**Preface**

This document describes basic steps for deploying TimeTrax. TimeTrax is an ASP.Net Web Application developed and maintained by InfoWorks. SQL Server 2016 Database Engine and Reporting Services are used for managing transactional data and reporting.

InfoWorks has three environments: development (DEV); user acceptance testing (UAT); and production (PROD). Please refer to the TimeTrax Network Architecture section in the appendix for the names of servers and databases for each environment.

As stated previously, this document describes the basic steps for deploying the TimeTrax solution. **While the author documented the names servers and databases at the time of writing, it is the responsibility of the individual/s performing the deployment to ensure the accuracy of the names based on the environment to which they are deploying.** The document also assumes that all components (i.e., reporting, database and Web application) are being deployed.

**NOTE: This document does not include the process for migrating the Windows Service created in project IW.TimeTrax.Services which is part of this solution. This service manages the downloading the CRM Opportunity data from the MSDynamics 365 CRM solution to table tblOpportunity in TimeTrax. The process is the same as when the CRM Administrator clicks the**

1. **Developer Tools Needed**
   1. **TortoiseSVN**: a Windows client for [Subversion (SVN)](https://tortoisesvn.net/) version control system
   2. **Visual Studio Professional 2015:** used for development and deployment of the ASP.Net Web application and the SQL Server Reporting Services (SSRS) reports
   3. **SQL Server Management Studio (SSMS)**: an integrated environment for managing any SQL infrastructure used to configure, manage, and administer Microsoft SQL Server
2. **General Source Control Actions:** *Refer to the Source Control section in the appendix for additional instructions and the TortoiseSVN Help documentation for detailed information*. Note that the application consists of two solutions: **TimeTrax** which has the Web, Database and supporting projects, and **TimeTrax-BI** which has the reports and supporting objects (e.g., datasource and shared datasets).

* 1. **Checkout**: To obtain a working copy, perform a checkout from a repository. Select a directory in File Explorer where you want to place your working copy; right click to pop up the context menu and select the command ***TortoiseSVN →* Checkout**. Ensure the URL of the repository is pointing to the **trunk** folder.
  2. **Update**: The process of getting changes from the server to your local copy is known as updating. To update, select the files and/or directories you want in File Exporer, right click and select ***TortoiseSVN →* Update**.
  3. **Commit**: Sending the changes you made to your working copy to the server repository is known as ***committing***. Before you commit, make sure that your working copy is up to date as described in 2.2. If your working copy is up to date and there are no conflicts, you are ready to commit your changes. Right click on any file and/or folders you want to commit in Windows Explorer, then ***TortoiseSVN → Commit***.
  4. **Tag:** When the application is stable and ready for deployment, tag the application as per guidelines in the appendix.

**Prep for Deployment Tasks**

1. **Identify tag name**: Identify the tag name to be used for the release…refer to the Tagging section in the appendix.
2. **Create two folders to store files required for deployment:**
   1. ***DB\_<tag name>***: Database files for deployment
   2. ***Web\_<tag name>:*** Web deployment package for deployment
3. **Database files…**located in Database project of TimeTrax solution (i.e., ...\IW\_TimeTrax\Database):
   1. **SQL scripts**: Create and/or update SQL scripts. SQL scripts are stored in the appropriate subfolder (e.g., DDL, DCL, DML) of the Database folder.
   2. **Batch file**: Create a batch file that will call the SQL scripts for modifying the database.
      1. Copy and rename the last deploy batch file using the naming convention ***deploy\_v#\_#\_#*.** For example, if this update is for version 4 modification 1 release 0, the filename will be *deploy\_v4\_1\_0.*
      2. Update header section
      3. Update values for version, modification and release
      4. Delete calls to previous scripts *except where instructed otherwise*
      5. Add calls to scripts as needed
      6. **Zip the Database folder**:
         1. Zip the Database folder (i.e., **...\IW\_TimeTrax\Database**). Ensure the solution is closed in Visual Studio
         2. Place zip file in ***DB\_<tag name>*** folder
      7. Rollback prep: rollback is accomplished by restoring database backup file
4. **Reporting** (i.e., *TimeTrax-BI.sln):* 
   1. Open TimeTrax-BI.sln in VS 2015
   2. Select the appropriate Solution Configuration (i.e., UAT or PROD)
   3. Right-click the solution and select Build
   4. Observe the process for any errors and/or warnings and take the appropriate action if needed
   5. Rollback prep: Develop plan to rollback changes to reporting project if needed
5. **Update TimeTrax Version Number**:
   1. Open TimeTrax in VS 2015
   2. Update the version number in the appSettings section of the **Web.UAT.config** if deploying to UAT or **Web.Prod.config** if deploying to production
   3. Build the solution using the appropriate Solution Configuration setting (i.e., UAT or PROD)
6. **Create the Web Deploy Package**
   1. Open the *TimeTrax solution* in VS 2015, right-click on the Web project **IW.TimeTrax.Web,** click Publish and select the appropriate profile:

