---

help

This is Regi, the command line tool to access the HANA Repository.

Usage:

```
regi ACTION [WHAT] [OPTION]...
```

All commandline parameters are case-insensitive.

Actions:

- Workspaces:
  - create workspace|ws
  - status
  - track allPackages
  - track package
  - unTrack allPackages
  - unTrack package
- Objects:
  - activate
  - checkOut
  - commit
  - discard
  - mergeTool
  - push
  - resolve
  - revert
  - takeOver
- Active/inactive objects:
  - delete allInactiveWorkspaces|allIWS
  - delete inactiveWorkspace|iWS
  - list activeObjects|aOs
  - list inactiveObjects|iOs
  - list inactiveWorkSpaces|iWS
- Object history:
  - list history
  - show object
- Packages and Delivery Units:
  - assign

    - show changeTrackingStatuslcts
- Miscellaneous:
    - help
    - version

Call:

- `regi help [ACTION] [WHAT] [--browse|--html|--txt]`

for help on a particular action.

"--browse" displays all help texts in the Browser specified in the environment variable "BROWSER" or "firefox" (default).

"--html" and "--txt" display the found help texts in the shell, in HTML and plain text format, respectively.

Topics:

- activationMode (activation modes)
- arguments (generic commandline arguments)
- changeTracking (Change Tracking)
- changeLog (Regi Change Log/Version History)
- cleanUp (clean up the Repository/Regi workspaces)
- conflicts (conflict resolution)
- deliveryUnits (delivery units)
- environment (generic environment variables)
- faq (frequently asked questions)
- hdbuserstore (the HANA client tool "hdbuserstore")
- lockFile (the lock file)
- logs (log files)
- mergeToolConfig (merge tool configuration)
- nutshell (Regi in a nutshell)
- objects (objects)
- packages (packages)
- regiignore (".regiignore" files)
- statusCalculation (local object status calculation)
- timeStamps (timestamps)
- traces (trace files)

Call:

- `regi help TOPIC [--browse|--html|--txt]`

for help on a particular topic.

Search the Regi help system:

Call:

- `regi help TERM1 [TERM2] -s [--browse|--html|--txt]`

to search the Regi help system.

Examples:

- `regi help`
  Display the main help page.

- `regi help --browse`
  Display the complete help in the Browser.

- `regi help activate`
  Display the help for "regi activate".

- `regi help create workspace`
  Display the help for "regi create workspacelws".

- `regi help resolve conflicts -s`
  Display the list of help texts including the terms "resolve" and "conflicts" in the shell.

- `regi help resolve conflicts -s --html`
  Display all help texts including the terms "resolve" and "conflicts" in the shell (in HTML format).

- `regi help resolve conflicts -s --txt`
  Display all help texts including the terms "resolve" and "conflicts" in the shell (in plain text format).

---

activationMode

Activation modes

The activation mode for regi activate, regi import DU and regi transport DU can be parametrized with the parameter --activationMode=N, where N is one of:

- `0`

  Default for regi activate.

  Name: ACTIVATION_CASCADE_ONE_PHASE

  Use case: Developers usually want the activation of their changes to fail as soon as one or more errors occur in the activation of the objects to be activated or in the activation of the affected objects.

  Behavior: Only commit any changes if the given objects are all OK, and the affected objects are also all OK.

  Pseudo code:

  ```
  {
    Source-Move
  ```

```
      GenerateActivatedObjects
      If no error so far: RegenerateAffectedObjects
    }(If no error: commit; else: rollback)
```

- `1`

  Not supported anymore by the Repository.

- `2`

  Not supported anymore by the Repository.

- `3`

  Name: ACTIVATION_CASCADE_TWO_PHASES

  Use case: Developers who want to activate their error-free changes despite of errors in the activation of the affected objects - especially if these errors are not related to the activated changes.

  Behavior: If the given objects are all OK, they are activated, and the affected objects are regenerated in a separate transaction.

  Pseudo code:

  ```
  {
    Source-Move
    GenerateActivatedObjects
  }(If no error: commit; else: rollback and exit)
    RegenerateAffectedObjects
  }(commit)
  ```

- `4`

  Default for regi import DU and regi transport DU.

  Name: ACTIVATION_IMPORT_MODE

  Use case: Administrators importing a DU usually want to get the import go through even if some of the imported objects have errors. Usually, the problems need to be fixed by others. This is much easier if the crucial objects are in active state, since then the inactive source need not be moved to someone else's workspace. The erroneous objects can easily be found since they are be marked as "broken" in the repository table.

  Behavior: Always commit, even if there are errors.

  Pseudo code:

  ```
  {
    Source-Move
    GenerateActivatedObjects
    RegenerateAffectedObjects
  }(commit)
  ```

- `5`

  Name: ACTIVATION_CHECK_ONLY

  Use case: Test object activation.

  Behavior: Everything is rolled-back at the end.

```
        {
          Source-Move
          GenerateActivatedObjects
          RegenerateAffectedObjects
        }(rollback)
```

.

---

arguments

Generic commandline arguments:

Most generic commandline arguments are matched by environment variables. Environment variables can be used to configure certain aspects of Regi permanently. See "regi help environment".

- Output:
    - `--debug|-d`
      Debug output.
    - `--debugPaths`
      Debug output for path handling.
    - `--debugRegiignore|--debugRi`
      Debug output for ".regiignore" files.
    - `--jsonRequest|--jReq`
      Display requests from Regi to the Repository.
    - `--jsonResponse|--jResp`
      Display responses from the Repository to Regi.
    - `--log|-l`
      Create log file in "._SYS_REGI_settings/logs".
    - `--noCommandLog`
      Do not create command log file in "._SYS_REGI_settings/logs".
    - `--perfTrace|--perf`
      Create Regi performance trace.
    - `--verbose|-v`
      Verbose output.
    - `--trace|-t`
      Create trace file in "._SYS_REGI_settings/logs".
- Connection:
    - `--key=KEY`
      Specify secure store key for server connection. Keys can be created with the HANA client tool "hdbuserstore".
    - `--hostName|--host=HOST:SQLPORT`
      Specify Host and SQL port for the server connection (if no key provided). SQLPORT = 30000 + 15 + 100*INSTANCE, e.g. 34215 for instance 42. For STFK connections, use the local proxy (e.g. "localhost:5001").

    - `--userName|--user=USER`
      Specify user for the server connection (if no key provided).
    - `--password|--passwd=PASSWORD`
      Specify password for the server connection (if no key provided).
    - `--readFromStdIn|--stdIn`
      Read password from standard input if it is not provided.
- Status calculation:
    - `--fast`
      Use timestamps and hashes for status calculation. Fast, but not recommended for writing scripts with Regi. Recommended for non-scripted Regi use.
    - `--noCompare|--nc`
      Compare neither timestamps nor hashes in status calculation, i.e., list all files as changed, even if they have not changed. This mode is useful for forced re-generation of objects.
    - `--veryFast`
      Only use timestamps for status calculation. Very fast, but not recommended for writing scripts with Regi.
- Other:
    - `--browse`
      Show the help texts in the Browser specified in the environment variable "BROWSER" or "firefox" (default).
    - `--ignoreLockFile|--ignoreLock`
      Ignore the Regi workspace lock file. See "regi help lockFile".
    - `--inactiveObjects|-i`
      List the inactive objects in "regi status".
    - `--lineEnd=LINEEND`
      Specify line end mode for writing files ("regi checkout"): "LOCAL": local, "UNIX": force LF, "WIN": force CRLF. Default: "LOCAL".
    - `--noBulkRead`
      Do not use bulk read; instead read the objects one at a time ("regi checkout").
    - `--noConfirm`
      Do not ask for confirmation of interactive merges (assume "y" if the merge tool has modified the file, "n" otherwise). ("regi mergetool", "regi resolve -m", "regi push -m").
    - `--noMultiImport`
      Import the DUs one after the other (do not use multi DU import) ("regi import du", "regi transport du").
    - `--noTips`
      Do not show tips, e.g. for conflict resolution.
    - `--simulate`
      Use simulation mode (read-only; neither writes to the file system nor to the Repository).
    - `--workSpace|--ws=WORKSPACE`
      Set workspace.

---

changeLog

Regi Change Log/Version History (important new features/fixes):

1.1.3 (April 29, 2016)

- Fixed startup crash.

1.1.2 (June 16, 2015)

- UTF-8 to Latin-1 conversion did not properly detect unsupported UTF-8 start and second bytes. Improved error message for invalid UTF-8 characters that cannot be converted to Latin-1.

1.1.1 (July 1, 2014)

- Added information about the specification of empty workspaces to online documentation.

1.1.0 (June 30, 2014)

- Added support for UTF8 user names and passwords.

1.0.19 (June 11, 2014)

- Tiny update for online documentation.

1.0.18 (June 10, 2014)

- More debug/trace information for checkout.

1.0.17 (June 6, 2014)

- Update for online documentation (FAQ) - how to use Regi on an STFK connection (CSS). Also updated documentation of create ws.

1.0.16 (May 19, 2014)

- Relaxed file name check (allow '%' for UI5 build).

1.0.15 (May 15, 2014)

- Update for online documentation (FAQ).

1.0.14 (April 16, 2014)

- Coverity fixes.

1.0.13 (April 9, 2014)

- Documentation of the history of statusCalculation.

1.0.12 (March 19, 2014)

- Support for checkIsDeliveryUnitBroken (show dustatus).
- Added "object_status", "change_number" and "released_at" to *.regiMeta files and to list aos.

1.0.11 (February 19, 2014)

- Support for native deletePackageRecursively.

1.0.10 (February 14, 2014)

- Added missing help for delete allInactiveWorkSpaces.

1.0.9 (February 13, 2014)

- Improved help text for activation modes.

1.0.7 (February 5, 2014)

- Added this change log to the Regi online help.

1.0.6 (February 3, 2014)

- Improved robustness of checkout in case packages are deleted but inactive objects remain.

1.0.5 (January 31, 2014)

- Improved robustness of discard (completely clears directories before attempting to delete them).

1.0.4 (January 27, 2014)

- Added support for unofficial import option determineXrefs.
- Important Fixes for create package and delete package (would create/delete wrong directories if called from a sub directory inside the workspace).

1.0.3 (January 16, 2014)

- Extended show object to show active and inactive versions in addition to historical versions.

1.0.2 (January 14, 2014)

- Important fix for Windows default browser retrieval.

1.0.1 (January 13, 2014)

- Added list activeObjects.

1.0.0 (December 26, 2013)

- Final polishing of the help system.

0.46.0 (December 25, 2013)

- Default status calculation reset to hash-only.

0.45.0 (December 17, 2013)

- Added support for show exportVersions and export option --exportVersion=N.

0.44.8 (December 17, 2013)

- export languages can now also be called without any language parameter (=all available languages).

0.44.6 (November 29, 2013) (HANA REVISION 71)

- Fix for --norec bug (DO NOT USE --norec prior to this version).

0.44.5 (November 26, 2013)

- Improved output of list change.
- show change now shows change meta data and contributors, list change only the entries.

0.44.0 (November 6, 2013)

- Added change tracking documentation.

0.43.1 (October 30, 2013)

- Better package parsing (correctly parses slash-separated packages also outside workspaces).

0.43.0 (October 30, 2013)

- First working version of change tracking.

0.42.2 (October 18, 2013)

- Added flat DU/inactive workspace listing to list dus and list iws if vendor/inactive workspace user provided (for shell scripting).

0.42.0 (October 15, 2013)

- Added stronger force mode for delete package (also deletes the contained active objects).
- Added stronger force mode for push.

0.41.2 (October 14, 2013)

- Added parameter synonym -f for --force.
- Fix for standard input password input with white space (create ws).

0.41.0 (October 4, 2013)

- Keeps unknown fields in .regiConfig, .regiStatus for HANA Studio Team Provider interoperability.

0.40.1 (September 30, 2013)

- Extended transport to support multiple DU parameters.
- Important fix for symbolic links (Linux).

0.40.0 (September 23, 2013)

- Extended help to support HTML and ASCII at the same time (HTML tags are stripped for ASCII text output).

0.39.0 (September 21, 2013)

- Added inactive objects parameter for checkout, commit, discard, mergetool, push, resolve, status for consistency.
- Added checkout of historical versions (--hv=N).
- Added transport command.

0.36.0 (September 16, 2013) SP7 VERSION (REVISION 70)

- Important fixes for checkout, discard, .regiignore, takeover, version.

0.35.0 (September 15, 2013)

- First complete version of new online help.
- Much improved .regiignore handling.

0.34.1 (September 12, 2013)

- Extended create package with new meta data inheritance.

0.34.0 (September 10, 2013)

- New help system: First version.

0.32.0 (August 30, 2013)

- Support for true multi import (new default).
- Multiple DU parameters for undeploy.

0.31.0 (August 29, 2013)

- New declarative command-line parser.

0.30.0 (August 6, 2013)

- Added support for bulk read (new default) for checkout (should be much faster on high-latency remote connections).

0.27.0 (July 30, 2013)

- Better handling of deletions (e.g. revert) by keeping deleted files in the .regiStatus file after committing.

0.26.0 (July 25, 2013)

- Vastly improved package/path resolution.
- status is now relative to the current directory.
- Added support for providing both packages and objects as parameters simultaneously.
- Command dispatch refactored.

0.24.0 (July 12, 2013)

- Added takeover.
- activate and checkout refactored.

0.22.1 (July 4, 2013)

- assign and unassign put back in.
- Important fix for commit (overwrite problem).
- checkout now also checks out empty packages.

0.21.0 (July 2, 2013)

- Added support for --force option for delete package.
- Much better conflict resolution guidance (tips).
- Support for all DU meta data attributes (including caption, lastUpdate, sp_PPMS_ID, ach).

0.20.0 (June 25, 2013)

- Lots of refactoring (local workspace handling plus X).
- Added base version to .regiStatus.
- Improved status command (more information).

0.19.1 (June 12, 2013)

- Added support for trace files.
- Important fix for commit bug with suffixless files.

0.18.2 (June 7, 2013)

- Added push compound command.
- Made environment variables and command-line arguments consistent.

- First full help completed.

0.17.2 (May 27, 2013)

- Improved update package and update DU.
- assign and unassign removed (update package can do the same).

0.17.1 (May 22, 2013)

- Fix for checkout --force (does not delete local-only files anymore).

0.17.0 (May 22, 2013)

- Added --historical parameter for checkout.

0.16.0 (May 18, 2013)

- Added discard.

0.15.3 (May 14, 2013)

- create ws only with explicit workspace argument.
- Unicode support for filenames on Windows.

0.15.0 (April 27, 2013)

- Re-implementations of checkout and revert: first version.

0.14.1 (April 13, 2013)

- Re-implementation of commit: first version.

0.14.0 (March 27, 2013)

- Re-implementation of status: first version.
- New modular status calculation.
- Set default for status calculation to time stamps plus hash (only compare hashes if time stamps differ).

0.13.3 (March 25, 2013) SP6 VERSION (REVISION 60-69)

- Removed unnecessary python trace functionality.

Beta01 SP3 VERSION (REVISION 30)

- Very first version of Regi.

---

changeTracking

Change Tracking:

Concepts

Starting with HANA SP7, the Repository supports "Change Tracking" for more advanced transport control. Change Tracking (CT) can be enabled or disabled for a HANA instance. If CT is enabled, each activated object is assigned to a so-called "change". A change contains arbitrary many objects which are called "change entries".

A change first has the "change status" "open", and can then either be "released" or deleted. This allows to control transports in a more advanced fashion than before: It is possible to restrict delivery unit exports to only those objects contained in released changes. An object can only be contained in at most one open change at a time.

A change also contains arbitrary many "contributors", i.e., participating users. Each contributor of a change has to set his/her status to "validated" before a change can be released.

A change is identified by a so-called "change ID". The change ID is composed of the HANA server SID (e.g. "HDB"), two forward slashes ("//") and a unique "change number". For example, the change ID "HDB//4711" is composed of the HANA server SID "HDB" and the change number "4711". Regi allows to specify change IDs either in full ("HDB//4711"), with the SID in lower case ("hdb//4711"), without the forward slashes ("hdb4711"), or, most conveniently, by just the change number ("4711").

Change Tracking Status

CT is disabled by default. To use it, it must be enabled explicitly using "regi set changeTrackingStatus|cts". CT cannot be disabled if there are still open changes. It is possible disable CT anyway with the parameter "--force|-f". You can show the change tracking status with "regi show changeTrackingStatus|cts".

