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

<Copyright data goes here>

**NATO UNCLASSIFIED**

Contents

[1 Purpose and Scope of the Document 6](#_Toc507762521)

[1.1 Introduction 6](#_Toc507762522)

[1.2 Scope 6](#_Toc507762523)

[1.3 Maintenance 6](#_Toc507762524)

[1.4 Maintenance of this Document 6](#_Toc507762525)

[1.5 Assumptions 6](#_Toc507762526)

[1.6 High Level Process 6](#_Toc507762527)

[1.7 Pre-Requisites 7](#_Toc507762528)

[2 Configuration Management Procedure 7](#_Toc507762529)

[2.1 Planning 7](#_Toc507762530)

[2.2 Identification 7](#_Toc507762531)

[2.2.1 Source Code 7](#_Toc507762532)

[2.2.2 Third Party Software 7](#_Toc507762533)

[2.2.3 Software Artefacts 8](#_Toc507762534)

[2.2.4 Configuration Data 8](#_Toc507762535)

[2.2.5 Change Requests 8](#_Toc507762536)

[2.2.6 Integration Points 8](#_Toc507762537)

[2.2.7 Environments 8](#_Toc507762538)

[2.2.8 Tenants and Customers 8](#_Toc507762539)

[2.2.9 Builds (Releases) 9](#_Toc507762540)

[2.2.10 Service Assets 9](#_Toc507762541)

[2.2.11 Software Licenses 9](#_Toc507762542)

[2.2.12 Client Estate 9](#_Toc507762543)

[2.3 Control 9](#_Toc507762544)

[2.4 Status Accounting 9](#_Toc507762545)

[2.5 Verification 10](#_Toc507762546)

[2.6 Audit 10](#_Toc507762547)

[3 Issue, Defect and Change Control Procedures 10](#_Toc507762548)

[3.1 Issues 10](#_Toc507762549)

[3.2 Defects 10](#_Toc507762550)

[3.3 Changes 10](#_Toc507762551)

[4 Tools and Techniques 11](#_Toc507762552)

[4.1 Tools 11](#_Toc507762553)

[4.1.1 Configuration Management System 11](#_Toc507762554)

[5 Recommendations 14](#_Toc507762555)

[5.1 R 14](#_Toc507762556)

[6 Reporting 14](#_Toc507762557)

[6.1 Current Reporting 14](#_Toc507762558)

[6.1.1 Release 14](#_Toc507762559)

[6.1.2 Centralised Reporting and Real Time Reporting 14](#_Toc507762560)

[7 Timing of Configuration Management and Change Control Activities 14](#_Toc507762561)

[7.1 CAB 14](#_Toc507762562)

[7.2 Integration / Development 14](#_Toc507762563)

[7.3 Environments 14](#_Toc507762564)

[7.4 CMDB 15](#_Toc507762565)

[7.5 Licencing 15](#_Toc507762566)

[7.6 Client Build Reports 15](#_Toc507762567)

[7.7 Hardware 15](#_Toc507762568)

[7.8 Stand Up Reviews 15](#_Toc507762569)

[7.9 Issue Reports / Defect Reports 15](#_Toc507762570)

[7.10 Release Review 15](#_Toc507762571)

[8 Roles and Responsibilities 15](#_Toc507762572)

[9 Scales for Priority and Severity 15](#_Toc507762573)

[Annex A. Project Deliverable Configuration Management 15](#_Toc507762574)

[Annex B. Milestone Lifecycle and CI Gates 15](#_Toc507762575)

[10 Appendix B – Heading 16](#_Toc507762576)

Index of Figures and Tables

[Diagram 2-1 – ITIL Based Configuration Management Procedure 7](#_Toc507762577)

[Table 0-1 Prerequisites Check List 16](#_Toc507762578)

Document Control

Changes History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version Number | Date | Author | Change | Status (Draft / Authorised) |
| 0.1 |  |  | 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
4. Agile

Definitions

| Term | Meaning |
| --- | --- |
| Prince2 | A Project Standard utilised in the production of this document |
| Jira |  |

