings are written with quotes around them, and cannot extend across a line. Empty strings are written as `""`. The following escape sequences are allowed within strings:

`\a`	alert (bell)
`\b`	backspace
`\h`	hard space
`\i`	hard tab
`\n`	newline
`\r`	carriage return
`\t`	horizontal tab
`\v`	vertical tab
`\\`	escaped \
`\"`	embedded "
`\`*ooo*	ASCII character with octal value *ooo*

Here are some sample strings:

```
"hello"    "\"embedded\" quotes"
"\\escaped backslash, with tab\t and newline\n"
```

A vector is list of integers or strings, separated by commas and surrounded by braces, as in

```
{ 1, 2, 3 }
{ "strings", 1, "and numbers" }
{ a++, ++c }
{ }
```

Because these are dynamic, the contents of the vector need not be constant expressions. For example, `a++` is valid in a vector list. Empty vectors are written as `{}`. You indicate a vector's elements with the bracket `[]` operator.

Indexable Objects

Both strings and vectors can be indexed -- indexing is 0-origin. Indexable objects have an associated cursor that indicates the current position of the index. This means you can treat any indexable object as a stream. (The cursor is actually associated with the value of a variable rather than with the variable itself; if you assign a new value to a variable, the cursor is reset to the origin.) The **rewind** statement lets you rewind the stream to the start of an object. Several functions are provided for dealing with indexable objects -- `read()`, `offset()`, and `tail()`.

```
# Setup the string:
stream="Alph" ;
# Read and print the first 3 elements;
# this prints "A l p"
println read(stream), read(stream), read(stream);

# Rewind the stream:
rewind(stream);
# Print the first two elements
# and the index of the next ("A l 2"):
println read(stream), read(stream), offset(a);
# Print the remainder of the string ("ph"):
println tail(stream);
```

To look ahead on a stream, use indexing combined with the offset. See *String/Vector Operations* for more information on the functions.

VARIABLES

A variable is a name with an associated value. The name follows the usual rules for C identifiers; the first character must be a letter or underscore (_), followed by zero or more letters, digits, or underscores.

Variables are created as they are first seen in **cc** input. Any variable can have any type of value; the type changes automatically if a value of a different type is assigned.

BUILT-IN VARIABLES

Several variables have pre-assigned values at the start of a **cc** program. You can change these values. The variables are:

`ARGV`
: This variable is assigned a vector containing the strings that appeared on the command line that called **cc**. Note that the name of the program is not provided. `ARGV` is typically examined, and used to build new argument lists for the compiler or linker that is being used.

`DIRSEPSTR`
: Contains the characters that can be used to separate path names into the component directories. On Windows systems, it is initially set to the string `/\:`. although the default `startup.mk` file modifies this based on the value of the ***SHELL*** environment variable. See the description of `filepath()`.

`ECHO`
: Originally assigned the empty (null) string, corresponding to a false expression. If it is later assigned a true value, the **exec** and **ovl_exec** operations (described later) print the commands as they are about to be executed.

`NORUN`
: Originally assigned the empty (null) string, corresponding to a false expression. If it is later assigned a true value, the **exec** and **ovl_exec** operations (described later) only echo their arguments, and **exec** returns as though the command succeeded, while **ovl_exec** terminates **cc** with a return value of 0. Setting `NORUN` in this way corresponds to running Make with the **-n** option. By assigning non-null values to both `ECHO` and `NORUN`, you can get **cc** to display the commands that it attempts to execute to compile a program.

`true`
: Assigned the integer value 1, `true` is used internally by **cc** to denote a true expression. You should not change this value; if you change the value of `true` to be an expression that is considered false (see *EXPRESSIONS*), **cc** may not function correctly.

PROGRAMS

A **cc** program consists of a number of statements, each of which is read and then executed. During the reading of a statement, a syntax error causes **cc** to print a diagnostic message, and then **cc** goes on to the next statement. During the execution of a statement, an error causes **cc** to print a diagnostic message. If **cc** finds a syntax error as it is reading a statement, it prints a diagnostic message and goes on to the next statement. While executing a statement, if **cc** finds an error, it prints a diagnostic message; if an expression is being calculated, then the null string is returned as the value of the expression.

STATEMENTS

This section describes the statements that may be used in input to **cc**. In the following, the characters {} are used to indicate zero or more occurrences of the enclosed items. The characters [] indicate an optional occurrence of the enclosed items.

The following table summarizes statements allowed in **cc**:

Description	Syntax

Advance	**advance** *name* ;
Assignment	*name* = *expr* ;
	name [*expr1*] = *expr2* ;
Assign & concatenate	*name* |= *expr* ;
	name [*expr1*] |= *expr2* ;
Break loop	**break** [*expr*] ;
Close file	**close** *filename* ;
Delete file	**delete** *filename* ;
Dump debugging info	**dump** ;
Empty statement	;
Execute command	**exec** *expr* ;
Exit	**exit** ;
For	**for** (*name* in *expr*)
	do *statements* **done**
For	**for** (*expr1* ; *expr2* ; *expr3*)
	do *statements* **done**
Functions	**function** *name* ([*arg1* { [, *arg2*]] })
	{ *statements* }
If	**if** (*expr1*) *statements*
	{ **elif** (*expr2*) *statements* }
	[**else** *statements*]
	fi
Open file	**open** *filename* ;
Overlay file	**ovl_exec** *cmd_expr1* { , *expr2* } ;
Print expression(s)	**print** *expr1* { , *expr2* } ;
Print line	**println** *expr1* { , *expr2* } ;
Print to file	**print** *expr1* { , *expr2* } -> *filename* ;
	println *expr1* { , *expr2* } -> *filename* ;
Rewind cursor	**rewind** *indexable_object*
While	**while** (*expr*) **do** *statements* **done**

Table 1: Summary of **cc** Statements

In this definition, *statements* means a list of statements, and *expr*, *expr1*, and *expr2* mean any legal expression as described a bit later. The expression value is used as a logical expression. See *EXPRESSIONS* for the description of what values are considered true and false.

Empty statement

An empty statement is written as a single semicolon (;). It has no effect.

If statement

An **if** statement has the form:

```
if ( expression ) statements
{ elif ( expression ) statements }
[ else statements ]
fi
```

Note that **fi** is required to end the **if** statement. Any number of **elif**'s can be inserted before the end of the **if** statement.

While statement

A while statement has the form:

```
while ( expression ) do statements done
```

The **done** is required to mark the end of the *statements*. The meaning is similar to the C while loop.

For loop

A **for** statement has one of these forms:

```
for ( name in expression ) do statements done
```

```
for ( expr1; expr2; expr3 ) do statements done
```

In the first form, the *name* must be a variable name, and the *expression* must evaluate to a vector or string. If the vector is not empty, the *statements* are executed repeatedly; on each iteration, *name* is assigned each successive value in the vector. The second form is much more like the C language **for** loop. Usually, the first expression initializes some variable, the second expression tests it, and the third expression increments or decrements it.

Advance

An advance statement has the form:

```
advance name ;
```

where *name* is a variable name used in an enclosing **for** loop. This statement immediately assigns to *name* the value it would normally have been given in the next iteration of the **for** loop, and subsequent iterations skip this value.

Break loop

A **break** statement has the form:

```
break [expr] ;
```

where *expr* is the number of levels to break out of. If no expression is specified, **break** exits the current loop (*expr* is 1).

Exit Program

The **exit** statement has the form:

```
exit expression ;
```

This terminates **cc** with the integer-valued expression as its exit value.

Dump Debugging Information

The **dump** statement is a debugging aid.

```
dump ;
```

outputs the names and values of all variables currently defined to the standard output.

Print Expression

There are several printing statement forms.

```
print expression { , expression } ;
```

prints the values of the given *expressions* on the standard output. If more then one *expression* is given, they are printed with a single blank separating each item. **print** does not put a newline character at the end of the output.

```
println expression { , expression } ;
```

is similar to the previous format, but does print a newline at the end of the list of expressions.

```
print expression { , expression } -> filename ;
println expression { , expression } -> filename ;
```

are similar to the other printing statements, but they print to the given file instead of to the standard output. The *filename* is given as a string-valued expression. If the file does not exist, it is created; if it already exists, output is appended to the end of the current contents.

Close File

To close a given file, use

```
close filename ;
```

The *filename* is given as a string-valued expression. If you use a printing statement to write to a file, it is important to close the file before **cc** runs a program that requires the contents of the file to be up-to-date.

Delete File

To delete a given file, use

```
delete filename ;
```

You can use this to get rid of temporary (work) files created by earlier statements in the **cc** program. The file name is given as a string-

valued expression.

Define Function

You can define a function using the **function** statement, which has the form:

```
function name ( [arg1{, arg2}] )
{
        statements
}
```

Execute a command

To execute a command, use

```
exec expression { , expression } ;
```

Each expression is converted into a string (if necessary) and then concatenated into a command string (with a single blank between each expression in the list). For example,

```
exec "cp","infile","outfile" ;
```

executes the command

```
cp infile outfile
```

The **exec** statement executes the command and then goes on to the next statement. (See the **exec** expression following for ways to get the return status of the executed command.) The behavior of **exec** can be changed by the values of the ECHO and NORUN variables.

Overlay Execute

The statement

```
ovl_exec expression { , expression } ;
```

is similar to an **exec** statement. In this case, the memory used by **cc** is overlaid with the command to be run, and **cc** is effectively terminated. This frees the memory space that **cc** is occupying for the executed program. The behavior of **ovl_exec** can be changed by the values of the ECHO and NORUN variables.

Rewind Stream

The **rewind** statement has the form:

```
rewind indexable_object ;
```

where *indexable_object* is a string or vector. This statement moves the cursor associated with *indexable_object* back to the origin. For more about indexable objects, see *Indexable Objects*.

EXPRESSIONS

Expressions are created by combining literal strings, integers, vectors, and variables, with the operators listed a bit later.

An expression has a type: integer, string, or vector. When an integer value is required and a string or vector is provided, some operators convert the vector into a string, and then interpret that string as an ASCII sequence representing a number. For example, `"123"` has the integer value `123` when used in arithmetic operations. By the same token, if a string is required, an integer value is converted into a string in the same way.