Examples:

- `regi show cts`
  Shows change tracking status.
- `regi set cts enabled`
  Enables change tracking.
- `regi set cts disabled`
  Disables change tracking.
- `regi set cts disabled -f`
  Disable change tracking even if there are still open changes.

Activation

Upon the activation of inactive objects, it is possible to either explicitly specify the change ID of an open change, or let Regi implicitly determine a suitable change ID. To this end, Regi first retrieves the "affected changes" of the specified objects, i.e., those open changes which contain a subset of the set of specified objects. This leads to three possible cases:

1. There is no open change containing any of the specified objects. In this case, a new change is created implicitly and the activated objects are assigned to this change.

2. There is precisely one open change containing a subset of the specified objects, i.e., no other open change contains any of the specified objects. In this case, the activated objects are assigned to this change and no new change is created.
3. There are more than one open changes containing subsets of the specified objects. In this case, the specified set of objects cannot be activated. It must be split up into subsets.

Important: If you activate entire (tracked) packages, the set of specified objects used for retrieving the affected changes includes all objects in the packages, not only the inactive objects. The aim of this is to keep the number of changes small.

It is possible to retrieve the set of affected changes without activating the objects by calling "regi activate" with the parameter "--getAffectedChanges|--ga".

Examples:

- `regi activate --ch=4711`
  Activates all inactive objects in the tracked packages and assigns the activated objects to change ID "HDB//4711" (given that the HANA server SID is "HDB").
- `regi activate --ga`
  Retrieves the set of affected changes of all objects in the tracked packages, but does not activate the objects.
- `regi activate`
  Activates all inactive objects in the tracked packages and assigns them to either a new, implicitly created change, or an open change containing a subset of the specified objects (given that there is no other open change containing a subset of the specified objects).

Export

It is possible to restrict the export of delivery units to only those objects which are contained in released changes with the parameter "--released|-r". It is also possible to include only those objects which are contained in changes which have been released up to a specific point in time with the parameter "--to=TIMESTAMP".

Examples:

- `regi export testdu -r`
  Exports only the released objects of the delivery unit "TESTDU".
- `regi export testdu -r --to=2013-11-06`
  Exports only the released objects of the delivery unit "TESTDU" up to the time stamp "2013-11-06".
- `regi transport testdu -r --to=2013-11-06 --tkey=myTargetKey`
  Transports only the released objects of the delivery unit "TESTDU" up to the time stamp "2013-11-06" to the target system with key "myTargetKey".

Managing changes

Changes can be explicitly created with "regi create change|ch", and explicitly deleted with "regi delete change|ch".

Open changes can be released with "regi release [change|ch]".

Two open changes can be merged, i.e., their sets of objects can be combined with "regi merge changes|chs".

Objects can be moved from one open change to another with "regi move".

The description of a change can be updated with "regi update change|ch".

Changes can be searched for with "regi list changes|chs".

The entries of a change can be listed with "regi list change|ch".

The meta data and the contributors of a change can be shown with "regi show change|ch".

Managing contributors

All contributors of a change have set their status to "validated" before a change can be released. This can be done with "regi validate [contributor|cont]".

Contributors can be created for a change with "regi create contributor|cont".

Contributors can be deleted from a change with "regi delete contributor|cont".

The description of a contributor can be updated with "regi update contributor|cont".

---

cleanUp

Clean up the Repository/Regi workspaces:

Delete delivery units:

Use "regi undeploy" to delete all objects in all packages of a delivery unit and the delivery unit itself.

Example:

- `regi undeploy testdu`
  Deletes all objects in all packages of delivery unit "TESTDU" and the delivery unit itself.

Delete packages and the contained objects:

Call "regi delete package PACKAGE PACKAGE... -f" to delete packages and the contained active and inactive objects in all workspaces.

Example:

- `regi delete package p1 p2 -f`
  Deletes the packages p1 and p2 and all the contained objects.

Delete inactive objects (also of other workspaces):

1. "regi list iws" to list the inactive workspaces,
2. "regi delete iws" to delete them.

Important: The user of the inactive workspaces to be listed and/or deleted is case-sensitive (i.e., must be written in upper case).

Example:

- `regi list iws --iuser=*`
  Lists the inactive workspaces of all users.

- `regi delete iws w1 w2 --iuser=THEUSER`
  Deletes the workspaces "w1" and "w2" of user "THEUSER".

Clean up Regi workspace:

1. Simply delete the Regi workspace directory to delete the workspace and re-create it using "regi create ws", or
2. Keep the Regi workspace directory and use "regi discard" to remove the meta data and content of the checked out objects.

Examples:

- `regi discard p1/`
  Discards the meta data and content of the objects in package "p1".

- `regi discard p1/ --nocontent`
  Discards the meta data but not the content of the objects in package "p1".

---

conflicts

Conflict resolution:

A file is marked as conflicted if "regi checkout" attempts to check out an object from the Repository which has been locally created, deleted, or modified.

To prepare conflict resolution, "regi checkout" retrieves:

1. the meta data of the object from the Repository (into "._SYS_REGI_settings/SYS/..."),
2. the content of the object from the Repository (into "._SYS_REGI_settings/diff/..."),
3. the content of the base version of the object (into "._SYS_REGI_settings/base/..."),
4. and marks the object as "conflicted" in "._SYS_REGI_settings/.regiStatus"

This allows you to resolve the conflicts locally, without having to be connected to the Repository. A conflict can either be resolved interactively using a three-way merge tool ("--mergetooll-m"), or non-interactively ("--local", "--base", "--remote").

Call:

1. "regi resolve --locall--basel--remotel--mergetooll-m for (non-)interactive conflict resolution.
2. "regi push --locall--basel--remotel--mergetooll-m" for (non-)interactive conflict resolution plus commit plus activate, or
3. "regi checkout --forcel-f" to overwrite your local changes with the currently active version, and thus resolve all conflicts to remote.

---

deliveryUnits

Delivery units:

Specification:

Delivery units (DUs) can be specified simply by their name. For convenience, DU names are automatically capitalized, i.e., they can also also be specified in lower case. In addition, you can use dashes "-" instead of underscores "_", e.g. SAPUI5-1 instead of SAPUI5_1.

The vendor does not have to be specified - the default vendor is the content vendor of the HANA instance. This can be overridden with the "--vendor" parameter.

Examples:

- `regi create du testdu`
  Creates DU "TESTDU" with vendor = content vendor.
- `regi create du testdu --vendor=test.sap.com`
  Creates DU "TESTDU" with vendor = "test.sap.com".

Creation:

DUs can be explicitly created with "regi create du".

Deletion:

Empty DUs can be explicitly deleted with "regi delete du" (a DU is empty if there are no packages assigned to it anymore). DUs can be undeployed (along with all contained packages and all contained objects in these packages) with "regi undeploy".

Modification:

The meta data of DUs can be modified with "regi update du". Packages can be assigned to/unassigned from delivery units with "regi assign" and "regi unassign", respectively.

Display:

The list of DUs on the system can be displayed with "regi list dus". The meta data of DUs can be displayed with "regi show du". The status of DUs can be shown with "regi show dustatus". The list of packages assigned to DUs can be displayed with "regi list du".

---

environment

Generic environment variables:

Most generic environment variables are matched by commandline arguments. See "regi help arguments".

Environment variables can be used to configure certain aspects of Regi permanently.

- Output:
    - `REGI_DEBUG=1`
      Debug output.
    - `REGI_DEBUG_PATHS=1`
      Debug output for path handling.
    - `REGI_DEBUG_REGIIGNORE=1`
      Debug output for ".regiignore" files.
    - `REGI_JSON_REQUEST=1`
      Display requests from Regi to the Repository.
    - `REGI_JSON_RESPONSE=1`
      Display responses from the Repository to Regi.
    - `REGI_LOG=1`
      Create log file in "._SYS_REGI_settings/logs".
    - `REGI_NO_COMMANDLOG=1`
      Do not create command log file in "._SYS_REGI_settings/logs".
    - `REGI_PERFTRACE=1`
      Create Regi performance trace.
    - `REGI_VERBOSE=1`
      Verbose output.
    - `REGI_TRACE=1`
      Create trace file in "._SYS_REGI_settings/logs".
- Connection:
    - `REGI_KEY=KEY`
      Secure store key for server connection. Keys can be created with the HANA client tool "hdbuserstore".
    - `REGI_HOST=HOST:SQLPORT`
      Host and SQL port for the server connection (if no key provided). SQLPORT = 30000 + 15 + 100*INSTANCE, e.g. 34215 for instance 42. For STFK connections, use the local proxy (e.g. "localhost:5001").
    - `REGI_USER=USER`
      User for the server connection (if no key provided).
    - `REGI_PASSWD=PASSWORD`
      Password for the server connection (if no key provided).
    - `REGI_STDIN=1`
      Read password from standard input if it is not provided.
- Status calculation:
    - `REGI_FAST=1`
      Use timestamps and hashes for status calculation. Fast, but not recommended for writing scripts with Regi. Recommended for non-scripted Regi use.
    - `REGI_NO_COMPARE=1`
      Compare neither timestamps nor hashes in status calculation, i.e., list all files as changed, even if they have not changed. Useful for forced re-generation of objects.

  - `REGI_VERYFAST=1`
    Only use timestamps for status calculation. Very fast, but not recommended for writing scripts with Regi.

- Other:

  - `REGI_ARGS=1|2`
    "1" to print the argument parse result and continue, "2" to print the argument parse result and stop.

  - `REGI_BROWSE=1`
    Show the help texts in the Browser specified in the environment variable "BROWSER" or "firefox" (default).

  - `REGI_IGNORE_LOCKFILE=1`
    Ignore the Regi workspace lock file. See "regi help lockFile".

  - `REGI_INACTIVE_OBJECTS=1`
    List the inactive objects in "regi status".

  - `REGI_LINEEND=LOCAL|UNIX|WIN`
    Specify line end mode for writing files ("regi checkout"): "LOCAL": local, "UNIX": force LF, "WIN": force CRLF. Default: "LOCAL".

  - `REGI_NO_BULKREAD=1`
    Do not use bulk read; instead read the objects one at a time ("regi checkout").

  - `REGI_NO_CONFIRM=1`
    Do not ask for confirmation of interactive merges (assume "y" if the merge tool has modified the file, "n" otherwise). ("regi mergetool", "regi resolve -m", "regi push -m").

  - `REGI_NO_MULTIIMPORT=1`
    Import the DUs one after the other (do not use multi DU import) ("regi import du", "regi transport du").

  - `REGI_NO_TIPS=1`
    Do not show tips, e.g. for conflict resolution.

  - `REGI_SIMULATE=1`
    Use simulation mode (read-only; neither writes to the file system nor to the Repository).

  - `REGI_WORKSPACE=WORKSPACE`
    Set workspace.

---

faq

FAQ:

1. **Q:** "I have created my "hdbuserstore" key, but when I try to create a Regi workspace with that key, I get an error starting with: "Connecting to the database failed". What can I do?"
   A: Re-check the definition of the "hdbuserstore" key. Unfortunately, "hdbuserstore" does not check the connection upon "hdbuserstore set", so errors in the key definition only come to light later when you try to create a Regi workspace. See also: "regi help hdbuserstore", "regi create ws".
2. **Q:** "I have tried to commit and activate, or push to transfer my files to the Repository, but all I get are "conflicts". What are these?"

A: Regi marks files as "conflicted" if it has attempted to check out new objects from the Repository which have been locally created, deleted or modified. See also: "regi help conflicts".

3. **Q:** "I have deleted a directory tree, committed and activated, or pushed the deletions, but the packages still remain in the Repository. Why?"
   A: Whereas Regi implicitly *creates* directories corresponding to packages upon "regi checkout", and it implicitly *creates* packages corresponding to directories upon "regi commit", Regi does not implicitly *delete* packages. This is due to an asymmetry between objects and packages in the Repository. After deleting the objects corresponding to the files in your deleted directory tree, you have to delete the packages explicitly with "regi delete package". See also: "regi help cleanup", "regi help packages".
4. **Q:** "I have interrupted a Regi call, but now Regi tells me that the workspace is locked. What can I do?"
   A: Either delete the lock file ".lock" in "._SYS_REGI_settings", or make Regi ignore the lock files altogether using the parameter "--ignoreLock" or the environment variable "REGI_IGNORE_LOCKFILE". See also: "regi help lockfile".
5. **Q:** "I am curious - how can I see the communication between Regi and the Repository?"
   A: Use either the parameter "--debug|-d", or "--jsonRequest|jReq" to see only the requests to the Repository, and/or "--jsonResponse|jResp" to see the responses from the Repository. With the parameter "--trace|-t", the Repository communication is written to the trace file "commands.trace" in "._SYS_REGI_settings/logs". See also: "regi help traces".
6. **Q:** "I think I have found a bug. What do you need to track it down?"
   A: At least provide the "commands.log" file located in "._SYS_REGI_settings/logs". If you can reproduce the bug, do so with the "--trace|-t" parameter attached to each command, or with the environment variable "REGI_TRACE" set to "1". Then, provide the "commands.trace" file also located in "._SYS_REGI_settings/logs". See also: "regi help traces".
7. **Q:** "I would like to activate a deletion but for some reason, the activation fails. What can I do to activate it anyway?"
   A: In such an emergency, use "regi activate --activationmode=4" - this makes the Repository activate all objects to be activated even if the activation has actually failed. Those objects are then marked as "erroneous" in the Repository. See also: "regi help activationmode".
8. **Q:** "I cannot delete a package because someone else has left inactive objects in there. What can I do?"
   A: You can either use the "-f" flag for "regi delete package", or you can use "regi list iws" and "regi delete iws" to delete the inactive objects in question. See also: "regi help cleanup", "regi help packages".
9. **Q:** "I just want to commit and activate my files anew, without having to actually modify them. I'd just like to use "touch" to touch the files. Is that possible?"
   A: Yes. Use the parameter "--veryfast" to make "commit" compare the files only based on their last modification timestamps. See also: "regi help statuscalculation".
10. **Q:** "I am writing a script using Regi, but for some reason it does not work - strangely, when I execute the steps manually, it works. Why?"
    A: You are using the "--fast" or "--veryFast" parameters to speed up the Regi status calculation in a script. This may fail on older file systems such as EXT3 which do not have sub-second resolution. Refrain from using "--fast" and "--veryFast" in scripts. See also: "regi help statuscalculation".
11. **Q:** "The tool "hdbuserstore" does not work or I cannot use it because I would like to work on a Regi workspace created by the HANA Studio Team Provider. What can I do?"
    A: The best way to create your workspace without "hdbuserstore" is:
    1. Set the environment variable "REGI_PASSWD" to the password,
    2. "regi create ws WORKSPACENAME --host=HOST:SQLPORT --user=USER". For STFK connections, use the local proxy (e.g. "localhost:5001"). Set "REGI_PASSWD" also if you want like to work on a Regi workspace created by the HANA Studio Team Provider.
12. **Q:** "I tried to create a DU but all I get is the error: "Delivery unit vendor is not a sub vendor of the content vendor". What does that mean?"

A: The vendor you give to a DU must be sub vendor of the content vendor. For example, "test.sap.com" is a sub vendor of "sap.com", and "sap.com" is a trivial sub vendor of itself. The content vendor can be configured in the HANA Studio. Simply double click on your system, proceed to the "Configuration" tab and search for "vendor". Then, enter the content vendor.

13. **Q:** "What is the intention behind the the commands "regi checkout ios", "regi commit ios", "regi discard ios" and "regi resolve ios", "regi status ios" (and also "regi mergetool ios" and "regi push ios")?"
A: Simple answer: You will not need them. Complex answer: These commands are there for consistency/completeness, as counterparts for "regi activate ios", "regi revert ios" and "regi takeover ios". In fact, some are actually used internally, e.g. "regi checkout ios" is used inside "regi revert ios" and "regi takeover ios". The intention is that the inactive objects of the workspace determine the set of objects eligible to the respective operation. For example, "regi commit ios" means that all files with corresponding inactive objects are to be committed.

14. **Q:** "I keep getting the error message "regi: Could not parse commandline arguments" when I try to create a workspace in the current directory. What can I do?"
A: You are probably used to the old Regi syntax which allowed to create workspaces inside of already created workspace directories using "regi create ws --key=KEY". Regi now requires a workspace path parameter for "create ws", so use the new syntax "regi create ws . --key=KEY", where the dot indicates that the current directory is the workspace path.

15. **Q:** "I want to commit/push files to the Repository regardless of whether they have been changed or not to regenerate the described catalog objects upon activation (a bit like "regenerate" in the HANA Studio). Can I do that?"
A: Yes. Use the parameter "--noCompare|--nc". See also: "regi commit", "regi push" and "regi status".