Acronyms

|  |  |
| --- | --- |
| Acronym | Meaning |
| CMS | Configuration Management System |
| CM | Configuration Management |
| RM | Release Management |
| DevOp | Development Operations |
| DM | Defect Management |
| ITIL | Information Technology Infrastructure Library |
| CI | Configuration Item |
| CR | Change Request |
| ChgMan | Change Management |
| CMDB | Configuration Management Database |
|  |  |

# Purpose and Scope of the Document

## Introduction

Configuration Management (v2, v3) Configuration Management (ITILv3): [Service Transition] The Process responsible for maintaining information about Configuration Items required to deliver an IT Service, including their Relationships. This information is managed throughout the Lifecycle of the CI (3).

This document will outline a review of the Configuration Management Strategy, based on Travis Perkins current business set up and the Momentum Project, and make recommendations based on commonly identified business practices, industry standards and best practice.

## Scope

This document will identify specific Configuration Items controlled within the project and will fulfil the criteria of the Prince2 structure of a Configuration Management strategy (2).

As such it will identify;

* How and where the project products are stored
* What storage and retrieval security are in place
* How the products and versions, along with variants are identified
* How changes to products (CI’s) are controlled.
* Where responsibilities for Configuration Management will lie.

## Maintenance

## Maintenance of this Document

## Assumptions

No Assumptions at present.

## High Level Process

|  |  |  |  |
| --- | --- | --- | --- |
| High Level Step | Description | Document Section | Applies to Servers |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Pre-Requisites

# Configuration Management Procedure

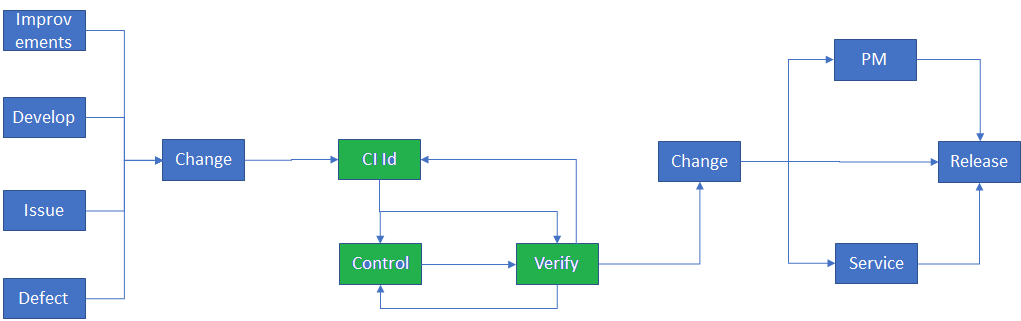


Diagram 2-1 – ITIL Based Configuration Management Procedure

This diagram demonstrates a ITIL based procedure for Configuration Management based on best practice. The section in green identifies the core process by which a change is applied to a CI, and the processes and procedures that feed into this process, and then out back into the project.

## Planning

A configuration management plan does not seem to exist for Momentum (other than for Infor).

## Identification

The following CI have been identified within Momentum that are under configuration management and control to varying extents. Each is detailed in terms of what is currently in use and commentary regarding issues around it.

Annex A will contain a list of potential other CI that could be considered Configuration Items, and subjected to Configuration control.

### Source Code

Source code relates to the lowest level of code that is utilised in the creation of software artefacts. All source code items used by Travis Perkins needs to be maintained under version control within a Configuration Management tool. This can be outside of a Configuration Management System – provided the code can be tracked to the artefacts produced from it.

A baselines of code sets, relating to applications, should be taken each time a change is applied to produce a new artefact.

### Third Party Software

Is software utilised by the project but produced externally by a separate party. Within TP and Momentum this consists of Commercial Off the Shelf software, produced by either subcontracted parties or third-party companies. This should include the Infor product. These

### Software Artefacts

The ‘end product’ of source code is deployable artefacts utilised in, or as, an application. All artefacts should be controlled within a baseline that relates to a common set usage to which they are applied, typically within a baseline of the ‘package’.

For the purposes of this document, a script utilised in an environment, or to build an environment is regarded as an Artefact.

