|  |  |
| --- | --- |
| 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 | 0.2 (Draft) |

<Copyright data goes here>

**NATO UNCLASSIFIED**

Contents

[1 Purpose and Scope of the Document 9](#_Toc508205036)

[1.1 Introduction 9](#_Toc508205037)

[1.2 Scope 9](#_Toc508205038)

[1.3 Maintenance of this Document 9](#_Toc508205039)

[1.4 Assumptions 9](#_Toc508205040)

[1.5 High Level Process 10](#_Toc508205041)

[2 Configuration Management Procedure 10](#_Toc508205042)

[2.1 Planning 11](#_Toc508205043)

[2.2 Identification 11](#_Toc508205044)

[2.2.1 Source Code 11](#_Toc508205045)

[2.2.2 Third Party Software 11](#_Toc508205046)

[2.2.3 Software Artefacts 11](#_Toc508205047)

[2.2.4 Configuration Data 12](#_Toc508205048)

[2.2.5 Change Requests 12](#_Toc508205049)

[2.2.6 Integration Points 12](#_Toc508205050)

[2.2.7 Environments 12](#_Toc508205051)

[2.2.8 Tenants and Customers 12](#_Toc508205052)

[2.2.9 Builds (Releases) 13](#_Toc508205053)

[2.2.10 Service Assets 13](#_Toc508205054)

[2.2.11 Software Licenses 13](#_Toc508205055)

[2.2.12 Client Estate 13](#_Toc508205056)

[2.3 Control 13](#_Toc508205057)

[2.4 Status Accounting 13](#_Toc508205058)

[2.5 Verification 14](#_Toc508205059)

[2.6 Audit 14](#_Toc508205060)

[2.6.1 Release Configuration Audits 14](#_Toc508205061)

[2.6.2 CM Repositories Audit 15](#_Toc508205062)

[2.6.3 Change Configuration Audit 15](#_Toc508205063)

[2.6.4 General CM Audit Questions 15](#_Toc508205064)

[3 Infor M3 Configuration Management (Momentum Specific) 16](#_Toc508205065)

[3.1 Configuration Management Plan and Documentation 16](#_Toc508205066)

[3.1.1 Processes 16](#_Toc508205067)

[3.1.2 Procedure 16](#_Toc508205068)

[3.1.3 Roles and Responsibility 16](#_Toc508205069)

[3.1.4 Change Management 17](#_Toc508205070)

[3.1.5 Requirements Management and Requirements Traceability Matrix 18](#_Toc508205071)

[3.1.6 Configuration Audits 18](#_Toc508205072)

[3.2 Standards and Methodologies 18](#_Toc508205073)

[3.3 Infor M3 Configuration Items 18](#_Toc508205074)

[3.3.1 Infor Source Base code 18](#_Toc508205075)

[3.3.2 Infor Configuration and Parameters 18](#_Toc508205076)

[3.3.3 Infor Datasets 19](#_Toc508205077)

[3.4 Tools 19](#_Toc508205078)

[3.4.1 Configuration Management System (CMS) 19](#_Toc508205079)

[3.4.2 Infor CMDB 19](#_Toc508205080)

[3.4.3 Automation tools 19](#_Toc508205081)

[3.4.4 Change Control of Configuration Items 19](#_Toc508205082)

[3.5 Lifecycle Diagram (Change to Release) 19](#_Toc508205083)

[3.5.1 RACI for Change to Release Lifecycle (3.5) 19](#_Toc508205084)

[3.5.2 Key Deliverables / Evidence for Change to Release Lifecycle 20](#_Toc508205085)

[3.6 Verification 20](#_Toc508205086)

[3.6.1 Unit Testing and Integration Testing (Pre-Release) 20](#_Toc508205087)

[3.6.2 Environments and provisioning 20](#_Toc508205088)

[3.7 Software Configuration Management 20](#_Toc508205089)

[3.7.1 Point of interface with Momentum / Travis Perkins 20](#_Toc508205090)

[3.7.2 Criteria for Release to Momentum / Travis Perkins (System Test and System Integration Test). 20](#_Toc508205091)

[3.8 Service Asset Configuration Management 20](#_Toc508205092)

[3.8.1 Interface with Travis Perkins SACM 20](#_Toc508205093)

[3.8.2 BAU Support into Travis Perkins and Service Desk 21](#_Toc508205094)