16. **Q:** "I'd like to use symbolic links in my workspace. Does Regi support that?"
A: Yes. However, there are a number of restrictions. On Linux, Regi fully supports symbolic links to directories and files. On Windows, Regi supports only symbolic links and junctions to directories; it does not support symbolic links to files, it does not support shortcuts, and it does not support symbolic links created with Cygwin ("ln -s"). You can create symbolic links and junctions to directories using the commands "mklink /d" or "mklink /j" in a Windows shell.

17. **Q:** "Why are there some parameters with one dash, e.g. "-v", and some with two dashes, e.g. "--tkey"?"
A: All parameters consisting of just one character must be preceded by one dash, and all parameters consisting of more than one character must be preceded by two dashes. This is just a convention, a bit Unix-like.

18. **Q:** "I get a weird connection error stating: "Connection failed (RTE:[-1] Kerberos error. Major: "No credentials were supplied [458752]"". What does this mean?"
A: Your credentials, either provided using "hdbuserstore" or directly provided, are at fault. The "Kerberos error" might just indicate that you specified no or not the correct password. Check your credentials and try again. See also: "regi help hdbuserstore", "regi create ws".

19. **Q:** "I get a weird error when checking out a large amount of files: "An error occurred: Failed to execute prepared statement: Session has been reconnected."". What can I do?"
A: The default behavior is to check out all files in one bulk, because this is more efficient, especially if the connection has some latency. In some circumstances, however, this default behavior does not work out with the HANA instance at hand. Use the parameter "--noBulkRead" in these cases, or set the environment variable "REGI_NO_BULKREAD" to "1". This applies to all commands calling "regi checkout", i.e.: "regi activate", "regi push", "regi revert" and "regi takeover".

20. **Q:** "How can I make a quick connection check in a workspace?"
A: Call "regi list ios".

21. **Q:** "When I try to discard packages in my Regi workspace, I keep getting a strange error: "An error occurred: Failed to delete directory DIRECTORY. System error code: 32". What does this mean and what can I do about it?"
A: Ok, it looks as if you are using Windows. "System error code 32" means: "The process cannot access

the file because it is being used by another process." It might be that the directory cannot be deleted because your current directory in the shell equals the directory DIRECTORY, or you have an Explorer window open in that directory.

22. **Q:** "I am working on a customer ticket and I am connected to the customer system via STFK. In the HANA Studio, I can connect to the system without problems, but when I switch to the command-line and try to work on the workspace with Regi, it cannot connect to the customer system. What can I do?"
A: For Regi, you need to set up the HANA connection using the local STFK proxy as the host (e.g. "localhost:5001"). Use this as the host for for hdbuserstore, or, if you do not use a hdbuserstore key, e.g. if you work on a Regi workspace created in the HANA Studio, set the environment variable REGI_HOST to the proxy (e.g. "set REGI_HOST=localhost:5001").
23. **Q:** "I'd like to access the workspace/inactive objects developed in the HANA Studio Modeler. How can I do this?"
A: Use the special workspace name "__empty__" to specify the "empty workspace" used by the HANA Studio Modeler.
24. **Q:** "How can I get the entire built-in Regi documentation in one HTML file? This would definitely be useful, also for printing it out."
A: Use the command "regi help --browse" or "regi help -s --html >regihelp.html".

---

hdbuserstore

The HANA client tool "hdbuserstore":

"hdbuserstore" can create "keys" which Regi uses to establish its connection to the HANA Repository.

"hdbuserstore" comes with the HANA client installation.

Type:

1. "hdbuserstore help" for help,
2. "hdbuserstore set KEY HOST:SQLPORT USER PASSWORD" to create a key in the secure store,
3. "hdbuserstore list" to list the keys in the secure store,
4. "hdbuserstore delete KEY" to delete a key. Note that "hdbuserstore" does not make a connection test upon "hdbuserstore set" - you have to call "regi create ws" to check whether the key you have defined really works.

---

lockFile

The lock file:

Regi prohibits concurrent access to Regi workspaces with lock files. The Regi lock file ".lock" is located in "._SYS_REGI_settings".

After Regi has been interrupted or has crashed unexpectedly, the lock file stays in place and must be deleted manually.

Workspace locking can be disabled by:

1. using the commandline argument "--ignoreLockFile|--ignoreLock", or
2. setting the environment variable "REGI_IGNORE_LOCKFILE" to "1".

---

logs

Log files:

Regi stores log files in "._SYS_REGI_settings/logs".

Logging can be enabled by:

1. using the commandline argument "--log|-l", or
2. setting the environment variable "REGI_LOG" to "1".

Logging can be enabled for:

- regi activate
- regi export du
- regi export la
- regi import du
- regi import la
- regi push
- regi transport

A log file called "commands.log" is always written and contains a history of the Regi commands called in the workspace.

---

mergeToolConfig

Merge tool configuration:

The default merge tool is "p4merge".

Others can be configured by:

1. using the commandline argument "--mergeToolConfig", or
2. setting the environment variable "REGI_MERGETOOL".

   Syntax:

   - `export REGI_MERGETOOL='MERGETOOL PARAMETERS'`
     where PARAMETERS include:
     - %b = (b)ase file in "._SYS_REGI_settings/base",
     - %r = (r)emote (active) version in "._SYS_REGI_settings/diff",

        - %l = (l)ocal version.

    Examples:

    - `export REGI_MERGETOOL='p4merge %b %r %l %l'`
      Default "p4merge" configuration: Configures "p4merge" to merge base, remote and local files, such that "p4merge" writes the merge result back to the local file.

    - `export REGI_MERGETOOL='kdiff3 %b %r %l -o %l'`
      Configures "kdiff3" to merge base, remote and local files, such that "kdiff3" writes the merge result back to the local file.

---

nutshell

Regi in a nutshell:

1. Use "regi create ws" to create a Regi workspace,
2. use "regi checkout" to check out packages from the Repository (automatically tracks the packages),
3. create/delete/modify objects,
4. Use "regi status" to see your changes,
5. Use "regi push" to commit and activate the creations/ deletions/modifications; use "regi push -m" to interactively resolve conflicts.

---

objects

Objects:

Upon "checkout", Regi retrieves:

1. the object content into the workspace directory,
2. the object meta data into "._SYS_REGI_settings/SYS/...",
3. the object status (its hash, the file modification timestamp and the base version) into "._SYS_REGI_settings/.regiStatus".

The base version is either:

1. the active version number if an active version has been checked out, or
2. the inactive version number - 1, if an inactive version has been checked out.

The object meta data is required to "commit" and "activate" modifications to the objects. When an object is committed, its inactive version number equals the checked out active version number + 1. When an object is activated, the Repository compares the inactive version number of the committed object with the currently active version version number. If the currently active version number is equal or higher than the inactive version number, the activation is rejected by the Repository, and Regi conflict resolution is required (see "regi help conflicts").

You must always check out the files before modifying and committing them, except if you are sure that the objects are new to the Repository. If you try to commit objects which already exist in the Repository without having checked them out before, it is guaranteed to fail: The Repository will reject your objects because they will have inactive version number 1, whereas the existing versions of the objects in the Repository will have equal or higher versions.

---

packages

Packages:

Specification:

Packages can either be specified:

1. dot-separated (e.g. "sap.hana.xs")
2. slash-separated (e.g. "sap/hana/xs")

Inside a Regi workspace, the slash-separated syntax is interpreted as a relative path.

Some commands like "regi checkout [objects]" accept both objects and packages. Specify the packages with the slash-separated syntax and a trailing slash.

Examples (inside workspace; current directory/package: "p1"):

- `regi checkout p2/p3/test.txt`
  Checks out object "test.txt" in package "p1.p2.p3".
- `regi checkout p2/p3`
  Checks out object "p3" in package "p1.p2" (no trailing slash).
- `regi checkout p2/p3/`
  Checks out package "p1.p2.p3" (trailing slash).
- `regi checkout package p1.p2.p3`
  Checks out package "p1.p2.p3".

Creation:

Packages can be either:

1. Implicitly created with "regi commit", or
2. Explicitly created with "regi create package".

Packages inherit their meta data from their super packages, if there exist any.

Inside a Regi workspace, if you explicitly create a package, the corresponding directory is created automatically.

Deletion:

Packages can only be explicitly deleted with "regi delete package".

Inside a Regi workspace, if you explicitly delete a package, the corresponding directory is automatically deleted.

If you use the --force|-f parameter, Regi deletes all sub packages and all contained active and inactive objects in all workspaces.

Modification:

The meta data of packages can be modified with "regi update package".

Display:

The list of packages on the system can be displayed with

1. "regi list packages". The list can be filtered.
2. "regi list rootPackages" displays the root packages.

Examples:

- `regi list packages`
  Lists all packages.
- `regi list packages p1*`
  Lists all packages starting with "p1".
- `regi list packages --textcoll=coll1`
  Lists all packages with textCollection "coll1".

The meta data of packages can be displayed with "regi show package".

---

regiignore

".regiignore" files:

Regi ignores files or entire directories specified by ".regiignore" files. Each directory in the Regi workspace may contain a ".regiignore" file, which specifies Perl Regular Expressions (PCRE) for the files/directories to be ignored in this directory or below.

Examples:

- `{"exclude":[".*~", ".*\.bak"]}`
  Ignores all files ending with "~" or ".bak"
- `{"exclude":[".*~", ".*\.bak"], "include":["do_not_exclude_me\.bak"]}`
  Ignores all files ending with "~" or ".bak", except for the file "do_not_exclude_me.bak".
- `{"exclude":["Debug/.*"]}`
  Ignores all files in the sub directory "Debug".

If a file is ignored, it will not show up in "regi status", and it cannot be committed anymore. It can still be checked out from the Repository (if an inactive or active version of it exists).

The "exclude" and "include" sets of ".regiignore" files in sub package directories are unioned with those of their parents. This allows you to specify more general exclusion and inclusion sets further up in the package hierarchy, and more specific sets further down. For instance, you could specify on the top-level of the package hierarchy that you want to exclude all files with the "html" suffix:

```
{"exclude":[".*\.html"]}
```

And then, some levels below in the package hierarchy (e.g. in package "p1.p2.p3"), you could specify that you wish to include (=not ignore) the file "index.html":

```
{"include":["index\.html"]}
```

You can have:

1. Shared ".regiignore" files, which should be placed inside one of your tracked packages, or
2. Local ".regiignore" files, which should be placed in the root directory of the Regi workspace.

To debug ".regiignore" files, use the parameter "--debugRi" or set the environment variable "REGI_DEBUG_REGIIGNORE" to "1". You might also try the various online regular expression testers on the Web (e.g. "http://regex.larsolavtorvik.com/")

---

statusCalculation

Local object status calculation:

Local object status calculation is used in the following actions:

- regi activate
- regi checkOut
- regi commit
- regi discard
- regi resolve
- regi revert and of course:
- regi status

to determine the locally created/deleted/modified objects. Default local object status calculation:

1. If a file exists in the file system but is not stored in the ".regiStatus" file in "._SYS_REGI_settings", it is *created*,
2. else if a file does not exist in the file system but it is stored in the ".regiStatus" file in "._SYS_REGI_settings", it is *deleted*,
3. else if the hash of the file in the file system and the hash in ".regiStatus" differ, it is *modified*,
4. else the file is *unchanged*.

If the parameter "--fast" is used, step 3) becomes:

3) else if the timestamp of the file in the file system and the timestamp in ".regiStatus" differ, and then if the hashes also differ, it is *modified*, This is faster than the default mode, and recommended for non-scripted Regi

use, but it is NOT recommended if Regi is used in scripts. Why? Comparing timestamps can be unreliable on older file systems without sub-second resolution. For example, EXT3 only has a timestamp resolution of one second - if a file is modified more than once in a second, the modifications cannot be detected by the "--fast" status calculation.

If the parameter "--veryfast" is used, step 3) becomes:

3) else if the timestamp of the file in the file system and the timestamp in ".regiStatus" differ, it is *changed*. This is the fastest mode, but also not recommended if Regi is used in scripts, for the same reasons as above ("--fast").

If the parameter "--noCompare|--nc" is used, step 3) becomes:

3) else it is *changed*. This is recommended for forcing Regi to commit files to the Repository in any case, e.g. for re-generating them.

History of default status calculation

- Up to SP6 (Regi Beta01 - Regi 0.13.3), the default status calculation uses pure hash comparison (no time stamps).
- In SP7 (Regi 0.36.0, 0.44.6), the default status calculation uses the "--fast" mode (first time stamp, then hash comparison). This can possibly break scripts using Regi on file systems without sub-second resolution (e.g. EXT3). For scripts, set the environment variable "REGI_HASHONLY" to "1".
- In SP8+ (Regi 1.0.12+), the default status calculation uses pure hash comparison (no time stamps) again.

Bottom line: If your script using Regi might use an SP7 version of Regi, set "REGI_HASHONLY" to "1" to make sure that the script works on all file systems.

---

timeStamps

Timestamps:

Timestamps e.g. for "regi checkout --hist=TIMESTAMP" have the format "YYYY-MM-DD HH:MM:SS.NNN". The time may be omitted: "YYYY-MM-DD".

---

traces

Regi trace files:

Regi stores trace files in "._SYS_REGI_settings/logs".

Tracing can be enabled by:

1. using the commandline argument "--trace|-t", or
2. setting the environment variable "REGI_TRACE" to "1". Tracing can be enabled for all commands.

---

activate

Activates objects in the Repository. If objects cannot be activated due to activation conflicts, checks them out to prepare conflict resolution. If change tracking is enabled and no change ID is provided, gets the affected changes of the specified objects prior to activation. If no change is affected, creates a new change for the inactive objects to be activated from the set of specified objects; else if precisely one change is affected, adds the inactive objects to it; else if more than one change is affected, activation fails.

Usage:

- activate inactiveObjectslios
- activate [trackedPackages]
- activate object OBJECT OBJECT...
- activate package PACKAGE PACKAGE...
- activate packages PACKAGE PACKAGE...
- activate [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--activationMode=N`
  Specify activation mode (default: N = 0). Call "regi help activationMode" for more information.
- `--changeid|--ch=CHANGEID`
  Specify change ID.
- `--getAffectedChanges|--ga`
  Do not activate but only print the affected changes, i.e., the changes including one or more of the specified objects.
- `--log|-l`
  Write check results to log file in "._SYS_REGI_settings/logs".
- `--noRecursive|--noRec`
  Do not recursively activate sub packages.
- `--noCheckOut`
  Do not check out objects.
- `--verbose|-v`
  Write check results to standard output.

Examples:

-
  ```
  regi activate
    = regi activate trackedPackages
  ```
  Activates all inactive objects in all tracked packages and their sub packages.

-
  ```
  regi activate --activationMode=4 -l
    = regi activate trackedPackages --activationMode=4 --log
  ```
  Activates all inactive objects in all tracked packages and their sub packages. Uses activation mode 4 and writes the check results to a log file.

-
  ```
  regi activate --noCheckOut
    = regi activate trackedPackages --noCheckOut
  ```

Activates all inactive objects in all tracked packages and their sub packages. Does not check out the objects in case of activation conflicts.

- `regi activate p1/test.txt p2/p3/`
  `= regi activate objects p1/test.txt p2/p3/`
  Activates the inactive object "test.txt" in package "p1" and activates all inactive objects in package "p2.p3" and its sub packages.

- `regi activate p1/test.txt p2/p3/ --norec`
  `= regi activate objects p1/test.txt p2/p3/ --norec`
  Activates the inactive object "test.txt" in package "p1" and activates all inactive objects in package "p2.p3" but not its sub packages.

- `regi activate ios`
  `= regi activate inactiveObjects`
  Activates all inactive objects in all packages.

- `regi activate object p1/test.txt`
  Activates the inactive object "test.txt" in package "p1".

- `regi activate package p2/p3/`
  Activates all inactive objects in package "p2.p3" and its sub packages.

- `regi activate --ga`
  `= regi activate trackedPackages --getAffectedChanges`
  Gets all affected changes, i.e., the changes including one or more of all objects in the tracked packages.

- `regi activate --ch=4711`
  `= regi activate trackedPackages --changeid=HDB//4711`
  Activates all inactive objects in all tracked packages and their sub packages, and includes them in the change with ID "HDB//4711".