* TimeTraxDEV for development…note that development may be done locally.
* TimeTraxUAT for UAT
* TimeTraxProd for Production
  1. Click the Publish button to build a deployment package which is located at
* …\IW.TimeTrax.Web\obj\DEV\Package for development
* …\IW.TimeTrax.Web\obj\UAT\Package for TimeTraxUAT
* …\IW.TimeTrax.Web\obj\PROD\Package for TimeTraxProd
  1. Copy the folder named **Package** to ***Web\_<tag name>*** folder
  2. Delete the folder named **PackageTmp** (in the ***Web\_<tag name>*** folder) leaving four files and one zip folder
* IW.TimeTrax.Web.deploy.cmd
* IW.TimeTrax.Web.deploy-readme.txt
* IW.TimeTrax.Web.SetParameters.xml
* IW.TimeTrax.Web.SourceManifest.xml
* IW.TimeTrax.Web.zip

1. **Tag solution**: Using TortoiseSVN, tag solution with the tag name identified in step 3
2. **Create App\_Offline.htm file**: Create a htm file named App\_Offline that will be placed in the root of the TimeTrax Web site content folder. App\_Offline.htm is a page that you can use to take your site offline when deploying updates or performing maintenance. The file allows you to completely shut down your website and route all requests to the app\_offline page. The file should have a HTML message indicating that the application is down for maintenance.
3. **Confirm contents of *Web\_<tag name>*** **folder:** 
   * **Web deployment package** created in step 8
   * **App\_Offline folder** containing App\_Offline.htm file from step 10
   * **Rollback folder**: A rollback is accomplished by restoring the Web files on the application server that are copied in step 13 during actual deployment.
4. **Confirm contents of *DB\_<tag name>*** **folder**
   * **Database zip folder** from step 5.2.6

**Deployment Tasks**

1. **Create a backup copy of the Web site:**
   1. Copy and paste the Web files (i.e., TimeTrax folder) to the Desktop in the event a rollback is required**.**
   2. Web site file locations:

* **DEV:** [\\app-tmtrx-d01.iwtn.com\D$\TimeTrax](file:///\\app-tmtrx-d01.iwtn.com\D$\TimeTrax)
* **UAT:** [\\app-tmtrx-t01.iwtn.com\D$\TimeTrax](file:///\\app-tmtrx-t01.iwtn.com\D$\TimeTrax)
* **PROD:** [\\app-tmtrx-p01.iwtn.com\D$\TimeTrax](file:///\\app-tmtrx-p01.iwtn.com\D$\TimeTrax)

1. **Put Web site in maintenance mode**: copy and paste the **App\_Offline.htm** file from the App\_Offline folder in the *Web\_<tag name>* folder to the root folder of the TimeTrax Web site
2. **Run deployment batch file (i.e., deploy\_v#\_#\_#) if there are updates to the database**
   1. Backup transactional database

* **UAT** database server and database names are **db-mssql-t01.iwtn.com** and **TimeTraxUAT**
* **Production** database server and database names are **db-mssql-p01.iwtn.com** and **TimeTraxProd**
  1. Unzip Database.zip folder from ***DB\_<tag name>*** folder to Desktop of database server
  2. Open a command prompt and set the directory path to **…\Database\deploy\deployments**
  3. Run the following command. Arguments passed are:
* Batch file name
* Server name
* Database name
* User account – Service account for environment (e.g., TimeTraxDevSvc, TimeTraxUATSvc, TimeTraxProdSvc)
* Password

The command will look something like this **example**:

**DEV – Local: deploy\_v4\_2\_0 “IWHQT-750-02” TimeTraxDev <User> <Pwd>**

**DEV – Remote: deploy\_v4\_2\_0 “db-mssql-d01.iwtn.com” TimeTraxDev <User> <Pwd>**

**UAT: deploy\_v4\_2\_0 “db-mssql-t01.iwtn.com” TimeTraxUAT <User> <Pwd>**

**PROD: deploy\_v4\_2\_0 “db-mssql-p01.iwtn.com” TimeTraxProd <User> <Pwd>**

* 1. Review log file output to **…\Database\deploy\deployments\logs** for errors

1. **Deploy reports if there are new or updated reports**:
   1. Open the *TimeTrax-BI solution* in VS 2015 and ensure the appropriate Solution Configuration setting is selected: **UAT** for test and **PROD** for production
   2. Build and deploy report project or individual report objects (e.g., reports, data source, datasets).

