JLT Web Rollout Project
SCM Process Document
Version: 1.1

Document Revision History			
Version	**Date**	**Author**	**Comments**
Draft	12 – Nov – 2013	Reddy, Kannan	Initial Version before KT for discussions
Draft	03 – Feb – 2014	Reddy	Updated version based on the KT discussions and understanding of the source code and repository
1.0	18 – Feb – 2014	Reddy	Incorporated Internal Review comments

Document Distribution Information		
Version	**Date**	**Distributed to**
Draft	12 – Nov – 2013	Jeanine and Justin
1.0	18 – Feb – 2014	Jeanine, Justin, Nigel and Mahesh

Table of Contents

1.0 INTRODUCTION

1.1 Background

Web Rollout project has been developed by a vendor. SiteCore is the main underlying technology platform, used as Content Management System (CMS). The scope of this development project includes the rollout of first 3 sites (JLT Group, Employee Benefits and Specialty) as well. Application Support and Maintenance *(On-going support and maintenance, further site rollouts and minor enhancements & change requests implementation)* of this project will be transitioned into JLT. There is a requirement for Phase 2 development as well. Vendor will continue the Phase 2 development while JLT is taking over the support & maintenance of Phase 1.

It is highly recommended to have a proper code management process and governance in place for this project since both teams (JLT and VENDOR) would require access to the code base for support and development respectively.

1.2 Purpose

The purpose of this document is to identify and describe the Software Configuration Management (SCM) process for the Web Rollout project, especially for the source code and other configurable items (SiteCore components, URLS, etc.) at the application level. This plan describes the required processes, in a simple, straightforward way, to ensure that the inevitable source code and other software configuration items changes occur within an identifiable and controlled environment.

1.3 Scope

The scope of this document is the identification of a top-level software configuration management process for the source repository only (Source Code, Site Templates, Security, Global settings and SiteCore components) at this stage. Environment (Hardware, Operating System, etc.) and project management (Project Plans, Schedules, Issue Tracking and other documents) related items are not covered and out of scope. The document can be extended to cover these items at a later stage.

The scope of both minor enhancements and change requests (alterations to the core functionality of the JLT solution) will be clarified as requirements are gathered for initial changes. This document will be expanded to capture the pre-requisites, boundaries for such work & exit-criteria to ensure continuity of the solution and collaborative development process.

Ongoing discussion between affected parties (JLT, VENDOR and if necessary the hosting provider, Attenda) will be required to evaluate change requirements and identify any impacts to the existing process definition, any subsequent improvements to the process will be captured within this document.

2.0 Source Code Management Process

2.1 Stakeholders

The following table provides the key stakeholders involved in this process and their primary responsibilities:

ID	Stakeholder Group	Organisation	Members	Responsibility
1	Development Team	JLT	Singapore: 1. Reddy 2. Balamurugan	• Application Support & Maintenance • Minor Enhancements / CR Development • Code Merging & Integration • Deployment & Release Management • JLT Dev & Staging Environment Ownership
2	Development Team	Vendor		• Phase 2 Development • Working closely with JLT Development Team to merge the source code and integrate into the main branch
3	QA Team	JLT	Mumbai: 1. Hema 2. Apurva	Regression and Integration Testing
4	Configuration Lead	JLT	Premal (Mumbai)	• Deployment & Release Management • SCM Owner • Regular / Periodic Audits / Reviews
5	Solution Specialists	JLT	Nigel (London)	Infrastructure Ownership (Prod & UAT)

2.2 Source Code Repository

GitHub is used as our source code repository. JLTi has already configured the repository in the cloud. VENDOR can upload the source code into this repository. **We require their Github user account details.** We will then provide the permission to their user account to access our repository *(the roles & permissions to the various branches is available in the sections below).*

JLTi GitHub repository: https://github.com/JLTiPL/WebRollout.git

2.3 Branching & Merging Strategy

Vendor will transfer the latest source code into JLTi's GitHub repository. JLT Development team will baseline it and will create appropriate branches for support and development. A development branch will be created for Phase 2 development with full admin privilege to Vendor.

Branching:

At a very high level, we maintain the following four branches. However, the respective development team can create sub branches to their requirements.

ID	Branch	Description / Purpose	Owner
1	Master	Master Copy, the production version of the application	JLT Development Team
2	Development	A main development branch where all hot fixes, development items should be merged / integrated and tested before promoting into Production. Any sub branches (for Vendor or JLT) should be created from this branch.	JLT Development Team
3	Dev / Support	This branch will be used by JLT Dev team for hot fixes and minor enhancements.	JLT Development Team
4	Phase 2 Dev	This branch will be given to Vendor for the Phase 2 Development.	Vendor

Merging & Conflicts Resolution:

As the Phase 1 support & minor enhancements and phase 2 will go in parallel, both teams (JLT Dev & Vendor) should get the latest code from Development branch to resolve all conflicts with their respective branch before merging the code back to development branch.