Related:

- checkOut (check out objects from the Repository)
- commit (commit files to the Repository)
- create change (create changes)
- delete change (delete changes)
- list activeObjects (list active objects)
- list change (list entries of changes)
- list changes (list changes)
- list inactiveObjects (list inactive objects)
- merge changes (merge two changes into one)
- mergeTool (interactively resolve conflicts)
- move (move objects from one change to another)
- push (push objects to the Repository)
- release change (release changes)
- resolve ((non-)interactively resolve conflicts)
- revert (revert inactive objects in the Repository)
- set changeTrackingStatus (set change tracking status)

- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- status (show the status of (files in) the Regi workspace)

---

assign

Assign packages to delivery units. Default vendor: content vendor.

Usage:

- assign [package] PACKAGE PACKAGE... DELIVERYUNIT

Options:

- `--noRecursive|--noRec`
  Do not recursively assign sub packages.

- `--vendor=VENDOR`
  Specify the vendor of the delivery units if necessary.

Examples:

-
  ```
  regi assign p1 p2.p3 testdu
    = regi assign package p1 p2.p3 testdu
  ```
  Assigns the packages "p1" and "p2.p3" to delivery unit "TESTDU". Vendor: content vendor.

-
  ```
  regi assign p1 p2.p3 testdu --vendor=test.sap.com
    = regi assign package p1 p2.p3 testdu --vendor=test.sap.com
  ```
  Assigns the packages "p1" and "p2.p3" to delivery unit "TESTDU". Vendor: test.sap.com.

-
  ```
  regi assign p1 p2.p3 testdu --norec
    = regi assign package p1 p2.p3 testdu --norec
  ```
  Assigns the packages "p1" and "p2.p3" to delivery unit "TESTDU"; does not recursively assign the sub packages of "p1" and "p2.p3" to "TESTDU". Vendor: content vendor.

Related:

- list deliveryUnit (list delivery unit)
- list packages (list packages in the Repository)
- show vendor (show content vendor)
- unAssign (unassign packages from their delivery units)
- update package (update meta data of packages)

---

checkOut

Checks out objects from the Repository. Default behavior: For each object, compares the active and inactive versions, and then retrieves the newer version of the two. If the corresponding file is locally modified, creates a conflict. Automatically tracks package arguments unless either "--notrack" or "--norec" is used.

Usage:

- checkOut inactiveObjects|ios
- checkOut [trackedPackages]
- checkOut object OBJECT OBJECT...
- checkOut package PACKAGE PACKAGE...
- checkOut packages PACKAGE PACKAGE...
- checkOut [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--active`
  Always retrieve the active version.
- `--base`
  Always retrieve the base (=last checked out active) version.
- `--force|-f`
  Overwrite the corresponding local objects in any case and resolve all conflicts to remote.
- `--inactive`
  If exists, always retrieve the inactive version, otherwise retrieve the active version. Same behavior as "regi checkout" in old versions of Regi.
- `--historical|--hist=TIMESTAMP`
  Retrieve the version older or equally old as TIMESTAMP, but newer than the version before.
- `--historicalObject|--histObj|--hO=OBJECT`
  Select an object to get the timestamp for historical check out. Required by --historicalVersion|--histVer|--hV if not exactly one object is denoted by the parameters.
- `--historicalVersion|--histVer|--hV=VERSION`
  Retrieve the version older or equally old as the timestamp of version VERSION of the selected object (--historicalObject|--histObj|-ho). If exactly one object is denoted by the parameters, the --historicalObject|--histObj|--ho parameter may be omitted.
- `--lineEnd=LINEEND`
  Specify line end mode for writing files: "LOCAL": local, "UNIX": force LF, "WIN": force CRLF.
- `--noBulkRead`
  Do not use bulk read; instead read the objects one at a time.
- `--noRecursive|--noRec`
  Do not recursively check out sub packages.
- `--noTrack`
  Do not track package arguments.
- `--overwrite`
  If a conflict is detected, overwrite the corresponding local files and meta data and do not mark them as conflicted.

- `--skip`
  If a conflict is detected, keep the corresponding local files and meta data and do not mark them as conflicted.

Examples:

- `regi checkout`
  `  = regi checkout trackedPackages`
  Checks out all objects in all tracked packages and their sub packages.

- `regi checkout p1/test.txt p2/p3/`
  `  = regi checkout objects p1/test.txt p2/p3/`
  Checks out the object "test.txt" in package "p1" and all objects in package "p2.p3" and its sub packages. Automatically tracks package "p2.p3" if it is not yet already tracked.

- `regi checkout object p1/test.txt`
  Checks out the object "test.txt" in package "p1".

- `regi checkout package p2/p3/`
  Checks out all objects in package "p2.p3" and its sub packages. Automatically tracks package "p2.p3" if it is not yet already tracked.

- `regi checkout package p2/p3/ --norec`
  Checks out all objects in package "p2.p3" but not its sub packages. Does not automatically track package "p2.p3" ("--norec" is used).

- `regi checkout package p2/p3/ --notrack`
  Checks out all objects in package "p2.p3" and its sub packages. Does not automatically track package "p2.p3" ("--notrack" is used).

- `regi checkout p1/test.txt --hv=42"`
  Checks out object test.txt in package "p1", Retrieves the historical version 42. As exactly one object is denoted by the parameters ("p1/test.txt"), the --historicalObject|--histObj|--hO parameter is omitted.

- `regi checkout p1/ --hv=42 --ho=p1/test.txt"`
  Checks out package "p1". Retrieves the versions older or equally old as the timestamp of version 42 of object "p1/test.txt"

- `regi checkout p1/test.txt --hist="2013-08-30 10:22:38.9430000"`
  Checks out object test.txt in package "p1". Retrieves the versions older or equally old as 2013-08-30 10:22:38.9430000.

Related:

- activate (activate inactive objects in the Repository)
- commit (commit files to the Repository)
- create workSpace (create Regi workspace)
- discard (discard checked out objects)
- list activeObjects (list active objects)
- list history (list history of objects)
- list inactiveObjects (list inactive objects)
- list rootPackages (list root packages)
- list packages (list packages in the Repository)

- mergeTool (interactively resolve conflicts)
- push (push objects to the Repository)
- resolve ((non-)interactively resolve conflicts)
- revert (revert inactive objects in the Repository)
- status (show the status of (files in) the Regi workspace)
- takeOver (take over inactive objects from other Repository workspaces)
- track allPackages (track all packages in the Repository)
- track package (track a package)

---

commit

Commits files to the Repository. Automatically creates packages if necessary.

Usage:

- commit inactiveObjectslios
- commit [trackedPackages]
- commit object OBJECT OBJECT...
- commit package PACKAGE PACKAGE...
- commit packages PACKAGE PACKAGE...
- commit [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--fast`
  Use timestamps and hashes for status calculation. Fast, but not recommended for writing scripts with Regi. Recommended for non-scripted Regi use.

- `--noCompare|--nc`
  Compare neither timestamps nor hashes in status calculation, i.e., commit all files, including those which have not been changed. Useful for forced re-generation of objects.

- `--noInherit`
  Package creation: Do not inherit meta data from super packages.

- `--noRecursive|--noRec`
  Do not recursively commit the files to the Repository.

- `--veryFast`
  Only use timestamps for status calculation. Very fast, but not recommended for writing scripts with Regi.

Examples:

-
  ```
  regi commit
    = regi commit trackedPackages
  ```
  Commits all files in all directories corresponding to the tracked packages and their sub packages.

- `regi commit p1/test.txt p2/p3/`
  `  = regi commit objects p1/test.txt p2/p3/`
  Commits the files "test.txt" in the directory corresponding to package "p1" and all files in the directories corresponding to the package "p2.p3" and its sub packages.

- `regi commit object p1/test.txt`
  Commits the file "test.txt" in the directory corresponding to the package "p1".

- `regi commit package p2/p3/`
  Commits the files in the directories corresponding to package "p2.p3" and its sub packages.

- `regi commit package p2/p3/ --norec`
  Commits the files in the directories corresponding to package "p2.p3" but not its sub packages.

- `regi commit package p2/p3/ --noinherit`
  Commits the files in the directories corresponding to package "p2.p3" and its sub packages. Does not inherit meta data from super packages (in this case, from "p2", if it already exists).

- `regi commit package p2/p3/ --fast`
  Commits the files in the directories corresponding to package "p2.p3" and its sub packages. Uses timestamps and hashes for status calculation.

- `regi commit package p2/p3/ --veryfast`
  Commits the files in the directories corresponding to package "p2.p3" and its sub packages. Only uses timestamps for status calculation.

Related:

- activate (activate inactive objects in the Repository)
- checkOut (check out objects from the Repository)
- create package (create packages)
- list inactiveObjects (list inactive objects)
- push (push objects to the Repository)
- revert (revert inactive objects in the Repository)
- status (show the status of (files in) the Regi workspace)
- takeOver (take over inactive objects from other Repository workspaces)

---

create change

Creates changes.

Usage:

- create change|ch DESCRIPTION DESCRIPTION...

Options:

None.

Examples:

-
  ```
  regi create ch desc1 desc2
    = regi create change desc1 desc2
  ```
  Creates two changes, the first with description "desc1" and the second with description "desc2".

Related:

- activate (activate inactive objects in the Repository)
- delete change (delete changes)
- list change (list entries of changes)
- list changes (list changes)
- merge changes (merge two changes into one)
- move (move objects from one change to another)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- update change (update meta data of changes)

---

create contributor

Creates change contributors.

Usage:

- create contributor|cont CHANGEID CONTRIBUTOR CONTRIBUTOR...

Options:

None.

Examples:

-
  ```
  regi create cont 42 USER1 USER2
    = regi create contributor HDB//42 USER1 USER2
  ```
  Creates the contributors "USER1" and "USER2" for the change "HDB//42".

Related:

- delete contributor (Delete change contributors)
- set changeTrackingStatus (set change tracking status)
- show changeTrackingStatus (show change tracking status)
- update contributor (update meta data of change contributors)

- validate contributor (set status of contributors to "validated")

---

create deliveryUnit

Creates delivery units. Default vendor: Content vendor.

Usage:

- create deliveryUnit|du DELIVERYUNIT DELIVERYUNIT...

Options:

- `--vendor=VENDOR`
  Set vendor (default: content vendor).

- `--caption=CAPTION`
  Set caption (default: empty).

- `--responsible|--resp=RESPONSIBLE`
  Set responsible (default: empty).

- `--version=VERSION`
  Set version (default: 0).

- `--versionSP|--version_sp=VERSIONSP`
  Set SP version (default: 0).

- `--versionPatch|--version_patch=VERSIONPATCH`
  Set patch version (default: 0).

- `--ppmsID=PPMSID`
  Set PPMS ID (default: empty).

- `--spPPMSID=SPPPMSID`
  Set SP PPMS ID (default: empty).

- `--ach=ACH`
  Set ACH (default: empty).

Examples:

- `regi create du testdu1`
  Creates delivery unit "TESTDU1". Vendor = content vendor.

- `regi create du testdu1 testdu2`
  Creates delivery units "TESTDU1" and "TESTDU2". Vendor = content vendor.

- `regi create du testdu1 testdu2 --vendor=test.sap.com`
  Creates delivery units "TESTDU1" and "TESTDU2". Vendor = "test.sap.com".

- `regi create du testdu1 testdu2 --version=1 --resp=me`
  Creates delivery units "TESTDU1" and "TESTDU2". Vendor = content vendor, version = 1, responsible = me.

Related:

- delete deliveryUnit (delete delivery units)
- list deliveryUnits (list delivery units)
- show deliveryUnit (show meta data of delivery units)
- show deliveryUnitStatus (show status of delivery units)
- show vendor (show content vendor)
- update deliveryUnit (update meta data of delivery units)

---

create package

Creates packages.

Usage:

- create package PACKAGE PACKAGE...
- create packages PACKAGE PACKAGE...

Options:

- `--deliveryUnit|--du=DELIVERYUNIT`
  Set delivery unit (default: empty).
- `--description|--desc=DESCRIPTION`
  Set description (default: empty).
- `--hintsForTranslation|--hints=HINTSFORTRANSLATION`
  Set hints for translation (default: empty).
- `--noInherit`
  Do not inherit meta data from super packages.
- `--origLang=ORIGLANG`
  Set original language (default: empty).
- `--responsible|--resp=RESPONSIBLE`
  Set responsible (default: empty).
- `--structural=STRUCTURAL`
  Set structural (default: 0).
- `--textCollection|--textColl=TEXTCOLLECTION`
  Set text collection (default: empty).
- `--textStatus=TEXTSTATUS`
  Set text status (default: empty).
- `--textTerminologyDomain|--textTerm=TEXTTERMINOLOGYDOMAIN`
  Set text terminology domain (default: empty).
- `--vendor=VENDOR`

Set vendor (default: content vendor if delivery unit provided, empty otherwise).

Examples:

- `regi create package p1`
  Creates package "p1".

- `regi create package p1 p2.p3`
  Creates packages "p1" and "p2.p3".

- `regi create package p1 p2.p3 --resp=me`
  Creates packages "p1" and "p2.p3". Sets "responsible" to "me".

- `regi create package p1 p2.p3 --structural=1`
  Creates structural packages "p1" and "p2.p3".

- `regi create package p1 p2.p3 --noinherit`
  Creates packages "p1" and "p2.p3". Does not inherit meta data from super packages (in this case, from "p2", if it already exists).

Related:

- commit (commit files to the Repository)
- delete package (delete packages)
- list packages (list packages in the Repository)
- show package (show meta data of packages)
- update package (update meta data of packages)

---

create workSpace

Creates a Regi workspace and makes a first connection check. Use "__empty__" for the "empty workspace" to access objects developed in the HANA Studio Modeler.

Usage:

- create workSpace|ws WORKSPACEPATH [KEY]

Options:

- `--key=KEY`
  Secure store key for server connection. Keys can be created with the HANA client tool "hdbuserstore".

- `--hostName|--host=HOST:SQLPORT`
  Specify Host:SQL port for the server connection (if no key provided). SQLPORT = 30000 + 15 + 100*INSTANCE, e.g. 34215 for instance 42. For STFK connections, use the local proxy (e.g. "localhost:5001").

- `--user=USER`
  Specify user for the server connection (if no key provided).

- `--password|--passwd=PASSWORD`

Specify password for the server connection (if no key provided).

- `--readFromStdIn|--stdIn`
  Read password from standard input if it is not provided.

- `--force|-f`
  Create the Regi workspace even if the corresponding Repository workspace already contains inactive objects.

Examples:

- `regi create ws myWS key`
  Creates workspace myWS with secure store key "key".

- `regi create ws __empty__ key`
  Creates "empty workspace" with secure store key "key".

- `regi create ws myWS --host=localhost:34215 --user=MYUSER --passwd=myPassword`
  Creates workspace myWS with host=localhost:34215, user=MYUSER, password=myPassword.

- `regi create ws myWS --host=localhost:34215 --user=MYUSER --stdin`
  Creates workspace myWS with host=localhost:34215, user=MYUSER, password is read from standard input.

- `regi create ws myWS key -f`
  Creates workspace myWS with secure store key "key". Creates the workspace even if the corresponding Repository workspace already contains inactive objects.

Related:

- checkOut (check out objects from the Repository)
- status (show the status of (files in) the Regi workspace)
- track allPackages (track all packages in the Repository)
- track package (track a package)

---

delete allInactiveWorkSpaces

Deletes all inactive workspaces. Inactive workspaces of other users can be deleted if the user has the System Privilege "REPO.WORK_IN_FOREIGN_WORKSPACE". Useful if packages cannot be deleted due to left inactive objects. Use "__empty__" for the "empty workspace".

Usage:

- delete allInactiveWorkSpaces|allIWS INACTIVEWORKSPACEUSER...

Options:

- `--inactiveWorkSpaceUser|--iUser=INACTIVEWORKSPACEUSER`
  The inactive workspace user (case sensitive; supports wild cards).

Examples:

- `regi delete alliws THEUSER`
  Deletes all inactive workspaces of user "THEUSER".

- `regi delete alliws "*"`
  `  = regi delete alliws --iuser=*`
  Deletes all inactive workspaces of all users.

Related:

- delete package (delete packages)
- delete inactiveWorkSpace (delete inactive workspaces)
- list inactiveObjects (list inactive objects)
- list inactiveWorkSpaces (list inactive objects)

---

delete change

Deletes changes.

Usage:

- delete change|ch CHANGEID CHANGEID...

Options:

None.

Examples:

- `regi delete ch 42 4711`
  `  = regi delete change HDB//42 HDB//4711`
  Deletes changes "HDB//42" and "HDB//4711".