### Configuration Data

Objects and data utilised for the specific set up of an environment, build, customer or tenant is maintained under configuration management control. This should be kept distinct from applications, infrastructure or architecture, as the application of Configuration Data should be what defines one system of similar function from another.

Where configuration data is captured as part of a build, then specific unique builds, rather than generic build to which configuration data is applied, should be identified and controlled.

Best practice recommendation is that Configuration Data is controlled separately from artefacts, infrastructure etc (the average lead time for configuration changes being much lower than those for changes to infrastructure, code or architecture).

### Change Requests

Any request for a change to be applied to the state of a CI will itself be treated as a CI. Change requests will come from four sources.

* Functional Delivery driven from the Design Specification.
* Issues Resolutions discovered in pre-Testing verification and shakedown tests
* Defects identified in formal milestone testing phases (ST, SIT, UAT etc) and those raised in live service.
* Improvements requested from within the project, or through service management requested changes.

### Integration Points

All Integration endpoints between M3 and the Travis Perkins estate, are be treated as a CI, under change control. Code produced related to the production of an integration must be treated as source code and artefacts (as applicable). Each integration point will be managed as a baseline and are under change control.

### Environments

Each of the environments are regard as a configuration item. However, these will be managed specifically by the environment management process. Builds to each environment are controlled using the DevOps solution. Changes to Environments are managed under change control.

### Tenants and Customers

Each tenant and customer are controlled under the environment management process as a CI as part of the Release Strategy. These are tracked on the Momentum EDP & Shakedown Report. Tenants and customers are implemented via the DevOps strategy, which utilises an automated build from controlled sources.

### Builds (Releases)

All builds produced for deployment into an environment are managed under a configuration management process within the Release Management Strategy. These are tracked in the ‘IT Integrated Momentum Deployment Plan’.

Within the wider Travis Perkins the following have been identified as Configuration Items and are under some degree of control.

### Service Assets

Assets used in the client environments of Travis Perkins existing ‘universe’ are under formal configuration management control. This includes hardware infrastructure deployed as part of the existing Travis Perkins solution. These are captured within a CMDB solution using automated discovery and ServiceNow sweeps. However, in several instances the reliability of the information within the CMDB relates only to the asset itself, rather than its specific configuration.

### Software Licenses

Software licence assets utilised within the Travis Perkins estate are monitored via the use of the Snow Application. This reports back on licences deployed within the estate (excluding only frequently rebuilt Virtual Machines).

### Client Estate

Clients deployed in the estate are managed using the SCCM application, based on specific controlled images. GPO’s are rolled out according to regulated change processes establishing a tight control over the estate.

## Control

Control of CI occurs across the Travis Perkins teams to varying degrees and procedures. The lack of any centralised CMS is likely the primary driver behind this. ServiceNow has been implemented for change control and is universal across the project (where even defects to be resolved are authorised by the change management process. To an extent this is mitigated by the utilisation of the DevOps solution, where configuration control is built into the functionality.

Momentum and Travis Perkins CI Control is fairly immature, being limited very specifically to deployable objects only, and not to the means by which they are controlled and managed. An example being the production content of a build is controlled. However the process items such as the spreadsheet producing that build, and the process by which its performed doesn’t seem to be treated as a CI.

## Status Accounting

Limited Configuration Status Accounting across Momentum and Travis Perkins is isolated to specific configuration items. Notably those managed by specific tool sets. Elsewhere the management of status accounting is limited to management via shared spreadsheets.

A notable exception to this exists within Travis Perkins rather than Momentum, where Jira and Confluence have been used as a means of controlling issues and tracking progress.

## Verification

All of the development facing Configuration Management stages utilise a verification stage prior to progress through the lifecycle.

There should be an output from all verification phases that fulfils some kind of approved report that demonstrates what verification was conducted, by whom, using what test scripts (or identified Test baseline).

## Audit

There is no schedule for configuration audits to be conducted. Configuration Audits should be conducted formally at least once per year. At present only the ServiceNow functions of the development facing Configuration Management would provide a reasonable audit trail for best practice industry standards. The utilisation of spreadsheets and the lack of a central shared configuration management system (and processes) make verifying an audit trail at best a laborious effort.