Some expressions are calculated by determining if operands are `true` or `false`. The following are considered false:

- the integer 0
- the null string `""`
- an empty vector `{ }`

Other objects are considered true. The built-in variable `true` can also be used as a condition.

Operators are grouped in precedence classes that are similar to those of the C programming language. Each section that follows describes a group of operators in a single-precedence class. The order of the sections gives the order of precedence.

Order of Operations	
A++ A-- ++A --A	pre- and post-increment and decrement
-A (A) V[a]	unary minus, grouping, array element

A*B A/B	integer multiplication and division
A+B A-B	addition and subtraction
A%B	modulus
:	
A==B A!=B	equality comparison
A<B A>B A<=B A>=B	relational comparison
A&&B A||B !A	logical AND, logical OR, logical negation
,	
->	file redirection
length(A)	length operation
A | B	string/vector concatenation
A=B A|=B	assignment
A and B are any expression.	
V is any vector expression.	

Table 2: cc Order of Operations

Primary Expressions

These expressions have the highest precedence.

- *expression*

gives the negative of an integer expression.

(*expression*)

gives the value of the expression inside the parentheses, and you use it to change the order of evaluation of expressions.

indexable_object [*expression*]

obtains the value of a specific indexable element. Indexable objects (vectors and strings) are indexed with 0-origin. If a vector subscript is out of range, an error message is displayed, and the null string is returned. As a special case, the 0-th element of a 0-length vector does not cause an error; instead, its value is the null string.

Arithmetic Operators

The standard multiplication operators are:

expression1 * *expression2*
expression1 / *expression2*
expression1 % *expression2*

representing integer multiplication, division, and the modulus. Non-integer operands are converted to integers in the usual way.

The standard addition operators are:

expression1 + *expression2*
expression1 - *expression2*

representing addition and subtraction. Non-integer operands are converted to integers in the usual way.

Equality Operators

The result of:

expression1 == *expression2*

is true if the two expressions have equal values.

expression1 != *expression2*

is true if the two expressions are not equal.

In both cases, if either operand is an integer, the other is converted into an integer for the purposes of comparison; otherwise, both expressions are converted into strings, and then compared.

Relational Operators

The relational operators are:

expression1 > *expression2*
expression1 < *expression2*
expression1 >= *expression2*
expression1 <= *expression2*

The value of these expressions is true if the given relationship is true. If one operand is an integer, the other is converted to an integer (if necessary) and the comparison takes place numerically; otherwise, operands are compared as strings, according to the ASCII collating sequence.

Logical Negation

The expression

! *expression*

is true if *expression* is false, and false if *expression* is true.

Logical Conditional Expressions

The conditional AND expression has the form:

expression1 && *expression2*

The value of this expression is false if *expression1* is false, in which case *expression2* is not evaluated; otherwise, if *expression1* is not false, the value of the expression is the value of *expression2*.

The conditional OR expression has the form:

expression1 || *expression2*

If *expression1* is not false, the value of the expression is that value, and *expression2* is not evaluated; otherwise, the value of the expression is the value of *expression2*.

Length Operation

The length operation has the form:

length (*expression*)

If *expression* is a vector, the value is the number of elements in the vector. If *expression* is a string, the value is the number of characters in the string; otherwise, the value is 0.

String/Vector Operators

The operator | concatenates strings or vectors. The form is

expression1 | *expression2*

where *expression1* must be a string or vector. If *expression1* is a string, the second argument is converted to a string and the two strings are concatenated for the result. For example, in

```
X = "hello";
X = X | " good" | "bye";
```

the variable X is assigned the value

```
"hello goodbye"
```

You can use the shorthand notation |= to append to the end of an existing string, as in

```
x = "hello";
x |= " good" | "bye";
```

which has the same result.

If the first argument of the | operator is a vector, the second argument is added as a new component to the vector. In this way, you can assign values to a vector incrementally. You can use the |= operator as a shorthand assignment. For example:

```
x = {}          # ensure x is a vector
x |= 1;         # x is now { 1 }
x |= "so";      # x is now { 1, "so" }
x |= 2 | 3;     # x is now { 1, "so", "23" }
# Remember, the '|' is applied to '2' and '3',
# forming a string, that is then added
x |= { 4, 5 };  # x is now { 1, "so", "23", 4, 5 }
```

Assignments

There are several assignment expressions.

name = *expression* ;

assigns the value of the *expression* to the given named variable.

name[*expression1*] = *expression2* ;

assigns the value of *expression2* to the element of the vector variable specified by *expression1*. It is an error to use an index greater then the length of the vector.

name |= *expression* ;
name[*expression1*] |= *expression2* ;

are concatenation assignments. The value of the right hand side is concatenated to the existing value of the left hand side. See *String/Vector Operators* for more about concatenation.

Because assignments are expressions, rather than statements, they can be used anywhere. For example, this is valid:

```
if (a = 2)
        println "true" ;
fi
```

String/Vector Operations

There are a number of other string/vector operators, summarized in this table:

Function	Description
access(*filename*, [*num* | *str*])	return true if *filename*'s mode is *str* or *num*
cd(*pathname*)	change to named directory
cinclude(*filename*)	include *filename*; return 0 if it does not exist
exec(*expression* {, *expression*})	execute command line
filepath(*expression*)	return vector containing path name components
getcwd()	return current working directory path
getenv(*name*)	return value of environment variable *name*
getopt([*str*[, *optind*]])	process options string
include(*filename*)	include *filename*; abort if *filename* does not exist
index(*str*, *expr*)	return index of first *expr* in *str*
isatty(*fileno*)	return true if *fileno* is a tty
offset(*indexable_object*)	return cursor offset in *indexable_object*
ovl_exec()	execute command line in overlaid memory
putenv()	alter value of environment variable
read(*indexable_object*)	return next token from *indexable_object*
replace(*str*, *old*, *new*)	return *str* with *old* replaced by *new*
rindex(*str*, *expr*)	return index of last *expr* in *str*
strerror()	return string corresponding to errno
substr(*expr*, *pos*)	return substring of *expr* starting at *pos*

substr(*expr*, *pos*, *len*)	return length *len* substring of *expr* starting at *pos*
tail(*indexable_object*)	return unread portion of *indexable_object*
tempfile([*dir_str*,] *pre_str*)	temporary file name
tolower(*expr*)	return *expr* in lowercase
toupper(*expr*)	return *expr* in uppercase

Table 3: Summary of **cc** Functions

access(*filename*, [*mode*])
: returns true if the access mode of *filename* is the same as the supplied *mode*. The *mode* should be passed as a string composed of the characters r, w, x, or f (for read, write, execute, or file existence). If no *mode* is supplied, f is assumed.

```
access("temp.af9","f")  -- checks if temp.af9 exists
access("temp.af9","rw") -- checks for read/write
                        permission on temp.af9
```

If the base operating system allows, *mode* can be passed as a number. This is non-portable, so you should use a string representation.

cd(*pathname*)
: changes the current directory to the value of *pathname*.

cinclude(*filename*)
: includes the specified file. No path search is performed. If *filename* is not an absolute path name, the current directory is searched. If the file does not exist, this function returns 0; otherwise, it returns 1.

exec(*expression* {, *expression*})
: is similar to the **exec** statement. The value of an exec expression is the exit status of the last command executed. This is an integer.

filepath(*expression*)
: returns a vector whose components are the strings corresponding to the path name components of the string *expression*. The built-in variable DIRSEPSTR is used to determine the components of the path.

getcwd()
: returns the current working directory path.

getenv(*name*)
: returns the string value of the environment variable indicated by the string expression *name*. If the environment variable is not set, the null string "" is returned.

getopt([*optstr* [, *optind*]])
: processes an option string; this simplifies the business of writing scripts that take options. *optstr* is a format string listing each option character; if the option character is followed by a colon, the option takes an argument. For example, the format string "aei:o:u" specifies that the options **a**, **e**, **i**, **o**, and **u** are valid, and the colons show that both **i** and **o** require arguments. The option being processed is stored in a variable; if there is an argument, it is stored in the global variable OPTARG.

optstr only needs to be specified on the first call. Subsequent calls to getopt() continue processing the same ARGV argument string. Here is a bit of code that sets flag variables for each option called:

```
while (opt = getopt("dl:rtx")) do
        if (opt == "d" ) d_flag++;
        elif (opt == "l" )
                l_flag++;
                l_arg = OPTARG;
        elif (opt == "r" ) r_flag++;
        elif (opt == "t" ) t_flag++;
        elif (opt == "x" ) x_flag++;
        fi
done
```

The getopt call does not remove items from the ARGV string, it just moves the OPTIND pointer through the ARGV vector. If you want to pass the tail of the argument string without the options, you can reset the value of ARGV with code like this:

```
members = {};
for (i=OPTIND; ARGV{i}; i++) do
        members |= { ARGV[i] );
done
ARGV = members;
```

include(*filename*)
: includes the specified file. No path search is performed. If *filename* is not an absolute path name, the current directory is searched. If the

file does not exist, the script aborts.

Note:

Expressions are compiled completely and then executed. As a result, if you use `include` in a function, the file is not included until an expression referencing the function is executed.

`index(`*expression*`,` *string-expression*`)`

returns the index of the first occurrence of *string-expression* in the *expression* If the first expression is a vector, it is first converted into a string. The index is 1-origin; a value of 0 is returned if the second expression is not a substring of the first.

`isatty(`*filenum*`\)`

returns `true` if *filenum* is a tty; *filenum* is an integer representing a file number. The values `0`, `1`, and `2` represent standard input, standard output, and standard error, respectively.

`offset(`*indexable_object*`)`

returns the current stream offset in an indexable object.

`ovl_exec(`*expression* `{,` *expression*`})`

constructs a command line in the same way, and then overlays **`cc`** with that command. **`cc`** exits with the exit code returned by that command.

`putenv(`*string*`)`

where *string* is a string expression of the form *name=value* assigns *value* to the environment variable indicated by the string expression *name* and adds that variable to the current environment.