Both teams are required to work together and communicate the series of changes being made into the source code **through the release meetings and release documents**.

We expect the teams to perform the impact analysis, and resolve the conflicts identified in the impact analysis before checking in the code into the development branch.

Note: Frequency of the meetings and the level of documents are yet to be discussed and agreed.

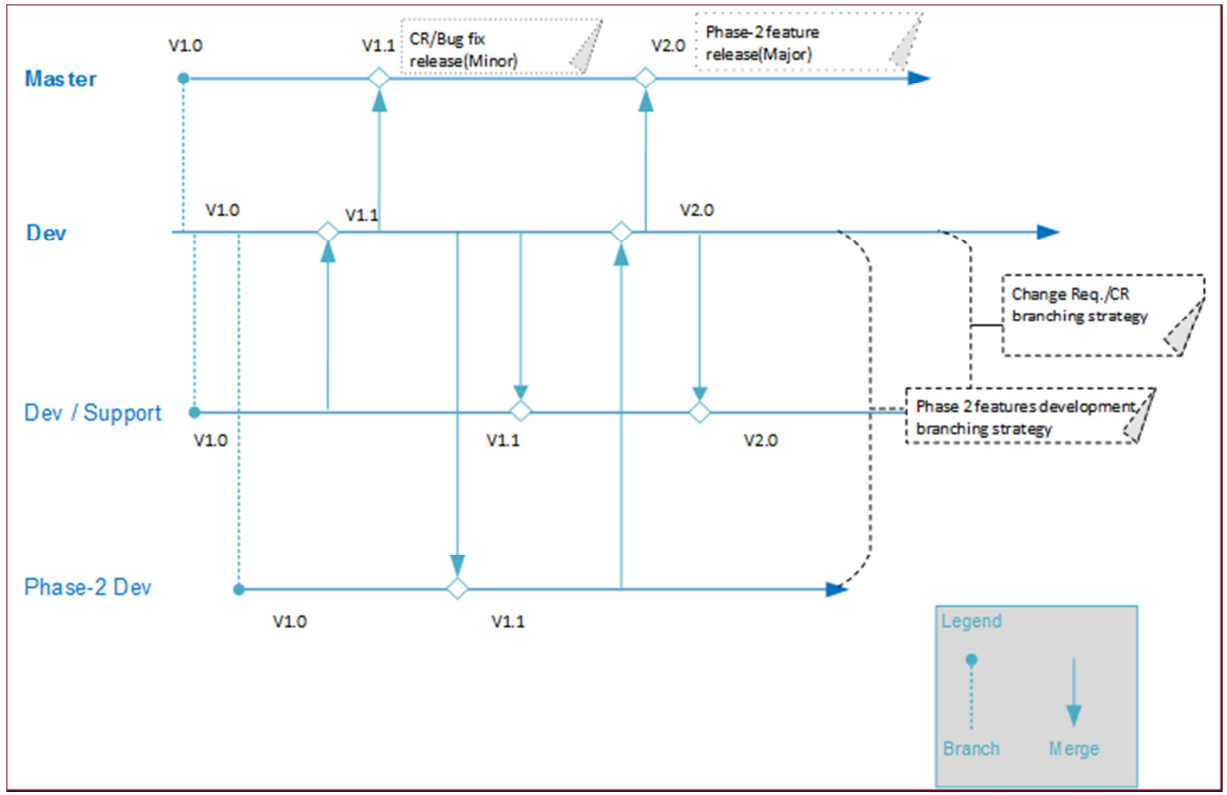
Integration & Release

Integration & Release the application to the UAT and production will be managed by JLT. We adopt the continuous integration process.

Continuous Integration (CI) is to make sure that the check-ins by various teams do not break the build. CI server will be part of staging server and will be used for creation of build for UAT and Production releases.

Each major branch (i.e. like development branch, phase-2 development branch) will have automated build process to maintain continuous integration within the teams.

The following diagram shows the branching & merging strategy:



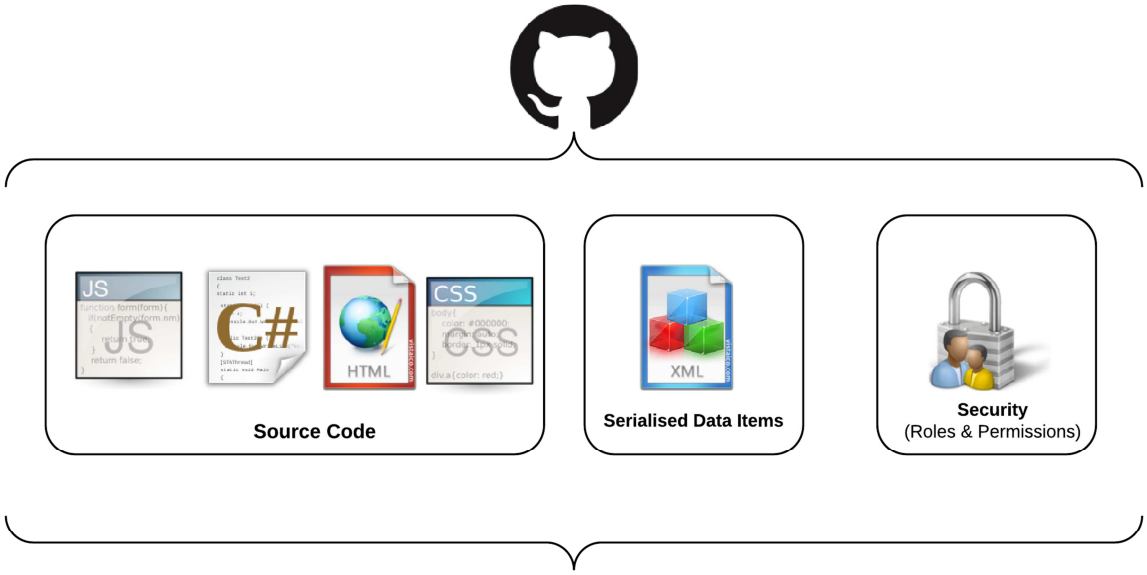
Shipping of database (Sitecore) content between branches (serialisation & Sitecore packages):

While merging changes between branches there will be the necessary step of transferring database changes using serialisation and derserialisation The process for managing these changes should be defined in detail to ensure all necessary changes are successfully shipped between branches and teams.

Note: Changes to setting related items and template/layout definitions are captured as part of the serialization process, however general website content is not included in the serialisation. Should the shipping of test content be required as part of the merge process the teams will have to use alternative means of moving these items e.g. via the Sitecore Package Manager tool.

2.4 Configuration Items

This project uses GitHub as the source repository. Source Code, Serialised Data Items and Security Configurations are the configuration items. Contents *(Individual site content items which are specific to the sites)* will not be part of source control repository.



2.4.1 Source code

Source code contains all source code files independent of Sitecore. It includes C# files, CSS, html and JavaScript files in addition to any .NET solution files and bespoke publish tasks, referenced third party assemblies etc. The source code includes all files required to deploy a full, working solution over a vanilla Sitecore installation.

2.4.2 Serialised Data Items

Serialised files are the Sitecore data items that include the Site templates, page types and all global settings data items. Files are stored in Sitecore's propriety, human readable format and are present in the root of the Presentation.Website project.

Note: Security permissions in Sitecore are applied to data items and stored as part of those items within the Sitecore database, they are therefore included in the serialisation of those items.

2.4.3 Security (Roles and Responsibility)

Roles can be serialised in Sitecore. These serialised roles will be stored in the source repository.

Note: Permissions are serialised as part of the data items. There is a dependency between serialised roles and serialised data items when modifying security. This dependency is currently not managed using automated methods, developers must ensure that both elements are serialised as and when security changes are made.

2.5 Access Control

We follow GitHub's organization repository permission model for this project. There are 4 types of repository access items available in this model: Owner, Admin, Write and Read. The following section provides the proposed permission level for both teams across the various branches:

2.5.1 Development Branch

Entity	Owner	Admin	Write	Read
JLT Dev Team	✓	✓	✓	✓
Vendor	X	X	✓	✓

2.5.2 JLT Dev / Support Branch

Entity	Owner	Admin	Write	Read
JLT Dev Team	✓	✓	✓	✓
Vendor	X	X	X	✓

2.5.3 Phase2 Development branch

Entity	Owner	Admin	Write	Read
JLT Dev Team	X	X	X	✓
Vendor	✓	✓	✓	✓

2.5.4 Master Branch

Entity	Owner	Admin	Write	Read
JLT Interactive	✓	✓	✓	✓
Vendor	X	X	X	X

APPENDIX: GITHUB PERMISSION LEVELS FOR ORGANISATION REPOSITORY

Repository action	Read access teams	Write access teams	Admin access teams	Owners team
Pull (read), push (write), and fork (copy) *all repositories* in the organization				✓
Create and delete any repository in the organization				✓
Create and delete the team's repositories			✓	✓
Change settings for repositories assigned to the admin team			✓	✓
Transfer repositories into, and out of, the organization account			✓	✓
Pull from (read) the team's assigned repositories	✓	✓	✓	✓
Push to (write) the team's assigned repositories		✓	✓	✓
Fork (copy) the team's assigned repositories	✓	✓	✓	✓
Send pull requests from forks of the team's assigned repositories	✓	✓	✓	✓
Merge and close pull requests		✓	✓	✓
Open issues	✓	✓	✓	✓
Close, reopen, and assign issues		✓	✓	✓
Apply labels and milestones		✓	✓	✓
Edit and delete their own comments on commits, pull requests, and issues	✓	✓	✓	✓
Edit and delete anyone's comments on commits, pull requests, and issues		✓	✓	✓
Edit wikis	✓	✓	✓	✓