****

|  |
| --- |
| Database Change Management Done Right  <http://projectroundhouse.org> |

|  |  |
| --- | --- |
| RoundhousE | Getting Started |

**Contents**

[Prerequisuites 3](#_Toc251737056)

[Database Installed 3](#_Toc251737057)

[.NET Framework 3.5 3](#_Toc251737058)

[Get RoundhousE 3](#_Toc251737059)

[Download 3](#_Toc251737060)

[Get the Source 4](#_Toc251737061)

[Run the Sample for Familiarization 4](#_Toc251737062)

[Sample Prerequisites 5](#_Toc251737063)

[Running It 5](#_Toc251737064)

[RoundhousE Change Log 6](#_Toc251737065)

[RoundhousE Output Folder 8](#_Toc251737066)

[Play It Again Sam 8](#_Toc251737067)

[Add RoundhousE to Your Project 8](#_Toc251737068)

[How To Structure Your Database Scripts 10](#_Toc251737069)

[RH init 10](#_Toc251737070)

[RoundhousE Configuration 10](#_Toc251737071)

[Barebone 10](#_Toc251737072)

[Full Configuration 10](#_Toc251737073)

[Version Your Database 10](#_Toc251737074)

[Environment Specific Scripts 10](#_Toc251737075)

Welcome to RoundhousE (RH)! RH is one of the most intelligent database change management (also known as migration) tools out there. Over the next couple of chapters you are going to learn how to effectively use RoundhousE to solve the complex business problem of managing change for your database. You will see that RH makes it really easy and as you learn the tool, you will wonder how you ever used anything else.

# Prerequisuites

## Database Installed

Whatever database you are going to use, be it SQL Server or be it MySQL, needs to be installed so you can do database change management.

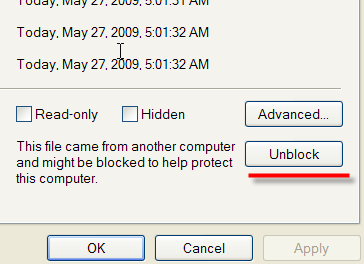
## .NET Framework 3.5

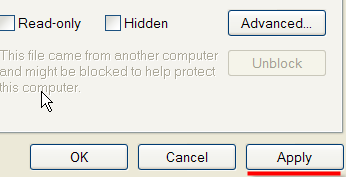
You will also need to have the .NET framework installed. Currently RH uses 3.5.

# Get RoundhousE

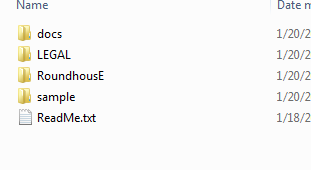
## Download

1. Head out to <http://code.google.com/p/roundhouse/downloads/list> and download the latest.
2. One thing to note, when you do download the zip file, the very first thing you will want to do (before extracting) is right click on that file and select [**Properties**]. On the **General** tab click {**Unblock**} if you have it. Otherwise you may have to unblock each file one at a time later.



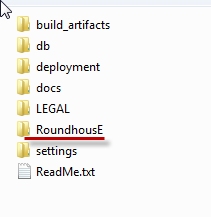


1. Unzip the files. What we need to use RH is in the RoundhousE folder.



## Get the Source

1. If you like living on the bleeding edge, then you can also get the source.
2. The other way to get RH is to point your favorite SVN client at <http://roundhouse.googlecode.com/svn/trunk/>.
3. Once it has completed downloading, just click on the build.bat (or open a command line in this directory and type build).
   1. If you are going to use the console, it is suggested you run ilmerge.bat instead. This will make the rh.exe one file instead of many.
4. Once it has completed, there is a folder called **code\_drop**. Inside of that folder is RoundhousE. This is where we get what we need to use RH.



# Run the Sample for Familiarization

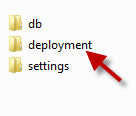
If you’ve downloaded the application, the best way to see the power of RoundhousE is to actually run the sample. It will also help you familiarize yourself with how RoundhousE works.

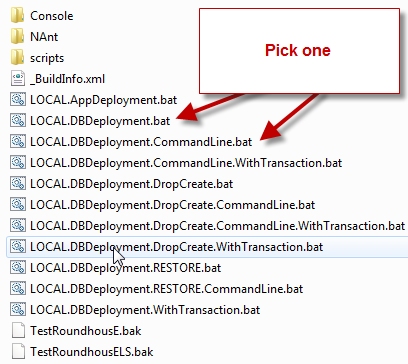
## Sample Prerequisites

The sample is run against SQL Server 2008, so you would need that installed. If you don’t (and you have SQL Server 2005), you can change the database type to sql2005 and use SQL Server 2005 instead. Unfortunately, those are really the only two choices for the sample. It is unknown whether it would work with SQL Server Express or not. If you find it does, please contact me and let me know.