**CAUTION: With moving to SSRS 2016, we discovered the logo.png file used in each report as an external image is not displayed in Report Manager or when a report is exported as a PDF. This is due to changes in SSRS 2016. The current countermeasure is to either**

**1) exclude the image file from the deployment if it has already been deployed and is being displayed**

**OR**

**2) open SSRS Report Manager; open Reports folder and upload the image file**

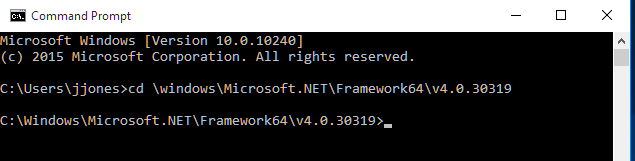
* 1. Observe the process for any errors and/or warnings and take the appropriate action if needed

1. **Deploy Web Package**
   1. Copy the ***Web\_<tag name>*** folder to the Desktop of the Web server
      * + DEV: app-tmtrx-d01.iwtn.com
        + UAT: **app-tmtrx-t01.iwtn.com**
        + PROD: app-tmtrx-p01.iwtn.com
   2. **If deploying to production**, edit **IW.TimeTrax.Web.SetParameters.xml;** replace “abc123” with the production database encryption key in the parameters list.

In the 03 TimeTrax Database/Databases.xlsx workbook.

* 1. Open a command window **as administrator** and change the directory to the Package folder (**i.e., …\Desktop\*Web\_<tag name>***\**Package**).
  2. Run the deployment file with the following command: **IW.TimeTrax.Web.deploy.cmd /Y**

1. [Secure ASP.Net Configuration](https://msdn.microsoft.com/en-us/library/zhhddkxy.aspx): Encrypt the secure settings in the [Web.config](https://msdn.microsoft.com/en-us/library/53tyfkaw.aspx) file by running following command (or use the batch file **EncryptSecureSettings.bat** on the server desktop) on Web server
   1. Open command prompt **as administrator** on appropriate Web server
   2. Set directory path to: **cd \windows\Microsoft.NET\Framework64\v4.0.30319**
   3. Run the following command: aspnet\_regiis -pef secureAppSettings D:\TimeTrax



1. Test application to ensure that the application is now running and accessible.
2. Clean up: Delete any folders/files on servers that are no longer needed.

**Steps for rollback**

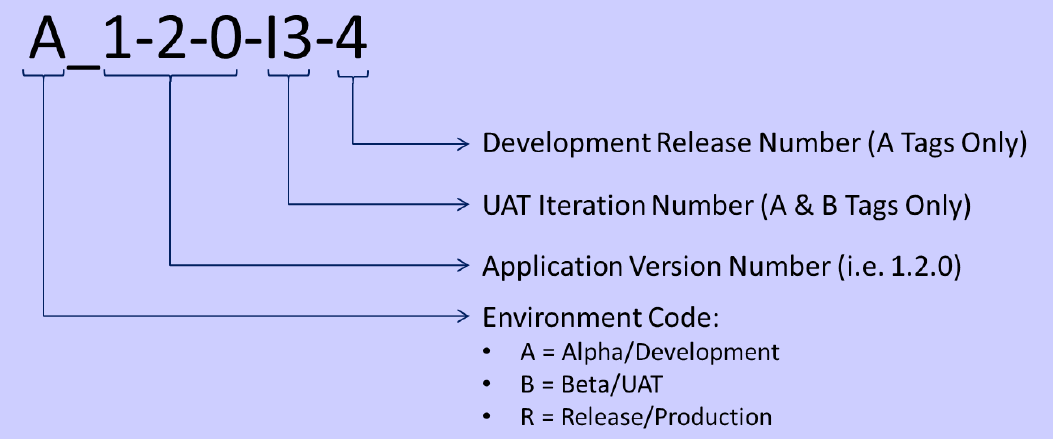
1. **Rollback**
   1. **Web site**: Place TimeTrax Web site in Maintenance Mode
   2. **Database**: Restore most recent backup files (.bak and .trn)
   3. **Reporting**: Execute plan to restore reporting solution to previous state
   4. **Web application**: Restore backup of Web site content made in step 12
   5. **Test the Web application**