[3.9 Priority and Severity Definitions 21](#_Toc508205095)

[3.9.1 Priority 21](#_Toc508205096)

[3.9.2 Severity 21](#_Toc508205097)

[3.10 Reporting 21](#_Toc508205098)

[3.10.1 Configuration Status Reporting 21](#_Toc508205099)

[3.10.2 Configuration Management Reporting (Audit) 21](#_Toc508205100)

[3.10.3 Change 22](#_Toc508205101)

[3.10.4 Problem Management 22](#_Toc508205102)

[3.10.5 Testing 22](#_Toc508205103)

[3.10.6 Releases 22](#_Toc508205104)

[4 Issue, Defect and Change Control Procedures 23](#_Toc508205105)

[4.1 Issues 23](#_Toc508205106)

[4.2 Defects 23](#_Toc508205107)

[4.3 Changes 23](#_Toc508205108)

[5 Tools and Techniques 23](#_Toc508205109)

[5.1 Tools 23](#_Toc508205110)

[5.1.1 Configuration Management System 23](#_Toc508205111)

[6 Reporting Presently 26](#_Toc508205112)

[6.1 Reported and Baseline Requirement for Reporting 27](#_Toc508205113)

[6.1.1 Heritage and Development 27](#_Toc508205114)

[6.1.2 Infor (CM) 27](#_Toc508205115)

[6.1.3 Integration 27](#_Toc508205116)

[6.1.4 Testing 27](#_Toc508205117)

[6.1.5 Release 27](#_Toc508205118)

[6.1.6 Deployment and Environments 27](#_Toc508205119)

[6.1.7 Software Asset 27](#_Toc508205120)

[6.1.8 Client Management 28](#_Toc508205121)

[6.1.9 Hardware Management 28](#_Toc508205122)

[6.1.10 CMDB 28](#_Toc508205123)

[6.2 Best Practice Recommended Reporting (KPI’s) 28](#_Toc508205124)

[6.2.1 Change Management 28](#_Toc508205125)

[6.2.2 Testing 28](#_Toc508205126)

[6.2.3 Problem Management 28](#_Toc508205127)

[6.2.4 Configuration Management Status Reporting. 28](#_Toc508205128)

[6.2.5 Release 29](#_Toc508205129)

[6.2.6 Configuration Management and Service Asset Configuration Management 29](#_Toc508205130)

[6.3 Centralised Reporting and Real Time Reporting 29](#_Toc508205131)

[7 Timing of Configuration Management and Change Control Activities 30](#_Toc508205132)

[7.1 CAB 30](#_Toc508205133)

[7.2 Integration and Platform DevOPs 30](#_Toc508205134)

[7.3 Environments 30](#_Toc508205135)

[7.4 CMDB 30](#_Toc508205136)

[7.5 Licencing 30](#_Toc508205137)

[7.6 Client Build Reports 30](#_Toc508205138)

[7.7 Hardware 30](#_Toc508205139)

[7.8 Stand Up Reviews 30](#_Toc508205140)

[7.9 Issue Reports / Defect Reports 30](#_Toc508205141)

[7.10 Release Review 31](#_Toc508205142)

[8 Roles and Responsibilities 31](#_Toc508205143)

[9 Scales for Priority and Severity 31](#_Toc508205144)

[9.1 Priority 31](#_Toc508205145)

[9.1.1 Urgency 31](#_Toc508205146)

[9.1.2 Impact 31](#_Toc508205147)

[9.2 ITIL Priority Matrix 32](#_Toc508205148)

[9.3 Severity 32](#_Toc508205149)

[9.3.1 Critical 32](#_Toc508205150)

[9.3.2 Major 33](#_Toc508205151)

[9.3.3 Medium 33](#_Toc508205152)

[9.3.4 Minor 33](#_Toc508205153)

[9.3.5 Cosmetic 33](#_Toc508205154)

[Annex A. Additional Configuration Items 34](#_Toc508205155)

[Annex B. Project Deliverable Configuration Management 34](#_Toc508205156)

[Annex C. Milestone Lifecycle and CI Gates 35](#_Toc508205157)

[Annex D. Configuration Types and Subtypes. 36](#_Toc508205158)

[Annex E. Configuration Authorities and Controls 36](#_Toc508205159)

