BuiltSteady Engineering Guide

Contents

[Source Control 3](#_Toc316300763)

[Repositories 3](#_Toc316300764)

[Documentation 3](#_Toc316300765)

[Structure 3](#_Toc316300766)

[Source Code 4](#_Toc316300767)

[Structure 4](#_Toc316300768)

[Enlisting and Making Changes 5](#_Toc316300769)

[Naming Conventions 5](#_Toc316300770)

[Build Environment 6](#_Toc316300771)

[Build 6](#_Toc316300772)

[Coding Conventions 7](#_Toc316300773)

[CSharp 7](#_Toc316300774)

[Namespace Declarations 7](#_Toc316300775)

[Using Statements 7](#_Toc316300776)

[JavaScript 7](#_Toc316300777)

[T-Sql 7](#_Toc316300778)

[Debugging 8](#_Toc316300779)

[WinPhone Client 8](#_Toc316300780)

[Deployment 9](#_Toc316300781)

[Domain Management 9](#_Toc316300782)

[Azure Accounts 9](#_Toc316300783)

[SQL Azure 9](#_Toc316300784)

[DAC 9](#_Toc316300785)

[Backup 9](#_Toc316300786)

[Generating Data Script 9](#_Toc316300787)

[Rename 10](#_Toc316300788)

[Test 11](#_Toc316300789)

# Source Control

## Repositories

All documentation and source code are stored in a GitHub repository.

Directions for setting up GitHub for Windows are located at: <http://help.github.com/win-set-up-git>

Directions for setting up GitHub for Mac OSX are located at: <http://help.github.com/mac-set-up-git>

## Documentation

Product documentation and corporate website are stored in a GitHub repository named builtsteady.

To enlist in the builtsteady repository, open a git bash command window and clone the repository to your local machine:

git clone git@github.com:ogazitt/builtsteady.git

### Structure

The builtsteady repository is divided into two primary directories; documents and corpweb.

The documents directory contains informational documentation for builtsteady corporation and the product. This includes legal documentation, marketing materials, presentations, videos, etc… This repository does not contain technical documentation, as that is stored in the zaplify repository with the source code. The documents tree is organized by area; legal, product, etc…

The corpweb directory contains the projects and source code for the corporate website.

## Source Code

The projects, source code, and technical documentation are stored in a GitHub repository named zaplify.

To enlist in the zaplify repository, open a git bash command window and clone the repository to your local machine:

git clone git@github.com:ogazitt/zaplify.git

### Structure

The zaplify repository is divided into four primary directories; source, test, tools, and documents.

The source directory contains the projects and code for the product. The source tree is organized by area: services, devices, packages, and shared. The packages directory contains external open-source code utilized by the product. The shared directory contains internally developed code utilized across areas of the product.

The test directory contains the projects and code for testing the product. The test tree is organized by area mirroring the source tree, with the addition of a clients directory for implementing test clients for various services.

The tools directory contains external and internal tools used to development, management, deployment, etc.. of the product.

The documents directory contains technical documentation pertaining to architecture, design, building and deploying the product, management of cloud services, test plans, etc...

## Enlisting and Making Changes

The following are the common git commands used for adding, removing, or updating files in a repository:

git add **filename** // add or update file in local changes

git add **directory** // add or update entire directory in local changes

git rm **filename** // remove file from local changes

git mv **filename** // rename file in local changes

git status // review local changes

git commit –m '**comment**' // commit local changes

git push // push committed changes to remote repository

## Naming Conventions

There are a few naming conventions that should be followed when adding new directories or files to the repository. External tools and packages are exempt from these conventions.

* Directory names **should** be lower-case and **must not** contain spaces. This is to eliminate confusion when interacting with GitHub which is case-sensitive.
* The names of files that contain code **should not** contain spaces. (includes code in source, test, and tools)
* The names of files that contain code **should** use pascal-casing. ( MyCodeModule.cs, Web.config, etc. )
* The names of files in a documents directory **may** contain spaces.

# Build Environment

# Build

# Coding Conventions

## CSharp

### Namespace Declarations

The namespace declaration should be the outermost containment for the document, with using statements and class definitions within. The convention for the namespace for a given source file is:

namespace {company}.{product}.{project}.{folder}

### Using Statements

The using statements should be placed at the top of the source file just inside the namespace declaration. By convention the system namespace references are declared first in alphabetical order, followed by third-party namespace references in alphabetical order, followed by the product namespace references in alphabetical order.