`read(`*indexable_object*`)`

reads the next item from *indexable_object* and returns it.

`replace(`*string*`,` *old*`,` *new*`)`

returns *string* with all occurrences of the first character of the string *old* replaced with the first character of the string *new*.

`rindex(`*expression*`,` *string-expression*`)`

is similar to `index` but returns the index of the *last* occurrence.

`strerror()`

Returns the string that corresponds to the current value of `errno`. If `errno` is not set, returns a null string.

`substr(`*expression*`,` *position*`)`

returns the substring of the first string, starting at the position given by the second argument (an integer expression). Strings are 1-origin; however, for convenience, if the value of *position* is 0, `substr` returns the substring beginning at 1. If *position* is negative, it is taken to be an offset from the end of the string. For example:

```
substr("hello",0) = substr("hello",1) -> "hello"
substr("hello",length("hello")) -> "o"
substr("hello",-3) -> "llo"
substr("hello",-1) -> "o"
```

`substr` may also take three arguments, as in:

`substr(`*expression*`,` *position*`,` *length*`)`

The *length* argument is an integer expression, giving the desired number of characters in the substring. If this value is negative, the substring is obtained by advancing to the indicated position, marking that the end, and then retreating by the specified amount. For example:

```
substr("hello",0,1) = substr("hello",1,1) -> "h"
substr("hello",-1,-3) -> "llo"
```

`tail(`*indexable_object*`)`

returns the unread portion of an *indexable_object*.

`tempfile([`*dir_str*`],` *pre*`)`

returns a string guaranteed to be an unused temporary file name in the directory specified by *dir_str*. If no *dir_str* is provided, then the directory specified by ***TMPDIR*** is used; if ***TMPDIR*** is not set, the system default is used (usually `/tmp`. If *pre* is not an empty string, it is taken as a prefix for the name of the temporary file.

`tolower(`*expression*`)`

returns the string *expression* with all uppercase characters translated to lowercase.

`toupper(`*expression*`)`
: does the opposite translation.

AVAILABILITY

MKS Toolkit for Developers
MKS Toolkit for Interoperability
MKS Toolkit for Professional Developers
MKS Toolkit for Enterprise Developers
MKS Toolkit for Enterprise Developers 64-Bit Edition
MKS Source Integrity Standard
MKS Integrity

SEE ALSO

Commands:

`cc`

diagnostics

applicable to MKS commands

DESCRIPTION

The exit status values for MKS commands (**si**, **im**, **tm**, **aa**, **integrity**) can be used by event triggers for automating processes with MKS Integrity. This reference page is provided as a guide to the exit status values you may see.

DIAGNOSTICS

Possible exit status values for MKS commands are:

`0`

Successful completion.

or

No differences between the files being compared (using **si diff**).

`1`

Command usage error.

`2`

Command was canceled by user. This does not include cancellations using CTRL-c, which overrides this exit status value. In those cases, the return code will be `130`.

`3`

Invalid element in the selection for the command.

`4`

Sandbox specified was ambiguous (using an **si** command). Command not executed.

`5`

Unable to create or utilize the selection for the command.

`6`

Unable to continue with the selection for the command because the program cannot find the next element.

`10`

Connection failed: a network error caused the command to terminate.

`16`

diff compared the files and found them to be different (using **si diff**).

`17`

Failure due to any of the following (using **si diff**):

- invalid command line argument
- cannot open one of the input files
- out of memory
- read error on one of the input files
- more than LINE_MAX characters between newlines

`19`

At least one of the files is a binary file containing embedded NUL (\0) bytes (using **si diff**).

`128`

General command failure.

`130`

Command was canceled by the user using CTRL-c.

`255`

Unknown exception or error code.

SEE ALSO

Miscellaneous:

ACL, options, preferences

NAME

envvar — standard environment variables

SYNOPSIS

export ***NAME***=*value*

set ***NAME***=*value*

echo ***$NAME***

DESCRIPTION

When a process is executed, it inherits a set of strings called the *environment*. It is conventional for these strings to have the form:

NAME=*value*

The **export** command built into the MKS KornShell can be used to set the variable ***NAME*** into the environment of every child process. The **set** command built into **command.com** or **cmd.exe** does the same. The **echo** command prints the value of environment variable ***NAME*** inside the MKS KornShell.

Note:
The shell maintains additional shell variables that are not exported to child processes; because these variables are not passed on, they are not environment variables.

The following environment variables are used throughout MKS Toolkit:

COLUMNS
If you set this variable to a numeric value, various commands use its value as the width of the output device in columns. This overrides the default.

COMSPEC
The value of this variable must point to the standard command interpreter (**command.com** or **cmd.exe**), if you want to use that command interpreter in any way. This variable is called ***ComSpec*** under Windows NT/2000/XP/2003.

ENV
The value of this variable is the name of a file of MKS KornShell commands, or else be null. When the MKS KornShell is invoked, the file named by ***ENV*** is executed before the MKS KornShell does anything else. Thus your ***ENV*** file may contain definitions of aliases, shell functions, and son on that may be used by shell scripts. Note that your ***ENV*** file is executed, whether or not the MKS KornShell is invoked as a login shell. This differs from older releases of the MKS KornShell.

HASHBANG
If this variable is set then the *#!* feature of the MKS KornShell is enabled.

HOME
This variable is set by the shell's default startup files. It contains the name of your home directory. Your home directory is the default directory for the **cd** command built into the MKS KornShell.

LINES
If you set this variable to a numeric value, various commands use its value as the number of lines available on the output device. This overrides the default.

LOGNAME
This variable is set by the shell's default startup files. It holds the user name of the current user.

MAILER
For commands that send mail, this variable points at a mail delivery program. If this variable is not set, then the default mailer, **mailx** is invoked.

PATH
This variable is set to a default value when you start the MKS KornShell. Normally, it is also set in your profile file. It lists the directories that are to be searched to find commands, as described in **sh**.

ROOTDIR
Because Windows systems have a multi-device file system, it is necessary to provide the location of the *standard* root directory for system files (for example, `/etc/profile.ksh` and `/tmp`). This variable contains a device name and possibly a directory where such files are found.

SHELL

This variable contains the full path name of the current shell. Note that if ***SHELL*** is not defined, all commands that require the full path name of the current shell use the contents of the environment variable ***COMSPEC***.

TERM

This variable contains the terminal type.

TK_NTLINKS_OFF

MKS Toolkit supports hard links under Windows NT/2000/XP/2003 on NTFS file systems. There is a slight loss of performance for this support. If you do not require hard link support then you should set and export the environment variable ***TK_NTLINKS_OFF*** to disable this support.

TK_NTSECURITYINFO_OFF

MKS Toolkit supports Windows NT/2000/XP/2003 security information on NTFS file systems. There is a slight loss of performance for this support. If you do not require any security information then you should set and export the environment variable ***TK_NTSECURITYINFO_OFF*** to disable this feature.

TK_NTSECURITYINFO_SID_TERSE

Under Windows NT/2000/XP/2003, when using **ls** with the **-l** option, files having an associated `SID`, whose name cannot be determined, display the value of the `SID` instead. `SID` values are usually very large. You should set and export the ***TK_NTSECURITYINFO_SID_TERSE*** environment variable. This causes all `SID` values to be shortened by replacing all the subauthority values, except the last one, by the string "-...-".

TK_PERL_USE_COMSPEC

When set, this environment variable causes **perl** to use the contents of the environment variable ***COMSPEC*** as the full path name of the current shell, whether or not the environment variable ***SHELL*** is set.

TMPDIR

By default, MKS Toolkit commands store temporary files under `/tmp`. To use a different directory for temporary files, set ***TMPDIR*** to the name of the directory you want to use.

TZ

Commands that print times (and dates) use this variable to determine the time zone. If the `TZ` variable is left undefined, the operating system's current time zone setting is used. See **timezone** for details. On Windows, the MKS Toolkit makes use of the built-in timezone support, and you should not set the ***TZ*** environment variable.

PORTABILITY

The standard names on Windows systems are a superset of those used by UNIX programs, with ***COMSPEC*** (***ComSpec*** under Windows NT/2000/XP/2003), ***ROOTDIR*** and ***TK_**** being the major additions.

AVAILABILITY

MKS Toolkit for Power Users
MKS Toolkit for System Administrators
MKS Toolkit for Developers
MKS Toolkit for Interoperability
MKS Toolkit for Professional Developers
MKS Toolkit for Enterprise Developers
MKS Toolkit for Enterprise Developers 64-Bit Edition
MKS AlertCentre
MKS Source Integrity Standard
MKS Source

SEE ALSO

Commands:

cd, **env**, **ls**, **sh**

Miscellaneous:

timezone

NAME

makefile — format of MKS Make file

DESCRIPTION

This reference page offers a summary of the syntax of a makefile's contents, using a style similar to BNF (Backus Naur Form). Items enclosed in [ ] are optional, while items enclosed in { } can appear zero or more times.

makefile
→ { *makefile-line* }
makefile-line
→ *macro-definition*
→ *target-definition*
→ *attribute-definition*
→ *conditional*

A line is a possibly empty sequence of characters, terminated by a nonescaped newline character; that is, one not immediately preceded by a backslash (\). A line can be extended over several input lines by placing a \ at the end of each line to be continued.

A comment begins with a # and extends to the end of the line. In the following, an *expression* is defined as

expression
→ *string*
→ *string* == *string*
→ *string* != *string*

where *string* is defined as a literal sequence of characters, or the sequence of characters resulting from the expansion of a macro.

A *conditional* is defined as

conditional →
.IF *expression*
makefile
{ .ELSIF *expression*
makefile
}
[.ELSE
makefile
]
.END

A *rule-definition* is defined as

rule-definition →
targets [*attributes*] *op* [*prerequisites*] [; *recipe*]
recipe
targets →
target { *target* }

A *target* may be a *special-target*.