Related:

- activate (activate inactive objects in the Repository)
- create change (create changes)
- list change (list entries of changes)
- list changes (list changes)
- merge changes (merge two changes into one)
- move (move objects from one change to another)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- update change (update meta data of changes)

---

delete contributor

Deletes change contributors.

Usage:

- delete contributor|cont CHANGEID CONTRIBUTOR CONTRIBUTOR...

Options:

None.

Examples:

- `regi delete cont 42 USER1 USER2`
  `  = regi delete contributor HDB//42 USER1 USER2`
  Deletes the contributors "USER1" and "USER2" from the change "HDB//42".

Related:

- create contributor (create change contributors)
- set changeTrackingStatus (set change tracking status)
- show changeTrackingStatus (show change tracking status)
- update contributor (update meta data of change contributors)
- validate contributor (set status of contributors to "validated")

---

delete deliveryUnit

Deletes delivery units. Default vendor: Content vendor.

Usage:

- delete deliveryUnit|du DELIVERYUNIT DELIVERYUNIT...

Options:

- `--vendor=VENDOR`
  The vendor (default: content vendor).

Examples:

- `regi delete du testdu1`
  Deletes delivery unit "TESTDU1". Vendor = content vendor.

- `regi delete du testdu1 testdu2`

Deletes delivery units "TESTDU1" and "TESTDU2". Vendor = content vendor.

- `regi delete du testdu1 testdu2 --vendor=test.sap.com`
  Deletes delivery units "TESTDU1" and "TESTDU2". Vendor = "test.sap.com".

Related:

- create deliveryUnit (create delivery units)
- list deliveryUnits (list delivery units)
- show deliveryUnit (show meta data of delivery units)
- show deliveryUnitStatus (show status of delivery units)
- show vendor (show content vendor)
- unDeploy (undeploy delivery units from the Repository)
- update deliveryUnit (update meta data of delivery units)

---

delete inactiveWorkSpace

Deletes inactive workspaces. Inactive workspaces of other users can be deleted if the user has the System Privilege "REPO.WORK_IN_FOREIGN_WORKSPACE". Useful if packages cannot be deleted due to left inactive objects. Use "__empty__" for the "empty workspace".

Usage:

- delete inactiveWorkSpace|iws INACTIVEWORKSPACE INACTIVEWORKSPACE...

Options:

- `--inactiveWorkSpaceUser|--iUser=INACTIVEWORKSPACEUSER`
  The inactive workspace user (case sensitive).

Examples:

- `regi delete iws w1 --iuser=THEUSER`
  Deletes the inactive workspace w1 of user "THEUSER".

- `regi delete iws w1 w2 --iuser=THEUSER`
  Deletes the inactive workspaces w1 and w2 of user "THEUSER".

Related:

- delete package (delete packages)
- delete allInactiveWorkSpaces (delete all inactive workspaces)
- list inactiveObjects (list inactive objects)
- list inactiveWorkSpaces (list inactive objects)

---

delete package

Deletes packages.

Usage:

- delete package PACKAGE PACKAGE...
- delete packages PACKAGE PACKAGE...

Options:

- `--force|-f`
  Delete the packages including all sub packages and all contained active and inactive objects in all workspaces. If the HANA Repository has API feature version under 16, the --force option only works if called from inside a Regi workspace, and the contained inactive objects are not deleted in any workspace.

- `--noDeletePackageRecursively`
  Do not use the native Repository functionality to recursively delete packages, even if the HANA Repository has API feature version 16 or higher. Use the non-native Regi functionality to recursively delete packages instead.

- `--noRecursive|--noRec`
  Do not recursively delete sub packages (or any contained active or inactive objects in any workspace).

- `--noUnTrack`
  Do not automatically untrack package arguments (presupposes --force and not --noRecursive.)

Examples:

- `regi delete package p1`
  Deletes package p1 and all sub packages.

- `regi delete package p1 p2.p3`
  Deletes packages p1 and p2.p3 and all sub packages.

- `regi delete package p1 p2.p3 -f`
  Deletes packages p1 and p2.p3 and all sub packages and all contained active and inactive objects.

- `regi delete package p1 p2.p3 --norec`
  Attempts to delete packages p1 and p2.p3, but does not delete their sub packages (or any contained active or inactive objects in any workspace).

Related:

- create package (create packages)
- list packages (list packages in the Repository)
- show package (show meta data of packages)
- unDeploy (undeploy delivery units from the Repository)
- update package (update meta data of packages)

---

discard

Discards checked out objects. Deletes all local meta data and the files corresponding to the objects themselves (similar to "remove from client" in Perforce). Automatically untracks package arguments unless either "--nountrack" or "--norec" is used.

Usage:

- discard inactiveObjectslios
- discard [trackedPackages]
- discard object OBJECT OBJECT...
- discard package PACKAGE PACKAGE...
- discard packages PACKAGE PACKAGE...
- discard [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--force|-f`
  Discard checked out objects in any case (even if the objects have been either created/deleted/modified).
- `--noContent`
  Delete only the meta data of the objects, not the objects themselves.
- `--noRecursive|--noRec`
  Do not recursively discard sub packages.
- `--noUnTrack`
  Do not automatically untrack package arguments.

Examples:

- `regi discard`
  `= regi discard trackedPackages`
  Discards all objects in all tracked packages and their sub packages.
- `regi discard p1/test.txt p2/p3/`
  `= regi discard objects p1/test.txt p2/p3/`
  Discards the object "test.txt" in package "p1" and all objects in package "p2.p3" and its sub packages.
- `regi discard object p1/test.txt`
  Discards the object "test.txt" in package "p1".
- `regi discard object p1/test.txt --nocontent`
  Discards the object "test.txt" in package "p1". Only deletes the meta data of the object, not the object itself.
- `regi discard package p2/p3/`
  Discards all objects in package "p2.p3" and its sub packages.
- `regi discard package p2/p3/ --norec`
  Discards all objects in package "p2.p3" but not its sub packages. Does not automatically untrack the package ("--norec" is used).
- `regi discard package p2/p3/ --nountrack`

Discards all objects in package "p2.p3" and its sub packages. Does not automatically untrack the package ("--nountrack" is used).

Related:

- checkOut (check out objects from the Repository)
- commit (commit files to the Repository)
- create workSpace (create Regi workspace)
- status (show the status of (files in) the Regi workspace)
- unTrack allPackages (untrack all tracked packages)
- unTrack package (untrack a package)

---

export deliveryUnit

Exports a delivery unit. Supports full export and patch export (timestamp based). Default vendor: content vendor.

Usage:

- export [deliveryUnit|du] DELIVERYUNIT [VENDOR] FILENAME

Options:

- `--alias=ALIAS`
  Set alias to anonymize the export.
- `--dUCheck=DUCHECK`
  Set DU check (0 = no check, 1 = version of DU is checked, 3 = version and SP is checked and both must match).
- `--exportVersion|--eV=EXPORTVERSION`
  Set export version to a value less than the maximum for downward compatibility.
- `--from=TIMESTAMP`
  Set lower time boundary for patch export.
- `--log|-l`
  Write export summary to log file in "._SYS_REGI_settings/logs".
- `--released|-r`
  Only export released objects (change tracking).
- `--to=TIMESTAMP`
  Set upper time boundary for patch/released export.
- `--vendor=VENDOR`
  Specify the vendor of the delivery unit if necessary.
- `--verbose|-v`
  Write export summary to standard output.

Examples:

- `regi export testdu testdu.tgz`
  Full DU export of "TESTDU" to file "testdu.tgz". Vendor: content vendor. Export version: maximum.
- `regi export testdu test.sap.com testdu.tgz`
  Full DU export of "TESTDU" to file "testdu.tgz". Vendor: test.sap.com. Export version: maximum.
- `regi export testdu testdu.tgz --alias=SAP`
  Full DU export of "TESTDU" to file "testdu.tgz". Vendor: content vendor. Export version: maximum. Anonymized with alias "SAP".
- `regi export testdu testdu.tgz --alias=SAP --ev=11`
  Full DU export of "TESTDU" to file "testdu.tgz". Vendor: content vendor. Export version: 11. Anonymized with alias "SAP".
- `regi export testdu testdu.tgz -l`
  Full DU export of "TESTDU" to file "testdu.tgz". Vendor: content vendor. Export version: maximum. Writes export summary to log file.
- `regi export testdu testdu.tgz -v`
  Full DU export of "TESTDU" to file "testdu.tgz". Vendor: content vendor. Export version: maximum. Writes export summary to standard output.
- `regi export testdu testdu.tgz --from=2013-09-01 --to=2013-09-07`
  Patch DU export of "TESTDU" to file "testdu.tgz". Vendor: content vendor. Export version: maximum. From: 2013-09-01, to: 2013-09-07.

Related:

- export languages (export languages delivery unit)
- import deliveryUnit (import delivery unit files)
- import languages (import languages delivery unit files)
- list change (list entries of changes)
- list changes (list changes)
- list deliveryUnits (list delivery units)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- show deliveryUnit (show meta data of delivery units)
- show deliveryUnitStatus (show status of delivery units)
- show exportVersions (show supported export versions)
- transport deliveryUnit (Transport delivery units from one system to another)
- transport languages (Transport languages delivery units)

---

export languages

Exports languages delivery unit. Language delivery units contain only translated texts, no content.

Usage:

- export languages|la DELIVERYUNIT VENDOR FILENAME LANGUAGE...

Options:

- `--alias=ALIAS`
  Set alias to anonymize the export.

- `--dUCheck=DUCHECK`
  Set DU check (0 = no check, 1 = version of DU is checked, 3 = version and SP is checked and both must match).

- `--exportVersion|--eV=EXPORTVERSION`
  Set export version to a value less than the maximum for downward compatibility.

- `--log|-l`
  Write export summary log file in "._SYS_REGI_settings/logs".

Examples:

- `regi export la testdu test.sap.com testdu_de.tgz`
  Exports translated texts of DU "TESTDU" to file "testdu_de.tgz" Vendor: "test.sap.com". Export version: maximum. Language: all languages available.

- `regi export la testdu test.sap.com testdu_de.tgz de`
  Exports translated texts of DU "TESTDU" to file "testdu_de.tgz" Vendor: "test.sap.com". Export version: maximum. Language: de.

- `regi export la testdu test.sap.com testdu_de.tgz de --ev=11`
  Exports translated texts of DU "TESTDU" to file "testdu_de.tgz" Vendor: "test.sap.com". Export version: 11. Language: de.

- `regi export la testdu test.sap.com testdu_de.tgz de -l`
  Exports translated texts of DU "TESTDU" to file "testdu_de.tgz" Vendor: "test.sap.com". Export version: maximum. Language: de. Writes export summary log file.

- `regi export la testdu test.sap.com testdu_de_fr.tgz de fr`
  Exports translated texts of DU "TESTDU" to file "testdu_de_fr.tgz" Vendor: content vendor. Export version: maximum. Languages: de, fr.

Related:

- export deliveryUnit (export delivery unit)
- import deliveryUnit (import delivery unit files)
- import languages (import languages delivery unit files)
- list deliveryUnits (list delivery units)
- show deliveryUnit (show meta data of delivery units)
- show deliveryUnitStatus (show status of delivery units)
- show exportVersions (show supported export versions)

- transport deliveryUnit (Transport delivery units from one system to another)
- transport languages (Transport languages delivery units)

---

import deliveryUnit

Imports delivery units. Supports both full DU exports and patch DU exports. Automatically uses multi DU import if available for >1 DUs.

Usage:

- import [deliveryUnit|du|file] FILENAME FILENAME...

Options:

- `--activationMode=N`
  Specify activation mode (default: N = 0).
- `--autoActivate=N`
  0: Do not activate objects after import, 1: Activate after import (default: N = 1).
- `--determineXrefs=N`
  0: Do not determine cross references and texts during import, 1: Determine cross references and texts during import (default: N = 0).
- `--forceRemove=N`
  0: If inactive versions of the objects already exist in the current inactive workspace, the import fails. 1: All inactive versions of the objects in the current workspace are reverted prior to import (default: N = 1).
- `--log|-l`
  Writes import summary to log file in "._SYS_REGI_settings/logs".
- `--noMultiImport`
  Import the DUs one after the other (do not use multi DU import)
- `--onlyTestImport=N`
  0: Normal import 1: Import is only simulated to see what objects are affected (default: N = 0).

Examples:

- `regi import testdu1.tgz testdu2.tgz`
  Imports DUs "testdu1.tgz" and "testdu2.tgz".
- `regi import testdu1.tgz --activationmode=0`
  Imports DU "testdu1.tgz" with activation mode "0".
- `regi import testdu1.tgz --autoactivate=0`
  Imports DU "testdu1.tgz" but does not activate the objects after import.
- `regi import testdu1.tgz --forceremove=0`
  Imports DU "testdu1.tgz" but does not revert the inactive objects prior to import.
- `regi import testdu1.tgz --onlytestimport=1`
  Tests the import of DU "testdu1.tgz".

- `regi import testdu1.tgz --determinexrefs=1`
  Imports DU "testdu1.tgz" and determines the cross references and texts.

- `regi import testdu1.tgz testdu2.tgz -l`
  Imports DUs "testdu1.tgz" and "testdu2.tgz". Writes import summary to log file.

- `regi import testdu1.tgz testdu2.tgz --nomultiimport`
  Imports DUs "testdu1.tgz" and "testdu2.tgz". Does not use multi DU import.

Related:

- activate (activate inactive objects in the Repository)
- export deliveryUnit (export delivery unit)
- export languages (export languages delivery unit)
- import languages (import languages delivery unit files)
- list deliveryUnits (list delivery units)
- show deliveryUnit (show meta data of delivery units)
- show deliveryUnitStatus (show status of delivery units)
- transport deliveryUnit (Transport delivery units from one system to another)
- transport languages (Transport languages delivery units)
- unDeploy (undeploy delivery units from the Repository)

---

import languages

Imports languages delivery unit files.

Usage:

- import languages|la FILENAME FILENAME...

Options:

- `--log|-l`
  Writes import summary to log file in "._SYS_REGI_settings/logs".

- `--onlyTestImport=N`
  0: Normal import, 1: Import is only simulated to see what objects are affected.

Examples:

- `regi import la testdu_de_fr.tgz`
  Imports languages delivery unit file "testdu_de_fr.tgz".

- `regi import la testdu_de_fr.tgz --onlytestimport=1`
  Tests the import of languages delivery unit file "testdu_de_fr.tgz"

.

Related:

- activate (activate inactive objects in the Repository)
- export deliveryUnit (export delivery unit)
- export languages (export languages delivery unit)
- import deliveryUnit (import delivery unit files)
- list deliveryUnits (list delivery units)
- show deliveryUnitStatus (show status of delivery units)
- transport deliveryUnit (Transport delivery units from one system to another)
- transport languages (Transport languages delivery units)
- unDeploy (undeploy delivery units from the Repository)

---

list activeObjects

Lists active objects. Faulty objects are marked as "broken".

Usage:

- list activeObjects|aos [PACKAGE]

Options:

- `--object=OBJECT`
  Set object name pattern (use "__empty__" for the empty object name).
- `--suffix=SUFFIX`
  Set object suffix (must not be a pattern; use "__empty__" for the empty suffix).

Examples:

- `regi list aos`
  Lists all active objects in the Repository.
- `regi list aos p1`
  Lists all active objects of the Repository in package "p1".
- `regi list aos 'p1*'`
  Lists all active objects of the Repository in packages matching "p1*".
- `regi list aos 'p1*' --object=test* --suffix=txt`
  Lists all active objects of the Repository in packages matching "p1*", with object names matching "test*" and suffix "txt".

Related:

- activate (activate inactive objects in the Repository)
- checkOut (check out objects from the Repository)

- list inactiveObjects (list inactive objects)
- status (show the status of (files in) the Regi workspace)

---

list trackedPackages

Lists the tracked packages.

Usage:

- list trackedPackages

Options:

None.

Examples:

- `regi list trackedpackages`
  Lists the tracked packages.

Related:

- status (show the status of (files in) the Regi workspace)
- track allPackages (track all packages in the Repository)
- track package (track a package)
- unTrack allPackages (untrack all tracked packages)
- unTrack package (untrack a package)

---

list change

Lists entries of changes.

Usage:

- list change|ch CHANGEID CHANGEID...

Options:

None.

Examples:

-
  ```
  regi list ch 42 4711
    = regi list change HDB//42 HDB//4711
  ```
  Lists the entries of changes "HDB//42" and "HDB//4711".