Auditability has been confirmed in relation to the process and procedures in the Integration / DevOps platform teams.

# Issue, Defect and Change Control Procedures

## Issues

Issues for the purposes of the Configuration Management Strategy relates to discovery of ‘defects’ within the pre-Test states – Such as discovery in unit testing or development. These should be formally reported, triaged, tracked and resolved through a formal process – Ultimately, effecting a change control procedure.

The methodology varies within Momentum, as no fixed approach or tool exists. These are unified in terms of deployment by being captured as change requests. Within the Travis Perkins projects the tendency is that Jira is utilised to manage issues. Within Momentum, these are managed through several different processes and tools, including ALM and Jira (Integration).

## Defects

For the purposes of this document, a defect refers to an issue, discovered in formal verification phases such as System Test. Defects are logged and managed via HP ALM and are within the responsibility of the Test Manager. Defects will however result in changes to CI and be governed by a Change Request before being deployed. The current situation results in HP ALM defects from Momentum being converted into Jira requests for tracking within Travis Perkins

## Changes

Changes are managed formally within Travis Perkins. Processes are in definition to tailor Momentum into existing processes by the Change Management team. Change management form the basis by which deployed configuration items are updated. The change process however only covers deployment of changes into estates, not the authorisation of changes. This does mean that Configuration Items can be updated outside of change management auspice.

# Tools and Techniques

## Tools

### Configuration Management System

There is at present no centralised configuration management system in the development facing Travis Perkins or Momentum Project. Systems exist for the control of some CI and are defined in the following paragraphs.

The primary role of a CMS is provided by the use of SVN and GitHub as a means of controlling source code control; meaning that configuration control is managed only in terms of software configuration management.

#### Heritage Systems

There is no CMS as such defined in use for Heritage systems. Applications are not managed in a controlled manner that would typically be described under CM best practices in the classic sense and mapping of relationships between systems and shared across brands are not clearly identified or separated.

#### DevOps Platform / Integration

Extensive and through control of CI is driven by the utilisation of the DevOps solution, its automated testing and deployment mechanisms. This team utilises a suit of applications for version and CI control across from the point of handover to release.

These are

##### Jenkins

For the scheduling of builds and automated deployments and functions.

##### GIT

For version control and identification of code.

##### Subversion

For source code control.

##### Artifactory and Sonatype Nexus

For the control of binary files.

##### Terraform

Deploy code as architecture

##### Ansible

For automation

##### Ubuntu

Utilised as a Linux Operating System

##### AWS

Used for VPC, EC2, ELB, Route 53 and S3)

##### Docker

Used for AWS ECS.

##### WSO2

Used for Integration, Microservices, Micro-Interactions and API Management.

##### Talend

Data Management and extraction, transformation and loading large data batch transactions.

##### SOA

Data Management and extraction, transformation and loading (non-large data transactions)

##### Jira

Used for Story Management (4) of requirement-controlled details, tracking issues and communication of issues and issue resolution within the change management protocol.

##### Confluence

Utilised for knowledge management. Contains details of process, procedures, check gates, reporting and knowledge relating to the integration and DevOps platform approach.

##### SFTP Server

A secure file transfer protocol server for the distribution and sharing of software code and artefacts.

##### MFT Go Anywhere

A secure file transfer protocol used for the distribution and sharing of code and artefacts in conjunction with SFTP Server.

##### Monitoring Tools

The Infrastructure / DevOps platform team monitor environments and service operational ability with Dynatrace, StackDriver, ElasticSearch, Logtash and Kibana.

##### Scripting Tools

The infrastructure / DevOps platform team use Python, Ruby and Bash for the production of scripts.

#### Infor

The Cis for the M3 Solution are controlled within JFrog. Which provides artefact management from source code, that fulfils some aspects of configuration control. The author is unfamiliar with this tool, so further investigation is required.