[Annex F. CI Relationships 36](#_Toc508205160)

[Annex G. Configuration Item Records. 37](#_Toc508205161)

[10 Appendix B – Quality Assurance 38](#_Toc508205162)

[Annex H. Service Support 38](#_Toc508205163)

Index of Figures and Tables

[Diagram 2-1 – ITIL Based Configuration Management Procedure 10](#_Toc508205164)

[Table 9.2-1 ITIL Priority Matrix 32](#_Toc508205165)

[Table 9.2-2 Priority Code Target and Resolution Times 32](#_Toc508205166)

[Table 6-1 Review Gate Process 35](#_Toc508205167)

Document Control

Changes History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version Number | Date | Author | Change | Status (Draft / Authorised) |
| 0.1 | *26/02/2018* | *Jamie Martin* | Initial draft | Draft |
| 0.2 | 05/03/2018 | Jamie Martin | Expanded out to include reports and expected reports, document library and control gates | 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. Agile Methodology
5. Software Engineering: A Practitioner's Approach, 2009
6. CMA-010 Configuration Management Plan, 0.2, 2017

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 |
| Momentum | A Travis Perkins Project |
| Prince2 | A Project Standard utilised in the production of this document |
| 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 (5) |
| 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 |
| 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 |

# Purpose and Scope of the Document

## Introduction

Configuration Management is the methodology responsible for maintaining information about Configuration Items required to deliver or maintain an IT Service, including their relationships and their transformation via controlled change. This information is managed throughout the Lifecycle of the CI (3). This is loosely defined in terms of Software Configuration Management (SCM) and Service Asset Configuration Management (SACM).

SCM more typically refers to the production and management of code and ‘set up data’ and relates to the development and production of changes to applications; where as SACM is more concerned with the operational functionality of existing systems.

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). It will serve to present the Configuration Management disciplines in use within Travis Perkins. It will identify how the process is currently operating and different elements of Configuration Management.

This document will utilise Prince2 Methodology (2) as its template but will explicitly refer to both ITIL and Agile best practices where relevant or useful. Where reference is made outside of these two principle sources, specific references will be made.

It will draw distinction between different aspects of Configuration Management (CM. SCM. SACM), and touch on Release, Deployment and Change Management, as key touch areas for effective configuration management.

The appendices will present Project Configuration Management, Document Management and present the concept of review gates as a means of project control points within a project or delivery lifecycle

## Maintenance of this Document

This document will be reviewed in line with changes made in process and procedure relation to SACM and SCM, within the project. It will be reviewed automatically on an annual basis in relation to Configuration Management Audits and in line with Quality Assurance process and procedures, within TP and Momentum, relating to the production and maintenance of documents. For the purposes of review and audit this document will be treated as a process document.

## Assumptions

This document assumes some familiarity with Configuration Management disciplines, beyond the specific definition of ITIL Service Asset Configuration Management.

## High Level Process

|  |  |  |
| --- | --- | --- |
| High Level Step | Description | Minimum Evidence – Deliverable of Completion |
| Requirements | Definition of Requirements for Project | Approved Requirements Specification |
| Design | Design of solution based on requirements | Approved Design Specification / System Requirements Specification. |
| Development | Production of solution application to fulfil requirements | Production of Unit Test Reports  Release notes  Development baseline |
| Integration | Production of application end points and integration points utilising development baseline | Integration Test Report |
| System Test | Non-Integrated Testing of application set against test cases / scenarios based on Functional Requirements | System Test Exit Report |
| System Integration Test | Fully integrated testing of application set against test cases / scenarios based on functional requirements | System Integration Exit Report |
| Deployment | Deployment of Application set outside of non-production environments (Pre-Prod, Production) | Approved Deployment strategy and Deployment Smoke Test report |
| User Acceptance Testing | User acceptance testing in which designated users implement test scenarios to replicate real world use of the system. Testing of Non-Functional Requirements and performance requirements | User Acceptance Test Exit Report  Non-Functional Requirement Test Exit Report  Performance Test Exit Report |
| Rehearsal 1-3 | Rehearsal Phases 1 to 3 to verify Go/No Go for Cut over to live service | Approved Rehearsal 3 Completion Report |
| Cutover | Go live for BSS | Approved Service Management Strategy |

