**IGHS – Continuous Delivery Pipelines**

**IGHS Continuous Delivery Pipelines**

**User Guide**

**IGHS Jaguars Team**

|  |  |
| --- | --- |
| Author: | Sathish Ramaraj,  Sandeep Saxena |
| Contributors: | |
| Project Manager: | |
| Document Version: | 1.0 |
| Initially Created: | 9 Dec, 2014 |
| Last Updated: | 9 Dec, 2014 |
| Signed Off by: | |
| Signed Off Date: | |

[**Deployment Pipeline**](http://geekswithblogs.net/TarunArora/archive/2011/06/06/unshelve-shelveset-created-from-one-branch-to-another.aspx)**s**

The deployment pipeline is the key pattern that enables continuous delivery. A deployment pipeline implementation provides visibility into the production readiness of your applications by giving feedback on every change to your system. It also enables team members to self-service deployments into their environments.

**IGHS** [**Pipelines Overview**](http://geekswithblogs.net/TarunArora/archive/2011/06/06/unshelve-shelveset-created-from-one-branch-to-another.aspx)

IGHS Pipelines are built using standard tool set of TFS, Jenkins and Nolio. Jenkins acts as an orchestrator and Nolio takes care of deployments.

The flow is:

1. The component manifest will be generated for each primary components build / TFS check-in.
2. Jenkins jobs will be triggered based on the schedule.
3. Jenkins triggers required Nolio workflow to deploy the changed components in DEV (**Pipelines are working only till DEV right now**).
4. Post deployment completion Jenkins sends a notification about successful deployment.

TFS Check-in

Component

Manifest

Deployment

Using Nolio CLI

**Manifest Template update from Nuget Package.cofig**

**Initiate Schedule deployment from Jenkins**

**Notification from Jenkins for Deployment status**

**Nuget Packages Update Automation**

The version of shared libraries which is used in code referencing should also be used to generate the manifest. **This is very important as the same version should be used for code reference and deployment**. The manifest generation build have flexibility to update the manifest template automatically for shared libraries version from Nuget package config file. The component manifest build initiates the Nuget update activity before generating the manifest.

**What is a primary Build?**

Primary build is one of the MSI package for the component manifest. The primary build will be initiating the manifest generation and the same manifest version should be picked up by build pipeline. The name of the Primary build definitions and Manifest Generation Build Definition is given in below table.

**NOTE: The manifest generation for below components is disabled in latest version of NUI Ext. In case someone need a manual deployment, please generate the manifest using the below shown build definitions in TFS.**

|  |  |  |
| --- | --- | --- |
| **Component Name** | **Primary Build definitions Name** | **Manifest Generation Build Definitions Name** |
| BOA | Tesco.Com.Web.BackOffice | RTC3.0\_Component\_Manifest\_BackOffice\_CD |
| Login | Tesco.Com.Web.Login | RTC3.0\_Component\_Manifest\_Login\_CD |
| LoginUIAssest | Tesco.Com.Web.Login.UIAssets | RTC3.0\_Component\_Manifest\_LoginUIAssets\_CD |
| Checkout | Tesco.Com.Web.UI.Grocery.Checkout | RTC3.0\_Component\_Manifest\_Grocery\_OrderCheckout\_CD |
| CustomerProfile | Tesco.Com.Web.UI.Grocery.MyAccount | RTC3.0\_Component\_Manifest\_Grocery\_CustomerProfile\_CD |
| BasketBuilding | Tesco.Com.Web.UI.Grocery.ShoppingCart | RTC3.0\_Component\_Manifest\_Grocery\_BasketBuilding\_CD |
| Home | Tesco.Com.Web.UI.Grocery.Home | RTC3.0\_Component\_Manifest\_Grocery\_Home\_CD |
| Delivery | Tesco.Com.Web.UI.Grocery.Delivery | RTC3.0\_Component\_Manifest\_Grocery\_Delivery\_CD |
| FindProducts | Tesco.Com.Web.UI.Grocery.Products | RTC3.0\_Component\_Manifest\_Grocery\_FindProducts\_CD |
| GroceryHost | Tesco.Com.Web.UI.GroceryHost | RTC3.0\_Component\_Manifest\_Grocery\_GroceryHost\_CD |

