Using CruiseControl As Continuous Build Software

As of Servoy 7.2 it is advised to use Jenkins instead of CruiseControl as Continous Build software. This page will no longer be maintained after the Servoy 7.2 release.

The continuous build software CruiseControl is open source, has a no-cost license, and is well suited for Java applications.

In This Chapter

- Overview
- Installing CruiseControl
- Configuration Tips
- Step-by-step Configuration for Using It with Servoy
  - Step 1 -- Create a Folder
  - Step 2 -- Install CruiseControl and Edit wrapper.conf
  - Step 3 - Download and Install SVNAnt
  - Step 4 -- Install Servoy Developer
  - Step 5 -- Install Command Line SVN Client
  - Step 6 -- Checkout Solution into Work Area
    - Alternate Steps If the SVN Does Not Contain the Solution
    - Using Servoy Developer to Checkout Solution
  - Step 7 - Configuration
  - Step 8 - Restart and Force Build/Commit Change

Overview

CruiseControl is a free continuous build software package released as open source software. It is managed by ThoughtWorks, whose employees created the original version. It is well suited for Java based projects, and some of the more notable features include:

- Plugins for builders such as Ant, NAnt, Maven, and Xcode; source control such as CVS and SVN; and notification via email and instant messaging.
- Other 3{^}rd^ part extensions for various report and usability.
- Open source so extensible and customizable.
- Web application

Installing CruiseControl



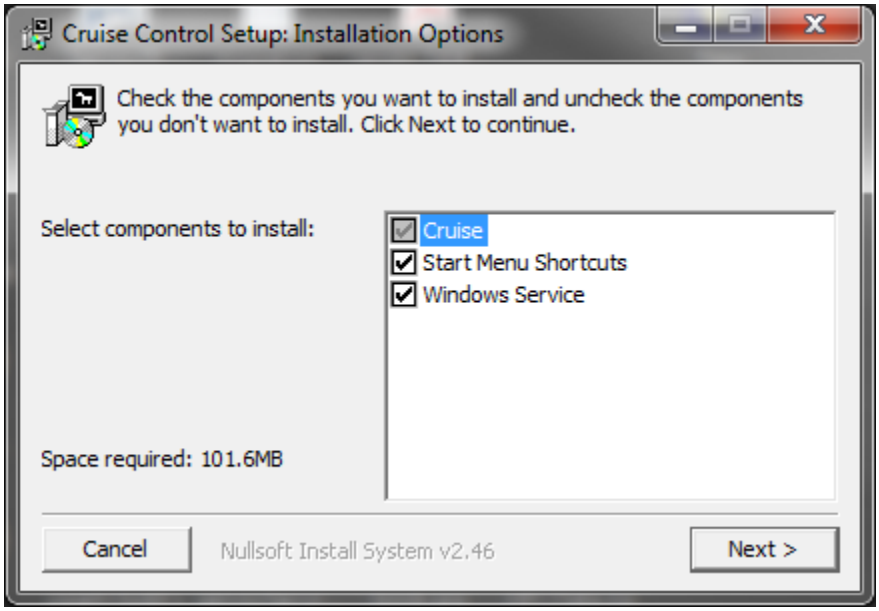
These instructions are for a Windows installation.

CruiseControl can be downloaded at the following location: http://sourceforge.net/projects/cruisecontrol/files/. At this location, there are three options available:

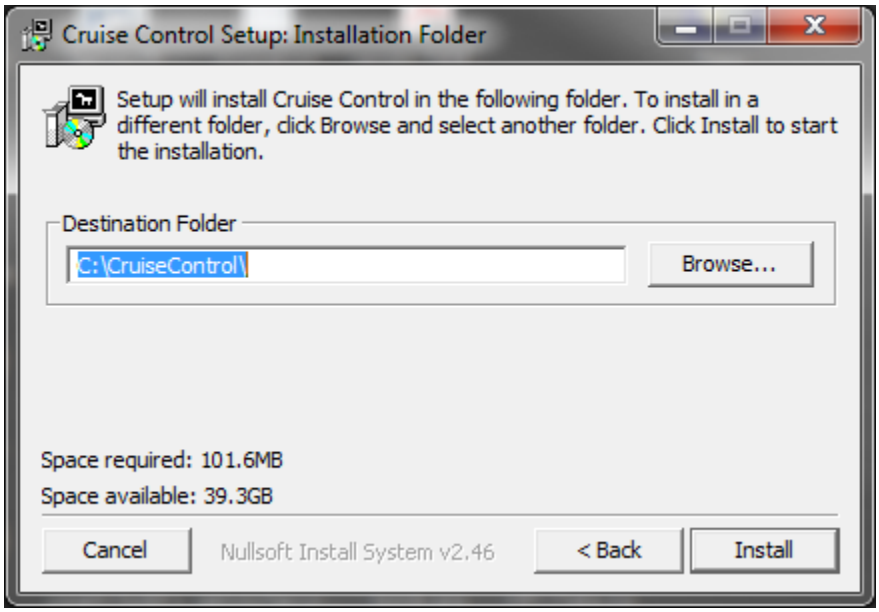
- Binary version that can be copied into a file location.
- Source code version that can be compiled with specific feature sets.
- Windows installer that will install a Windows service with the installation.

These instructions will concentrate on using the Windows installer. The other two options are available if a more customized installation is desired or necessary.

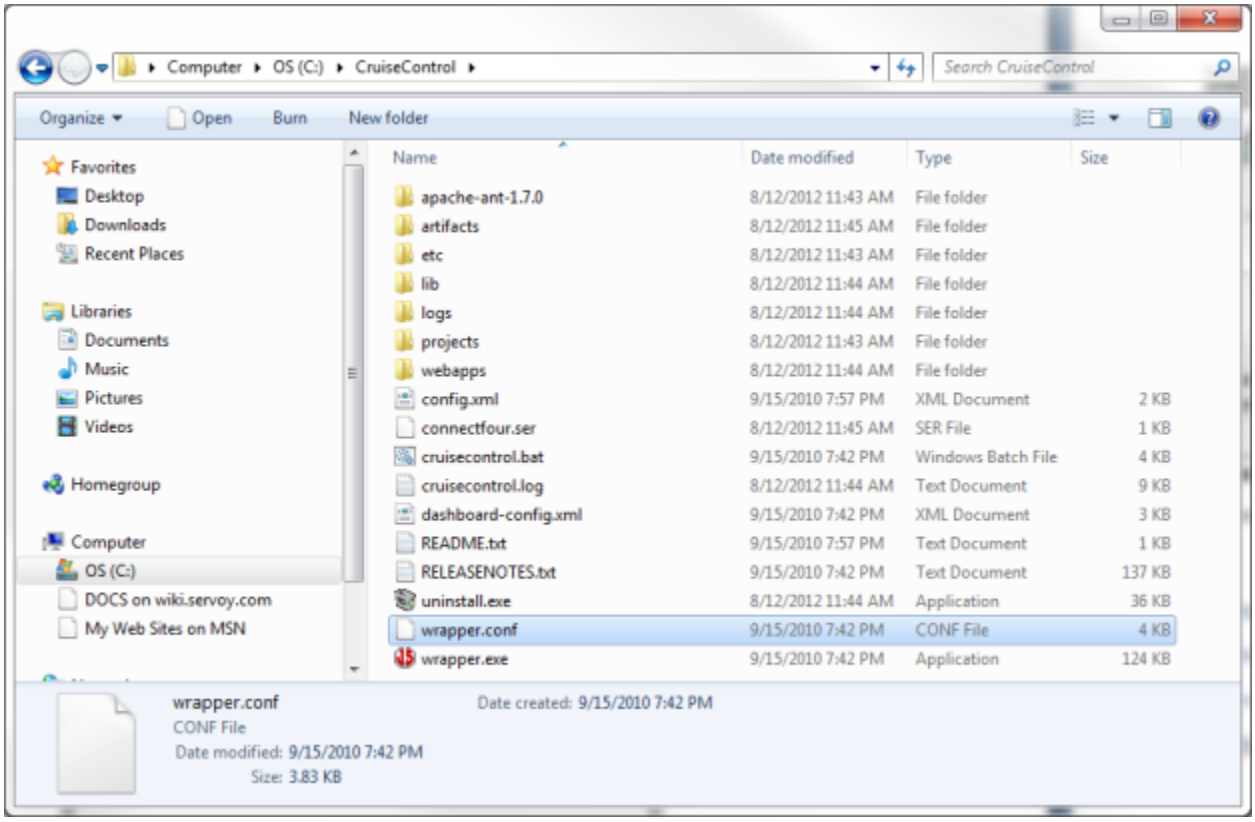
Once the installation file has been downloaded, executing the installer should open a window