# Configuration Management Procedure

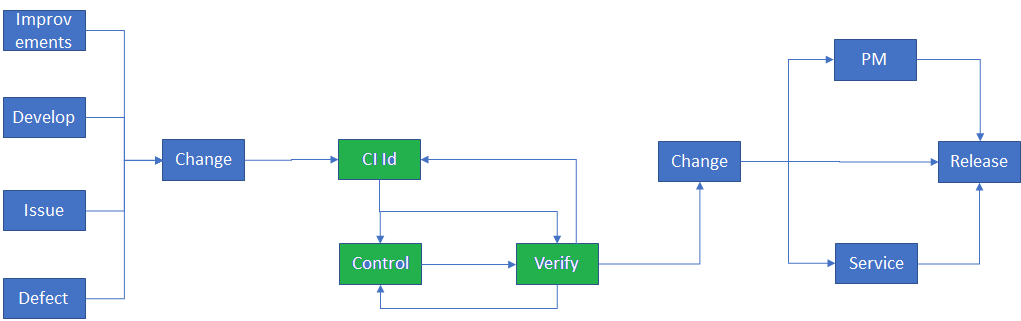


Diagram 2-1 – ITIL Based Configuration Management Procedure

This diagram demonstrates a ITIL based procedure for Configuration Management based on best practice (3). 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

All areas of configuration management will have an approved Configuration Management Plan document that identifies (2)

* How and where the project products are stored (Repositories / CMS)
* What storage, backup 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.

Annex D to Annex G contains best practice identification relating to the production and management of CI’s that should appear in any Configuration Management plan.

## Identification

The following core CI have been identified within Momentum / Travis Perkins that are under configuration management and control. 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 should 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 (CMS) – provided the code can be tracked to the artefacts produced from it.

A baselines of code sets, relating to applications, will 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.

Use of third party software should be controlled in both development and operation environments with a view to rationalisation of applications which produce similar or an identical purpose. Additionally, management of licences and support contracts relating to third party software will need to be captured in relation to SACM control. Documents relating to use of third party software should also be considered CI’s where they have a key training function.

### 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.

Artefacts may also be drawn from the contents of third party (COTS) software.

### 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 strategy. Builds to each environment are controlled using the DevOps solution to ensure replication of build.

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.

A customer / tenant shall consist of the architecture and the configuration necessary to define it from other builds. Configuration data should not be held within the tenant, customer or environment.

### 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 limited 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 are generally outside of CI control.

## 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. There are risks inherent in the use of spreadsheets in that they lack an audit trail and produce a considerable burden on the owner in terms of maintenance and distribution. ­­

The lack of a Configuration Management System makes configuration audit status accounting problematic. This could be resolved with the application of Jira within Momentum (as this is already in use within Travis Perkins). Whilst not strictly a configuration management system, without further Atlassian products, it is reasonable to utilise Jira as a means of presenting configuration audit status accounting across any lifecycle driven aspect of a project and tie that into the use of a revision control tool such as an instance of TortoiseSVN to ‘mimic’ a CMS.

## Verification

All Configuration Management processes will utilise a verification stage prior to progress through the lifecycle.

There should be an output from all verification phases that fulfils approved report that demonstrates what verification was conducted, by whom, using what test scripts (or identified Test baseline). For example, a Unit report, Integration Test, System Test Report, SIT Report, Shakedown test report etc. These should be official documents related to an owning plan and process, that identify what was verified, the methodology, defects / issues raised, tests carried out, scripts run etc.

Verification without evidence, should not be considered approved.

## Audit

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.

Configuration audits should be conducted at least once per year (ideally every six months, driven by the QA Manager / Lead). Audit should establish;

### Release Configuration Audits

* Does the release documentation clearly define the scope of release, including the CRs that should be incorporated?
* Are all dependencies / bugs been documented?
* Is there adequate documentation that identifies the environment to recreate the release?
* Is there adequate documentation that specifies the components and the versions of those components that comprise the release?
* Are all the items of the release in sync with each other?
* Has the release been created using the right versions of the right components from the right repositories?

### CM Repositories Audit

* Are the repositories defined as per the CM plan?
* Have the items been put into the correct repositories?
* Are the required items present in the repositories?
* Have the items been named according to the conventions specified in the CM plan?
* Are the version numbers of items according to the CM plan?
* Have all items been put in the repositories according to the events defined in the CM plan?
* Do the items have required ‘documentation’ to identify the item, version and the change history?

