CreamingSoda

VERSION 2.0

Hello.

Welcome to CreamingSoda!

Table of Contents

About the Manual

Welcome to CreamingSoda. This document is designed to get you up to speed as quick as possible on how to use the CreamingSoda Automatic Revision Control System with your software or document project.

CreamingSoda is a desktop Revision Control Tool that runs on Microsoft Windows, Apple Mac OS X and various Linux distributions.

To begin using CreamingSoda, you'll need to install it to your computer and either use a valid license key or have an active trial.

Thanks for choosing CreamingSoda!

01. Installation

Download the latest version of CreamingSoda from the download page and follow the instructions listed for your platform. The trial installer can be used by Standard and Professional edition licensees too.

A. License Registration

Once installed, you will be presented with the **Licensing Page**. If you wish to run CreamingSoda as a 14 Day trial, simply click the **Begin Trial** button, otherwise enter your license key into the text box and click **Activate License**

B. Command Line Installation

If you have opted to install the command-line only version of CreamingSoda, you will not be able to begin a trial license from the command line. If you require a command-line only installation on a trial basis, please contact us for an evaluation code.

To register a command line version simply use the -register command line switch followed by a space and your license key.

You will need to run the command with administrative privileges.

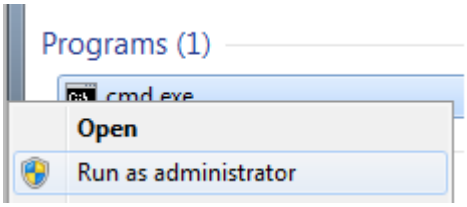
Starting a command prompt with administration privileges:

Windows 7 & Below:

1. Click the Start Orb and in the search field type **cmd**

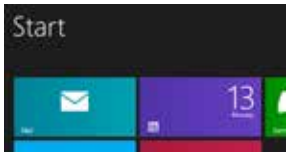
2. Right click the **cmd.exe** program

3. Select **Run as Administrator** from the popup menu



Windows 8:

1. Open the Start Screen

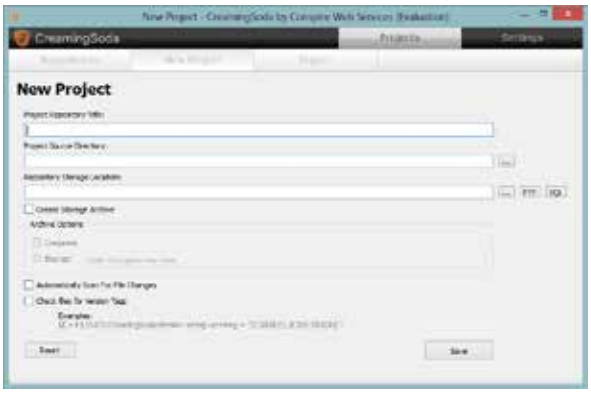


2. Type **cmd** to begin an application search

3. Right click the **Command Prompt** tile and click **Run as Administrator**

02.

Creating Repositories



The New Project screen allows you to create the parameters required to add a new repository to your CreamingSoda installation.

Repositories you add will only be added to your repository list, other users on your computer will be able to import them into their user account if they are stored in a folder that is accessible by all users.

Repository Title

The name of the repository.
This will be used when accessing the repository from the command line, so it's advisable to keep it relatively short.

Source Directory

The location of your source files.

Anything in this folder will be processed when CreamingSoda is working - the files can be anything you desire (software code, text files, documents, graphics or even binary data).

To keep performance snappy, we recommend keeping recursive directories to a minimum and minimize the amount of large files to be processed.

Storage Location

This specifies where CreamingSoda should store the builds and revision files for your project.

By default the storage location is in a sub-file in your home directory. If multiple users are likely to be working on the project, a mapped network drive can be used.

Professional Edition licensees are able to specify FTP or SQL server parameters to make sharing repositories across enterprise deployments easier.

Automatic Scan

Determines whether CreamingSoda should add the project to the list of repositories it automatically scans.

If the project is included in automatic revision scanning, any changes to the directory will be automatically copied to the repository.

This should be disabled for large projects, or projects where builds and revisions will be performed manually.