Select all the components. Next, the installer will ask for a location. This location should have sufficient rights for read/write

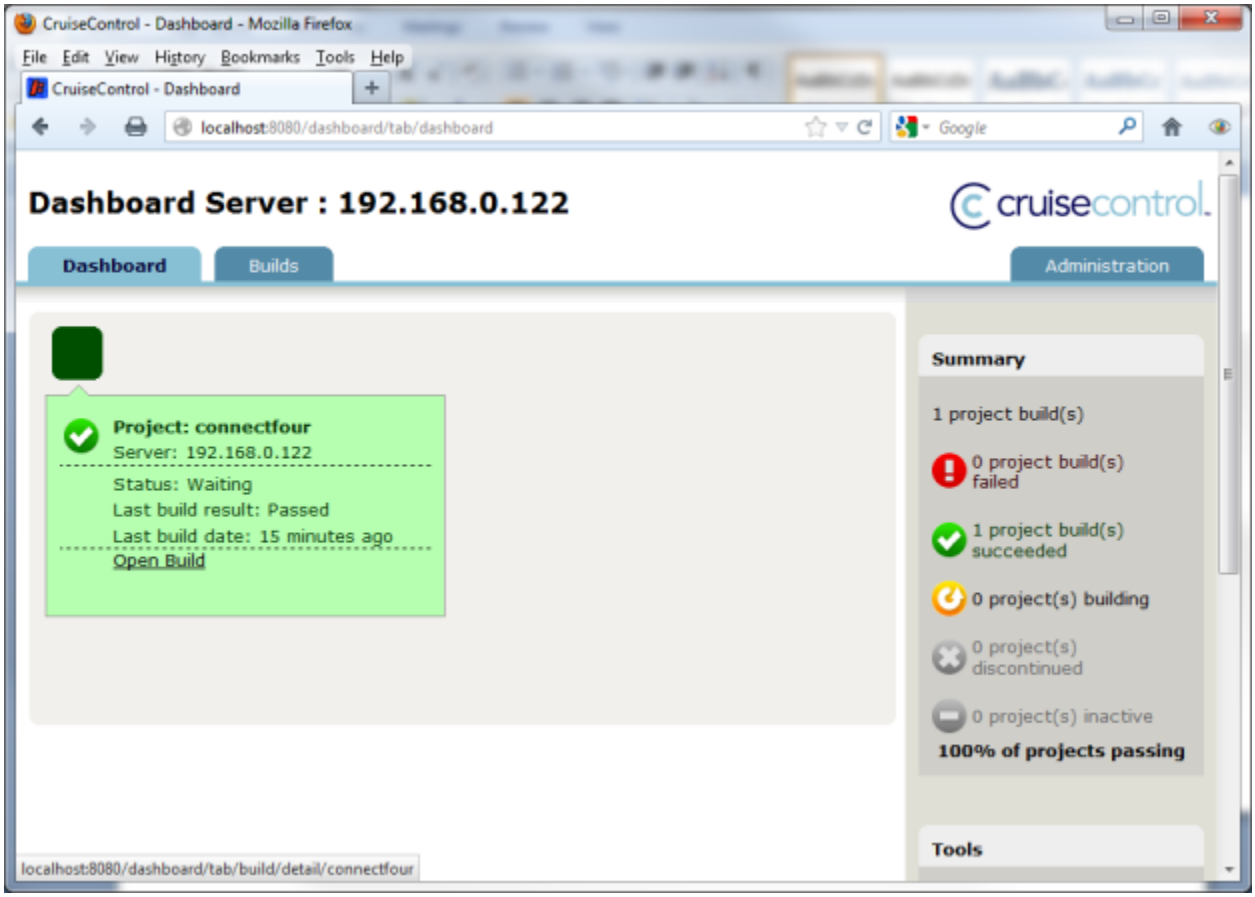


Clicking **Install** will start the installation. After the installation is complete, there should be a folder containing the program contents



Also, if you open the Local Services control panel for the machine, there should be a service named 'CruiseControl Service' available that has not been started yet. Go ahead and start the service to start CruiseControl.

CruiseControl ships with a default Java project named ConnectFour. When CruiseControl starts, it will try to build this project (using Ant) and return a result. To see if this has occurred, open the CruiseControl dashboard at the following location: `http://{myHost}:8080/dashboard`. The application dashboard will launch and you should see a screen like below



If the build has passed, then CruiseControl is configured and running properly in its default state.

Some troubleshooting tips if installation does not work:

- If you cannot launch the dashboard, make sure Servoy is not running (see configuration tips below)
- If the dashboard is running, but there are build errors, as indicated in the builds tab, one issue could be that your Java environment is not properly configured. The connectfour project depends on Ant being configured correctly, and if there are issues with the program reaching Ant, or if there are multiple versions on your machine, there might be issues with the build. Check your Java environment and environment variables to ensure that Ant runs properly.
- Make sure the service actually starts. If not, check to make sure the install actually completed and that you have proper permissions in the installation location.

Configuration Tips

As CruiseControl and Servoy are both Java driven applications, they share some Java technologies, and by default, share port configurations as well. If you plan to run CruiseControl and Servoy Application Server on the same machine, port changes will need to be made one application or the other in order for the applications to work properly.

