

***Deployment Strategy (Infrastructure, Technical Architecture, Code & Build Management)***

**DOCUMENT INFORMATION**

File Name: Deployment Strategy

|  |  |  |
| --- | --- | --- |
| Author & Updates | | |
| Name | Version | Date |
| Christian Cabrera | 1.0 | 5/27/2016 |
| Tawhid Rahman | 2.0 | 6/13/2016 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**TABLE OF CONTENTS**

[1 Objective 4](#_Toc453589350)

[2 Process Overview 4](#_Toc453589351)

[3 Environments 4](#_Toc453589352)

[4 Code development and Check-ins 5](#_Toc453589353)

[5 Release/Build versioning 5](#_Toc453589354)

[6 Code Reviews 6](#_Toc453589355)

[7 Release Schedule 6](#_Toc453589356)

[Tuesday (New Code cutoff/Deploy to Constest) 6](#_Toc453589357)

[Wednesday (Deploy to Test for business testing) 6](#_Toc453589358)

[Thursday (Testing continues) 6](#_Toc453589359)

[Friday (Testing continues) 6](#_Toc453589360)

[Monday (Production Deployment) 6](#_Toc453589361)

[8 Planned release Schedule for Deployment to a Production Environment 7](#_Toc453589362)

[9 Hotfixes 7](#_Toc453589363)

[Overview 7](#_Toc453589364)

[Process Steps 7](#_Toc453589365)

[Triage call 7](#_Toc453589366)

[10 Build Process (Update Code) 8](#_Toc453589367)

[11 Deployment Steps/Checklist 9](#_Toc453589368)

[12 Release Documentation 10](#_Toc453589373)

[13 Post Deployment Health Check 10](#_Toc453589375)

[14 Deployment Back out Process 10](#_Toc453589376)

[15 Administration Roles and Responsibilities 10](#_Toc453589377)

[16 SLA’s 11](#_Toc453589378)

[17 Appendix A 11](#_Toc453589379)

# Objective

This document details the overall process and controls used by Bottomline Technologies to successfully migrate the AX solution from development to other preproduction environments.

# Process Overview

The Microsoft Dynamics AX solution at Bottomline Technologies needs to be periodically moved from development environment to the other preproduction environments. This document defines the deployment processes that will be followed to help ensure a successful deployment of the solution to the other preproduction environments on a weekly basis (or as decided based on the release cycle requirements). The steps below detail the process that needs to be followed for deploying the code artifacts from development to the Bottomline environments.

# Environments

| **Environment** | **Environment Purpose** |
| --- | --- |
| **Sandbox** | Testing of configurations which may be disruptive to the primary test activities |
| **CONSTest** | Consultant build verification |
| **ConvDAT** | Data Migration/Testing |
| **Development Environments** | Code Development |
| **Build Environment** | Compilation and generation of model stores |
| **Master/Staging** | Repository for approved setup/configurations |
| **Test** | Requirements Analysis and Functional Testing |
| **UAT** | Process, End to End and User Acceptance Testing |
| **Training** | End user training |
| **Production** | Production Environment |
| **Disaster Recovery** | Intended to take over as production use in case of a surprise issue at the main site. |
| **High Availability** | Intended as a temporary take over in case of a problem with the main machine. (Clustering) |
| **ISR Test** |  |
| **Q2C Environments** | TBD |

# Code development and Check-ins

All code to be included in a build must be checked into the source control (TFS) before the build process is initiated. Development needs to follow the check-in procedure for all code being check-in before this time.

The code MUST compile and it is the responsibility of the Developer who checked in to ensure that the code compiles and all source that is needed is checked in, in its entirety.

All check-ins MUST NOT break functionality in the solution. It is the responsibility of the developer checking in the code to clean up any breaks caused by their check in. The person checking in the code must fix the dependent code when they check-in or the developer must work with the other developer to coordinate the check-in.

The source control MUST NOT be left in a broken state at any time.

If any of the above conditions cannot be met, the code SHOULD NOT be checked into the source control.

# Release/Build versioning

The release/build will be versioned in 2 ways in order to both track and determine which release version an environment is currently running. First during the build process, see Build Process section, the source files in TFS will be labelled with the version.

The Version numbering consists of a four-part number separated by periods (e.g. 6.0.1108.670). Below is the break down for each part of the version number.

1. Major AX release
2. Feature pack
3. CU version
4. Build

In the above the 4th value will be updated each time a build is cut and deployed.

Following this structure, the 6th build for Customer with CU3 and no feature pack would be identified as “6.0.3.6”. This version is the Label in TFS as well as the version of the VAR, VAP, CUS, and USR models.

# Code Reviews

The Technical Lead or Development lead will review code changes prior to deployment to verify that development standards have been followed, and that programming logic is sound. The code should also be run through CAT (LCS Code Analysis Tool). The ISV code will be reviewed by the Bottomline development team.