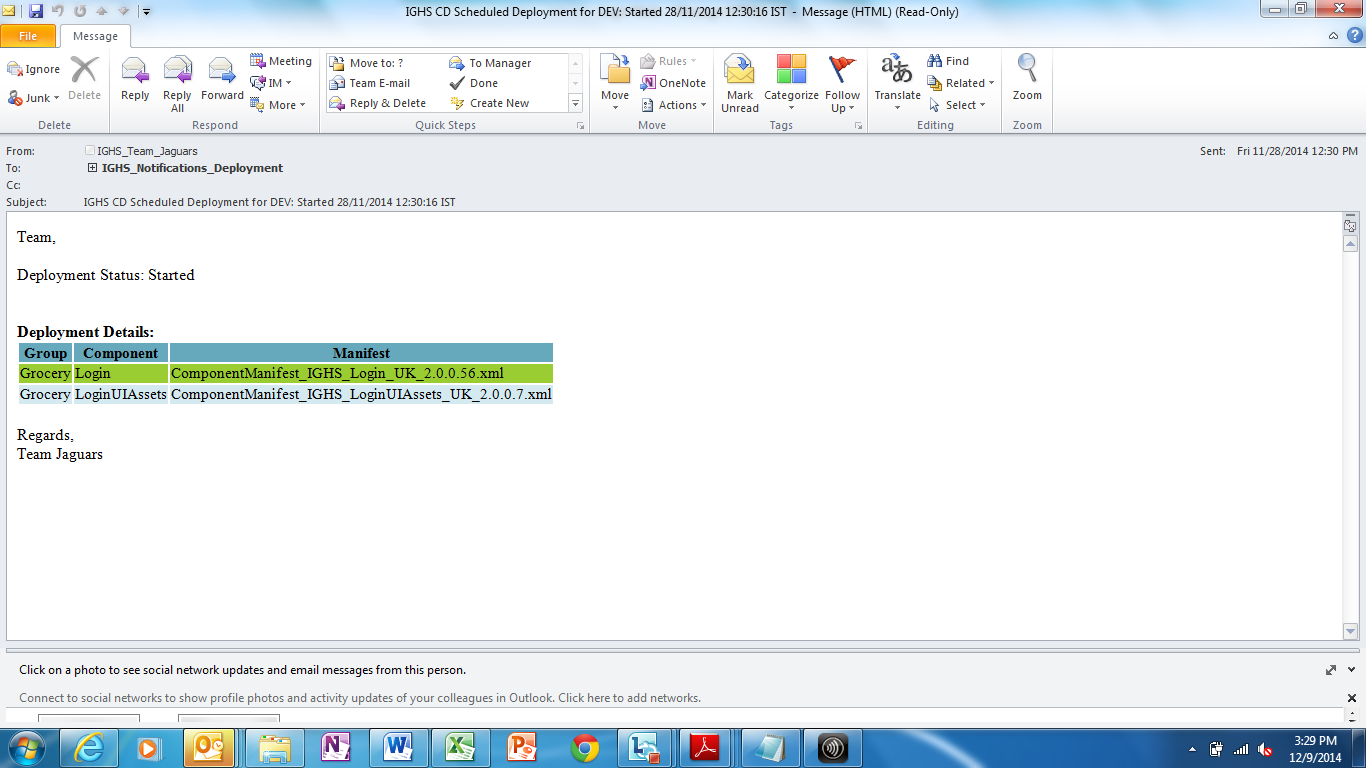
🛈 **If you want to deploy some components without check-in of primary build, you should trigger manually the manifest generation build definition. And the new manifest version would pick the latest artifacts which will be deployed automatically using build pipeline.**