Create Storage Archive

Professional Edition Only

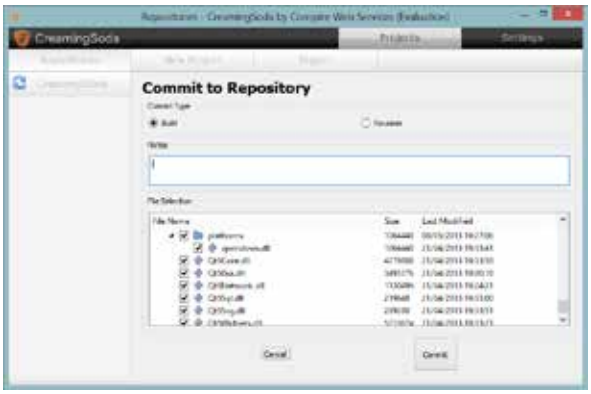
If enabled, CreamingSoda will create a single standalone file to form the basis of the repository storage. Revisions and Builds will be added to this file over time. Standalone files can be compressed or encrypted if desired.

Version Tags

CreamingSoda can automatically update the build and revision number in your project, handy for software builds.

When enabled, every time a commit is performed, CreamingSoda will check for a special version tag (usually in a commented line for software projects) and update the following line with the current Build and Revision numbers.

This is great for software developers since it means you can easily track down any issues your customers may be facing by looking back at that specific build number.



03.

Committing Files

When committing files, you'll be able to select whether or not to add files to the repository as a build or a revision, enter some notes and select which files to include and which files to ignore.

The difference between Builds and Revisions are: Builds are a snapshot of the entire project source folder at that time, and should you ever need to at any point, checking out a build will give you a fully working copy of your source files.

Revisions only include the files which have been changed on the file system and If no files have been changed a revision will fail. File changes are detected in a number of ways, usually by checking the file contents or last modified times. CreamingSoda will include all files by default and requires implicit instructions on which files to exclude. Once excluded, a file will not be included again unless you have instructed CreamingSoda to do so.

04. Environment Integration.

Integrating CreamingSoda into any environment is a relatively seamless process and one that is certainly easy to master.

For document editing, such as Word Processing, Presentations or Spreadsheets, simply setting up a repository that uses your documents directory as the file source should suffice. Every time you click the save button in your application, CreamingSoda will make a revision automatically. It doesn‘t even need to be confined to office documents either - anything from CAD files to musical masters can be automatically scanned.

For software developers, CreamingSoda can integrate into your build environment. Not only will your source files be revised every time you save, but a full snapshot of your source directory can be committed every time you click the build button. Ever need to go back to the source code of a specific build after a customer has encountered an error? Now it‘s as easy as clicking the view files button!

Build environment integration is handled by the command line utility provided with your CreamingSoda installation - great, because it means it‘s easy to integrate it into nearly any build system.

While the ability to commit builds and revisions is included, advanced users can even create new repositories from the command line for tighter integration into your build system. A full list of command line switches is available here.

After setting up your repository, committing a build via the command line looks something like:

```
cscmd.exe commit -repo “repositoryname” -type build
```

Where **repositoryname** can be replaced with the name of the repository as it appears in your CreamingSoda repository list, or the full path to the repository descriptor file.

Optionally, you can also pass the **-notes** parameter which lets you override the build notes. If you‘d prefer to commit the files as a revision, replace **-type build**, with **-type revision**.

If you are a Professional Edition user and have opted for an encrypted archive file, you may pass **-encrypt encryptionkey** to decrypt your archive file to commit the build or revision.

Note If your file is not encrypted already, passing -encrypt will encrypt your repository. Passing an incorrect encryption key will cause the commit to fail.

A. Qt Creator Setup

Qt Creator provides a uniform platform for building Qt based applications across a multitude of platforms. Using CreamingSoda with Qt Creator is easy and once you‘ve set a couple things up, everything is automatic.

1. Create your application project in Qt Creator

2. Create a repository in CreamingSoda and use the project source directory from Qt Creator in the New Repository wizard.

