Document Reference

|  |  |
| --- | --- |
| Company/ Address: | Travis Perkins Lodge Way House  Lodge Way  Harlestone Way Northampton  NN5 7UG |
| Workstream /Project: | Service Transition |
| Document Reference: | Doc Ref |
| Role / Name: | Configuration Manager (Jamie Martin) |
| Address: | Reihill House, Northampton |
| Issue Date: |  |
| Authorisation: |  |
| Revision and Status | 0.1 (Draft) |

<Copyright data goes here>

**NATO UNCLASSIFIED**

Contents

[1 Purpose and Scope of the Document 7](#_Toc508877072)

[1.1 Purpose 7](#_Toc508877073)

[1.2 Scope 7](#_Toc508877074)

[1.3 Maintenance of this Document 7](#_Toc508877075)

[2 Roles and Responsibilities 8](#_Toc508877076)

[2.1 Project Manager 8](#_Toc508877077)

[2.2 Product Owner 8](#_Toc508877078)

[2.3 Development Team 8](#_Toc508877079)

[2.4 Scrum Master 9](#_Toc508877080)

[2.5 Configuration Manager 9](#_Toc508877081)

[2.6 DevOps and Platform Lead 9](#_Toc508877082)

[2.7 Unit Test Lead 10](#_Toc508877083)

[2.8 Environment Manager 10](#_Toc508877084)

[3 Software Configuration Management and Scrum Events 10](#_Toc508877085)

[3.1 Configuration Identification 10](#_Toc508877086)

[3.2 Configuration Status Accounting 10](#_Toc508877087)

[3.3 Configuration Audit 10](#_Toc508877088)

[3.4 Version Control 10](#_Toc508877089)

[3.5 Workspace Management 11](#_Toc508877090)

[3.6 Build Management 11](#_Toc508877091)

[3.7 Release Management 11](#_Toc508877092)

[3.8 Naming Conventions 12](#_Toc508877093)

[3.9 Branching Strategy 12](#_Toc508877094)

[4 Code Management 12](#_Toc508877095)

[4.1 Production 12](#_Toc508877096)

[4.2 Development 12](#_Toc508877097)

[5 Build Environment 13](#_Toc508877098)

[5.1 Build Environments 13](#_Toc508877099)

[5.2 Security and Permissions 13](#_Toc508877100)

[6 Build Process 13](#_Toc508877101)

[6.1 Build Process 13](#_Toc508877102)

[6.2 Build Versioning 13](#_Toc508877103)

[6.3 Build Schedule 13](#_Toc508877104)

[6.4 Build Logs and Tracking 13](#_Toc508877105)

[6.5 Build Scripts 13](#_Toc508877106)

[6.6 Build Maintenance 13](#_Toc508877107)

[7 Build Artefacts 13](#_Toc508877108)

[8 Deployments 13](#_Toc508877109)

[9 Appendix A – Branching and Merging Code 13](#_Toc508877110)

[10 Appendix B – Heading 13](#_Toc508877111)

Index of Figures and Tables

Document Control

Changes History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version Number | Date | Author | Change | Status (Draft / Authorised) |
| 0.1 | *14/03/2018* | *Jamie Martin* | Initial draft | Draft |
|  |  |  |  |  |

Authorised by

|  |  |  |
| --- | --- | --- |
| Name | Role | Date |
|  |  |  |
|  |  |  |

Distribution

|  |  |
| --- | --- |
| Status | Location |
| Electronic master (all other copies are uncontrolled) |  |
| Purchaser and Supplier personnel |  |
| Supplier personnel |  |

References

1. Concise Oxford English Dictionary, 11th edition
2. Managing Successful Projects with PRINCE2, 2009
3. ITIL v3, 2011
4. CMMI v1.3, 2011
5. Scrum, 2017
6. Software Engineering: A Practitioner's Approach, 2009
7. CMA-010 Configuration Management Plan, 0.2, 2017
8. IT Core Release Management Strategy, 0.1, 2017
9. Momentum Defect Management Process, 2.0, 2018
10. TES-010 Momentum Test Strategy, 1.2, 2017
11. TP-ChM-P002 – Change Management Process, 1.2, 2016
12. Momentum Configuration Management Strategy, 0.2, 2018
13. Configuration Management Plan, xx, 2018