**Dev. sheet updation in TFS**

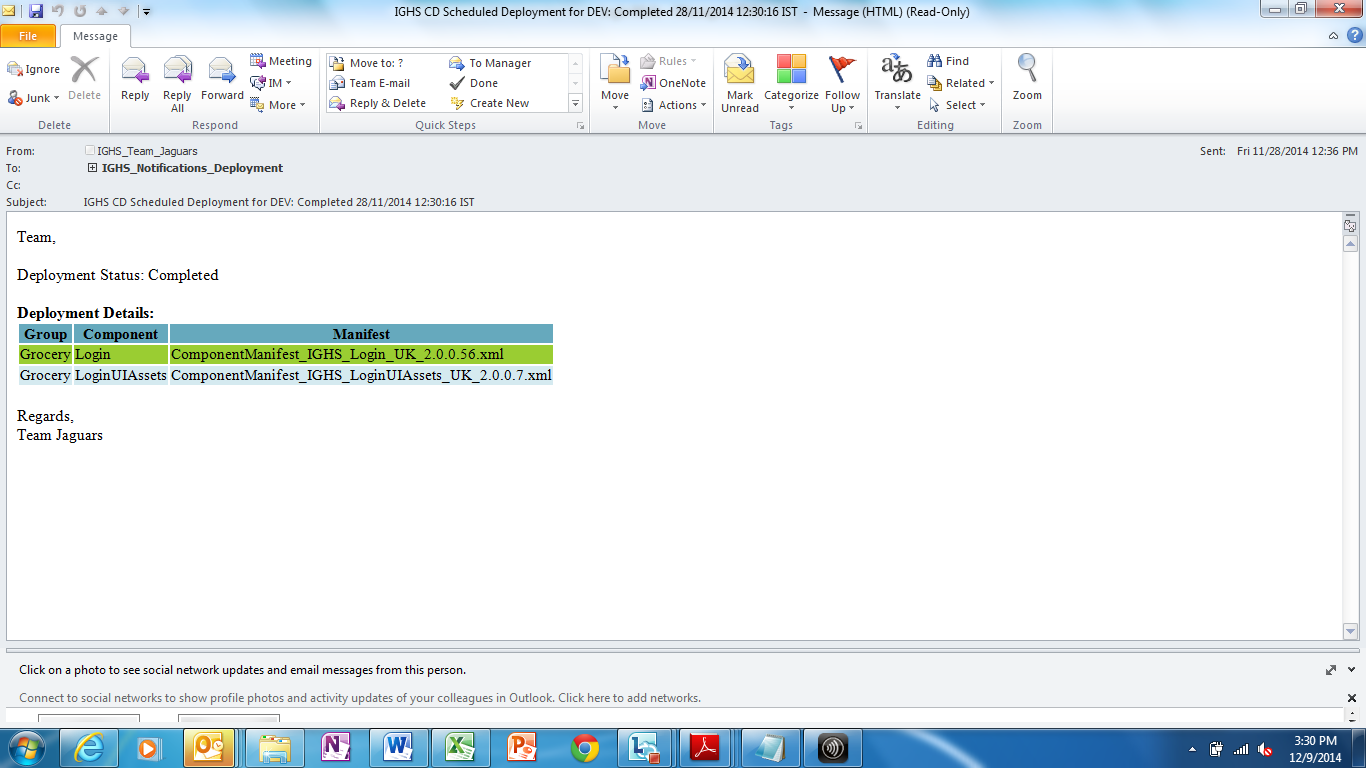
For every component’s deployment you would get a notification mail from build pipeline, which contains deployment status and manifest version. Please carry out sanity in DEV and then update the manifest version in Dev. sheet in TFS *($/InternationalDeployment/Deployments/R2.0/DEV/ DEV\_Component\_Versions.xls)* and the same would be picked for the STG deployments.

**Deployment Notifications from Pipelines**

***Deployment Started :*** Only components which are highlighted in green are changed and getting deployed*.*



***Deployment Completed***



**Next Steps for Pipelines:**

1. As of now following components are put on continuous delivery pipelines

OrderCheckout, CustomerProfile, GroceryHost, Delivery, BasketBuilding, FindProduct, Home, BOA, Login, LoginUIAssest

1. Currently pipelines are working only till DEV.
2. Next step is to integrate the automated DEV test sanity packs in the pipelines and extend the pipelines to STG
3. Another next step is to build the pipelines for rest of the components including the API Services.