Any number of *attributes* may follow a *target* in a *rule-definition*.

attribute
→ .IGNORE
→ .SILENT
→ .PRECIOUS
→ .LIBRARY
→ .PROLOG
→ .EPILOG
→ .SETDIR=*path*

An *attribute-definition* is like a *rule-definition*, except that a *recipe* is not allowed. An *attribute-definition* is given as

attribute-definition
→ *attribute* { *attribute* } : *targets*

The *op* defines the type of rule.

op → :

→ ::
→ :^
→ :!
→ :-
→ :|

The *prerequisites* are a list (possibly empty) of targets that must be brought up-to-date before making the current target. The *prerequisites* may optionally be followed by a semicolon (`;`) and a *recipe*.

A recipe normally follows; it consists of a number of lines, all starting with at least one tab character, or a *group-recipe*, which is written between brackets (`[ ]`).

recipe
→ { [+@-] *line* }
→ [[+@-]*lines*]

A *special-target* is a keyword that is given as a *target* in a *rule-definition*. The following special targets are allowed:

special-target
→ .BRACEEXPAND
→ .ERROR
→ .EXPORT
→ .GROUPEPILOG
→ .GROUPPROLOG
→ .IMPORT
→ .INCLUDE
→ .INCLUDEDIRS
→ .MAKEFILES
→ .NOAUTODEPEND
→ .REMOVE
→ .SOURCE
→ .SOURCE.*suffix*
→ .SUFFIXES
→ .POSIX

A *macro-definition* is recognized as a *string* followed by an `=` character. There are three forms of macros:

macro = *string*
assign *string* to *macro*
macro := *string*
expand macro definitions *string*, then assign
macro += *string*
append *string* to end of *macro*

If *string* is empty, *macro* is assigned the null string (unless you use the `+=` operator; in this case, the macros contents are not touched). White space on either side of the `=`, `:=`, or `+=` is ignored.

Macros are expanded by `$(`*macro*`)` or `${`*macro*`}`. If a macro name is a single letter, it can also be expanded by `$`*n*, where *n* is the name.

If the `.BRACEEXPAND` special target is set, **make** expands a token sequence (separated by white space) prepending a prefix string and appending a suffix string to each token:

string1{ *token-list* }*string2*

Future versions of MKS Make may not support this obsolescent feature.

A macro can also be modified as it is expanded. The syntax for this is

$(*macro* {:*modifier*})

where any number of *modifiers* can be supplied, separated by `:`.

modifier
→b
→d
→f
→l
→ s/*string*/*new_string*/
→ *suffix_string*=*new_suffix_string*
→ t"*string*"
→u
→^"*string*"
→+"*string*"

The `:s` modifier is a substitute operator; occurrences of the given pattern that are matched are replaced by the indicated value. The `:t` modifier

tokenizes the macro string definition, placing the given *string* between each name in the indicated string. That string may contain the following escape sequences:

```
\"      →       "
\\      →       \
\a      →       alert or <bel>
\b      →       <backspace>
\f      →       <form feed>
\n      →       <newline>
\r      →       <carriage return>
\t      →       <tab>
\v      →       <vertical tab>
```

The `:u` modifier maps all characters in the expansion to uppercase.

The `:l` modifier maps all characters in the expansion to lowercase.

The `:^` modifier adds a prefix to all of the tokens in the expanded macro.

The `:+` modifier adds a suffix to all of the tokens in the expanded macro.

A number of built-in macros are defined by MKS Make.

$@	full target name
$$@	dynamic expansion of $@
$%	full target name, or name of archive library
$$%	dynamic expansion of $%
$*	$@ without suffix; same as `$(@:db)`
$$*	dynamic expansion of $*
$>	library name for current target (library member)
$$>	dynamic expansion of $>
$?	all recently changed prerequisites
$<	in normal rules, those prerequisites prompting execution of current rule; in inference rules, the prerequisite of the current rule
$&	all prerequisites
$^	prerequisites in current rule

MKS Make also uses *control-macros*, whose value is either maintained by **`make`**, or used to control the corresponding *attribute*.

control-macros

```
→ DIRSEPSTR
→ .EPILOG
→ GROUPFLAGS
→ GROUPSHELL
→ GROUPSUFFIX
→ .IGNORE
→ INCDEPTH
→ MAKE
→ MAKECMD
→ MAKEDIR
→ MAKEFLAGS
→ MAKESTARTUP
→ MFLAGS
→ NULL
→ .PRECIOUS
→ .PROLOG
→ PWD
→ SHELL
→ SHELLFLAGS
→ SHELLMETAS
→ .SILENT
```

For example, the `.SILENT` macro is like the `.SILENT` attribute, except that using the macro assigns the `.SILENT` attribute to all targets.

AVAILABILITY

MKS Toolkit for Developers
MKS Toolkit for Interoperability
MKS Toolkit for Professional Developers
MKS Toolkit for Enterprise Developers
MKS Source Integrity Standard
MKS Integrity

SEE ALSO

Commands:

options

applicable to all MKS commands

DESCRIPTION

This reference page contains the following information:

Specifying Members, Sandboxes and Projects for MKS Integrity Commands

There are three types of MKS Integrity configuration management commands, and therefore the way you specify the *member* varies:

1. Some commands can only be executed on members in the context of a Sandbox, because they manipulate the member's working file. These are noted as requiring a *sandbox member...*, such as **si ci**, **si co**, and **si merge**. These take a Sandbox member, and you may not perform the operation against a project member. If you try to specify a project for these commands, you will see an error message.
2. Other commands do not manipulate a member's working file, and therefore can be performed directly in the context of a project. If you specify a Sandbox, it is simply used as a pointer to find the project itself. The fact that you specify the Sandbox is only incidental. These are noted as requiring a *project member...*, and most of the commands that say *member...* fall into this category; for example, **si updaterevision**, **si updatearchive**, and **si addlabel**.
3. There are also some commands that perform differently depending on whether they are given a *sandbox* or a *project*. Examples for this would be **si diff** and **si edit**.

Specifying Sandboxes Explicitly or Implicity for MKS Integrity Commands

You can explicitly specify either **-P** or **-S** options for most commands. For **-P** you must specify a project or subproject, the **-P** option does not accept the filename of a Sandbox. For **-S** you must specify a Sandbox or sub Sandbox, **-S** does not accept the filename of a project.

MKS Integrity also allows for implicit Sandbox selection. This means that you can use commands without explicitly specifying a **-S** *sandbox* option, and MKS Integrity determines the Sandbox to operate on based on the directory you're working in.

For example, suppose you are working in the directory `C:/test/sbx/sub1/sub2`, and suppose you have a Sandbox only in the `/sbx` directory. Using the **-S** option to explicitly specify the Sandbox, you might enter the following to check in a file:

```
si ci -S c:/test/sbx/project.pj header.c
```

Using implicit Sandbox location to check in a file, you might enter:

```
si ci header.c
```

If the Sandbox is not specified explicitly through **-S**, MKS Integrity tries to locate one by starting in the current working directory and seeing if there is a Sandbox registered in that directory. If there isn't, it searches up the directory tree until it either finds one, or until it reaches the root of the drive. In the example provided, MKS Integrity first determines there is no Sandbox in `/sub2`, then `/sub1`, then finds the Sandbox in `/sbx` and uses that Sandbox.

Now suppose you have Sandboxes in each of `/sbx`, `/sub1`, and `/sub2`. This implicit Sandbox selection allows you to work with multiple Sandboxes in one command line entry, and without having to explicitly specify lengthy locations for each one. If you're working in the `C:/test/sbx` directory and decide to check in files to each Sandbox, for example, you might enter:

```
si ci header.c sub1/comp.c sub1/sub2/img.c
```

This checks in the file `header.c` to the Sandbox at `C:/test/sbx` , checks in the file `comp.c` to the Sandbox at `C:/test/sbx/sub1`, and checks in the file `img.c` to the Sandbox at `C:/test/sbx/sub1/sub2`.

A requirement for implicit Sandbox location to operate correctly is that there can be no more than one Sandbox (or sub Sandbox) in a single directory. If you create two or more Sandboxes in the same directory, the implicit Sandbox location algorithm cannot unambiguously determine which Sandbox to use in that directory, and it prompts you to clarify by specifying the name of the Sandbox that you want to use in that case. In general, you shouldn't create multiple Sandboxes in the same directory.

Note:
Certain **si** commands do not operate on Build Sandboxes, which are created as read-only for the purpose of building a programming artifact. Using inappropriate commands with a Build Sandbox causes error messages to appear.

Specifying Projects

This section provides information on the two available syntaxes for specifying projects, followed by examples of their usage. The following two syntaxes are available:

- *Source Configuration Path*
  A keyword-based string that provides the ability to specify subprojects within the context of a project tree (see examples that follow).
- *Flat Path*
  The legacy syntax that may not be supported in future releases. It takes the form of a simple pathname string, possibly accompanied by a development path name or a project checkpoint.

WELL FORMED PROJECTS: MKS recommends using well formed projects wherever possible. A well formed project is one where every directory contains a subproject (if not possible, then all members should belong to the nearest enclosing subproject), and that subproject is named `project.pj`.
BENEFIT: Well formed projects use more compact paths. Using a well formed project elminates the need to use hash (#) values for specifying projects (refer to Path ambiguity example).

SCENARIOS WHERE SOURCE CONFIGURATION PATH IS SUPERIOR TO FLAT PATH SPECIFICATION

The following scenarios are documented in this section, and illustrate how using the source configuration path syntax is the superior choice compared to flat path specification:

- Subproject not on main development path
- Path ambiguity
- Ambiguous co-located subprojects

The following keywords are used in the examples:

- The **#** keyword specifies the well-formed project or subproject name. Well-formed project and subproject names end with `project.pj`.
- The **#d** keyword specifies the development path name.
- The **#s** keyword specifies the subproject in a poorly-formed project tree. A poorly-formed project tree has co-located subprojects or subprojects located more than one directory level deep. Using this keyword, you can only specify one subproject for each occurrence of the keyword.

For a description of all keywords, see the **-P** option under General Options.

SUBPROJECT NOT ON MAIN DEVELOPMENT PATH

The flat path cannot handle the case where the object (in this case a sub project on a variant) MKS Integrity is locating does not exist in the main project tree (on the main devpath).
In the following diagram, user needs to specify project `sub2` on devpath Dev.