# Release Schedule

Below is the release schedule for moving code to ConsTest and Test. All other Bottomline environments including ConvDAT and Master Configuration will be updated with the latest code release as needed. The other environments will have scheduled updates as to not impact activities and taking into the consideration that the Data Model changes during the stabilization phase of the project are not very frequent if at all. Again this is an example of what the schedule would look like once the release

Example: (Note this is a Tuesday through Monday schedule. Testing starts on Tuesday and the build deployment is pushed on Monday)

## Tuesday (New Code cutoff/Deploy to Constest)

* Code cut off @ 9am EDT Time
  + All reviewed and approved code needs to be checked into TFS
* Deploy build to ConsTest during the afternoon/evening
* Pre-Release notes will be generated and sent to Core Team @ 9pm EDT Time
* Custom code development, fix generation, code remediation

## Wednesday (Deploy to Test for business testing)

* Testing of changes included in release in ConsTest by core team
* Deploy build to Test @6pm EDT
* Release notes sent to users immediately following the build deployment to Test
* Custom code development, fix generation, code remediation

## Thursday (Testing continues)

* Testing of changes included in release in Test
* Custom code development, fix generation, code remediation

## Friday (Testing continues)

* Testing of changes included in release in Test
* Custom code development, fix generation, code remediation

## Monday (Production Deployment)

* Code review by Technical/Development Lead to take place in the morning EDT time
  + This is to include running the model through the CAT tool in LCS
* Custom code development, fix generation, code remediation
* Deploy approved build to Production @ 8pm EDT

# Planned release Schedule for Deployment to a Production Environment

* Deploy Build to ConsTest No Later than 3 weeks prior to agreed Go Live. QA tests AX as determined in appropriate Functional Design Document.
* Once QA confirms successful test, deploy to TEST environment no later than 2 weeks prior to agreed Go Live. User Acceptance Testing is then performed until QA satisfied complete.
* Deploy Code from TEST to PROD only when QA confirms completion of UAT. Release to Production Environment to be scheduled on a Saturday starting at 12:00 GMT to avoid disruption to Global Users. Deployment Tested in Production once complete by QA. In the event of roll back deployment will have to be fully complete by 05:00am on the Following Sunday, prior to Israel business starting.

# Hotfixes

## Overview

If a Critical issue (Test stop) is encountered, a triage call needs to be put together once a fix is generated in order to discuss the need for fast track deployment, and testing in ConsTest. The fix can be deployed to ConsTest via an XPO. Once those conditions have been met as determined in the triage call, it will be determined if it needs to be deployed via XPO to Test and Production ahead of the next release or if it can be released on the next release cycle. All code movement needs to be approved by the Technical/Development Lead.

## Process Steps

1. Critical Stop issue encountered
2. Issue is investigated by Bottomline EA team
3. Issue reported to Functional Consultant
4. Functional Consultant determines if it is a setup issue and/or if it can be reproduced in Test
5. If Functional Consultant can reproduce and determines it is not a setup issue, the issue is than reported to a Technical Consultant(Developer)
6. Technical Consultant troubleshoots issue and determines if it requires a code fix, product fix, or infrastructure fix
7. In the event that the resolution of the issue requires a code fix whether from a Product Hotfix or development fix, the fix is first put into the a developer machine and checked into TFS
8. Technical Consultant loads Fix into Test
9. Functional Consultant tests issue in Test
10. If the issue is resolved in Test than a Triage call is initiated else go back to step 6.
11. If determined in the Triage call that the fix needs to be moved forward prior to the normal release schedule, it will be scheduled to move into Process Test as a hotfix at an agreed upon time.
12. The hotfix will be included in the release notes and designated as a hotfix that was promoted and on what date.

## Triage call

The Triage call that is setup as part of the hotfix process will be used to determine;

* Severity of the issue
* Issue impact on the system
* Fixes impact on the system
* Cost of moving the fix prior to the next scheduled release

The participants of the Triage call must include the following;

* Functional Consultant who diagnosed and raised the issue
* Technical Consultant who created the hotfix
* Technical/Development Lead
* Delivery Architect
* Representation from PMO (MCS PjM, EM, or other governance representative)

All issues in regards to process, severity, and impact will be tracked in TFS before and after go-live then handed off to Bottomline Technologies helpdesk system.