Definitions

| Term | Meaning |
| --- | --- |
| Agile Methodology | A software development philosophy, rather than a specific methodology |
| Atlassian | A software vendor that produces the Jira and Confluence applications |
| Capability Maturity Model Integration | An IT Standard and methodology for Software Development that has explicit Configuration Management standards. |
| Change Request (CR) | An authorised process means of capturing change to Configuration Items. |
| Company | A company (one or more) is hosted in a tenant. |
| Confluence | A wiki based knowledge base tool integrated with Jira produced by Atlassian – Allows for live documentation control. |
| Infor | A software vender, supplier of the M3 Solution |
| JFrog | An Artefactory application utilised by Infor for Configuration Control |
| Jira | A configurable issue management tool suited to multiple project applications. Produced by Atlassian |
| Increment | Scrum term, relating to work done as part of a sprint – From increment in relation to version. |
| Momentum | A Travis Perkins Project |
| Prince2 | A Project Standard utilised in the production of this document |
| Product Backlog | In SCRUM the product Backlog consists of tasks or developments that need to be progressed within the project. |
| Service Asset Configuration Management | ITIL Service Asset and Configuration Management aims to maintain information about Configuration Items (CIs) required to deliver an IT service, including their relationships. (3) |
| ServiceNow | Service Management Software |
| Software Configuration Management | is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of Configuration Management (6) |
| Tenant | Tenant is the equivalent cloud term for ‘environment’. |
| TortoiseSVN | A popular GUI version control tool that operates on a subversion database |

Acronyms

|  |  |
| --- | --- |
| Acronym | Meaning |
| API | Application Programming Interface |
| AWS | Amazon Web Services |
| BAU | Business as Usual |
| BSS | British Steam Supplies (a Travis Perkins brand) |
| CAB | Change Acceptance Board |
| CCB | Configuration Change Board |
| ChgMan | Change Management |
| CI | Configuration Item |
| CM | Configuration Management |
| CMDB | Configuration Management Database |
| CMMI | Capability Maturity Model Integration |
| CMS | Configuration Management System |
| Cont.Dev | Continuous Development |
| Cont.Int | Continuous Integration |
| Cont.Test | Continuous Test |
| COTS | Commercial off the Shelf |
| CR | Change Request |
| DAG | Design Authority Group |
| DevOp | Development Operations |
| DM | Defect Management |
| DSL | Definitive Service Library |
| EDP | Electronic Data Processing |
| GPO | Group Policy |
| HP ALM | Hewett Packard Application Lifecycle Management tool – typically utilised for the management for defects in testing |
| ISO | International Organisation for Standardisation |
| IT | Information Technology |
| ITIL | Information Technology Infrastructure Library |
| KBPS | Kilobytes per second |
| KPI | Key Performance Indicators |
| OS | Operating System |
| QA | Quality Assurance |
| QM | Quality Management |
| RACI | Responsible, Accountable, Consulted Informed |
| RM | Release Management |
| SACM | Software Asset and Configuration Management |
| SCM | Software Configuration Management |
| SIT | System Integration Test |
| SOA | Service Orientated Architecture |
| SRS | System Requirements Specifications |
| ST | System Test |
| SVN | Subversion |
| TP | Travis Perkins |
| UAT | User Acceptance Test |
| VM | Virtual Machine |

# Purpose and Scope of the Document

## Purpose

In software engineering, software configuration management (SCM or S/W CM) is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management.

SCM practices include revision control and the establishment of baselines. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many host (6).

This document will provide a plan that will be utilised in the determination of processes for how Software Configuration Management (SCM) will be defined within Travis Perkins; along with the Momentum project. It will form one of two plans for Configuration Management (CM) for Travis Perkins, the other providing a CM plan for Service Asset Configuration Management (SACM).

## Scope

The purpose of this plan is to provide basic guidelines for how source code and software builds will be managed.

This document will cover source code administration, build environment standards and define the process by which new components will be added to builds, and a common understanding of how to manage broken builds.

The SCM Plan will ensure consistent software quality across projects by ensuring that:

* All components are managed in a source control tool
* All components are compiled on an independent and controlled build machine
* Build machine specifications are controlled and documented
* Establish the basis of SCM Processes and Procedures

## Maintenance of this Document

This document will be maintained in accordance with Travis Perkins

# Roles and Responsibilities

The methodology for development operations within Travis Perkins is defined by Scrum, 2017(5). Full definition of the Scrum can be found in Confluence ([link](https://travisperkins.atlassian.net/wiki/spaces/IN/pages/35258826/Scrum+Guide)).

## Change Manager

All deployments into the integration team will be managed through TP-ChM-P002 – Change Management Process, 1.2, 2016 (11). An approved, and fully impacted CR is required for the handover of code from Development phases.