3. Back in Qt Creator, click the projects button down the left-hand side.

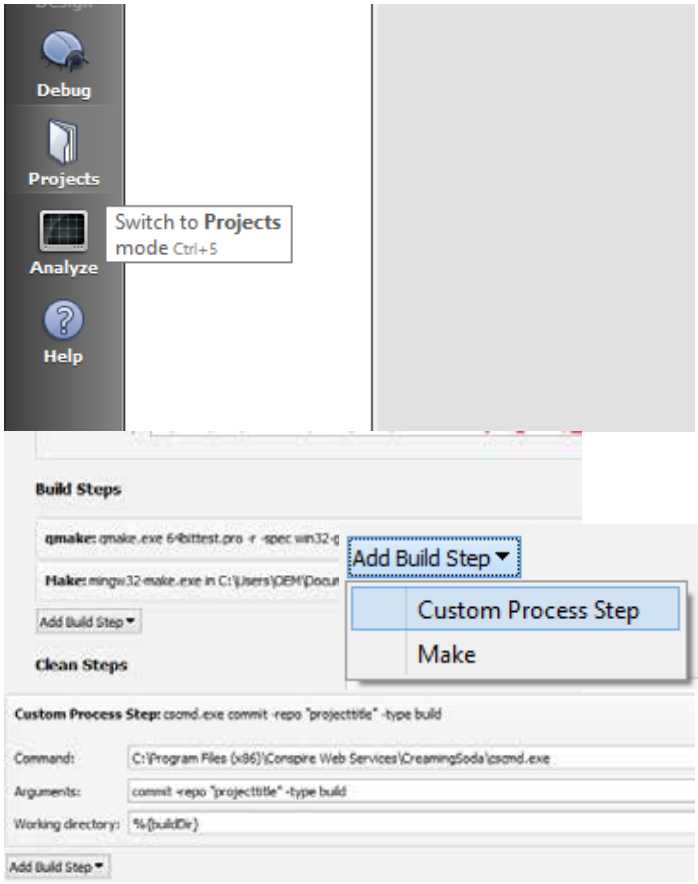
4. In the Build Steps section, click ‘Add Build Step’

5. Select ‘Custom Build Step’ from the menu.

6. In the ‘Custom Build Step’ area, locate ‘cscmd.exe’ from your CreamingSoda installation directory. Under arguments enter:

```
commit -repo ‘project’ -type build
```

(where ‘project’ is the name of your CreamingSoda repository, or the full / relative path to the repository descriptor file)



7. You can optionally move the build step up or down the chain depending on your needs. We tend to leave it as the last step, so only successful compilations trigger a CreamingSoda commit.

B. Visual Studio Integration

Microsoft Corporation‘s Visual Studio is the premier development studio for the Microsoft Windows platform. Customers who wish to use CreamingSoda with Visual Studio are in luck - it couldn‘t be any easier!

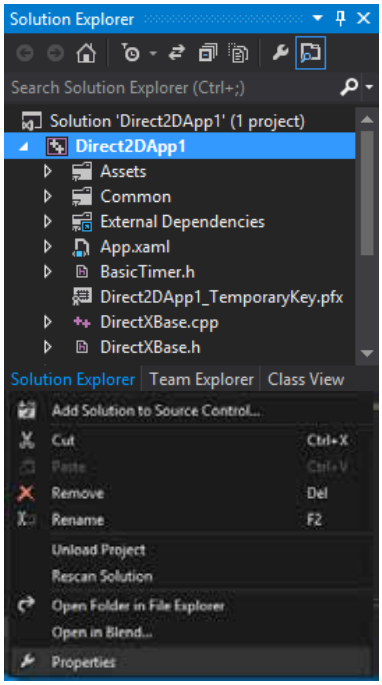
1. Create and save your Visual Studio Project