### Change[[1]](#footnote-1) Configuration Audit

* Have all the required CR[[2]](#footnote-2)s been closed?
* Do CRs identify all items to be changed?
* Have all items identified for change in the CR been changed?
* Is it possible to isolate the changes between any two versions of the items?
* Is the documentation in the items adequate to trace the changes back to the appropriate CR?
* Is there adequate means to go back to a previous means?
* Are there any changes between two versions of an item that are not traceable to an approved CR?
* Are the CRs documented before making changes in items?
* Are CRs analysed, evaluated and approved prior to making of the change in items?

### General CM Audit Questions

* Are appropriate back-ups of repositories been taken?
* Has the recovery from back-up been tested?
* Are there any unauthorized components available in the working directories of the team members?
* Is there adequate security/authorization to ensure that only authorized team members perform the check-in and check-out?

# Infor M3 Configuration Management (Momentum Specific)

## Configuration Management Plan and Documentation

A configuration management plan exists for the Infor M3 Solution (CMA-010 Configuration Management Plan. This is currently version 0.2 Draft. This document should be in an approved state; and approved and accepted by Travis Perkins.

The CM strategy will need to be compatible with existing Travis Perkins operations such as SACM, Change Management, Support and Service Management and this section of the document will approach questions where that needs to be considered.

Particular attention needs to be focused on how M3 will be supported.

### Processes

Documented, approved process for Configuration Management and changes

Reference in CMA-010 Configuration Management Plan, 0.2, 2017 (section 3.3.3). This should be referenced.

Reference to the Data Migration Process in CMA-010 Configuration Management Plan, 0.2, 2017 (section 3.1.3)

Reference to the Process for Change / CCB in CMA-010 Configuration Management Plan, 0.2, 2017 (section 3.3.1 and 4.2)

Reference to Processes for Branching, Roll Back and Update in CMA-010 Configuration Management Plan, 0.2, 2017 (section 3.3.3)

### Procedure

Documented and approved procedure Configuration Management and Configuration Change Procedures – These need to dovetail into existing TP procedures as much as possible – except where there is clear separation between the functionality of M3 in regard to Infor and TP BAU procedure.

### Roles and Responsibility

Roles and Responsibilities are outlined in CMA-010 Configuration Management Plan, 0.2, 2017 (6 section 3.3). These are;

#### Configuration Change Control Board

This serves as a Change Control Board for CI affected or potentially affected by CR’s under Infor control. Identification of specific roles to be invited and frequency needs to be established along with agenda and minutes of for each meeting. There should be CM representation from Travis Perkins on each of these boards.

A Forward Schedule of Change should serve as the agenda driver for this meeting (3).

#### Project Managers

* PM’s Chair the CCCB.
* Project Managers own the responsibility for CM activities,
* identification of CI’s,
* communication of CM activities,
* participation in the CCCB and re-baselining of any changes to existing CI baselines.

#### Configuration Manager (Key Role)

Assigned by the Design Authority Group (DAG). The Configuration Manager is responsible for

* Overall management of the Configuration Management Processes.
* Identification of CIs
* Providing configuration standards and templates to the team
* Providing any required configuration training
* Approving or Rejecting FastTrack Change Requests
* Ensuring the process for update, roll back and branching of configurations is followed
* Providing audits of the configuration data and communicating the results to interested parties.

#### Solution Architect

All identified CIs are assigned to a Solution Architect.  The assigned Solution Architect is responsible for:

* Designating a focus group to develop major change requests
* Ensure all major change requests comply with organizational templates and standards prior to the CCCB
* Identification of CIs

#### Workstream Leads / Technical Consultants / Business Consultants

Each CI will be assigned to a focus group which may consist of several business and technical consultants.  Each member of the focus group will provide input to major change requests prior to submitting the change request to the Solution Architect for review and presentation at the CCB.

Business and Technical consultants and Workstream Leads are also responsible for identification and submission to the Configuration Manager any cosmetic or minor change requests, complying with organizational templates and standards.

### Change Management

Infor identify three types of Change Requests for Infor (6 section 4.2), that relate to Configuration Changes. Workstream leads are responsible for capturing changes in Test and updating the ‘Changes in TST Document’.

Only Major CR’s will be submitted to the CCCB (3.1.3.13.1.3.1 above).