> Only one or the other needs to be modified, not both.

Changing ports for a Servoy Application Server installation is detailed on Network Related Settings.

To change port configurations for CruiseControl instead, modify the wrapper.conf file found in the CruiseControl main folder. See the section below:

```
Application parameters. Add parameters as needed starting from 1
wrapper.app.parameter.1=CruiseControlWithJetty
wrapper.app.parameter.2=-webport
wrapper.app.parameter.3=8080
wrapper.app.parameter.4=-jmxport
wrapper.app.parameter.5=8000
wrapper.app.parameter.6=-configfile
wrapper.app.parameter.7=config.xml
wrapper.app.parameter.8=-rmiport
wrapper.app.parameter.9=1099
```

Note that parameter 3 and parameter 9 are conflicts with Servoy Application Server default ports. Change these ports to any other available port on the machine.

Step-by-step Configuration for Using It with Servoy

Step 1 -- Create a Folder

Create a folder for the software factory: `c:/SoftwareFactory`

> File structure for this folder can be found in SofwareFactory.zip

Step 2 -- Install CruiseControl and Edit wrapper.conf

Install CruiseControl into the `c:/SoftwareFactory/CruiseControl` folder. (Instructions)

This is also a good time to make configuration changes to the wrapper.conf file (Instructions) To work in this environment, we need to change and add some lines to wrapper.conf for the run directory and classpath:

Add two lines right above the `wrapper.java.classpath` lines in the wrapper.conf

```
wrapper.working.dir=c:/SoftwareFactory/workArea
set.CC_INSTALL=c:/SoftwareFactory/CruiseControl
```

Modify three lines as follows:

```
wrapper.java.classpath.1=%CC_INSTALL%/lib/wrapper/*.jar
wrapper.java.classpath.2=%CC_INSTALL%/lib/*.jar
wrapper.java.library.path.1=%CC_INSTALL%/lib/wrapper
```


To avoid a bug that makes the dashboard not show build status, use Java 6 instead of 7; modify the following line to point to Java 6:

```
wrapper.java.command=c:/Program Files/Java/jre6/bin/java
```

Save the wrapper.conf file.