Tree 1	Tree 2
/aurora_project/project.pj	/aurora_project/project.pj (Development Path Dev)
|	|
|	|
--- sub/project.pj	---sub/project.pj
	|
	|
	---sub2/project.pj

Flat Path: `-P /aurora_project/sub2/project.pj --devpath Dev`
This syntax does not work because MKS Integrity attempts to find `/aurora_project/sub2/project.pj` on the main devpath first, and then jumps from there into the development path, but in this case that subproject does not exist in the main devpath.

Source Configuration Path: `-P"#/aurora_project#d=Dev#sub2"`
This syntax works because it instructs MKS Integrity to start with /aurora_project (which does exist in main devpath), then jump to devpath Dev, then move into `sub2`.

Summary: In this scenario, there is no way to specify sub2 using the flat path syntax because it is not on the main development path. You must use the source configuration path syntax to specify `sub2`.

PATH AMBIGUITY

In some cases, the flat path lacks the ability to specify the desired project with no ambiguity.
In the following diagram, the user needs to specify subproject `beta/project.pj` (but only from the location in Tree 1 below).

Tree 1	Tree 2
/aurora_project/project.pj | | ---codebase/project.pj (Development Path Dev1) | | ---beta/project.pj (Development Path Dev2)	/aurora_project/codebase/project.pj [Reg as top-level proj] | | ---beta/project.pj

Flat Path: `-P/aurora_project/codebase/beta/project.pj`
This syntax is ambiguous, because it could be specifying the project on either node. MKS Integrity picks the first project it finds in the registry, which may not be the one desired. The contents of each project are not the same because one is on devpath Dev2, while the other is on the main devpath.

Source Configuration Path:
`-P "#/aurora_project#codebase#beta"`
`-P "#/aurora_project/codebase#beta"`
This syntax method is unambiguous. It is completely clear which beta subproject location is specified.

NOTE: The short form is used because the projects are assumed to be well formed. The long form example would be: `-P #p=/aurora_project/project.pj#s=codebase/project.pj#s=beta/project.pj`
`-P #p=/aurora_project/codebase/project.pj#s=beta/project.pj`

Summmary: In this scenario, the flat path method is ambiguous while the source configurion path is not. The source configuration path can never be ambiguous.

AMBIGUOUS C0-LOCATED SUBPROJECTS

The flat path syntax cannot specify co-located suprojects (subprojects located in same directory).
In the following diagram, two subprojects are located in the same directory.

Tree 1	Tree 2
/aurora_project/source_code/root.pj /colocatedsub.pj	/aurora_project/source_code/base.pj /colocatedsub.pj
NOTE: Project colocatedsub.pj is in the same directory as root.pj but is the subproject of root.pj in hierarchy.	**NOTE:** Project colocatedsub.pj is in the same directory as base.pj but is the subproject of base.pj in the hierarchy.

In this scenario, `root.pj`, `base.pj`, and `colocated.pj` are all in the same directory.

File Path: `-P /aurora_project/source_code/colocatedsub.pj`
This syntax method is ambigious because it is unclear which project the co-located project belongs to. There might be different policies that affect how `colocatedsub.pj` is administered or used.

Source Configuration Path `-P #/aurora_project/source_code/root.pj#s=colocatedsub.pj`
This syntax method clearly specifies a registered project and a subproject where there are co-located subprojects

Summary: In this scenario, the flath path syntax method is unable to specify the co-located subproject. Only the source configuration path syntax can specify the desired path.

Rules for Jumps

When jumping to a specific configuration in a project path, the following rules apply:

- You cannot jump anywhere from a build project
- You can jump from a normal project to a variant only if it is the root of the variant (the project through which the development path was created)
- You cannot jump to a variant if it differs from the closest variant higher in the project hierarchy (if there is a higher variant). When no subprojects are configured as variants in the hierarchy, the closest variant is the variant of the top-level project. When at least one subproject in the hierarchy is configured as a variant, the closest variant is the variant of the lowest configured subproject. This does not include the variant of the subproject on which the jump is specified, if it is currently configured as a variant.

The last two rules are verified based on the type of the parent project. You can always jump to the current configuration of a subproject, even if it violates the rules listed above.

The following provides examples of how jump rules are applied when jumping to a variant. If you had the following project setup:

`/projects/aurora_project/source_code/savings_tool/project.pj`

where `source_code` is a subproject currently configured as `beta_variant` and `savings_tool` is a shared subproject currently configured as `normal`.

The following jump would be allowed:

`-P #/projects/aurora_project#source_code/savings_tool#d=beta_variant`

The following jump would not be allowed:

`-P #/projects/aurora_project#source_code/savings_tool#d=prod_variant`

You can specify a jump to `beta_variant` from the subproject `savings_tool` because it is the same as the variant for `source_code`, and because as a shared subproject it is accepted as the local variant root (the project through which the development path was created). You cannot jump to `prod_variant` because it is different than the variant of `source_code`.

The following jumps would also be allowed:

`-P #/projects/aurora_project#d=SP4#source_code#d=SP4`

`-P #/projects/aurora_project#d=SP4#source_code#d=beta_variant`

The following jump would not be allowed:

`-P #/projects/aurora_project#d=SP4#source_code#d=prod_variant`

You can specify a jump to `SP4` from the subproject `source_code` because it is the same as the variant for `aurora_project`. You can specify a jump to `beta_variant` because `source_code` is currently configured as `beta_variant`. You cannot jump to `prod_variant` because it is different than the variant of `aurora_project`.

Note: If you are using a case-insensitive database repository, you can use case-insensitive keyword-based strings.

TIP: If the path contains a hash character (#), use a second hash character to escape it. For example:

`-P #/projects/C##/aurora_project#d=SP4#source_code#d=SP4`

General Options

Some MKS Integrity configuration management commands share the following general options.

`--devpath=`*path*
identifies the development path of a variant project. This is a label that was associated with a branch of the project by **`si createdevpath`**. Paths that include spaces must be enclosed by quotes. The following characters may not be used in a development path: \n, \r, \t, :, ['], #.

Note: This option is always used in conjunction with the **-P** option and a flat string project path. It cannot be used if you specify a project using a keyword string for the **-P** option. It is also mutually exclusive with the **--projectRevision** and **--sandbox** options.

--cpid=*ID*
--changePackageId=*ID*
identifies a change package that is notified of this action, for example, *1452:1*. Note the following about using this option:

- This option can only be specified if change packages are enabled.
- You must specify this option if you have requested to obtain a lock and your administrator has set up locks to be tracked in change packages.
- You must specify this option if your administrator has made change packages mandatory.
- If your administrator has given you permission, you can bypass mandatory change packages by specifying **--changePackageId=***:bypass*.
- If change packages are enabled but it is not mandatory to specify a change package, or if no change package is applicable, you must specify **--changePackageId=***:none*.

--[no]failOnAmbiguousProject
if you specify the project using a flat string for the **-P** option, this option displays an error message when multiple projects correspond to the specified path.

--filter=*filteroptions*
allows you to select members for all commands that take a list of members, using *filteroptions*, which can be one or more of the following:

archiveshared
selects members that share another member's archive. This option is valid only with the database repository.

attribute*:name[=value]*
selects members based on an attribute name and, optionally, value.

changed *[:working|:sync|:newer|:size|:missing|:newmem|:all]*
selects changed members based on: members with changes to working files, members that are out of sync with the project, members where a newer revision exists in the project, members where the working file size differs, members with a missing working file, members where a new member revision is available, or all changes.

rule *[:memberrevdiffers|:defined|:invalid]*
selects members based on a revision rule filter. **:memberrevdiffers** selects all members for which the rule does not match the member revision. **:defined** selects all members with a revision rule. **:invalid** selects all members for which the rule does not expand to any existing revision.

file*:expression*
selects members with a specific file name. This allows you to specify wild cards for file naming, such as the asterisk (*) to match any number of characters, and the question mark (?) to match a single character. For example, `*.java` or `*RB.properties` would be valid expressions.

caseinsensitivefile*:expression*
selects members with a specific case-insensitive file name. This allows you to specify wild cards for file naming, such as the asterisk (*) to match any number of characters, and the question mark (?) to match a single character. For example, `*.java` or `*rb.properties` would be valid expressions.

frozen
selects frozen members.

label*[:name]*
selects any member whose member revision has the specified label.

anylabel*[:name]*
selects any member that contains a revision that has the specified label.

locked*[:name]*
selects all locked members or those locked by a particular user.

locktype*[:exclusive|:nonexclusive|:any]*
selects members that are locked with the specified lock type. If no lock type or *any* is specified, all locked members are displayed.

state*[:name]*
selects members based on state.

format*[:text|:binary]*
selects members based on storage format.

workingbranch
selects members where the working file is on a branch from a given development path that is not the trunk development path.

Note: This filter applies only to Sandboxes.

deferred*[:add|:addfromarchive|:checkin|:drop|:import|:move|:rename|:updaterevision|:all]*
selects deferred members based on: add, addfromarchive, checkin, drop, import, move, rename, updaterevision, or all operations.

memberonbranch
shows only members that are off the main development trunk.

unresolvedmerges
selects members affected by unresolved merges.

pending*[:add|:addfromarchive|:drop|:import|:movememberfrom|:movememberto|:renamefrom|:renameto|:update|:updaterevision|:all]*
selects pending members based on add, addfromarchive, drop, import, movememberfrom, movememberto, renamefrom, renameto, update, updaterevision, or all operations.

workinprogress
combines the `deferred` (all), `locked` (all), and `changed` (all) filters to select members that are considered work in progress.

sparsecontents
shows only existing working files and deferred operations in a sparse Sandbox.

Using commas between the *filteroptions* serves to build logical "OR" statements between them, allowing you to create powerful filters. You may also specify multiple **--filter=***filteroptions* on the command line, which effectively creates logical "AND" statements between them.

For example, you can resynchronize all modified JAVA files through:

si resync --filter=changed --filter=file:*.java

or you can resynchronize all files with label a or b through:

si resync --filter=label:a,label:b

You can also negate a filter using the ! character.