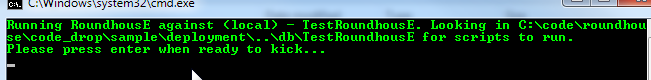
## Running It

1. Go to **code\_drop/deployment** (if you have the source) or to **sample** if you downloaded the release. Go into the **deployment** folder.

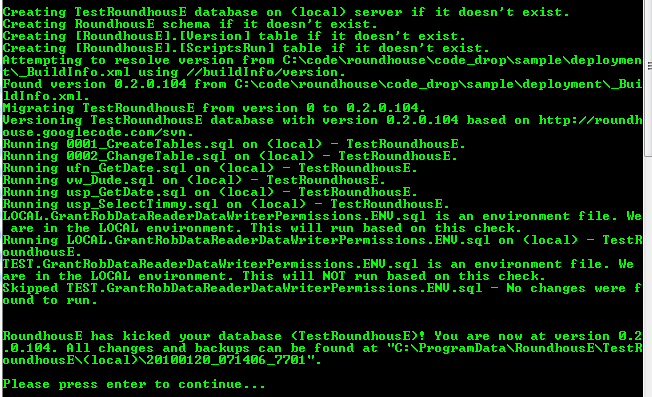




1. Pick one of these two items to run. I’m going to show screen shots for the console.bat.



1. Press **{Enter}**. Notice all of the things it does. It just created our database, versioned it, then applied all of the database scripts (marking them as part of this version). Then it saved the output of what it ran to a folder.



## RoundhousE Change Log

RH does quite a bit when it runs. Let’s take a more in depth look.

Running RoundhousE against (local) - TestRoundhousE. Looking in C:\code\roundhouse\code\_drop\sample\deployment\..\db\TestRoundhousE for scripts to run.

Please press enter when ready to kick...

The first thing you notice is that it tells you what server and what database it is going to run on. The database does not have to exist prior to run for SQL Server. It will create it automatically. Then it mentions where it is going to look for scripts. Next and slightly important, when it is in interactive mode (the default), it stops and gives you the opportunity to double check to be sure this is correct.

Creating TestRoundhousE database on (local) server if it doesn't exist.

Creating RoundhousE schema if it doesn't exist.

Creating [RoundhousE].[Version] table if it doesn't exist.

Creating [RoundhousE].[ScriptsRun] table if it doesn't exist.

Now it creates the database if it doesn’t exist. Then it creates its schema and two tracking tables. Version is where it keeps track of versioning. It also creates a ScriptsRun table.

Attempting to resolve version from C:\code\roundhouse\code\_drop\sample\deployment\\_BuildInfo.xml using //buildInfo/version.

Found version 0.2.0.104 from C:\code\roundhouse\code\_drop\sample\deployment\\_BuildInfo.xml.

Migrating TestRoundhousE from version 0 to 0.2.0.104.

Versioning TestRoundhousE database with version 0.2.0.104 based on http://roundhouse.googlecode.com/svn.

Versioning is important. The next step for RH is to version based on two elements. A repository path (if provided) and a version number (if provided). It is preferred that you version based on the same way you version your code. In this sample, versioning is done with source control revisions as the last number in the version. Notice that RH is looking at an XML file and XPath for versioning. It can also get the file version from a DLL. It is intelligent enough to know the difference when you put the version file name in during configuration.

Looking for Update scripts in "C:\CODE\roundhouse\code\_drop\deployment\..\db\TestRoundhousE\up". These should be one time only scripts.

Running 0001\_CreateTables.sql on (local) - TestRoundhousE.

Running 0002\_ChangeTable.sql on (local) - TestRoundhousE.

Looking for Run First After Update scripts in "C:\CODE\roundhouse\code\_drop\deployment\..\db\TestRoundhousE\runFirstAfterUp".

Looking for Function scripts in "C:\CODE\roundhouse\code\_drop\deployment\..\db\TestRoundhousE\functions".

Running ufn\_GetDate.sql on (local) - TestRoundhousE.

Looking for View scripts in "C:\CODE\roundhouse\code\_drop\deployment\..\db\TestRoundhousE\views".

Running vw\_Dude.sql on (local) - TestRoundhousE.

Looking for Stored Procedure scripts in "C:\CODE\roundhouse\code\_drop\deployment\..\db\TestRoundhousE\sprocs".

Running usp\_GetDate.sql on (local) - TestRoundhousE.

Running usp\_SelectTimmy.sql on (local) - TestRoundhousE.