Related:

- activate (activate inactive objects in the Repository)
- create contributor (create change contributors)
- create change (create changes)
- delete change (delete changes)
- delete contributor (Delete change contributors)
- export deliveryUnit (export delivery unit)
- list changes (list changes)
- merge changes (merge two changes into one)
- move (move objects from one change to another)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- transport deliveryUnit (Transport delivery units from one system to another)
- update change (update meta data of changes)
- update contributor (update meta data of change contributors)
- validate contributor (set status of contributors to "validated")

---

list changes

Lists changes.

Usage:

- list changes|chs

Options:

- `--changeId|--ch=CHANGEID`
  Specify the change ID.

- `--srcSystem=SRCSYSTEM`
  Specify the source system.

- `--changeNumber=CHANGENUMBER`
  Specify the change number.

- `--changeStatus=CHANGESTATUS`
  Specify the change status (0 = undefined, 1 = open, 2 = released, 3 = abandoned, 4 = eoct = end of change tracking).

- `--open|-o`
  Specify change status 1 (open change).

- `--released|-r`
  Specify change status 2 (released change).

- `--abandoned|-a`
  Specify change status 3 (abandoned change).

- `--contributor|--cont=CONTRIBUTOR`
  Specify the contributor.

- `--contributorStatus=CONTRIBUTORSTATUS`
  Specify the contributor status (0 = undefined, 1 = open, 2 = validated).

- `--createdBy=CREATEDBY`
  Specify created by.

- `--releasedFrom=RELEASEDFROM`
  Specify released from.

- `--releasedTo=RELEASEDTO`
  Specify released to.

Examples:

-
  ```
  regi list chs
    = regi list changes
  ```
  Lists all changes.

-
  ```
  regi list chs -o
    = regi list changes --open
    = regi list changes --changestatus=1
  ```
  Lists all open changes.

-
  ```
  regi list chs -o --cont=USER1
    = regi list changes --open --cont=USER1
    = regi list changes --changestatus=1 --cont=USER1
    = regi list changes --changestatus=1 --contributor=USER1
  ```
  Lists all open changes in which user "USER1" is a contributor.

Related:

- activate (activate inactive objects in the Repository)
- create change (create changes)
- delete change (delete changes)
- list change (list entries of changes)
- merge changes (merge two changes into one)
- move (move objects from one change to another)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- update change (update meta data of changes)

---

list deliveryUnit

Lists the packages in a delivery unit. Default vendor: content vendor.

Usage:

- list deliveryUnit|du DELIVERYUNIT DELIVERYUNIT...

Options:

- `--vendor=VENDOR`
  Specify the vendor of the delivery unit if necessary.

Examples:

- `regi list du testdu`
  Lists packages in delivery unit "TESTDU". Vendor: content vendor.

- `regi list du testdu1 testdu2 --vendor=test.sap.com`
  Lists packages in delivery units "TESTDU1" and "TESTDU2". Vendor: "test.sap.com".

Related:

- assign (assign packages to delivery units)
- unAssign (unassign packages from their delivery units)
- update package (update meta data of packages)

---

list deliveryUnits

Lists the delivery units in the Repository.

Usage:

- list deliveryUnits|dus

Options:

- `--vendor=VENDOR`
  Specify the vendor of the delivery units to list.

Examples:

- `regi list dus`
  Lists the delivery units in the Repository. If as in this case no vendor is provided, the output is sorted by the vendors of the delivery units.

- `regi list dus --vendor=sap.com`
  Lists those delivery units in the Repository which have vendor "sap.com". If as in this case a vendor is provided, the output is a plain list of delivery units.

Related:

- create deliveryUnit (create delivery units)
- delete deliveryUnit (delete delivery units)
- import deliveryUnit (import delivery unit files)
- unDeploy (undeploy delivery units from the Repository)

---

list history

Lists history of objects.

Usage:

- list history OBJECT OBJECT...

Options:

None.

Examples:

- `regi list history p1/test.txt p2/p3/test.html`
  Lists history of the object "test.txt" in package "p1", and object "test.html" in package "p2.p3".

Related:

- checkOut (check out objects from the Repository)
- show object (prints a version of an object to standard output)

---

list inactiveObjects

Lists inactive objects of a Repository workspace. Default: Workspace name and workspace user corresponding to the Regi workspace.

Usage:

- list inactiveObjectslios [PACKAGE]

Options:

- `--inactiveWorkSpace|--iWorkSpace|--iWS=INACTIVEWORKSPACE`
  Set inactive workspace name (use "__empty__" for the "empty workspace").
- `--inactiveWorkSpaceUser|--iUser=INACTIVEWORKSPACEUSER`
  Set inactive workspace user (case sensitive).

- `--object=OBJECT`
  Set object name pattern (use "__empty__" for the empty object name).

- `--suffix=SUFFIX`
  Set object suffix (must not be a pattern; use "__empty__" for the empty suffix).

Examples:

- `regi list ios`
  Lists inactive objects of the Repository workspace corresponding to the Regi workspace.

- `regi list ios p1`
  Lists inactive objects of the Repository workspace corresponding to the Regi workspace. Restrict the search to package p1.

- `regi list ios 'p1*'`
  Lists inactive objects of the Repository workspace corresponding to the Regi workspace. Restrict the search to all packages matching "p1*".

- `regi list ios 'p1*' --object=test* --suffix=txt`
  Lists inactive objects of the Repository workspace corresponding to the Regi workspace. Restrict the search to all packages matching "p1*", with object names matching "test*" and suffix "txt".

- `regi list ios --iws=otherW --iuser=OTHERUSER`
  Lists inactive objects of the Repository workspace identified with workspace name "otherW" and WorkSpace user "OTHERUSER".

Related:

- activate (activate inactive objects in the Repository)
- checkOut (check out objects from the Repository)
- commit (commit files to the Repository)
- delete allInactiveWorkSpaces (delete all inactive workspaces)
- delete inactiveWorkSpace (delete inactive workspaces)
- list activeObjects (list active objects)
- list inactiveWorkSpaces (list inactive objects)
- revert (revert inactive objects in the Repository)
- status (show the status of (files in) the Regi workspace)
- takeOver (take over inactive objects from other Repository workspaces)

---

list inactiveWorkSpaces

Lists inactive workspace in the Repository. Default inactive workspace user: Workspace user corresponding to the Regi workspace user. Supports wild cards for the workspace user. Inactive workspaces of other users can be listed if the user has the System Privilege "REPO.WORK_IN_FOREIGN_WORKSPACE". "__empty__" stands for the "empty workspace".

Usage:

- list inactiveWorkSpaces|iWS [INACTIVEWORKSPACEUSER]

Options:

- `--inactiveWorkSpaceUser|--iUser=INACTIVEWORKSPACEUSER`
  Set inactive workspace user (case sensitive; supports wild cards).

Examples:

- `regi list iws`
  Lists inactive workspaces of the workspace user corresponding to the Regi workspace user. If as in this case no wild card is used, the output is a plain list of inactive workspaces.

-
  ```
  regi list iws "*"
    = regi list iws --iuser=*
  ```
  Lists inactive workspaces of all users. If as in this case a wild card is used, the output is sorted by the inactive workspace users.

Related:

- delete allInactiveWorkSpaces (delete all inactive workspaces)
- delete inactiveWorkSpace (delete inactive workspaces)
- list inactiveObjects (list inactive objects)
- takeOver (take over inactive objects from other Repository workspaces)

---

list rootPackages

Lists the root packages in the Repository.

Usage:

- list rootPackages

Options:

None.

Examples:

- `regi list rootpackages`
  Lists the root packages in the Repository.

Related:

- list packages (list packages in the Repository)
- track allPackages (track all packages in the Repository)
- track package (track a package)

---

list packages

Lists packages in the Repository. The list can be filtered (supports wild cards).

Usage:

- list packages [PACKAGE]

Options:

- `--deliveryUnit|--du=DELIVERYUNIT`
  Filter by delivery unit (default: "*").

- `--description|--desc=DESCRIPTION`
  Filter by description (default: "*").

- `--hintsForTranslation|--hints=HINTSFORTRANSLATION`
  Filter by hints for translation (default: "*").

- `--origLang=ORIGLANG`
  Filter by original language (default: "*").

- `--responsible|--resp=RESPONSIBLE`
  Filter by responsible (default: "*").

- `--srcSystem=SRCSYSTEM`
  Filter by source system (default: "*").

- `--textCollection|--textColl=TEXTCOLLECTION`
  Filter by text collection (default: "*").

- `--textStatus=TEXTSTATUS`
  Filter by text status (default: "*").

- `--textTerminologyDomain|--textTerm=TEXTTERMINOLOGYDOMAIN`
  Filter by text terminology domain (default: "*").

- `--vendor=VENDOR`
  Filter by vendor (default: "*").

Examples:

- `regi list packages`
  Lists all packages in the Repository.

- `regi list packages --desc=myDescription`
  Lists all packages with description "myDescription".

- `regi list packages --desc=myDescription --origLang=en`
  Lists all packages with description "myDescription" and original language "en".

Related:

- create package (create packages)

- delete package (delete packages)
- list rootPackages (list root packages)
- track allPackages (track all packages in the Repository)
- track package (track a package)

---

merge changes

Merges two changes into one.

Usage:

- merge changes|chs SOURCECHANGEID TARGETCHANGEID

Options:

None.

Examples:

-
  ```
  regi merge ch 42 4711
    = regi merge changes HDB//42 HDB//4711
  ```
  Merges changes "HDB//42" and "HDB//4711".

Related:

- activate (activate inactive objects in the Repository)
- create change (create changes)
- delete change (delete changes)
- list change (list entries of changes)
- list changes (list changes)
- move (move objects from one change to another)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- update change (update meta data of changes)

---

mergeTool

Interactively resolves conflicts using a configurable three-way merge tool. Alias for "regi resolve --mergetoollm".

Usage:

- mergeTool inactiveObjectslios
- mergeTool [trackedPackages]
- mergeTool object OBJECT OBJECT...
- mergeTool package PACKAGE PACKAGE...
- mergeTool packages PACKAGE PACKAGE...
- mergeTool [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

See "regi resolve".

Examples:

See "regi resolve".

Related:

- activate (activate inactive objects in the Repository)
- checkOut (check out objects from the Repository)
- commit (commit files to the Repository)
- push (push objects to the Repository)
- resolve ((non-)interactively resolve conflicts)
- status (show the status of (files in) the Regi workspace)

---

move

Moves objects from one change to another.

Usage:

- move inactiveObjectslios SOURCECHANGEID TARGETCHANGEID
- move [trackedPackages] SOURCECHANGEID TARGETCHANGEID
- move object SOURCECHANGEID TARGETCHANGEID OBJECT OBJECT...
- move package SOURCECHANGEID TARGETCHANGEID PACKAGE PACKAGE...
- move packages SOURCECHANGEID TARGETCHANGEID PACKAGE PACKAGE...
- move [objects] SOURCECHANGEID TARGETCHANGEID OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--noRecursive|--noRec`
  Do not recursively move objects.

Examples:

- `regi move 42 4711`
  `  = regi move HDB//42 HDB//4711 trackedPackages`
  Moves all objects from change "HDB//42" to "HDB//4711" which are contained in one of the tracked packages.

- `regi move 42 4711 p1/test.txt p2/p3/`
  `  = regi move objects HDB//42 HDB//4711 p1/test.txt p2/p3/`
  Moves all objects from change "HDB//42" to "HDB//4711" which either equal the object "test.txt" in package "p1" or which are contained in package "p2.p3" and its sub packages.

- `regi move 42 4711 p1/test.txt p2/p3/ --norec`
  `  = regi move objects HDB//42 HDB//4711 p1/test.txt p2/p3/ --norec`
  Moves all objects from change "HDB//42" to "HDB//4711" which either equal the object "test.txt" in package "p1" or which are contained in package "p2.p3" but not its sub packages.

- `regi move 42 4711 ios`
  `  = regi move HDB//42 HDB//4711 inactiveObjects`
  Moves all objects from change "HDB//42" to "HDB//4711" which are contained in the set of inactive objects.

- `regi move package 42 4711 p2/p3/`
  Moves all objects from change "HDB//42" to "HDB//4711" which are contained in package "p2.p3" and its sub packages.

Related:

- activate (activate inactive objects in the Repository)
- create change (create changes)
- delete change (delete changes)
- list change (list entries of changes)
- list changes (list changes)
- merge changes (merge two changes into one)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- update change (update meta data of changes)

---

push

Pushes objects to the Repository. 1) Commits, 2) Activates, 3) If a) There are conflicts, and b) Either --base, --local, --remote or --mergetooll-m Then 4) (Non-)interactively resolves, 5) Commits, 6) Activates.

Usage:

- push inactiveObjectslios

- push [trackedPackages]
- push object OBJECT OBJECT...
- push package PACKAGE PACKAGE...
- push packages PACKAGE PACKAGE...
- push [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--activationMode=N`
  Specify activation mode (default: N = 0). Call "regi help activationMode" for more information.

- `--base`
  Non-interactively resolve to the base (=last checked out active) version.

- `--force|-f`
  Force = --local + --noCompare|--nc + --activationMode=4.

- `--local`
  Non-interactively resolve to the local version.

- `--log|-l`
  Write check results to log file in "._SYS_REGI_settings/logs".

- `--mergetool|-m`
  Interactively resolve using a configurable three-way merge tool.

- `--mergeToolConfig=MERGETOOLCONFIG`
  Configure the three-way merge tool (see "regi help mergetoolconfig")

- `--noCompare|--nc`
  Compare neither timestamps nor hashes in status calculation, i.e., push all files, including those which have not been changed. Useful for forced re-generation of objects.

- `--noConfirm`
  Do not ask for confirmation of interactive merges (assume "y" if the merge tool has modified the file, "n" otherwise).

- `--noRecursive|--noRec`
  Do not recursively push sub packages.

- `--remote`
  Non-interactively resolve to the remote version (=the active version at the time of check out).

- `--verbose|-v`
  Write check results to standard output.

Examples:

-
  ```
  regi push
    = regi push trackedPackages
  ```
  Pushes all objects in all tracked packages and their sub packages to the Repository.

-
  ```
  regi push -m
    = regi push trackedPackages -m
  ```
  Pushes all objects in all tracked packages and their sub packages to the Repository. In case of conflicts, interactively resolves using the configured three-way merge tool and then commits and activates again.

- `regi push --local`
  ```
    = regi push trackedPackages --local
  ```
  Pushes all objects in all tracked packages and their sub packages to the Repository. In case of conflicts, non-interactively resolves to the local version and then commits and activates again.

- `regi push -f`
  ```
    = regi push trackedPackages -f
  ```
  Pushes all objects in all tracked packages and their sub packages to the Repository. In case of conflicts, non-interactively resolves to the local version and then commits and activates again. Pushes all files, including those which have not been changed. Sets activation mode to 4.

- `regi push p1/test.txt p2/p3/`
  ```
    = regi push objects p1/test.txt p2/p3/
  ```
  Pushes the object "test.txt" in package "p1" and all objects in package "p2.p3" and its sub packages to the Repository.

- `regi push object p1/test.txt`
  Pushes the object "test.txt" in package "p1" to the Repository.

- `regi push package p2/p3/`
  Pushes all objects in package "p2.p3" and its sub packages to the Repository.

- `regi push package p2/p3/ --norec`
  Pushes all objects in package "p2.p3" but not its sub packages to the Repository.

Related:

- activate (activate inactive objects in the Repository)
- checkOut (check out objects from the Repository)
- commit (commit files to the Repository)
- mergeTool (interactively resolve conflicts)
- resolve ((non-)interactively resolve conflicts)
- status (show the status of (files in) the Regi workspace)

---

release change

Release changes.

Usage:

- release [change|ch] CHANGEID CHANGEID...

Options:

None.

Examples:

- `regi release 42 4711`
  ```
    = regi release change HDB//42 HDB//4711
  ```

Releases changes "HDB//42" and "HDB//4711".

Related:

- activate (activate inactive objects in the Repository)
- create change (create changes)
- delete change (delete changes)
- export deliveryUnit (export delivery unit)
- list change (list entries of changes)
- list changes (list changes)
- merge changes (merge two changes into one)
- move (move objects from one change to another)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- transport deliveryUnit (Transport delivery units from one system to another)
- update change (update meta data of changes)

---

resolve

(Non-)interactively resolves conflicts.

Usage:

- resolve inactiveObjectslios
- resolve [trackedPackages]
- resolve object OBJECT OBJECT...
- resolve package PACKAGE PACKAGE...
- resolve packages PACKAGE PACKAGE...
- resolve [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--base`
  Non-interactively resolve to the base (=last checked out active) version.
- `--local`
  Non-interactively resolve to the local version.
- `--mergetool|-m`
  Interactively resolve using a configurable three-way merge tool.
- `--mergeToolConfig=MERGETOOLCONFIG`
  Configure the three-way merge tool (see "regi help mergetoolconfig")

- `--noConfirm`
  Do not ask for confirmation of interactive merges (assume "y" if the merge tool has modified the file, "n" otherwise).