#### Change Management

Change Management across the TP system is managed through the use of ServiceNow, and operates under a single set of controlled processes defined by the change management team. ServiceNow has been identified as the tool to be utilised by Momentum under the same change management teams process and procedures,

#### Issue Management

Issue management within TP is handled through Jira with resolutions back into CI being merged through the authority of the change management process. Defects raised in HP ALM on the team are raised as Jira and tracked separately. Confluence is also in use and has been used to create knowledge bases within Integration and other areas.

#### Service Asset and Configuration Management

Service Assets are managed utilising a combination of tools.

##### Client Management

Clients are managed utilising remote build protocols for new assets, and management of Group Policy Updates for supporting existing operating systems in regard to patching vulnerabilities, resolving OS known issues etc. The estate is managed by Microsoft System Centre Configuration Manager. This is the defined TP means of deploying builds into the client estate.

##### Licence Management

Licence management is captured with the utilisation of Snow a licence management reporting application that is part of the core client builds. This reports back on all assets to which it has been deployed – identifying licenced software deployed on the client device. The exception to this remains with regularly rebuilt virtual environments, where in the daily rebuild would distort reporting.

##### Hardware Asset Management

##### Configuration Management Database (CMDB)

The CMDB operating within Travis Perkins is based in the ServiceNow application. This system operates on the basis of weekly Discovery sweeps and extract imports from the estate. The degree to which this captures the environment accurately is questionable. Third party assets aren’t fully captured, periodic sweeps mean only 60% approximately of laptops are captured, and the software does not have the capacity to analyse some historical assets (due to their age) and ‘third party’ devices.

As such its problematic in producing a reliable and accurate database of the operating estate.

#### Release Management

Release management tooling is contained within the integrated dev ops solution. A duplicate Jenkins / Git is available to the Momentum Release Management and Environment team in order to produce environments, to create an independence from the Integration / DevOps platform team.

# Recommendations

## R

# Reporting

## Current Reporting

### Heritage

All Heritage reports are produced and controlled through the use of Jira.

### Release

The following reports are generated by Release

#### IT Integrated Momentum Release Deployment Plan

[IT Integrated Momentum Release Deployment Plan](https://docs.google.com/spreadsheets/d/1N5ViaKhlo4iU6PP2HRzrrFm7QHKUkCWwae7cJY_b2rY/edit?usp=sharing) - Sent weekly from the governance meeting.

#### Momentum R1 – EDP and Shakedown Report – System Test

[Momentum R1 - EDP & Shakedown Report - \* System Test \*](https://docs.google.com/spreadsheets/d/1QzQN0_mq0b6T4LuFC1kt477Wfj9Nvow-foonAvgc3V4/edit?usp=sharing)-Sent weekly from the governance meeting.

#### Current Integration System Test Environment Shakedown and Deployment Status

[Current Integration System Test Environment Shakedown and deployments Status](https://docs.google.com/presentation/d/1e2TRvrIu2v-65fogNBbvkHljhsRu83s75hnd7jV82_Y/edit?usp=sharing)-Sent weekly Friday's.

### Centralised Reporting and Real Time Reporting

The utilisation of Jira and Confluence within Travis Perkins allows for centralised reporting, generation of dashboard, and real time reporting from Jira across the Integration and DevOps platform teams – This includes large screen environment status reports displayed in the team area.

# Timing of Configuration Management and Change Control Activities

## CAB

The CAB meets every Thursday, with a deadline submission of close of business Wednesdays. CAB only discusses changes which have not been approved by impactors, and to provide a formal centralised point of approval. An agenda is generated for this meeting.

## Integration / Development

## Environments

The IT Release and Environments Governance board meets at 1.30pm every Wednesday to centralise communication regarding the current status and issues relating to Releases and Environments.

There are scheduled customer refreshes of builds conducted every Monday, with a provision for additional refreshes on Wednesdays.

## CMDB

The CMDB runs a discover sweep every 5 days, supported by imports of data provided from third parties and exports from systems that cannot support Discovery.

## Licencing

## Client Build Reports

## Hardware

## Stand Up Reviews

There is a daily Configuration Management stand up meeting scheduled for 9.30 am every day to discuss issues relating to Momentum and Configuration Management hosted by the Infor Configuration Manager.