For example, you can check out all JAVA files that are *not* labelled *Beta* by typing:

si co --filter=file:*.java --filter=!label:Beta

--issueid=*id*
specifies the issue ID that corresponds to the change package that records the changes(s). This option can only be specified if the integration between MKS Source and MKS Integrity is enabled. See your administrator for details on using the MKS Integrity integration.

Note: The terms item and issue refer to the same MKS Integrity object and are indistinguishable. Issue is a term embedded in legacy command and option names; therefore, item and issue are used interchangeably in the CLI documentation.

Note: If you have an issue assigned to you that contains only one open change package, you can specify the issue ID instead of the change package ID.

-P *project*
--project=*project*
specifies the path and name of a project. You can specify the project using a flat string or a keyword string. It is recommended that you use a keyword string, especially when you are writing scripts, since flat strings can be ambiguous as to which project is being specified. Use the following keywords to identify the project.

#=value
#wp=value
#wproject=value
specifies the well-formed project or subproject name. Well-formed project or subproject names end with `project.pj`. For example, `#/aurora_project/source_code`. You do not need to specify a trailing `project.pj`.

#p=value
#project=value
specifies the full name of the project, when it does not end with `project.pj`. For example, `#p=/aurora_project/source_code/root.pj`

#=value
#ws=value
#wsubs=value
specifies the subproject in a well-formed project tree. A well-formed project tree has one subproject per directory. Using this keyword, you can specify several levels of subprojects at the same time. For example, `#/aurora_project/source_code/#applications/savings_tool`. You do not need to specify a trailing `project.pj`.

#s=value
#sub=value
specifies the subproject in a poorly-formed project tree. A poorly-formed project tree has co-located subprojects or subprojects located more than one directory level deep. Using this keyword, you can only specify one subproject for each occurrence of the keyword. For example: `#/aurora_project/source_code/#s=applications/savings_tool/project.pj#s=colocated.pj`.

Note:
The **#s** and **#** keywords do not interpret sub subprojects in the same way. For example, `#/aurora_project#source_code/applications` is not the same as `#/aurora_project#s=source_code/applications/project.pj` but is the same as `#/aurora_project#s=source_code#s=applications/project.pj`

#d=value
#devpath=value
specifies the development path name, for example, `#/aurora_project/source_code/#applications/savings_tool#d=beta_variant`. You can only jump to a variant for a subproject if the subproject is the root of the variant (the project rhough which the development path was created). You cannot jump to a variant if it differs from the closest variant higher in the project hierarchy (if there is a higher variant).

#b=value
#build=value
specifies the number, label or symbolic of the revision, for example, `#/aurora_project/source_code/#applications/savings_tool#b=head`.

#n=
#normal=
specifies that the subproject is a normal subproject. Do not enter a value.

#l=value
#location=value
specifies the absolute path of the target subproject (rather than the configuration path). This keyword can be used for commands where the subproject context is not needed and the subproject is not part of any configuration (**si configuresubproject** and **si sharesubproject**).

Note the following about the use of keywords:

- The order of the keywords is important. Keywords are processed from left to right to build the project specification.
- If you need to specify a `'#'` or `'='` symbol in a keyword value, specify the symbol twice (`'##'`, `'=='`).
- If you are specifying a variant subproject, you must specify its path starting at the root of the variant project (the project through which the development path was created).

--projectRevision=*rev*
identifies a particular revision of a build project.

Note: This option cannot be used if you specify a project using a keyword string for the **-P** option. This option is also mututally exclusive with the **--devpath** option.

-R
--[no|confirm]recurse
controls whether to recursively apply this command to any subprojects; used in all commands which take a list of members.

-r *rev*
--revision=*rev*
uses a specified revision for the member. *rev* can be a valid revision number or a label. You may also facilitate automation with special keyword identifiers, specified using a colon (:) prefix (except for the `state:`,`devpath`, `build` or `link` keywords). Acceptable identifiers are:

:head
identifies the head revision.

:member
identifies the member revision.

:locked
identifies locked revisions.

:master
identifies the member revision in the master project. This option is only applicable to variant projects.

:time:*timestamp*
uses the most recent revision on any branch at the specified timestamp. For example, `-rtime:December 22, 2007 3:33:34 PM GMT-05:00`. MKS Integrity recognizes all current timezones whatever your locale (country), for example, CEST, CET, EDT, PST, or GMT+/-*hours*:*minutes*. The following examples illustrate North American and German timestamps recognized by MKS Integrity:

Example 1: US GMT -5 (where `E` is the day of the week, `M` is the month, `d` is the numerical day of the month, `y` is the year, `h` is the time in hours, `m` is the time in minutes, `s` is the time in seconds, `a` is AM/PM, `z` is the timezone difference from Greenwhich Mean Time.

```
EEEE, MMMM d, yyyy h:mm:ss a z   |  Wednesday, April 28, 2007 3:33:45 AM GMT-05:00
EEEE, MMMM d, yyyy h:mm:ss a z   |  Wednesday, April 28, 2007 3:33:45 AM GMT-05:00
EEEE, MMMM d, yyyy h:mm:ss a     |  Wednesday, April 28, 2007 3:33:45 AM
EEEE, MMMM d, yyyy h:mm a        |  Wednesday, April 28, 2007 3:33 AM
MMMM d, yyyy h:mm:ss a z         |  April 28, 2007 3:33:45 AM GMT-05:00
MMMM d, yyyy h:mm:ss a z         |  April 28, 2007 3:33:45 AM GMT-05:00
MMMM d, yyyy h:mm:ss a           |  April 28, 2007 3:33:45 AM
MMMM d, yyyy h:mm a              |  April 28, 2007 3:33 AM
MMM d, yyyy h:mm:ss a z          |  Apr 28, 2007 3:33:45 AM GMT-05:00
MMM d, yyyy h:mm:ss a z          |  Apr 28, 2007 3:33:45 AM GMT-05:00
MMM d, yyyy h:mm:ss a            |  Apr 28, 2007 3:33:45 AM
MMM d, yyyy h:mm a               |  Apr 28, 2007 3:33 AM
M/d/yy h:mm:ss a z               |  4/28/07 3:33:45 AM GMT-05:00
M/d/yy h:mm:ss a z               |  4/28/07 3:33:45 AM GMT-05:00
M/d/yy h:mm:ss a                 |  4/28/07 3:33:45 AM
M/d/yy h:mm a                    |  4/28/07 3:33 AM
h:mm:ss a z                      |  3:33:45 AM GMT-05:00
h:mm:ss a z                      |  3:33:45 AM GMT-05:00
h:mm:ss a                        |  3:33:45 AM
h:mm a                           |  3:33 AM
EEEE, MMMM d, yyyy               |  Wednesday, April 28, 2007
```

```
MMMM d, yyyy                  |  April 28, 2007
MMM d, yyyy                   |  Apr 28, 2007
M/d/yy                        |  4/28/07
MMM d, yyyy - h:mm:ss a       |  Apr 28, 2007 - 3:33:45 AM
MMM d, yyyy - h:mm a          |  Apr 28, 2007 - 3:33 AM
```

Example 2: Germany CEST (where `E` is the day of the week, `M` is the month, `d` is the numerical day of the month, `y` is the year, `H` is the time in hours, `m` is the time in minutes, `s` is the time in seconds, `'Uhr` is a string literal, `'z` is the timezone (for example, CEST).

```
EEEE, d. MMMM yyyy H.mm' Uhr 'z |  Montag, 26. Juli 2007 22.26 Uhr CEST
EEEE, d. MMMM yyyy HH:mm:ss z   |  Montag, 26. Juli 2007 22:26:29 CEST
EEEE, d. MMMM yyyy HH:mm:ss     |  Montag, 26. Juli 2007 22:26:29
EEEE, d. MMMM yyyy HH:mm        |  Montag, 26. Juli 2007 22:26
d. MMMM yyyy H.mm' Uhr 'z       |  26. Juli 2007 22.26 Uhr CEST
d. MMMM yyyy HH:mm:ss z |  26. Juli 2007 22:26:29 CEST
d. MMMM yyyy HH:mm:ss   |  26. Juli 2007 22:26:29
d. MMMM yyyy HH:mm      |  26. Juli 2007 22:26
dd.MM.yyyy H.mm' Uhr 'z |  26.07.2007 22.26 Uhr CEST
dd.MM.yyyy HH:mm:ss z   |  26.07.2007 22:26:29 CEST
dd.MM.yyyy HH:mm:ss     |  26.07.2007 22:26:29
dd.MM.yyyy HH:mm                |  26.07.2007 22:26
dd.MM.yy H.mm' Uhr 'z   |  26.07.07 22.26 Uhr CEST
dd.MM.yy HH:mm:ss z     |  26.07.07 22:26:29 CEST
dd.MM.yy HH:mm:ss               |  26.07.07 22:26:29
dd.MM.yy HH:mm          |  26.07.07 22:26
H.mm' Uhr 'z            |  22.26 Uhr CEST
HH:mm:ss z              |  22:26:29 CEST
HH:mm:ss                |  22:26:29
HH:mm                   |  22:26
EEEE, d. MMMM yyyy              |  Montag, 26. Juli 2007
d. MMMM yyyy            |  26. Juli 2007
dd.MM.yyyy              |  26.07.2007
dd.MM.yy                        |  26.07.07
MMM d, yyyy - h:mm:ss a |  Jul 26, 2007 - 10:26:29 PM
MMM d, yyyy - h:mm a    |  Jul 26, 2007 - 10:26 PM
```

:timeonbranch:*timestamp*@*branchnumber*
: uses the most recent revision on a specific branch at a specific timestamp. `-rtimeonbranch:`*timestamp* uses the most recent revision on the branch where the member revision currently resides. For example, `-rtimeonbranch:December 22, 2007 3:33:34 PM GMT-05:00`. `-rtimeonbranch:`*timestamp*`@`*branchnumber* uses the most recent revision on the specified branch at the specified timestamp. For example, `-rtimeonbranch:December 22, 2007 3:33:34 PM GMT-05:00@1.5.1.1`. MKS Integrity recognizes all current timezones whatever your locale (country), for example, CEST, CET, EDT, PST, or GMT+/-*hours*:*minutes*. For timestamp examples, see the :time:*timestamp* option.