2. Create a repository in CreamingSoda and use the project source directory from Qt Creator in the New Repository wizard

3. In the Visual Studio ‘Solution Explorer’ right click the project

4. Select ‘Properties’ from the menu

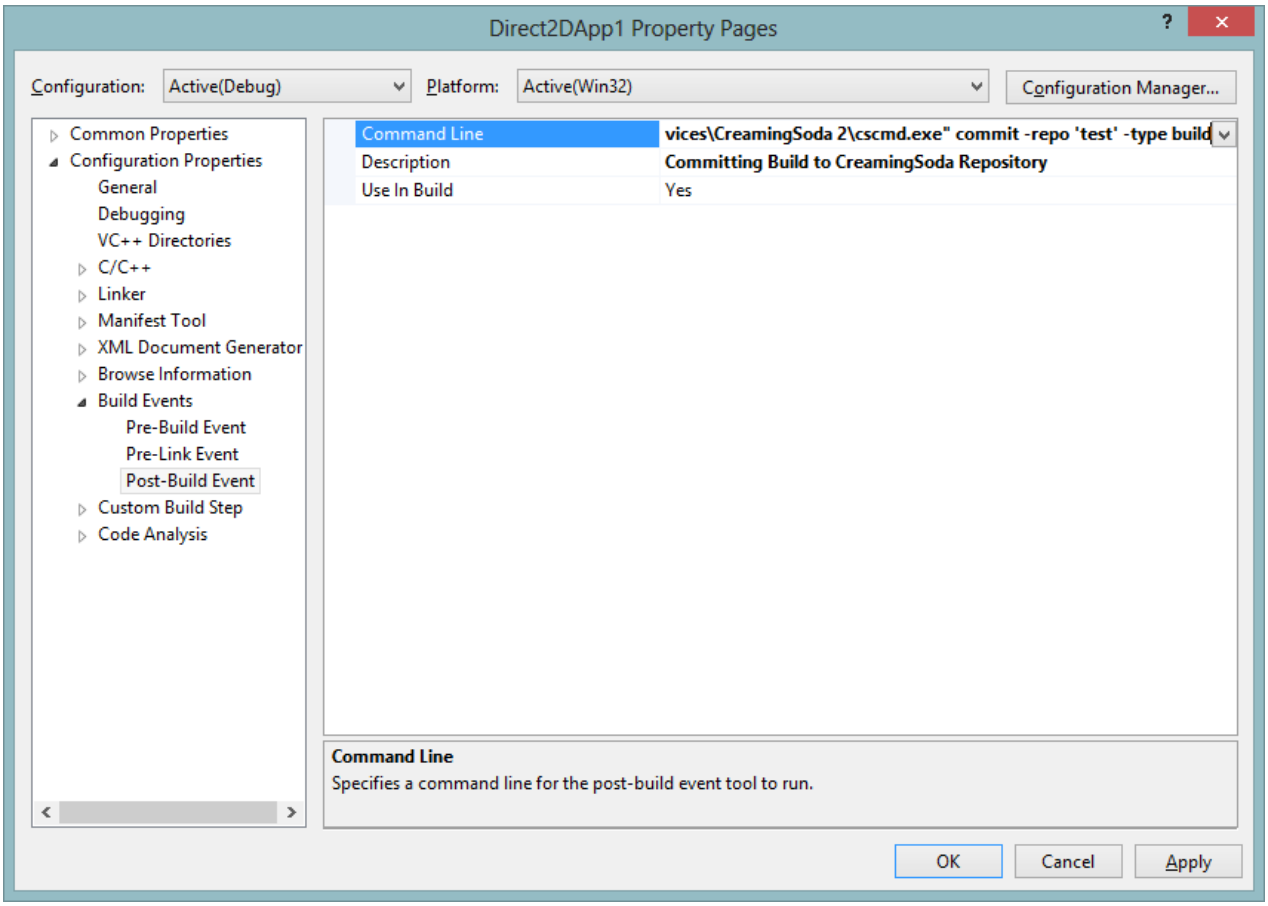
5. In the ‘Command Line’ text field, enter the path to your CreamingSoda installation path and cscmd.exe (in double quotes if necessary), followed by:




```
commit -repo ‘Direct2DApp1’ -type build
```

(In this case ‘Direct2DApp1’ is our project name in both Visual Studio and CreamingSoda)

6. Optionally, change the ‘Description’ field to ‘Performing CreamingSoda Build Step’ or similar to help distinguish the event in the Build logs.