- `--noRecursive|--noRec`
  Do not recursively resolve sub packages.

- `--remote`
  Non-interactively resolve to the remote version (=the active version at the time of check out).

Examples:

- `regi resolve --local`
  `  = regi resolve trackedPackages --local`
  Non-interactively resolves all objects in all tracked packages and their sub packages to the local version.

- `regi resolve -m`
  `  = regi resolve trackedPackages -m`
  Interactively resolves all objects in all tracked packages and their sub packages using the configured three-way merge tool.

- `regi resolve p1/test.txt p2/p3/ --remote`
  `  = regi resolve objects p1/test.txt p2/p3/ --remote`
  Non-interactively resolves the object "test.txt" in package "p1" and all objects in package "p2.p3" and its sub packages to the remote version.

- `regi resolve object p1/test.txt --base`
  Non-interactively resolves the object "test.txt" in package "p1" to the base version.

- `regi resolve object p1/test.txt --local`
  Non-interactively resolves the object "test.txt" in package "p1" to the local version.

- `regi resolve object p1/test.txt -m`
  Interactively resolves the object "test.txt" in package "p1" using the configured three-way merge tool.

- `regi resolve package p2/p3/ --remote`
  Non-interactively resolves all objects in package "p2.p3" and its sub packages to the remote version.

- `regi resolve package p2/p3/ --base --norec`
  Non-interactively resolves all objects in package "p2.p3" but not its sub packages to the base version.

Related:

- activate (activate inactive objects in the Repository)
- checkOut (check out objects from the Repository)
- commit (commit files to the Repository)
- mergeTool (interactively resolve conflicts)
- push (push objects to the Repository)
- status (show the status of (files in) the Regi workspace)

---

revert

Reverts inactive objects (of the current workspace) in the Repository. 1) deletes the inactive versions of the objects in the Repository. 2) checks out the base version of the objects, if there is one. (unless "revert inactiveObjects|ios" is called or --noCheckOut is used)

Usage:

- revert inactiveObjects|ios
- revert [trackedPackages]
- revert object OBJECT OBJECT...
- revert package PACKAGE PACKAGE...
- revert packages PACKAGE PACKAGE...
- revert [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--noCheckOut`
  Do not check out the base versions of the objects.
- `--noRecursive|--noRec`
  Do not recursively revert.

Examples:

-
  ```
  regi revert
    = regi revert trackedPackages
  ```
  Reverts all inactive objects in all tracked packages and their sub packages.

-
  ```
  regi revert --noCheckOut
    = regi revert trackedPackages --noCheckOut
  ```
  Deletes the inactive versions of the objects in all tracked packages and their sub packages. Does not check out the base versions of the objects.

-
  ```
  regi revert p1/test.txt p2/p3/
    = regi revert objects p1/test.txt p2/p3/
  ```
  Reverts the inactive object "test.txt" in package "p1" and reverts all inactive objects in package "p2.p3" and its sub packages.

-
  ```
  regi revert p1/test.txt p2/p3/ --norec
    = regi revert objects p1/test.txt p2/p3/ --norec
  ```
  Reverts the inactive object "test.txt" in package "p1" and reverts all inactive objects in package "p2.p3" but not its sub packages.

-
  ```
  regi revert ios
    = regi revert inactiveObjects
  ```
  Deletes all inactive objects in all packages. Does not check out the base versions of the objects.

-
  ```
  regi revert object p1/test.txt
  ```
  Reverts the inactive object "test.txt" in package "p1".

-
  ```
  regi revert package p2/p3/
  ```
  Reverts all inactive objects in package "p2.p3" and its sub packages.

Related:

- activate (activate inactive objects in the Repository)
- commit (commit files to the Repository)
- list inactiveObjects (list inactive objects)
- status (show the status of (files in) the Regi workspace)
- takeOver (take over inactive objects from other Repository workspaces)

---

set changeTrackingStatus

Sets change tracking status (enabled/disabled).

Usage:

- set changeTrackingStatus|cts enabled|disabled

Options:

- `--force|-f`
  Disable change tracking even if there are still open changes.

Examples:

-
  ```
  regi set cts enabled
    = regi set changetrackingstatus enabled
  ```
  Enables change tracking.

-
  ```
  regi set cts disabled
    = regi set changetrackingstatus disabled
  ```
  Disables change tracking.

-
  ```
  regi set cts disabled -f
    = regi set changetrackingstatus disabled --force
  ```
  Disables change tracking even if there are still open changes.

Related:

- activate (activate inactive objects in the Repository)
- show changeTrackingStatus (show change tracking status)

---

show change

Shows meta data and contributors of changes.

Usage:

- show change|ch CHANGEID CHANGEID...

Options:

None.

Examples:

- `regi show ch 42 4711`
  `  = regi show change HDB//42 HDB//4711`
  Shows meta data and contributors of changes "HDB//42" and "HDB//4711".

Related:

- create change (create changes)
- delete change (delete changes)
- list change (list entries of changes)
- list changes (list changes)
- merge changes (merge two changes into one)
- move (move objects from one change to another)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show changeTrackingStatus (show change tracking status)
- update change (update meta data of changes)

---

show changeTrackingStatus

Shows change tracking status.

Usage:

- show changeTrackingStatus|cts

Options:

None.

Examples:

- `regi show cts`
  Shows change tracking status.

Related:

- activate (activate inactive objects in the Repository)
- set changeTrackingStatus (set change tracking status)

---

show deliveryUnit

Shows meta data of delivery units. Default vendor: content vendor.

Usage:

- show deliveryUnit|du DELIVERYUNIT DELIVERYUNIT...

Options:

- `--vendor=VENDOR`
  Specify the vendor of the delivery units if necessary.

Examples:

- `regi show du testdu`
  Shows meta data of DU "TESTDU". Vendor: content vendor.

- `regi show du testdu1 testdu2 --vendor=test.sap.com`
  Shows meta data of DUs "TESTDU1" and "TESTDU2". Vendor: "test.sap.com"

Related:

- create deliveryUnit (create delivery units)
- delete deliveryUnit (delete delivery units)
- show deliveryUnitStatus (show status of delivery units)
- update deliveryUnit (update meta data of delivery units)

---

show deliveryUnitStatus

Shows the status of delivery units. The status is "broken" if one of the objects in the delivery unit could not be successfully activated (successfully activated = object status "0"). The status is "ok" otherwise. Default vendor: content vendor.

Usage:

- show deliveryUnitStatus|duStatus DELIVERYUNIT DELIVERYUNIT...

Options:

- `--vendor=VENDOR`
  Specify the vendor of the delivery units if necessary.

Examples:

- `regi show dustatus testdu`

Shows status of DU "TESTDU". Vendor: content vendor.

- `regi show dustatus testdu1 testdu2 --vendor=test.sap.com`
  Shows status of DUs "TESTDU1" and "TESTDU2". Vendor: "test.sap.com"

Related:

- create deliveryUnit (create delivery units)
- delete deliveryUnit (delete delivery units)
- show deliveryUnit (show meta data of delivery units)
- update deliveryUnit (update meta data of delivery units)

---

show exportVersions

Shows the minimum required and the maximum supported export versions.

Usage:

- show exportVersions|evs

Options:

None.

Examples:

-
  ```
  regi show evs
    = regi show exportversions
  ```
  Shows the minimum required and the maximum supported export versions.

Related:

- export deliveryUnit (export delivery unit)
- export languages (export languages delivery unit)
- transport deliveryUnit (Transport delivery units from one system to another)
- transport languages (Transport languages delivery units)

---

show object

Prints a version of an object to standard output.

Usage:

- show [object] OBJECT VERSION

Options:

None.

Examples:

- `regi show p1/test.txt a`
  `= regi show object p1/test.txt active`
  Prints the active version of object "test.txt" in package "p1" to standard output.

- `regi show p1/test.txt i`
  `= regi show object p1/test.txt inactive`
  Prints the inactive version of object "test.txt" in package "p1" to standard output.

- `regi show p1/test.txt 42`
  `= regi show object p1/test.txt 42`
  Prints historical version 42 of object "test.txt" in package "p1" to standard output.

Related:

- checkOut (check out objects from the Repository)
- list history (list history of objects)

---

show package

Shows meta data of packages.

Usage:

- show package PACKAGE PACKAGE...

Options:

None.

Examples:

- `regi show package p1 p2.p3`
  Shows meta data of packages "p1" and "p2.p3".

Related:

- create package (create packages)
- delete package (delete packages)
- update package (update meta data of packages)

---

show vendor

Shows the content vendor of the Repository.

Usage:

- show vendor

Options:

None.

Examples:

- `regi show vendor`
  Shows the content vendor of the Repository.

Related:

- create deliveryUnit (create delivery units)
- delete deliveryUnit (delete delivery units)
- export deliveryUnit (export delivery unit)
- export languages (export languages delivery unit)
- list deliveryUnit (list delivery unit)
- show deliveryUnit (show meta data of delivery units)
- show deliveryUnitStatus (show status of delivery units)
- unDeploy (undeploy delivery units from the Repository)
- update deliveryUnit (update meta data of delivery units)

---

status

Shows the status of (files in) the Regi workspace.

Usage:

- status inactiveObjects|ios
- status [trackedPackages]
- status object OBJECT OBJECT...
- status package PACKAGE PACKAGE...
- status packages PACKAGE PACKAGE...
- status workSpace|ws
- status [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--fast`

Use timestamps and hashes for status calculation. Fast, but not recommended for writing scripts with Regi. Recommended for non-scripted Regi use.

- `--noCompare|--nc`
  Compare neither timestamps nor hashes in status calculation, i.e., list all files as changed, even if they have not changed.

- `--inactiveObjects|-i`
  List inactive objects.

- `--noRecursive|--noRec`
  Do not recursively show the status of the files.

- `--veryFast`
  Only use timestamps for status calculation. Very fast, but not recommended for writing scripts with Regi.

Examples:

- `regi status`
  `  = regi status trackedPackages`
  Shows the status of all files in all directories corresponding to the tracked packages and their sub packages.

- `regi status --fast`
  `  = regi status trackedPackages --fast`
  Shows the status of all files in all directories corresponding to the tracked packages and their sub packages. Uses the fast mode for status calculation.

- `regi status p1/test.txt p2/p3/`
  `  = regi status objects p1/test.txt p2/p3/`
  Shows the status of the files "test.txt" in the directory corresponding to package "p1" and all files in the directories corresponding to package "p2.p3" and its sub packages.

- `regi status object p1/test.txt`
  Shows the status of the file "test.txt" in the directory corresponding to package "p1".

- `regi status package p2/p3/`
  Shows the status of the files in the directories corresponding to package "p2.p3" and its sub packages.

- `regi status package p2/p3/ --norec`
  Shows the status of the files in the directories corresponding to package "p2.p3" but not its sub packages.

Related:

- activate (activate inactive objects in the Repository)
- checkOut (check out objects from the Repository)
- commit (commit files to the Repository)
- discard (discard checked out objects)
- list activeObjects (list active objects)
- list trackedPackages (list tracked packages)
- list inactiveObjects (list inactive objects)
- mergeTool (interactively resolve conflicts)
- push (push objects to the Repository)
- resolve ((non-)interactively resolve conflicts)

- revert (revert inactive objects in the Repository)
- takeOver (take over inactive objects from other Repository workspaces)
- track allPackages (track all packages in the Repository)
- track package (track a package)
- unTrack allPackages (untrack all tracked packages)
- unTrack package (untrack a package)
- version (show Regi version)

---

takeOver

Takes over inactive objects from other Repository workspaces. 1) Takes over inactive objects, 2) Checks out the inactive objects (unless "takeover inactiveObjectslios" is called or --noCheckOut is used).

Usage:

- takeOver inactiveObjectslios
- takeOver [trackedPackages]
- takeOver object OBJECT OBJECT...
- takeOver package PACKAGE PACKAGE...
- takeOver packages PACKAGE PACKAGE...
- takeOver [objects] OBJECT_OR_PACKAGE OBJECT_OR_PACKAGE...

Options:

- `--inactiveWorkSpace|--iWorkSpace|--iWS=INACTIVEWORKSPACE`
  Set inactive workspace name.
- `--inactiveWorkSpaceUser|--iUser=INACTIVEWORKSPACEUSER`
  Set inactive workspace user (case sensitive).
- `--noCheckOut`
  Do not check out the inactive versions of the objects.
- `--noRecursive|--noRec`
  Do not recursively take over.

Examples:

-
  ```
  regi takeover --iws=w1 --iuser=USER1
    = regi takeover trackedPackages --iws=w1 --iuser=USER1
  ```
  Takes over all inactive objects in all tracked packages and their sub packages from the Repository workspace "w1" of user "USER1".

-
  ```
  regi takeover --noCheckOut --iws=w1 --iuser=USER1
    = regi takeover trackedPackages --noCheckOut --iws=w1 --iuser=USER1
  ```
  Takes over the inactive versions of the objects in all tracked packages and their sub packages from the Repository workspace "w1" of user "USER1". Does not check out the taken over inactive objects.

- `regi takeover p1/test.txt p2/p3/ --iws=w1 --iuser=USER1`
  `  = regi takeover objects p1/test.txt p2/p3/ --iws=w1 --iuser=USER1`
  Takes over the inactive object "test.txt" in package "p1" and all inactive objects in package "p2.p3" and its sub packages from the Repository workspace "w1" of user "USER1".

- `regi takeover p1/test.txt p2/p3/ --iws=w1 --iuser=USER1 --norec`
  `  = regi takeover objects p1/test.txt p2/p3/ --iws=w1 --iuser=USER1 --norec`
  Takes over the inactive object "test.txt" in package "p1" and all inactive objects in package "p2.p3" but not its sub packages from the Repository workspace "w1" of user "USER1".

- `regi takeover ios --iws=w1 --iuser=USER1`
  `  = regi takeover inactiveObjects --iws=w1 --iuser=USER1`
  Takes over all inactive objects in all packages from the Repository workspace "w1" of user "USER1". Does not check out the taken over inactive objects.

- `regi takeover object p1/test.txt --iw=w1 --iuser=USER1`
  Takes over the inactive object "test.txt" in package "p1" from the Repository workspace "w1" of user "USER1".

- `regi takeover package p2/p3/ --iws=w1 --iuser=USER1`
  Takes over the inactive objects in package "p2.p3" and its sub packages from the Repository workspace "w1" of user "USER1".

Related:

- checkOut (check out objects from the Repository)
- commit (commit files to the Repository)
- list inactiveObjects (list inactive objects)
- list inactiveWorkSpaces (list inactive objects)
- revert (revert inactive objects in the Repository)
- status (show the status of (files in) the Regi workspace)

---

track allPackages

Tracks all root packages in the Repository. Tracking a package means that the package and all of its sub packages are tracked.

Usage:

- track allPackages|rootPackages

Options:

None.

Examples:

- `regi track allpackages`
  Tracks all root packages in the Repository.

Related:

- create workSpace (create Regi workspace)
- list trackedPackages (list tracked packages)
- list rootPackages (list root packages)
- list packages (list packages in the Repository)
- status (show the status of (files in) the Regi workspace)
- track package (track a package)
- unTrack allPackages (untrack all tracked packages)
- unTrack package (untrack a package)

---

track package

Tracks a package. Tracking a package means that the package and all of its sub packages are tracked.

Usage:

- track [package|packages] PACKAGE PACKAGE...

Options:

None.

Examples:

-
  ```
  regi track p1 p2.p3
    = regi track package p1 p2.p3
  ```
  Tracks packages "p1" and "p2.p3" and all of their sub packages.

Related:

- create workSpace (create Regi workspace)
- list trackedPackages (list tracked packages)
- list rootPackages (list root packages)
- list packages (list packages in the Repository)
- status (show the status of (files in) the Regi workspace)
- track allPackages (track all packages in the Repository)
- unTrack allPackages (untrack all tracked packages)
- unTrack package (untrack a package)

---

transport deliveryUnit

Transports a delivery units from one system to another. Combines export and import. Connection keys/host names/user names/passwords default to the currently set keys/host names/user names/passwords. Default vendor: content vendor.

Usage:

- transport [deliveryUnit|du] DELIVERYUNIT DELIVERYUNIT...