Note: Updating a revision by timestamp makes the most recent revision at the specified timestamp the member revision.

:memberbranchtip
: identifies the tip revision on the member revision branch.

:working
: identifies the working revision (Sandbox only).

:trunktip
: identifies the tip revision on the trunk.

state:*statename*
: identifies the state, for example, ***Beta***. This option is useful when you want to select revisions in a project that are in a specific state.

For each project member, MKS Integrity searches from the member revision on the development path to the root of the archive to find a revision that corresponds to the specified state. If the member revision is on a branch, MKS Integrity starts from the tip revision and searches to the root of the archive; other branches in the archive are ***not*** searched. If no revision on the development path matches the specified state, the command fails, stating "Revision does not exist."

devpath: *devpathname*
: identifies the member revision in the specified development path. This keyword only operates on member commands.

build: *revisionnumber*
: identifies the member revision in the build project with the specified build revision number, which must be a valid project checkpoint number or project label in which a given member is contained. Must specify a registered project. This keyword only operates on member commands.

:rule
: identifies a rule defined with the `si setmemberrule` command.

link:p=*project*[::d=*devpath*][::m=*member*][::recurse] [::b=*buildrevnumber*]
: allows you to set the member revision to whatever is the member revision for the corresponding member in a specific external project configuration (normal, variant, build). Links the project that the member belongs to (the target project) with the master project where:

project is the master project

devpath is the development path for the master project

member is a member in the target project. If not provided, the project is searched for a member with the same backing archive. If recurse is specified, the search is recursive throughout the subprojects. There must be exactly one backing archive for each member.

A possible application is to update all members to the same revision, even if they don't have the same backing archive.

-S *sandbox*
--sandbox=*sandbox*
: specifies the location of a Sandbox. In some cases, the commands that take this option do something with the Sandbox contents themselves. In other cases, specifying the Sandbox location is simply a way to locate, or "point to", the corresponding project file. This option is mutually exclusive with **-P** *project*|**--project =***project*.

Note:
: Locations that include spaces must be enclosed by quotes.

Universal Options

The following universal options apply to all commands (**si, im, aa, integrity**).

--user=*name*
: identifies the user to use for connecting to the Integrity Server.

--hostname=*server*
: identifies the name of the host server where the Integrity Server is located.

--password=*password*
: identifies the password to use for connecting to the Integrity Server.

--port=*number*
: identifies the port on the host server where the Integrity Server is located.

--[no]batch
: controls batch mode. Batch mode forces the application to process commands without prompting for responses.

--cwd=*directory*
: acts as if the command is executed in the specified directory. In particular, any files and members in the selection are treated as being relative to that directory.

Suppose you are working in the `c:\sandbox` directory and you want to issue the check out command so that the implicit Sandbox selection will work in a subdirectory, rather than having to specify the complete path for subdirectory Sandbox names. You could use the **--cwd** option to do this, for example:

si co --cwd=./demoapp/controls demoappctrl.c

makes MKS Integrity work in the `c:\sandbox\demoapp\controls` directory and follows implicit Sandbox selection rules from there to find the appropriate Sandbox, then checks out the `demoappctrl.c` file.

-F *file*
--selectionFile=*file*
provides an alternative way to specify the selection. The specified *file* is a text file containing a list of file names, members, projects, or Sandboxes, one per line. The command operates on all the listed files.

Note:
Be careful to avoid duplications. In some cases if a file, member, project or Sandbox is listed twice in the *file*, the command may report an error.

--forceConfirm=*[yes|no]*
-N
--no
-Y
--yes
controls the responses of either "yes" or "no" to all prompts. Specifying "yes" or "no" can be an easy way to accomplish the same thing as specifying other command options with *[no|confirm]* prefixes, for example the **--[no|confirm]overwriteChanged** option in the **si co**, **si resync**, and **si revert** commands. Specifying **--yes** or **--no** accomplishes the same thing for **--overwriteChanged** and **--nooverwriteChanged**, but further responds "yes" or "no" to all other questions asked.

Note:
Be careful to use specific options if you want variations in your responses to prompts. The **--yes** and **--no** options in particular are wide-ranging types of responses and should be used only in rare circumstances.

-g
--gui
allows user interaction to happen through the GUI (graphical user interface).

--width
controls the width in pixels of the graphical user interface.

--height
controls the height in pixels of the graphical user interface.

-x
specifies the x location in pixels of the graphical user interface window.

-y
specifies the y location in pixels of the graphical user interface window.

--quiet
controls the status display to silence most information messages.

--settingsUI=*[gui|default]*
controls the GUI for command options.

--status=*[none|gui|default]*
controls the status display.

-?
--usage
shows usage for the command.

DIAGNOSTICS

See the **diagnostics** reference page for possible exit status values.

SEE ALSO

Miscellaneous:
ACL, **diagnostics**, **preferences**

preferences

preferences applicable to MKS Integrity setprefs and viewprefs commands

DESCRIPTION

The **setprefs** and **viewprefs** commands refer to the same group of commands and preference keys for the MKS Integrity component you are configuring (**si**, **im**, **aa**, **integrity**). This reference page is provided as a guide to the preferences you can configure. To see each component's specific keys, simply append the command to the **viewprefs** command. For example,

```
si viewprefs --command=co
```

displays the following output:

```
co: mergeType
co: onMergeConflict
co: allowImplicitIdentification
co: branchIfVariant
co: changePackageID
co: cwd
co: failOnAmbiguity
co: filter
co: forceBranch
co: forceConfirm
co: includeFormers
co: includePending
co: issueID
co: keywordExpand
co: lock
co: lockOnFreeze
co: mdiMode
co: merge
co: mks.useSystemTrayIcon
co: onLock
co: overwriteIfChanged
co: overwriteIfDeferred
co: recordAsWorkInProgress
co: recurse
co: restoreTimestamp
co: revision
co: sandboxName
co: selectionFile
co: updateMemberRev
co: command.batchMode
co: sandbox.allowProjectSelection
co: sandbox.allowSandboxSelection
co: sandbox.forceSandboxSelectionIfAmbiguous
co: server.credential
co: server.hostname
co: server.port
co: server.user
co: socksProxyHost
co: socksProxyPort
co: status.popupDelay
```

Preference Keys

The preference keys you can specify for the above commands are often similar, and some are global preferences such that when you change it for one command, it changes for all. The following are some common preference keys:

allowImplicitIdentification
: controls whether to allow the implicit identification and selection of Sandboxes. This means that you can use commands without explicitly specifying a **-s** Sandbox option, and MKS Integrity determines the Sandbox to operate on based on the directory you're working in. Valid values are *true* or *false*.

cwd
: identifies a working directory for the command. In particular, any relative files and members in the selection are treated as being in that directory.

command.batchMode
: a global key that controls batch mode. Batch mode forces the application to process commands without prompting for responses. Valid values are *false*, *true*.

developmentPath
: specifies a particular development path to use with the command all the time.

filter
: specifies a filter to use with the command all the time.

forceConfirm
: specifies whether to force the application to request confirmation for the particular command.

includeFormers
specifies whether to include members that have been dropped from the project.

projectName
specifies a particular MKS Integrity project to work with all the time.

projectRevision
specifies a particular MKS Integrity configuration management project checkpoint to work with all the time.

recurse
controls whether to recurse into subprojects.

sandbox.allowProjectSelection
a global key that controls whether to allow MKS Integrity configuration management project selection. Valid values are *false*, *true*.

sandbox.allowSandboxSelection
a global key that controls whether to allow Sandbox selection. Valid values are *false*, *true*.

sandboxName
specifies a particular Sandbox to be used all the time.

selectionFile
specifies a file that lists member files to work with.

server.credential
a global key that identifies the credential, or password, for logging into the Integrity Server.

server.hostname
a global key that identifies the hostname for the Integrity Server.

server.port
a global key that identifies the port for the Integrity Server.

server.user
a global key that identifies the user name for logging into the Integrity Server.

socksProxyHost
a global key that identifies the SOCKS proxy host.

socksProxyPort
a global key that identifies the SOCKS proxy port.

status.popupDelay
a global key that controls the duration, in milliseconds (ms), that the commands status appears in the graphical user interface or the command line interface.

SEE ALSO

Miscellaneous:
diagnostics, options

rcsedit

MKS Integrity editing facilities for descriptions and log messages

DESCRIPTION

Whenever MKS Integrity asks you to enter a description or log message, you may make use of editing commands to help you enter the message. Editing commands must be on a separate line and the first character of the line must be a tilde `~` character. Here are the recognized commands:

`~?`
: Displays a summary of editing commands.

`~d`
: Restore message from *dead.let* file.

`~e`
: Writes the current message to a temporary file and invokes a text editor to edit that file. The name of the editor is taken from the ***EDITOR*** environment variable. If there is no such variable, **`ed`** is used. When you are finished editing the message, write it back out to the temporary file and quit the editor. You return to MKS Integrity.

`~p`
: Displays the current text of the message.

`~q`
: Copies the current message to a file named *dead.let* and quits.

`~r` *filename*
: Reads the contents of the given file and appends them to the current message.

`~v`
: Similar to `~e`, except that it invokes the visual editor identified by the ***VISUAL*** environment variable. The default is **`vi`**. (You can only use **`vi`** if you have a UNIX system or the MKS Toolkit.)

`~w` *filename*
: Writes the current message to the specified file.

`~!` *command*
: Executes the given system command.

If a line begins with a tilde, MKS Integrity assumes it is an editing command. If you want to enter a text line that begins with a tilde, type two, as in

```
~~ This line begins with a single tilde.
```

EXAMPLES

```
These are the changes I have made:
~r changes
.
```

reads in text from a file named `changes` and adds that text to the message being composed. The period (`.`) to mark the end of the message is still required.

ENVIRONMENT VARIABLES

The MKS Integrity editing facilities use the following environment variables:

EDITOR
: contains the name of the editor invoked by the `~e` command.