RH then looks through the scripts folders and executes scripts.

Looking for Permission scripts in "C:\CODE\roundhouse\code\_drop\deployment\..\db\TestRoundhousE\permissions".

LOCAL.GrantRobDataReaderDataWriterPermissions.ENV.sql is an environment file. We are in the LOCAL environment. This will run based on this check.

Running LOCAL.GrantRobDataReaderDataWriterPermissions.ENV.sql on (local) - TestRoundhousE.

TEST.GrantRobDataReaderDataWriterPermissions.ENV.sql is an environment file. We are in the LOCAL environment. This will NOT run based on this check.

Skipped TEST.GrantRobDataReaderDataWriterPermissions.ENV.sql - No changes were found to run.

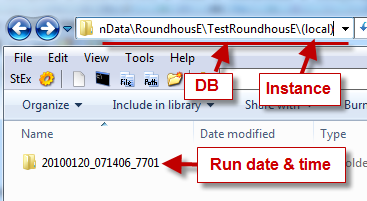
This is pretty interesting. We have a set of permissions scripts here. One called LOCAL.*something*.ENV.sql and another called TEST.*something*.ENV.sql. Only one of these runs based on the environment we are in. Yes, RH is environment aware (as long as you provide it in the configuration). If you take a look at these scripts again, one is called LOCAL.*something*.ENV.sql and the other is called TEST.*something*.ENV.sql. It’s the “**.ENV.”** that tells RH that it is an environment file. Now RH starts looking for a match in the name of the file to the environment it is in.

RoundhousE has kicked your database (TestRoundhousE)! You are now at version 0.2.0.104. All changes and backups can be found at "C:\ProgramData\RoundhousE\TestRoundhousE\(local)\20100120\_071406\_7701".

The last thing it does is report success. It also tells you where you can find the changes it made in an output folder.

## RoundhousE Output Folder

Below is the output folder. The structure of the output folder is *databasename*\*server\yyyyMMdd\_HHmmss\_ffff*.



## Database Look

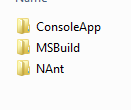
## Play It Again Sam

So now you’re starting to see some of the abilities of RH. Now run the sample again. Some migrations tools fall down right here.

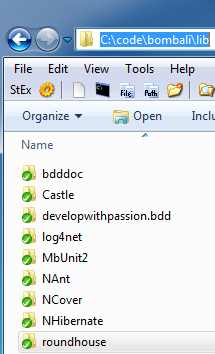
# Add RoundhousE to Your Project

Now that we have RoundhousE, we need to decide what technology we are going to use to do migrations. Are we going to use MSBuild, NAnt, or the console application (or a combination of these)?

1. Go into the RoundhousE folder and choose ConsoleApp, MSBuild, or NAnt.



1. In your project, you must decide where this is to go. I like to keep RH in my lib (tools) folder and move it to the deployment folder when I build my project. In some cases, you may already want it in your deployment folder to begin with. That is entirely up to you. If I was to make a recommendation, I would keep it in the lib folder and set up instructions for it to be moved to the deployment folder at build time.
2. Add it to either your deployment scripts folder or your lib folder. These instructions are for adding it to your lib (references/libs/thirdparty/3rdparty/etc) folder.



1. If you added it to the lib folder, there is a sample build customization script (in NAnt, specifically written for [UppercuT](http://projectuppercut.org/)) that will copy over RoundhousE to a NAnt folder. This would be using NAnt for deploys/migrations.

<target name="package\_roundhouse\_in\_deploy\_folder">

<echo message="Moving RoundhousE over to deployment folder" />

<copy todir="${dirs.drop}\${folder.deployment}\NAnt" >

<fileset basedir="${dirs.current}\..\lib\RoundhousE">

<include name="\*\*/\*.\*" />

</fileset>

</copy>

</target>

1. You could also do the same with the console.

<target name="copy\_roundhouse\_files\_to\_deploy\_console">

<echo message="Copying roundhouse assemblies to ${dirs.deployment}\Console."/>

<copy todir="${dirs.deployment}\Console">

<fileset basedir="${dirs.current}\..\lib\RoundhousE">

<include name="\*\*/\*.\*" />

</fileset>

</copy>

</target>

1. Now we are set up to start using RH and we need to set up our database scripts.

# How To Structure Your Database Scripts

## RH init

THIS FEATURE IS NOT YET AVAILABLE. When you are ready to start using it, open a command line and type rh init. Follow the instructions to create a database project structure that is set up for RH to use during a migration.

# RoundhousE Configuration

## Barebone

## Full Configuration

# Version Your Database

# Environment Specific Scripts