**Appendix**

1. **Source control:** All development files are managed by Subversion (SVN): reports, database scripts and application source code. Access to the repository and performing all SVN operations is via the Windows client TortoiseSVN. What follows is are descriptions of tasks performed with SVN at a high level**…please refer to TortoiseSVN Help document for detailed information.**
   1. **Checkout**: A Checkout will download the code from the SVN repository to the local machine creating a local working copy. Permission is required to access the SVN server; contact SVN administrator Jenny Baldwin if needed. A Checkout is performed from the SVN repository at the Trunk level.
      1. In File Explorer, create a new folder where you want to store the repository contents
      2. Right-click the folder and select SVN Checkout
      3. Enter the URL https://svn.infoworks-tn.com/svn/IW\_TimeTrax/trunk to connect the repository and select OK
      4. Enter the user name and password if needed
   2. **Update**: Periodically, you should ensure that changes done by others get incorporated in your local working copy. The process of getting changes from the server to your local copy is known as updating. Note that it will be necessary to resolve conflicts if the update contains changes to the same section of code that you have modified. To update, select the files and/or directories you want in File Explorer, right click and select TortoiseSVN → Update.
   3. **Commit**: Committing is the act of sending the changes you made to your working copy back to the repository. Before you commit, make sure that your working copy is up to date as described in 6.2. If your working copy is up to date, stable and there are no conflicts, you are ready to commit your changes. Note that performing a commit is expected when committing to the trunk or a branch but not a tag version of the code; you will be prompted if you try to commit to a tag. Right click on any file and/or folders you want to commit in Windows Explorer, then TortoiseSVN → Commit.
   4. **Trunk, Branch, Merge, and Tag:**
      1. **Trunk**: The main line of development.
      2. **Branch**: A feature of version control systems to isolate changes onto a separate line of development. This line is known as a *branch*. Branches are often used to try out new features without disturbing the main line of development with compiler errors and bugs. As soon as the new feature is stable enough then the development branch is *merged* back into the main branch (trunk).
      3. **Merge:** Where branches are used to maintain separate lines of development, at some stage you will want to merge the changes back into the trunk. Merging always takes place within a working copy (i.e., the copy on your computer).
      4. **Tags**: A feature of version control systems to mark revisions (e.g. a release version), so you can at any time recreate a certain build or environment. Tags can be created from a branch or the trunk. **Be aware that changes made in a branch are not in the trunk and vice versa unless a merge has taken place.** While creating tags for UAT may be appropriate, changes in the branch should be merged into the trunk prior to creating a release tag and web package for production.

**It is important to understand how branching and merging works in SVN before you start using it**, as it can become quite complex. It is highly recommended that you read the sections on Branching and Merging in the **TortoiseSVN Help** document before performing these operations.

* 1. **Switching**: At any given time, there will be different versions of the code stored in the tags, branches and trunk folders. Using the Switch feature allows the developer to switch between versions of code within these folders. To perform a switch, select the top-level folder of your project in File Explorer, right click and select TortoiseSVN → Switch. Although Subversion itself makes no distinction between tags and branches, the way they are typically used differs a bit.
  2. **Tagging**: Another feature of version control systems is the ability to mark revisions (e.g. a release version), so you can at any time recreate a certain build or environment. This process is known as tagging. InfoWorks tags releases for UAT and Production. To identify the tag name:
     1. Check the last Tag name from the SVN repository Tags folder for the project for the Beta environment. Note that tags for the Beta/UAT environment will begin with the letter B and production begin with the letter R.
     2. Increment the Version, Modification, Release or Iteration number as appropriate
     3. Tag naming format: Additional information is in the file named ‘Source Control…’ in folder “05 Training and Documentation” on TimeTrax team site in SharePoint.



* 1. **Making revisions to a release:** Tags are typically used to create a static snapshot of the project at a specific stage. As such they are not normally used for development - that's what branches are for, which is the reason for the /trunk /branches /tags repository structure. ***Working on a tag revision is not a good idea***, but because your local files are not write protected there is nothing to stop you doing this by mistake. However, if you try to commit to a path in the repository which contains /tags/, TortoiseSVN will warn you.

**It may be that you need to make further changes to a release which you have already tagged.** **Using tag B\_4-1-0-I1 as an example**, the recommended way to handle this is to:

* + 1. Create a new branch (e.g. F\_4-1-0) from the tag B\_4-1-0-I1
    2. Switch to the new branch F\_4-1-0
    3. Make the changes on this branch and commit
    4. Create a **new tag** from the branch
       1. If the release is another iteration for UAT, the tag would be B\_4-1-0-I2
       2. If the release is for production, the tag would be R\_4-1-0
    5. **REMEMBER TO MERGE CHANGES IN BRANCH INTO TRUNK.** If you modify a working copy created from a branch and commit, all those changes are committed to the branch and not the trunk. Only the modifications are stored. The rest remains a cheap copy. To get the changes into the trunk, you must:
       1. Make the trunk your working copy by switching to the trunk version
       2. Merge the changes from the branch into your working copy (i.e., trunk)

1. **Network Architecture (as of 2018/04/25)**