## Integration Lead

The integration lead will be responsible for the adoption and processing of code provided by TP Development Teams or Third-Party suppliers and its conversion into artefact CI’s. The Integration lead will be responsible for

* Ensuring that QA sign off is obtain from the provider – This should include
  + - * A release note detailing the delivery (see 3.7 below)
      * Scripts necessary for installation / deployment are supplied
      * Proof of unit testing being conducted successfully by the provider – in the case of a third-party supplier will consist of a full test report.
* Establishing a baseline of artefacts from source code (tagging code and artefacts)
* Updating the Configuration Status Accounting for all CI’s accepted from development.

## Product Owner

The Product Owner is responsible for maximizing the value of the product resulting from work of the Development Team. How this is done may vary widely across organizations, Scrum Teams, and individuals.

The Product Owner is the sole person responsible for managing the Product Backlog. Product Backlog management includes:

* Clearly expressing Product Backlog items;
* Ordering the items in the Product Backlog to best achieve goals and missions;
* Optimizing the value of the work the Development Team performs;
* Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next; and,
* Ensuring the Development Team understands items in the Product Backlog to the level needed.
* Identifying Backlog items to defined requirements
* Reporting on progress of Product Backlog to the Project Manager
* Management of Change requests to backlog priority

The Product Owner may do the above work or have the Development Team do it. However, the Product Owner remains accountable.

## Development Team

Scrum identifies developers as a team, rather than individuals with specific specialisation. The Development Team consists of professionals who do the work of delivering a potentially releasable Increment of "Done" product at the end of each Sprint. A "Done" increment is required at the Sprint Review. Only members of the Development Team create the Increment (5)

* Turning Product Backlog into deliverable content
* Production of product increment or requirement fulfillment
* Raising Issues for Problem Management
* Conducting unit testing of product increment
* Maintaining Development environment
* Implementation of process and procedure
* Establishing the new baseline of product

## Scrum Master

The Scrum Master is responsible for the day to day running of sprints and assigning development teams to specific pieces of work during the sprint (5)

* Ensuring that goals, scope, and product domain are understood by everyone on the Scrum Team
* Ensuring techniques for effective Product Backlog management;
* Understanding product planning in an empirical environment;
* Ensuring the Product Owner knows how to arrange the Product Backlog to maximize value;
* Understanding and practicing agile methodology and Quality
* Defining criteria for Development Teams
* Escalating issues that impede the Development Team’s progress
* Establishing and ensuring Scrum methodology and Quality in the Development Team
* Serve as point of communication for stakeholders regarding development
* Management of Scrum Events

## Configuration Manager

The Configuration Manager will own this plan and related processes. In specific relation to the SCM Plan they will;

* Consulting with the Scrum Master, Development team and Product Owner to establish Configuration Management activities.
* Establishing Configuration Management activities to the Product owner – These are;
  + - * Configuration Item Identification
      * Configuration Status Accounting provisions
      * Configuration Audits
* Establishing Configuration Management activities to the Development team;
  + - * Version Control
      * Workspace Management
      * Build Management
      * Change Management
      * Release Management
* Review and Approval of Configuration Audits from Product Owner
* Identification of builds for production of build environments
* Approving changes and updates to Scrum practices in regard to Configuration Management standards.

## DevOps and Platform Lead

The DevOps and Platform solution incorporates most of the requirements of a SCM operation in terms of management of code, production of audit, version control of code/artefacts, automated testing and build deployment. For, the purpose of, this document, DevOps is represented by the DevOps lead.

* Ownership of the DevOps Solution
* Ownership of repositories, version control and artefact repositories
* Provisioning of approved CM builds to Environment Manager for deployments
* Responsibility for ensuring automated processes fulfill process requirements
* Implementation of the DevOps process
* Serve as a point of contact between DevOps and the wider project.

## Unit Test Lead

The Unit Test Lead role will be represented within the Development Team (5). The Unit Test Lead will be the individual (s) who either conduct Unit Testing against developed code, or the individual who assures that Unit Testing has been conducted and signs off test reports.

* Confirming the successful verification of unit testing to the Configuration Manager
* Management of automated test script baseline
* Management of automated test tools
* Conducting and planning unit test runs.
* Production of the Unit Test Report

## Environment Manager

The Environment Manager will own all non-development environments, in addition to;

* Booking environments for Testing and Cut over activities
* Ownership of the Deployment process
* Ownership of all non-development environments
* Assuring the process of environment set up is utilized for each environment
* Shakedown Testing of environments
* Raising and ownership of issues tickets and problem tickets relating to environment deployments

# Software Configuration Management activities and Scrum Events

There is no specific definition for the activities of SCM or a Configuration Manager within Scrum. As such, this section of the document will identify the responsibilities and activities associated with SCM as activities and responsibilities that need to be conducted within Scrum Events.