## Issue Reports / Defect Reports

## Release Review

The IT Release and Environments Governance board meets at 1.30pm every Wednesday to centralise communication regarding the status of Releases and Environments.

The Momentum Release lead also hosts a daily stand up meeting covering Environment shakedown.

# Roles and Responsibilities

# Scales for Priority and Severity

1. Project Deliverable Configuration Management

All projects operate on a basis of identified milestones. At key milestones specific deliverables, typically documented processes must be produced. As these deliverables affect the completion of a milestone and are inform decisions and direction throughout the lifecycle of the project; it is recommended that they be kept under control as Configuration Items, subject to formal change control.

As an example, if we consider a design document. This effectively is produced from approved requirements, to produce a design that will determine the production of artefacts that themselves form part of a service. It becomes essential then that requested changes to a requirement, will affect a change to the design document, that in turn will result in potential changes to artefacts and service.

By treating project deliverables as CI’s, it is possible to identify and effectively communicate changes in revision to a deliverable to a document, and single out other affected objects that need to be considered potentially affected by those changes – in the same way that Configuration Management would utilise in singling out affected code and systems within a software change. As noted, changes to deliverables can themselves require review of traditional software CIs (the introduction of a change or new function, will affect requirements, the design documentation, as well as software artefacts and new code – as well as potentially requiring changes to existing integration end points). As such, it is advisable that milestone deliverables are kept within the Configuration Management System, as CI’s with strict change control protocols.

By treating these deliverables as CI it is possible to create a system where by the success of each milestone can be quantified in terms of objective metrics. As a project matures in its lifecycle, the more important these relationships become, as the range of relationships expand, and failure to capture changes in a formal manner, creates a greater risk.

Something as simple as a single requirement change, will result in impacts throughout the project lifecycle, at every level (The design, development code, integration, test cases, plans and scenarios are all driven by requirements – and subject to impact from changes to a requirement).

Additionally, in controlling project deliverables a baseline of these key objects becomes available to the project based on a single source of reference. This allows the project to determine the degree of success at each milestone, and determine activities necessary to rectify issues, as well as to establish points at which future deliverables should be started.

The primary means of achieving this is defined as a ‘release gate’, and is detailed in Annex B below.

* 1. Document Management

Document management -

1. Milestone Lifecycle and CI Gates

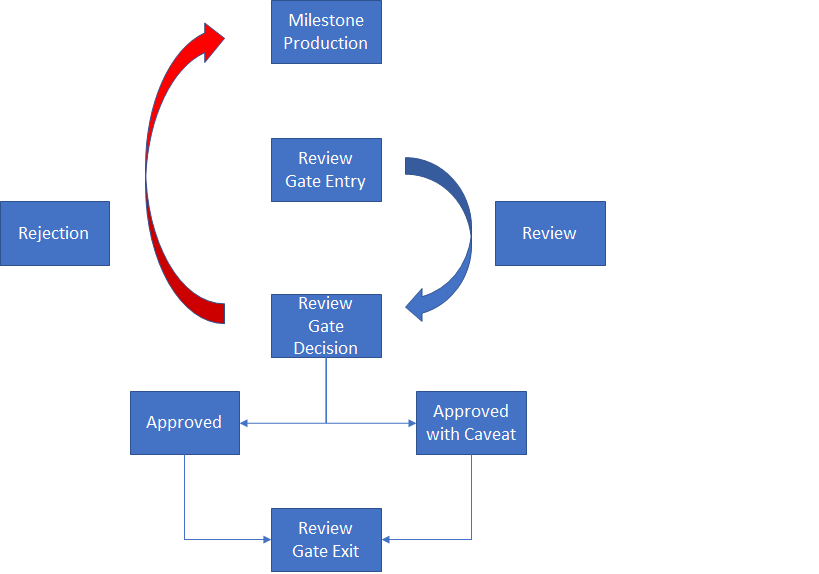


Table 5-1 Review Gate Process

# Appendix B – Heading