Example:

namespace BuiltSteady.Zaplify.Website.Resources

{

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Runtime.Serialization.Json;

using System.Runtime.Serialization;

using System.Web.Security;

using Newtonsoft.Json;

using BuiltSteady.Zaplify.Website.Models;

using BuiltSteady.Zaplify.ServerEntities;

using BuiltSteady.Zaplify.ServiceHelpers;

…

}

## JavaScript

## T-Sql

# Debugging

## WinPhone Client

To debug both the web server and winphone client at the same time, you should not use F5 to launch the startup project. Instead, use right-click, debug, start new instance on the website project first. Once the browser is launched, use right-click, debug, start new instance on the winphone project. The phone emulator will launch and the winphone application will start. One can exit the phone application, but the emulator SHOULD NOT be shutdown. To restart the winphone application for debugging, simply right-click, debug, start new instance on the winphone project again.

# Deployment

## Domain Management

Domain registration for both builtsteady.com and zaplify.com is managed at <http://www.whois.com>. Login using the corporate email address, [builtsteadysoftware@gmail.com](mailto:builtsteadysoftware@gmail.com) and click on the “My Account” link.

In the “Manage Free Services” section, enter the domain name you want to manage and select the service you want to manage ( DNS mapping, email forwarding, etc. )

## Azure Accounts

BuiltSteady Inc. is registered as a Microsoft BizSpark member, providing access to free software and an Azure account with limited resources available for up to eight developers. There are currently four developer accounts registered with BizSpark ( <http://www.microsoft.com/bizspark> ) and Windows Azure ( <http://windows.azure.com> ).

[stevemillet@comcast.net](mailto:stevemillet@comcast.net), [ogazitt@hotmail.com](mailto:ogazitt@hotmail.com), [dev1@buildsteady.com](mailto:dev1@buildsteady.com), [dev2@builtsteady.com](mailto:dev2@builtsteady.com)

Login into Windows Azure using one of these registered Live IDs to access the hosted services, storage accounts, and SQL Azure used to deploy the product.

## SQL Azure

### DAC

The database projects produce DAC packages which can be used to deploy to SQL Azure. Once initially deployed, a database may be upgraded, retaining the existing data in the database. Use SSMS to deploy a DAC package to SQL Azure. Connect SSMS to the SQL Azure server and select Data-tier Applications under Management and right-click to deploy a new Data-tier Application. Once deployed, one may right-click an existing Data-tier Application to either delete or upgrade the deployment.

NOTE: Currently there are issues trying to upgrade a SQL Azure database using a DAC package. An alternative is to generate a script to extract the data from the current database and apply that to a new database. This assumes the schema for the current and new database are compatible.

### Backup

Before upgrading any SQL Azure database, one should create a backup copy which can be restored in case there are problems during the upgrade process. Using SSMS or the SQL Azure management console, attach to the master database for the server and run the following SQL command:

CREATE DATABASE {backup} as copy of {target}

The process of copying the database may take a while depending on the size of the database being backed up. To check the status of the backup database, attach to the master database for the server and run the following SQL command:

SELECT DATABASE name, state\_desc from sys.databases

The backup database state will change from COPYING to ONLINE when the copy is complete.

### Generating Data Script

One can generate a script which will extract the data from one database to apply to another that has a compatible schema. In SSMS, select the database to extract the data from and right-click, Generate Scripts… This will launch the wizard for generating a database script.

* Choose to script the entire database and hit the Next button
* Enter the name of the file to save the script into and hit the Advanced button
* Set Convert UDDTs to Base Types to ‘True’
* Set Script for database engine type to ‘Sql Azure Database’
* Set Types of data to script to ‘Data only’
* Hit the OK button and the Next buttons to generate the script

The generated script can now be applied to a new database to populate it with the data extracted from the original database, assuming the schema of the new database is compatible.

### Rename

When upgrading, it may be useful to rename an existing SQL Azure database. To rename a database, attach to the master database for the server and run the following SQL command:

ALTER DATABASE {current\_name} MODIFY NAME = {new\_name}

# Test