Full definition of the Software Configuration Management activities, such as Configuration Identification, Status Account etc are defined in Configuration Management Plan, xx, 2018 (13) and Momentum Configuration Management Strategy, 0.2, 2018 (12).

## Configuration Identification

Part of each Sprint Planning needs to include the identification of configuration items by the Product Owner, assisted with the Configuration Manager. This means that new CI may be identified on a 2-4-week basis.

Configuration Identification and setting controls will occur as part of the Sprint Planning.

## Configuration Status Accounting

Configuration Status Accounting is the responsibility of the Product Owner. Updates to Status accounts will be driven through the daily Scrum. Changes reported by the development team to statuses of CI will be updated in line with the developers face to face report. It will be an entry requirement into integration, that all Configuration Statuses represent the required status for Integration handover to occur.

## Configuration Audit

Configuration Audits will be undertaken by the Product Owner as part of the Scrum Washup. The Configuration Manager will provide to the Product Owner a set of criteria, and instruction into conducting Configuration Audits. The Configuration Manager will review and approved Product Owner Configuration Audits.

## Version Control

Responsibility for version control will reside with the Scrum Team / Development team. Update and version control, using the appropriate tool, will be made by the development team on a day to day basis. All version control activities must be completed as part of the sprint completion, and the responsibility of the Product Owner is to verify that this has occurred, before the Sprint is committed for the creation of a baseline for testing. This will be a requirement for entry into Integration.

## Workspace Management

Workspace management of dev environments will reside with the Scrum Team / Development team. Each Developer within the team is responsible for management of their workspace and its maintenance – Details for the construction of workspaces are detailed in the Integration Development process, recorded in Confluence ([here](https://travisperkins.atlassian.net/wiki/spaces/IN/pages/36307132/Getting+Started+with+Integration+Development)).

## Build Management

Build management will occur in two phases. During the Sprint Planning the Product Owner will identify all dependencies within, and outside the build. Each developer is then responsible for updates within the build in the production of a new build. Upon completion as part of the Sprint completion, the Product Owner will confirm that all events to produce a new version of the build have been completed, individual developer tests have been completed, and that dependency requirements have been fulfilled or documented. This will be an entry requirement into integration

Each Developer within the Scrum team will be responsible for the production of issue and problem reports, and ensuring that they are assigned, identified and resolved in a timely manner. Issues and problems that are not resolved within a Sprint will be reported as part of the Sprint completion to the Product Owner, as these will need to be added to the Product Backlog – and if carried forward accepted by the wider project.

A build will consist of

* Configuration Items produced or changed
* Scripts necessary for deployment
* Configuration files and associated documents for application
* All installation documentation
* Evidence of QA such as test reports, requirements traceability matrix etc.

Upon completion of a build that is to be considered progression in the Software Delivery Lifecycle the Product owner will be responsible for raising the Change Request (11).

## Release Management

This section details release management responsibilities, inherent in the SCM Plan, it does not define Release Management across Momentum or Travis Perkins (which can be found in IT Core Release Management Strategy, 0.1, 2017 (8).

Responsibility to produce release baselines for progression in the SDLC will be development team (see The Product Owner will assume responsibility for approval of all documentation and the Release Note.

Where a CR for progression is raised, there will be a release note produced that will document the following (at minimum).

* Baseline ID
* Version
* Project
* Release Version
* Creation Date
* Reference to Sprint
* All identified dependencies
* CI Delivered, updated or removed as part of the release.
* Reference to related documentation
* Identification of Functionality / Requirements delivered and updated
* Identification of any Defects resolved
* Identification of all problems or issues resolved
* A list of all outstanding issues, defects and problems.
* Proof of successful development testing and quality assurance.

## Naming Conventions

It is the responsibility for the development team to adhere to the approved naming conventions. This will be reviewed as part of the audit by the product owner.

## Branching Strategy

The development team is responsible for implementing the branching strategy. The Product Owner will establish any branching activity required as part of the Sprint Planning. The Scrum team will use the approved TP Branching and Strategy (reference required).

# Code Management

## Development

Development code will refer to code that has not yet been integrated via the DevOps platform process, and is in the process of being created, modified or updated in response to the Product Backlog. Responsibility for management development code will lie with the Scrum Team. The expectation is that it will be stored and controlled with a version management system, such as Subversion, or within an identified company configuration management system.

This will be managed in line with sections 3.1 to 3.9 above, and as specified in terms of Configuration Management principles (13; 12).

The basis for any development will always be based on the current approved production code, which will be used to provide the basis for all new developments. Where an existing Pre-Production code base exists, that is also based on the same production code, then the development code will need to be based on that pre-production code. Where artefacts share source code, this will necessitate a branch.