Direct2DApp1 Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
VC++ Directories
C/C++
Linker
Manifest Tool
XML Document Generator
Browse Information
Build Events
Pre-Build Event
Pre-Link Event
Post-Build Event
Custom Build Step
Code Analysis
Command Line
vices\CreamingSoda 2\cscmd.exe" commit -repo 'test' -type build
Description
Committing Build to CreamingSoda Repository
Use In Build
Yes
Command Line
Specifies a command line for the post-build event tool to run.
OK
Cancel
Apply

```
CreamingSoda
By Conspire Web Services
------------------------
Usage:
   cscmd [action] [options]

   OPTIONS:
       create
           -name [projectname]
               REQUIRED
               Specifies the name of the project.

           -source [srcpath]
               REQUIRED
               Specifies the location of the source files.

           -store [storedir]
               Optional (Default %HOMEDIR%/CreamingSoda/%PROJECTNAME%)
               Specifies where to store the repository.
               Local/Network folders are supported as paths.
               (PRO) FTP Servers are supported as ftp://username@server/.
               (PRO) SQL Servers can be used with:
                         {driver}://{username}@{server}/{dbname}
                         You will be prompted for a password if required.
                         Example: mysql://dbuser@localhost/repositorydb

           -autoscan [ON/OFF]
               Optional (Default ON)
               When ON, Automatic revisions will be enabled.

           -versioncheck [ON/OFF]
               Optional (Default ON)
               Determines if source files will be scanned for version tags.

           -archive [ON/OFF]
               Optional (Default OFF)
               If ON, the repository will be stored in a standalone archive.
               This feature is only availble in the Professional Edition.

           -compress [ON/OFF]
               Optional (Default OFF)
               When ON, stored file revisions stored will be compressed.
               This option is only valid when creating storage archives.

           -encrypt [CRYPTKEY]
               Optional.
               When set, stored file revisions stored will be encrypted.
               This option is only valid when creating storage archives.
               This feature is only availble in the Professional Edition.

       info
           -repo [reponame]
               REQUIRED.
               The repository to print information on.
               Can be the repository name, or the storage directory.
```

APPENDIX. Command Line Parameters

```
commit
    -repo [reponame]
        REQUIRED.
        The repository to work with.
        Can be the repository name, or the storage path.
        Local/Network folders are supported as paths.
        (PRO) FTP Servers are supported as ftp://username@server/.
        (PRO) SQL Servers can be used with: sql://username@server/.

    -type [revision/build]
        REQUIRED.
        Sets whether or not this commit is a revision, or build.
        Revisions only work with modified files from the source.
        Builds take a complete snapshot of the source.

    -notes [notes]
        Optional (Defaults to timestamp)
        The notes to be displayed with the commit.
        Must be encased in ‘ or “ Quotes.

    -force
        Optional.
        If passed, revisions will be submitted, even if no
        source files were modified. Ignored for build commits.

    -ignore [FILENAME]
        Optional.
        Any filenames listed will not be included in the revision.
        Ignoring files once will ignore them for all future commits.
        Wildcards will be matched.
        Multiple ignore arguments are allowed.

    -include [FILENAME]
        Optional.
        Includes a previously ignored file.
        Including a file once will remove any ignore statements.
        Wildcards will be matched.
        Multiple include arguments are allowed.

    -encrypt [CRYPTKEY]
        Optional.
        When set, stored file revisions stored will be encrypted.
        Use when the repository has encryption enabled.
        [CRYPTKEY] must match previously set encryption key.
        This feature is only availble in the Professional Edition.

    -quick
        Optional.
        If passed, the file scanner will skip checking file contents.
        This is quicker, but less accurate.
        No parameters are required.
        This switch is ignored when using archival storage.
        Repository setting: Automatic Version Checking is ignored when
        using this switch.
```

```
checkout
        -repo [reponame]
            REQUIRED.
            The repository to work with.
            Can be the repository name, or the storage directory.

        -path [checkoutpath]
            REQUIRED.
            The location to checkout to.

        -build [buildid]
            Optional (Defaults to the latest build + revisions)
            If passed, revisions newer than the build will be submitted,
            even if no source files were modified.

        -ignore [FILENAME]
            Optional.
            Any filenames listed will not be included in the revision.
            Ignoring files once will ignore them for all future commits.
            Wildcards will be matched.
            Multiple ignore arguments are allowed.

        -include [FILENAME]
            Optional.
            Includes a previously ignored file.
            Including a file once will remove any ignore statements.
            Wildcards will be matched.
            Multiple include arguments are allowed.

        -decrypt [CRYPTKEY]
            Optional.
            When set, file revisions will be decrypted.
            This feature is only availble in the Professional Edition.
```