Options:

(export)

- `--alias=ALIAS`
  Set alias to anonymize the export.
- `--dUCheck=DUCHECK`
  Set DU check (0 = no check, 1 = version of DU is checked, 3 = version and SP is checked and both must match).
- `--exportVersion|--eV=EXPORTVERSION`
  Set export version to a value less than the maximum for downward compatibility.
- `--from=TIMESTAMP`
  Set lower time boundary for patch export.
- `--released|-r`
  Only export released objects (change tracking).
- `--sourceKey|--skey=SOURCEKEY`
  Secure store key for the source system connection.
- `--sourceHostName|--sourceHost=SOURCEHOST`
  Host:SQL port for the source system connection (if no key provided)
- `--sourceUserName|--sourceUser=SOURCEUSER`
  User for the source system connection. (if no key provided)
- `--sourcePassWord=SOURCEPASSWORD`
  Password for the source system connection. (if no key provided)
- `--sourceReadFromStdIn|--sourceStdIn`
  Read password for the source system connection from standard input if necessary.
- `--to=TIMESTAMP`
  Set upper time boundary for patch/released export.
- `--vendor=VENDOR`
  Specify the vendor of the delivery unit if necessary.
- `--verbose|-v`
  Write export summary to standard output.

(import)

- `--activationMode=N`
  Specify activation mode (default: N = 4).
- `--autoActivate=N`

0: Do not activate objects after import, 1: Activate after import (default: N = 1).

- `--forceRemove=N`
  0: If inactive versions of the objects already exist in the current inactive workspace, the import fails. 1: All inactive versions of the objects in the current workspace are reverted prior to import (default: N = 1).

- `--noMultiImport`
  Import the DUs one after the other (do not use multi DU import)

- `--onlyTestImport=N`
  0: Normal import 1: Import is only simulated to see what objects are affected (default: N = 0).

- `--targetKey|--tkey=targetKEY`
  Secure store key for the target system connection.

- `--targetHostName|--targetHost=TARGETHOST`
  Host:SQL port for the target system connection (if no key provided)

- `--targetUserName|--targetUser=TARGETUSER`
  User for the target system connection. (if no key provided)

- `--targetPassWord=TARGETPASSWORD`
  Password for the target system connection. (if no key provided)

- `--targetReadFromStdIn|--targetStdIn`
  Read password for the target system connection from standard input if necessary.

(both)

- `--log|-l`
  Write export/import summary to log file in "._SYS_REGI_settings/logs".

Examples:

- `regi transport testdu --skey=mySourceKey`
  Full DU transport of "TESTDU" from the source system with key "mySourceKey" to the current system. Vendor: content vendor. Export version: maximum.

- `regi transport testdu1 testdu2 --tkey=myTargetKey`
  Full DU transport of "TESTDU1" and "TESTDU2" from the current system to the target system with key "myTargetKey". Vendor: content vendor. Export version: maximum.

- `regi transport testdu --skey=mySourceKey --tkey=myTargetKey`
  Full DU transport of "TESTDU" from the source system with key "mySourceKey" to the target system with key "myTargetKey". Vendor: content vendor. Export version: maximum.

- `regi transport testdu --skey=mySourceKey --tkey=myTargetKey --ev=11`
  Full DU transport of "TESTDU" from the source system with key "mySourceKey" to the target system with key "myTargetKey". Vendor: content vendor. Export version: 11.

-
  ```
  regi transport testdu --tkey=myTargetKey
    --from=2013-09-01 --to=2013-09-07
  ```
  Patch DU Transport of "TESTDU" to the target system with key "myTargetKey". Vendor: content vendor. Export version: maximum. From: 2013-09-01, to: 2013-09-07.

Related:

- export deliveryUnit (export delivery unit)

- export languages (export languages delivery unit)
- import deliveryUnit (import delivery unit files)
- import languages (import languages delivery unit files)
- list changes (list changes)
- list change (list entries of changes)
- list deliveryUnits (list delivery units)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- show deliveryUnit (show meta data of delivery units)
- show deliveryUnitStatus (show status of delivery units)
- show exportVersions (show supported export versions)
- transport languages (Transport languages delivery units)

---

transport languages

Transports languages delivery units. Language delivery units contain only translated texts, no content. Combines export and import. Connection keys/host names/user names/passwords default to the currently set keys/host names/user names/passwords Default vendor: content vendor.

Usage:

- transport languages|la DELIVERYUNIT DELIVERYUNIT...

Options:

(export)

- `--alias=ALIAS`
  Set alias to anonymize the Transport.

- `--dUCheck=DUCHECK`
  Set DU check (0 = no check, 1 = version of DU is checked, 3 = version and SP is checked and both must match).

- `--exportVersion|--eV=EXPORTVERSION`
  Set export version to a value less than the maximum for downward compatibility.

- `--language|--la=LANGUAGE`
  Add language to the set of languages to export.

- `--sourceKey|--skey=SOURCEKEY`
  Secure store key for the source system connection.

- `--sourceHostName|--sourceHost=SOURCEHOST`
  Host:SQL port for the source system connection (if no key provided)

- `--sourceUserName|--sourceUser=SOURCEUSER`
  User for the source system connection. (if no key provided)

- `--sourcePassWord=SOURCEPASSWORD`
  Password for the source system connection. (if no key provided)

- `--sourceReadFromStdIn|--sourceStdIn`
  Read password for the source system connection from standard input if necessary.

- `--vendor=VENDOR`
  Specify the vendor of the delivery unit if necessary.

(import)

- `--onlyTestImport=N`
  0: Normal import 1: Import is only simulated to see what objects are affected.

- `--targetKey|--tkey=targetKEY`
  Secure store key for the target system connection.

- `--targetHostName|--targetHost=TARGETHOST`
  Host:SQL port for the target system connection (if no key provided)

- `--targetUserName|--targetUser=TARGETUSER`
  User for the target system connection. (if no key provided)

- `--targetPassWord=TARGETPASSWORD`
  Password for the target system connection. (if no key provided)

- `--targetReadFromStdIn|--targetStdIn`
  Read password for the target system connection from standard input if necessary.

(both)

- `--log|-l`
  Write export/import summary log file in "._SYS_REGI_settings/logs".

Examples:

- `regi transport la testdu --skey=mySourceKey`
  Transports translated texts of DU "TESTDU" from the system with key "mySourceKey" to the current system. Vendor: content vendor. Export version: maximum. Language: all languages available.

-
  ```
  regi transport la testdu --vendor=test.sap.com --la=de
    --tkey=myTargetKey
  ```
  Transports translated texts of DU "TESTDU" from the current system to the target system with key "myTargetKey". Vendor: "test.sap.com". Export version: maximum. Language: de.

-
  ```
  regi transport la testdu --vendor=test.sap.com --la=de
    --ev=11 --tkey=myTargetKey
  ```
  Transports translated texts of DU "TESTDU" from the current system to the target system with key "myTargetKey". Vendor: "test.sap.com". Export version: 11. Language: de.

-
  ```
  regi transport la testdu1 testdu2 --la=de --la=fr
    --skey=mySourceKey --tkey=myTargetKey
  ```
  Transports translated texts of DUs "TESTDU1" and "TESTDU2" from the system with key "mySourceKey" to the target system with key "myTargetKey". Vendor: "test.sap.com". Language: de, fr.

Related:

- export deliveryUnit (export delivery unit)
- export languages (export languages delivery unit)
- import deliveryUnit (import delivery unit files)
- import languages (import languages delivery unit files)
- list deliveryUnits (list delivery units)
- show deliveryUnit (show meta data of delivery units)
- show deliveryUnitStatus (show status of delivery units)
- show exportVersions (show supported export versions)
- transport deliveryUnit (Transport delivery units from one system to another)
- transport languages (Transport languages delivery units)

---

unAssign

Unassigns packages from their delivery units.

Usage:

- unAssign [package] PACKAGE PACKAGE...

Options:

- `--noRecursive|--noRec`
  Do not recursively assign sub packages.

Examples:

-
  ```
  regi unassign p1 p2.p3
    = regi unassign package p1 p2.p3
  ```
  Unassigns the packages "p1" and "p2.p3" from their delivery unit.

-
  ```
  regi unassign p1 p2.p3 --norec
    = regi unassign package p1 p2.p3 --norec
  ```
  Unassigns the packages "p1" and "p2.p3" from their delivery unit; do not recursively unassign the sub packages of "p1" and "p2.p3".

Related:

- assign (assign packages to delivery units)
- list deliveryUnit (list delivery unit)
- list packages (list packages in the Repository)
- show vendor (show content vendor)
- update package (update meta data of packages)

---

unDeploy

Undeploys delivery units from the Repository Default vendor: content vendor.

Usage:

- unDeploy [deliveryUnit|du] DELIVERYUNIT DELIVERYUNIT...

Options:

- `--vendor=VENDOR`
  Specify the vendor of the delivery units if necessary.

Examples:

-
  ```
  regi undeploy testdu
    = regi undeploy deliveryUnit testdu
  ```
  Undeploys delivery unit "TESTDU" from the Repository. Vendor: content vendor.

-
  ```
  regi undeploy testdu --vendor=test.sap.com
    = regi undeploy deliveryUnit testdu --vendor=test.sap.com
  ```
  Undeploys delivery unit "TESTDU" from the Repository. Vendor: "test.sap.com".

Related:

- delete deliveryUnit (delete delivery units)
- delete package (delete packages)
- import deliveryUnit (import delivery unit files)
- import languages (import languages delivery unit files)
- list deliveryUnits (list delivery units)
- show vendor (show content vendor)

---

unTrack allPackages

Untracks all tracked packages.

Usage:

- unTrack allPackages|rootPackages

Options:

None.

Examples:

- `regi untrack allpackages`
  Untracks all tracked packages.

Related:

- create workSpace (create Regi workspace)
- list trackedPackages (list tracked packages)
- list rootPackages (list root packages)
- list packages (list packages in the Repository)
- status (show the status of (files in) the Regi workspace)
- track allPackages (track all packages in the Repository)
- track package (track a package)
- unTrack package (untrack a package)

---

unTrack package

Untracks a package.

Usage:

- unTrack [package|packages] PACKAGE PACKAGE...

Options:

None.

Examples:

-
  ```
  regi untrack p1 p2.p3
    = regi untrack package p1 p2.p3
  ```
  Untracks packages "p1" and "p2.p3" (and all of their sub packages).

Related:

- create workSpace (create Regi workspace)
- list trackedPackages (list tracked packages)
- list rootPackages (list root packages)
- list packages (list packages in the Repository)
- status (show the status of (files in) the Regi workspace)
- track allPackages (track all packages in the Repository)
- track package (track a package)
- unTrack allPackages (untrack all tracked packages)

---

update change

Updates meta data of changes.

Usage:

- update change|ch CHANGEID CHANGEID...

Options:

- `--description|--desc=DESCRIPTION`
  Set change description.

Examples:

-
  ```
  regi update ch 42 4711 --desc=desc1
    = regi update change HDB//42 HDB//4711 --desc=desc1
  ```
  Updates the description of changes "HDB/42" and "HDB//4711" to "desc1".

Related:

- create change (create changes)
- delete change (delete changes)
- list change (list entries of changes)
- list changes (list changes)
- merge changes (merge two changes into one)
- move (move objects from one change to another)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show change (show meta data and contributors of changes)
- show changeTrackingStatus (show change tracking status)
- update change (update meta data of changes)

---

update contributor

Updates meta data of change contributors.

Usage:

- update contributor|cont CHANGEID CONTRIBUTOR CONTRIBUTOR...

Options:

- `--description|--desc=DESCRIPTION`
  Set change contributor description.

Examples:

-
  ```
  regi update cont 42 USER1 USER2 --desc=desc1
    = regi update contributor HDB//42 USER1 USER2 --desc=desc1
  ```
  Updates the change contributor description of change "HDB//42" and users "USER1" and "USER2" to "desc1".

Related:

- create contributor (create change contributors)
- delete contributor (Delete change contributors)
- set changeTrackingStatus (set change tracking status)
- show changeTrackingStatus (show change tracking status)
- validate contributor (set status of contributors to "validated")

---

update deliveryUnit

Updates meta data of delivery units. Default vendor: Content vendor.

Usage:

- update deliveryUnit|du DELIVERYUNIT DELIVERYUNIT...

Options:

- `--vendor=VENDOR`
  Specify the vendor of the delivery unit if necessary. The vendor cannot be updated with this action; to do this, the DU must be deleted and then re-created.

- `--caption=CAPTION`
  Set caption.

- `--responsible|--resp=RESPONSIBLE`
  Set responsible.

- `--version=VERSION`
  Set version.

- `--versionSP|--version_sp=VERSIONSP`
  Set SP version.

- `--versionPatch|--version_patch=VERSIONPATCH`
  Set patch version.

- `--ppmsID=PPMSID`
  Set PPMS ID.

- `--spPPMSID=SPPPMSID`
  Set SP PPMS ID.

- `--ach=ACH`

Set ACH.

Examples:

- `regi update du testdu1 testdu2 --version=1 --resp=me`
  `Updates delivery units "TESTDU1" and "TESTDU2".`
  Vendor = content vendor. Updates "version" to "1" and "responsible" to "me".

- `regi update du testdu1 testdu2 --vendor=test.sap.com --version=42`
  Updates delivery units "TESTDU1" and "TESTDU2". Vendor = "test.sap.com". Updates "version" to "1" and "responsible" to "me".

Related:

- create deliveryUnit (create delivery units)
- delete deliveryUnit (delete delivery units)
- show deliveryUnit (show meta data of delivery units)
- show deliveryUnitStatus (show status of delivery units)
- show vendor (show content vendor)

---

update package

Updates meta data of packages.

Usage:

- update package PACKAGE PACKAGE...
- update packages PACKAGE PACKAGE...

Options:

- `--deliveryUnit|--du=DELIVERYUNIT`
  Set delivery unit.

- `--description|--desc=DESCRIPTION`
  Set description.

- `--hintsForTranslation|--hints=HINTSFORTRANSLATION`
  Set hints for translation.

- `--origLang=ORIGLANG`
  Set original language.

- `--responsible|--resp=RESPONSIBLE`
  Set responsible.

- `--structural=STRUCTURAL`
  Set structural.

- `--textCollection|--textColl=TEXTCOLLECTION`
  Set text collection.

- `--textStatus=TEXTSTATUS`
  Set text status.

- `--textTerminologyDomain|--textTerm=TEXTTERMINOLOGYDOMAIN`
  Set text terminology domain.

- `--vendor=VENDOR`
  Set vendor (default: content vendor if delivery unit provided, empty otherwise)

Examples:

- `regi update package p1 p2.p3 --resp=me`
  Updates packages "p1" and "p2.p3". Sets "responsible" to "me".

- `regi update package p1 p2.p3 --structural=1`
  Updates packages "p1" and "p2.p3". Sets "structural" to "1".

Related:

- assign (assign packages to delivery units)
- create package (create packages)
- delete package (delete packages)
- list deliveryUnit (list delivery unit)
- show package (show meta data of packages)
- unAssign (unassign packages from their delivery units)

---

validate contributor

Sets the status of contributors of a change to "validated".

Usage:

- validate [contributor|cont] CHANGEID CONTRIBUTOR CONTRIBUTOR...

Options:

- `--description|--desc=DESCRIPTION`
  Sets the status to "validated" with description DESCRIPTION.

Examples:

-
  ```
  regi validate cont 4711 USER1 USER2 --desc=desc1
    = regi validate contributor HDB//4711 USER1 USER2 --description=desc1
  ```
  Sets the status of the contributors "USER1" and "USER2" of change "HDB//4711" to "validated" with description "desc1".

Related:

- create contributor (create change contributors)

- delete contributor (Delete change contributors)
- list change (list entries of changes)
- list changes (list changes)
- release change (release changes)
- set changeTrackingStatus (set change tracking status)
- show changeTrackingStatus (show change tracking status)
- update contributor (update meta data of change contributors)

---

version

Shows Regi version.

Usage:

- version

Options:

- `--clientFormat|--client`
  Show client format version number (components)
- `--plain`
  Show plain version number.
- `--major`
  Show major version number component.
- `--minor`
  Show minor version number component.
- `--micro`
  Show micro version number component.

Examples:

- `regi version`
  Shows full Regi version.
- `regi version --client`
  Shows full Regi client format version.
- `regi version --plain`
  Shows plain Regi version number components.
- `regi version --major --minor`
  Shows major and minor Regi version number components.
- `regi version --client --micro`
  Shows micro Regi client format version number component.

Related:

- status (show the status of (files in) the Regi workspace)