An approved Change Request (11) is required for Development code to become Test Code.

## Testing Code

Testing code will refer to a development code baseline, that has been integrated and deployed to internal testing environments, but not yet delivered into production service. Test code is managed through the Integration / DevOps Platform approach (detailed in Confluence [here](https://travisperkins.atlassian.net/wiki/spaces/IN/pages/35258826/Scrum+Guide)). This primarily exists a CI repository with revision control applied from Github. These effectively form individual baselines of code for deployment for Testing and Release

## Production

Production code will refer to the management of code that has been deployed into UAT or Production. It represents the code that is established as ‘Live’. It represents code that has been through the SDLC.

This is managed in the same tool as Test Code but identified separately by the status identifying that it has been released to Production.

A Change Request (11) is necessary to authorise change to be implemented to Production Code.

# Build Environment

## Build Environments

Build environments are provisioned as Virtual Machines (VM) provisioned from the DevOps platform solution. Full details for provisioning a VM for Development are detailed on the Development Confluence page [here](https://travisperkins.atlassian.net/wiki/spaces/IN/pages/36307132/Getting+Started+with+Integration+Development).

Specific details of environment configuration are detailed on the Development Confluence page [here](https://travisperkins.atlassian.net/wiki/spaces/IN/pages/7996312/Environment).

# Build Process

## Build Process

Processes for Integration and Build are detailed on the Confluence pages [here](https://travisperkins.atlassian.net/wiki/spaces/IP/pages/158171590/Processes)

## Build Versioning

Builds are automatically version controlled within the DevOps solution. This in effect creates a continual integration process that constructs a new baseline, identified by labels generated in the repository.

The primary build is the 700 Configuration Management Build (that forms the basis of the 710 and 720 company build). This build is then utilised to refresh identified environment builds throughout the DevOps estate.

Momentum maintains a freeze on environment refresh during ST, SIT, UAT, NFR testing phases.

## Build Logs and Tracking

### Issues

The process for handling development / integration bugs is detailed in Confluence ([here](https://travisperkins.atlassian.net/wiki/spaces/IP/pages/159810963/Integration+Internal+Bug+Management))

### Problems

### Requests for Change

Requests for Change will be managed directly through ServiceNow via the Change Management Process (11).

Details of the Momentum Request for Change Process can be found in Confluence ([here](https://travisperkins.atlassian.net/wiki/spaces/IP/pages/109871649/Momentum+RFC+Process)).

### Defects

Whilst Defects are recorded and Managed in ALM for Momentum (9) – these are converted and tracked as Jira tickets within the TP development and integration domains. This is defined for Integration / Development in Confluence ([here](https://travisperkins.atlassian.net/wiki/spaces/IP/pages/159810963/Integration+Internal+Bug+Management))

Full breakdown of the Defect Management process within TP can be found in Confluence ([here](https://travisperkins.atlassian.net/wiki/spaces/IP/pages/57573614/Defect+Management))

### Build Log Reporting

All activities conducted relating to the DevOps platform, are also automated. These are captured and recorded against each Jira ticket and ServiceNow objects creating an automated history trail of success or failure, at each stage of the automated process.

### Status Reporting

Status reports for each Momentum project can be found within Confluence ([here](https://travisperkins.atlassian.net/wiki/spaces/IR1/pages/158335312/Projects)).

## Build Scripts

Build scripts necessary to conduct operations for build and deployment, will exist as Jenkins job scripts. These will be deliverables from development teams (internal and third party supplied).

These will be maintained within the Jenkins tool set. These will be updated as necessary in line with changes and revisions to the existing code baselines.

## Build Maintenance

Changes to an integration / developed build or build component will only be accepted through the change control process (11). Changes to the Infor M3 build are controlled and defined through CMA-010 Configuration Management Plan, 0.2, 2017 (7).

Changes to code or artefacts within a build will generate a new revision of the 700 Company, and will only be accepted in to service through IT Core Release Management Strategy, 0.1, 2017 (8), with an approved CR for deployment being agreed.

# Deployments

All deployments will be handled by the Environment Manager, and will only progress with his explicit approval that a deployment is ready to proceed, which will be approved against a checklist of required criteria. Deployments will be managed as part of the IT Core Release Management Strategy, 0.1, 2017 (8), and will be only be handed over for use following successful Shakedown testing being conducted against the deployment.

All deployments will be conducted utilising identified builds, drawn from the DevOps solution requested by the Environment Manager. These deployments are automatically deployed into an identified environment tenant, with a pre-defined client and company.

# Appendix A – Branching and Merging Code

# Appendix B – Heading