Question – Emergency Fast Track and Fast Track Change should have impact beyond the CM and Project Director – as neither has reasonable knowledge to reasonably understand impact.

Criteria for determining severity and priority missing and what exists is outside of best practice for problem management.

The three key CR’s are outlined below

#### Fast Track

These are minor and cosmetic changes that require the approval of the Momentum Configuration Manager and the Infor Configuration Manager. These are to be implemented within 24 hours and captured in the ‘Changes in TST document’. A fast track change process is referenced (6 Section 4.2)

#### Major Change

Major changes are identified as software patches (code changes) and significant configuration changes. No major change process is identified. These will be impacted on a wider basis (undefined) and be conducted weekly (no reference to how these are captured).

#### Emergency Fast Track

Emergency Fast Track Changes are to resolve issues that are holding up testing (presumably critical changes). Process is referenced (6 Section 4.2) and require programme director approval.

Provision needs to be defined in terms beyond testing – where regular rebuilds are acceptable and data continuity isn’t an issue. This process would be high risk in Pre-Prod and Production and is unsuited to management of an active or UAT service.

### Requirements Management and Requirements Traceability Matrix

The CM Plan makes reference to changes to Requirements in section 4.2 – Requirements are a CI that needs to be managed as a baseline, as this is what everyone is operating from.

This means that Infor Requirements need to be formally managed, and reported on, and placed under formal change control.

### Configuration Audits

Frequency. For more details on Configuration Audits see section 2.6 above.

## Standards and Methodologies

Key Methodology (Agile / ITIL / CMMI / Lean), Cont. Integration, Cont. Development, Cont. Testing etc. These need to be identified and adhered to.

## Infor M3 Configuration Items

The following objects are identified as Configuration Items in CMA-010 Configuration Management Plan, 0.2, 2017 (6).

### Infor Source Base code

This is defined in CMA-010, section 3.1.1 as being the source code for the M3 Application. Reference is made here to Infor standard products (M3BE and MCE’s) along with product fixes and patches.

There is a specific statement that it is ‘Travis Perkins responsibility to identify and test any changes which may have an impact on their business model’ (6 section 3.1.1). This presumably refers to conducting System Testing and UAT testing etc. I would however be reasonable to expect that any such change should it come from outside of Travis Perkins would be reported via the Change Control Process.

As such the Infor CCB process needs to tie into the TP change management procedure.

### Infor Configuration and Parameters

This is defined in CMA-010, section 3.1.2 in a very broad scope as parameter settings within the application and stored as data within the database.

### Infor Datasets

This is defined in CMA-010, section 3.1.3 as application data that is necessary to initiate any testing cycle.

#### Recommended CI

On initial assessment the applications in 6 4.1.1.2 should be considered CI as they relate to SACM and BAU impact as third party applications and can be subject to change both from Travis Perkins / Momentum and Infor. Configuration control on the tenant and Company level present situations where in the entire Infor system is redelivered for any change – which is functional for Dev but not supporting patching (especially resolution of critical and major service issues).

Additionally, referring to 6 section Appendix A – there is a list of version-controlled Items that should be considered Configuration Items, as these make up the Infor configuration management baseline. Change to any of these applications, whether from within TP, Momentum or Infor would result in a necessary change to the operational baseline (as well as potentially other factors).

The ‘Changes in Test’ Document is a CI as are CRs and PRs and the output of verification of CI changes are CI’s

Finally, in my professional opinion, its necessary to consider as CI other elements outside of typical IT objects such as documents which relate to, produce or inform control of configuration items, including key deliverable documents.

## Tools

### Configuration Management System (CMS)

No reference is made to a CMS in CMA-010 Configuration Management Plan, 0.2, 2017. JFrog is reported as being in use, however this is not strictly speaking a Configuration Management System tool (used primarily for Artefact management and control).

Use of a CMS would integrate change control of CI’s, allow greater depth in terms of CI control (down to code and application levels) as well as providing real time configuration status accounting.

### Infor CMDB

This is referenced in CMA-010 Configuration Management Plan, 0.2, 2017, section 3.3.1. No statement about the CMDB, what its is, what it runs on, were it sits and how its managed.

### Automation tools

What automation tools are in use in Infor for Software Configuration Management and Release Management. How do these engage with Travis Perkins existing process.