# Build Process (Update Code)

| **Task** | **Description** | **Complete** | **Completion Date** | **Consultant** |
| --- | --- | --- | --- | --- |
| **Check-in source to TFS** | Ensure all developers have checked in all source and verify via pending objects inquiry. | No |  |  |
| **Initiate build process on the build server** | Perform build process, see appendix A for the Automated Build Scripts Quick Guide | No |  |  |
| **AOS Stop** | Stop the AOS service in Target (e.g. ConsTest) Environment | No |  |  |
| **Stop Management Reporter** | Stop the management reporter integrations in target environment | No |  |  |
| **Backup target environment** | Do target Backup in case of rollback | No |  |  |
| **Import Model store to target** | Import the Model Store into Target (e.g. ConsTest) | No |  |  |
| **AOS Start** | Start AOS in Environment | No |  |  |
| **Run Through Software Update Checklist** | See specific steps in appendix A for the Automated Build Scripts Quick Guide | No |  |  |
| **Deploy Reports, Cubes, EP, and DLLs** | These components are on an as needed basis, if the release has changes to them they will be redeployed | No |  |  |
| **Start Management Reporter** | Start the management reporter integrations in the target environment | No |  |  |
| **Validate Environment** | Quick Smoke Test in Environment | No |  |  |

# Deployment Steps/Checklist



The deployment process involves taking the AX Solution and deploying it to various Bottomline Technologies environments. Depending on the target environment the deployment steps can very. Also depending on if the solution has been deployed to the environment or not there may be additional steps to get the complete solution installed and up to date, see Appendix for an example of the plan that was put together for updating the environments with CU3 and associated ISV, and customized code.

Below are the steps for updating an Environment:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Description** | **Complete** | **Completion Date** | **Consultant** |
| **Export Model Store from Build Environment** | Export the latest model store from build box | No |  |  |
| **Stop the Target AOS** | Stop the target environment’s AOS | No |  |  |
| **Stop Management Reporter** | Stop the management reporter integrations in target environment | No |  |  |
| **Make a backup of the Target Database** | Make a of the database which includes data and code | No |  |  |
| **Apply Model Store to Target** | Apply the latest Model Store | No |  |  |
| **AOS Start** | Start AOS in Environment | No |  |  |
| **Run Through Software Update Checklist** | See specific steps in appendix A for the Automated Build Scripts Quick Guide | No |  |  |
| **Deploy Reports, Cubes, and EP** | Deploy necessary BI and portal artifacts | No |  |  |
| **Start Management Reporter** | Start the management reporter integrations in the target environment | No |  |  |
| **Validate Environment** | Quick Smoke Test in Environment | No |  |  |

# Release Documentation

As part of releasing the build to the team a Release Notes Document will be sent out to the team. This document will contain all changes to the solution. These changes typically include Product Hotfixes, Bug fixes, and ISV updates.



# Post Deployment Health Check

[Description: The Testing Deliverables and Responsibilities section is used to identify the teams and individuals who will be both managing and executing the testing process and the associated deliverables. This information may be placed in a table or matrix that identifies each key element of the testing process and the people who will participate in that process. This section will also define responsibilities for compiling data to be used for testing. It is useful to use a resource assignment matrix (such as RACI) to show what role or organization is responsible, accountable, consultative and informed.]

Once an environment has been update with a new release/build the Release Manager and Delivery Architect will check the environment to make sure things are working properly before releasing the environment to others. Things that will be check are ISV solutions, Reports, and other key functional components.

# Deployment Back out Process [Description: In this section, describe the nature of the testing data for test execution and responsibilities for provision and acceptance of the data.]

A back out process should be handled on a case by case basis. In the event that a build needs to be backed out in the backup of the environment that was taken just prior to the code deployment should be restored bringing the build back to the previous release.

# Administration Roles and Responsibilities

With many environments that are many roles that are a necessity to drive the deployment forward. The following are the roles and responsibilities that we have identified:

* AX Application Administrator (BT: Krista Goodyear) (MS: Christian Cabrera)
  + Handles all aspects of the application infrastructure respective to the AX application
* Build manager (Bottomline: Victor Tang) (MS: Vipriya Gurav)
  + Reviews and merges code from AX, custom development, ISVs, and provides a model store for AX admins to import
* Core infrastructure administrator (BT: Bob Love’s Team) (MS: Christian Cabrera)
  + Management of servers, VM/Hyper-V administration, network administration, etc.
* Security administrator (BT: Krista Goodyear/Thomas Joy) (MS: Christian Cabrera)
  + Determines the roles, duties, and privileges for users/groups (Audit/SOX)
* Developers (BT: Victor Tang, Steve Bredall, TBD) (MS: Offshore Development Team)
  + Developers who will be developing code in AX
* QA Lead (BT: Cain Allan) (MS: Michael Herold)
  + Determination of sign-off build release

# SLA’s

The determination of Service Level Agreements need to be counted upon. Usually there are levels of degrees of setup.

* Level I – Frontline helpdesk issues w accounts/access
* Level II – Business analysts, system administration, process issues
* Level III – Development issues

Along with the levels of support, a business impact of severity should be created such as:

* Severity 1 – Critical Impact/System Down
* Severity 2 – Significant Impact/Urgent
* Severity 3 – Minor Impact/Normal
* Severity 4 - Low

# Appendix A

The document below describes the change and lifecycle management process for AX including generating builds from source control in TFS.