**Till we extend the pipelines to STG, Jaguars has developed a tactical approach for automated STG deployments which does not need any manual intervention for deployment. This approach is not part of the Continuous Delivery pipelines and would not be required once pipelines are extended till STG. The approach is explained in the next section.**

**Schedule Deployment for STG**

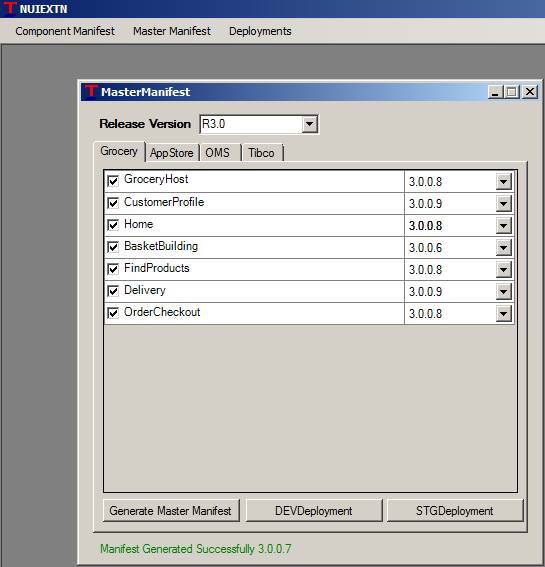
The Scheduled Deployments has been implemented for STG environment. This Deployment would be picking the signed off manifests for STG deployment from TFS and Build Output folder. The scheduled workflows can only deploy the master manifest – so you would need to generate a master manifest using NUI for the components which need to be deployed to STG (even for single component).

**Creating Master Manifest for STG Schedule**

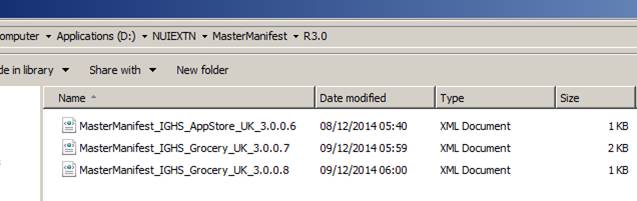
Master manifest generation is required only for Grocery/Appstore components.

***Master Manifest Generation:***

**Step 1:** Generate the master manifest from NUIEXTN as given below. Even If you need to deploy single Grocery/Appstore component, you need to generate master manifest.



**Step 2:** Generated master manifest will be dropped to your local system *@ D:\NUIEXTN\MasterManifest.*



**Step 3:** These manifest files need to check-in to TFS at below given paths in TFS. There is a TFS build configured which will copy manifest file from TFS to Drop location ([*\\Uktee01-clusdb\IGHSBuildOutput\IGHS\_Manifest\ScheduldedDeployment\STG\<ComponentName*](file:///\\Uktee01-clusdb\IGHSBuildOutput\IGHS_Manifest\ScheduldedDeployment\STG\%3cComponentName)*>).* And this manifest files would be used for the scheduled STG deployment.

|  |  |
| --- | --- |
| **Component Name** | **TFS Path for Scheduled Deployment Manifest Check-in** |
| Appsotore | $/InternationalDeployment/ScheduldedDeployment/STG/AppStore |
| BOA | $/InternationalDeployment/ScheduldedDeployment/STG/BOA |
| Grocery | $/InternationalDeployment/ScheduldedDeployment/STG/Grocery |
| Login | $/InternationalDeployment/ScheduldedDeployment/STG/Login |
| LoginUIAssets | $/InternationalDeployment/ScheduldedDeployment/STG/LoginUIAssets |
| UIAssets | $/InternationalDeployment/ScheduldedDeployment/STG/UIAssets |

**Note:** If you want to deploy Other Components (BOA, Login, LoginUIAssets) for which there is no Master Manifest please pick the component manifest file from DEV server for TFS check-in.

**!**