VISUAL
: contains the name of the editor invoked by the `~v` command.

PORTABILITY

These facilities are extensions to traditional implementations of RCS and are non-portable.

SEE ALSO

Commands:

`si ci`

Product Notices

U.S. Government Rights. The software and documentation (the "Software and Documentation") are "commercial items" developed exclusively at private expense, consisting of "commercial computer software" and "commercial computer software documentation" as such terms are defined or used in the applicable U.S. acquisition regulations. If you are the U.S. Government or any agency or department thereof the Software and Documentation are licensed hereunder (i) only as a commercial item and (ii) with only those rights as are granted to all other end users pursuant to the terms and conditions of the Software License Agreement ("Agreement") accompanying the Software. The Software or Documentation shall not be used, duplicated, or disclosed in any way not specifically permitted by the terms of the Agreement. Nothing in the Agreement requires MKS to produce or furnish technical data for or to the U.S. Government or any agency or department thereof.

This product contains Qooxdoo version 0.7.4. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. Copyright © 1991, 1999 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

This product contains JSON-RPC JavaScript client software v. 1.0. Copyright (c) 2003-2004 Jan-Klaas Kollhof. Copyright (c) 2005 Michael Clark, Metaparadigm Pte. Ltd. This code is based on Jan-Klaas' JavaScript lait library (jsolait). Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product contains JIDE stock icons. Copyright (c) 2002-2008 JIDE Software, Inc. All rights reserved.

This product contains Apache Derby v. 10.3.3 software developed by The Apache Software Foundation (http://www.apache.org/). Copyright 2004-2008 The Apache Software Foundation. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Portions of Derby were originally developed by International Business Machines Corporation and are licensed to the Apache Software Foundation under the "Software Grant and Corporate Contribution License Agreement",informally known as the "Derby CLA". The following copyright notice(s) were affixed to portions of the code with which this file is now or was at one time distributed and are placed here unaltered. (C) Copyright 1997,2004 International Business Machines Corporation. All rights reserved. (C) Copyright IBM Corp. 2003. The portion of the functionTests under 'nist' was originally developed by the National Institute of Standards and Technology (NIST), an agency of the United States Department of Commerce, and adapted byInternational Business Machines Corporation in accordance with the NIST Software Acknowledgment and Redistribution document at http://www.itl.nist.gov/div897/ctg/sql_form.htm.

This product contains TreeTableModel.java, AbstractTreeTableModel.java, TreeTableModelAdapter.java, AbstractCellEditor.java and JTreeTable.java. Copyright 1997, 1998 Sun Microsystems, Inc. All Rights Reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistribution in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of Sun Microsystems, Inc. or the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission. This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES OR LIABILITIES SUFFERED BY LICENSEE AS A RESULT OF OR RELATING TO USE, MODIFICATION OR DISTRIBUTION OF THIS SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.You acknowledge that this software is not designed, licensed or intended for use in the design, construction, operation or maintenance of any nuclear facility.

This product contains Xinha which is licensed under the htmlArea License (based on BSD license). Copyright (c) 2002-2004, interactivetools.com, inc. Copyright (c) 2003-2004 dynarch.com. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1) Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2) Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3) Neither the name of interactivetools.com, inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT

OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product contains JGo software. Copyright (c) 1999-2004 Northwoods Software Corporation. All rights reserved.

This product contains NLog 1.0. Copyright (c) 2004-2006 Jaroslaw Kowalski (jaak@jkowalski.net). All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of Jaroslaw Kowalski nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product contains Runtime Modules of IBM(R) 32-bit (or 64-bit) SDK for AIX(TM), Java(TM) Technology Edition, Version 6
(c) Copyright Sun Microsystems Inc, 1992, 2008
(c) Copyright IBM Corporation, 1998 - 2008
(c) Copyright The Apache Software Foundation, 1999-2007
All Rights Reserved.

This product contains Apache Jakarta Commons HttpClient software developed by The Apache Software Foundation (http://www.apache.org/). Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product contains Apache Tomcat v. 5.5.17 software developed by The Apache Software Foundation (http://www.apache.org/). Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product includes the Extreme! Computing XPP3-1.1.3.2. software developed by the Indiana University Extreme! Lab (http://www.extreme.indiana.edu/). Copyright (c) 2002 Extreme! Lab, Indiana University. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Indiana University Extreme! Lab (http://www.extreme.indiana.edu/)."; Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Indiana University" and "Indiana University Extreme! Lab" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact http://www.extreme.indiana.edu/. 5. Products derived from this software may not use "Indiana Univeristy" name nor may "Indiana University" appear in their name, without prior written permission of the Indiana University. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHORS, COPYRIGHT HOLDERS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product contains Glazed Lists software developed by publicobject.com and O'Dell Engineering. Copyright © 2003-2006, publicobject.com, O'Dell Engineering Ltd. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. Copyright © 1991, 1999 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

This product contains Bean Scripting Framework software version 2.2 developed by The Apache Software Foundation (http://www.apache.org/). Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product contains IBM Toolbox for Java – JT Open software. Copyright © up to and including 2009, International Business Machines Corporation ("IBM") and others. All Rights Reserved. Licensed under the IBM Public License Version 1.0 (the "License"); you may not use this file except in compliance with the License. See the License at http://www-128.ibm.com/developerworks/library/os-ipl.html for the specific language governing permissions and limitations under the License. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,

DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. IN NO EVENT SHALL THE CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product contains ICU v. 1.8.1. Copyright © 1995-2008 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE. Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

This product contains Java BEEP Core Version 0.9.08 developed by The Apache Software Foundation (http://www.apache.org/) licensed under the Apache License, Version 1.1. Copyright (c) 1999-2001 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowlegement may appear in the software itself, if and wherever such third-party acknowlegements normally appear 4. The names "The Jakarta Project", "Commons" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product contains Java BEEP Core Version 0.9.08. Blocks Public License Copyright © 2000-2001, Invisible Worlds, Inc. All Rights Reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name, trademarks, or trade names of Invisible Worlds, Inc., nor the names, trademarks, or trade names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INVISIBLE WORLDS OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Cryptix General License Copyright © 1995, 1997, 1998, 2000. The Cryptix Foundation Limited. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistribution of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. THIS SOFTWARE IS PROVIDED BY THE CRYPTIX FOUNDATION LIMITED AND CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE CRYPTIX FOUNDATION LIMITED OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This package is a SSLv3/TLS implementation written by Eric Rescorla (ekr\@rtfm.com) and licensed by Claymore Systems, Inc. Redistribution and use in source and binary forms, with or

without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by Claymore Systems, Inc. Neither the name or Claymore Systems, Inc. nor the name of Eric Rescorla may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product contains Java Service Wrapper v.3.2.3 developed by Tanuki Software. Copyright © 1999, 2006 Tanuki Software Inc. Permission is hereby granted, free of charge, to any person obtaining a copy of the Java Service Wrapper and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sub-license, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE. Portions of Java Service Wrapper have been derived from source code developed by Silver Egg Technology under the following license: Copyright © 2001 Silver Egg Technology. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

This product contains JavaScript for Java which is subject to the Netscape Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/NPL/. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is Rhino code, released May 6, 1999. The Initial Developer of the Original Code is Netscape Communications Corporation. Portions created by Netscape are Copyright (C) 1997-1999 Netscape Communications Corporation. All Rights Reserved. Contributor(s): Norris Boyd

This product contains JBoss Application Server version 4.2.0., JBoss AOP version 1.5.5.GA and JBoss Web version 2.0.0.GA. Copyright © 2006, Red Hat Middleware LLC, and individual contributors as indicated by the @author tags. See the copyright.txt file in the distribution for a full listing of individual contributors This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. Copyright © 1991, 1999 Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

This product contains JFree Chart version 1.0.1. Copyright 2000-2006, by Object Refinery Limited and Contributors. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. Copyright © 1991, 1999 Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

This product contains jTDS Drivers version 1.2.2 for MS SQL Server. This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version. Copyright © 1991, 1999 Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

This product contains software developed by JXML, Inc. and its contributors:http://www.jxml.com/mdsax/contributors.html Copyright (c) 1998, 1999, JXML, Inc. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. All product materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by JXML, Inc. and its contributors: http://www.jxml.com/mdsax/contributors.html. Neither name of JXML nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY JXML, INC. AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.

IN NO EVENT SHALL JXML OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product contains Log4J software v. 1.2.15 developed by The Apache Software Foundation (http://www.apache.org/). Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0. Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

This product contains NSPR Library software. The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is the Netscape Portable Runtime (NSPR). The Initial Developer of the Original Code is Netscape Communications Corporation. Portions created by Netscape are Copyright (C) 1998-2000 Netscape Communications Corporation. All Rights Reserved.

This product contains OpenSSL software and is licensed under the terms of the OpenSSL License and the SSLeay License. Copyright © 1998-2008 The OpenSSL Project. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit. (http://www.openssl.org/)" 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact openssl-core@openssl.org. 5. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project. 6. Redistributions of any form whatsoever must retain the following acknowledgment: "This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/)" THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE OpenSSL PROJECT OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com). Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved. Original SSLeay License. Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved. This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be left out if the routines from the library being used are not cryptographic related. 4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement: "This product includes software written by Tim Hudson (tjh@cryptsoft.com)" THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product includes Sapia Ubik software developed by Sapia Open Source Software. Copyright © 2002, 2003 Sapia Open Source Software (http://www.sapia-oss.org/). All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by Sapia Open Source Software Organization (http://www.sapia-oss.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. The names "Sapia", "Sapia Open Source Software" and "Sapia OSS" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact info@sapia-oss.org. Products derived from this software may not be called "Sapia", nor may "Sapia" appear in their name, without prior written permission of Sapia OSS. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE SAPIA OSS ORGANIZATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF

SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This product contains software code that is Copyright 2007, Microsoft Corporation. All rights reserved.

This product contains XML Parsing Library for C version 2.6.32. Copyright (C) 1998-2003 Daniel Veillard. All Rights Reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software. THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE DANIEL VEILLARD BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.