Step 3 - Download and Install SVNAnt

SVNAnt is a helper application used by ANT to update projects from the SVN. Download the application from: http://subclipse.tigris.org/files/documents/906/49042/svnant-1.3.1.zip. Extract the contents of the zip file into `c:/SoftwareFactory/svnant-1.3.1`

Step 4 -- Install Servoy Developer

We need an installation of Servoy Developer. Install Servoy Developer with the following options set

- Install Servoy in the folder `c:/SoftwareFactory/Servoy`
- Install with a separate PostgreSQL database, create a repository connection and create the repository.
- Create the workspace as `c:/SoftwareFactory/DeveloperWorkspace`

Step 5 -- Install Command Line SVN Client

SVNAnt works better with a command line client. Recommended: slikSVN

Step 6 -- Checkout Solution into Work Area

At this time, it is a good idea to create the workArea file structure, create a folder name workArea and build the following structure (note that some of the files needed are provided later in this document). This is where most of the work will occur during the operation. CruiseControl will be configured to run out of this folder.



A zip file containing this file structure as well as example files is attached, see SoftwareFactory.zip

- antscripts
    - test_runner
        - cc_build.xml
        - export_and_test.xml
        - export_and_test.properties
        - servoy.properties
- checkout
    - test_runner
- logs
- config.xml
- cruisecontrol.properties
- dashboard-config.xml

Using the command line client installed, check out the solution (or solutions) from the SVN into the workspace file located at c:/SoftwareFactory/workArea/checkout/testrunner. For details on how to do this, please refer to the SVN client you have chosen.

Alternate Steps If the SVN Does Not Contain the Solution

If your SVN does not contain the solution yet, and you have the solution in a .servoy file, do the following:

1. Open Servoy Developer.
2. Install Subclipse plugin if you have not done so already and restart.
3. Import the solutions into the current workspace (should be `c:/SoftwareFactory/DeveloperWorkspace`)
4. Share the solution on SVN.
5. Close Servoy Developer

Using Servoy Developer to Checkout Solution

As an alternative, you can use Servoy Developer to check out the solution. (NOTE: you will still need the command line client for operation of the software factory)

1. Make sure the solution is in the SVN repository.
2. Open Servoy Developer, verify Subclipse 1.8 is installed.
3. Change workspace to `c:/SoftwareFactory/workArea/checkout/testrunner`.
4. Checkout the solution(s) from SVN repository.
5. Change the workspace back to `c:/SoftwareFactory/DeveloperWorkspace`
6. Close Servoy Developer.



If you are using a SSL repository, the SVN server certificate must be trusted. Otherwise, the software factory will fail.

Step 7 - Configuration

A file structure should be started now (example files in SoftwareFactory.zip)

A few items need to be configured at this point.

cruisecontrol.properties -- a few items in this file need to be configured for your environment

- Mail properties -- so CruiseControl can send emails on build status
- SVN properties -- username, password
- Directories -- workArea, and others.

config.xml - configure the modification sets; these lines need to be configured for every solution in the workspace

jenkins.xml

```
<svn localworkingcopy="${svnDir}/solution1" property="solution1_changed" />
```

cc_build.xml - this file needs to be constructed to actually run unit tests and exports. Follow the structure for the example and just modify to suit your specific situation. Keep in mind that it is design to do a unit test and export if any changes occur in any module of the main solution or the resources project. The example provided shows for two main solutions, a single module for the second solution, and a resources project.

servoy.properties - a separate properties file for any db connections used to run unit tests, as well as a repository for the unit tests.

Step 8 - Restart and Force Build/Commit Change

After making any changes to properties, the CruiseControl service must be restarted. After restarting, or starting if you have not started the service, try to make a build of the project. If everything succeeds, there should be a clean .servoy file in your passed exports folder.