### Change Control of Configuration Items

How are changes to CI’s controlled, measured, produced and Reported. Is this via a tool, does this engage with the CMS. This needs to be dovetailed into TP Change Management.

## Lifecycle Diagram (Change to Release)

For Infor from inception to release (SIT is identified at the last environment for which Infor engagement is mentioned).

### RACI for Change to Release Lifecycle (3.5)

Responsibility, Accounting, Consulted and Informed.

### Key Deliverables / Evidence for Change to Release Lifecycle

Evidence and proof at each point of status change in 3.5

## Verification

### Unit Testing and Integration Testing (Pre-Release)

#### Test Scripts, Cases, data and baseline.

Criteria for success? Storage, Unit Test Exit Report, Tests Run and Test Plan etc.

### Environments and provisioning

For development, unit testing and pre-release testing Infor utilise the Dev Tennant. The Test Tennant is also used for Infor System Test and System Integration Test.

Question: Should Infor have access to the Test tenant? Risk to baseline from capacity of Infor to conduct changes on the environment – As the test tenant is built from the CM master to which Infor changes have already been applied. System Test and System Integration Test should be the responsibility of the Momentum Test Manager (not Infor) to verify.

Environment provision services exist within Travis Perkins – This should be utilised to stand up all non-development environments.

## Software Configuration Management

### Point of interface with Momentum / Travis Perkins

CM Plan (6) references Quality Gates (example is presented in Annex C). These need TP CM input. Evidence of criteria fulfilment needs to be established.

### Criteria for Release to Momentum / Travis Perkins (System Test and System Integration Test).

CM Plan (6) references Quality Gates (example is presented in Annex C). These need TP and Momentum criteria input to verify each release. Evidence of criteria fulfilment needs to be established.

## Service Asset Configuration Management

### Interface with Travis Perkins SACM

#### Software Licencing

New applications relating to M3 need to be identified and the cloud solution will need to be subject to Snow based monitoring.

#### Hardware and Asset Managements

Should be unaffected due to the lack of hardware relating to M3

#### Client Management (Minimum build requirements for M3)

A minimum specification for Client based operation needs to be defined.

#### CMDB

The CMDB Discovery sweep will need to identify the Cloud solution and applications in order to capture them.

### BAU Support into Travis Perkins and Service Desk

The solution presented for CM presents a concern in terms of support and management as changes will need to be applied on the application configuration level for BAU maintenance.

## Priority and Severity Definitions

### Priority

An ITIL Based example of establishing Priority is presented in 9.1 below

#### Urgency

#### Impact

### Severity

An ITIL based example of establishing severity is presented in 9.3

## Reporting

### Configuration Status Reporting

Infor will report once weekly (Friday) basis on Configuration Status identifying (6);

* Software
* Data
* Documentation

All items reported should be a CI, as its an essential component in the production of a release.

Annex D to Annex G forms a best practice basis on what should be reportable in terms of Configuration Status Reports.

### Configuration Management Reporting (Audit)

ITIL (3) identifies the following KPI for reporting Configuration Management

* Verification Frequency
* Number of Incidents owing to inaccurate CMS Information
* Effort for CMS Verifications
* CMS Coverage
* Number of unauthorized Changes detected automatically
* Number of CMS Errors

### Change

ITIL (3) identifies the follow Key Performance Indicators for Change Management

* Number of major changes
* Number of CAB meetings
* Time for Change Approval / Rejections
* Change Approval vs Rejections
* Number of Emergency Changes

Infor reporting on change identifies the following criteria (6 Section 4.3)

* How long a CR has been open
* Distribution of CR’s to Group / Owner
* Trending – What areas approved changes are occurring in.

### Problem Management

ITIL identifies the following KPI’s for Verification

* Number of Problems
* Problem Resolution Time
* Number of unresolved Problem
* Number of Incidents per Known Problem
* Time until Problem Identification
* Problem Resolution Effort

### Testing

ITIL identifies the following KPI’s for Verification

* Percentage of failed Release Component Acceptance Tests
* Number of identified Errors
* Time for Error Fixing
* Incidents caused by New Releases
* Percentage of failed Service Acceptance Tests

### Releases

ITIL identifies the following KPI’s for Release Management

* Number of Releases
* Duration of Major Deployments
* Number of Release Backouts
* Number of releases which had to be reversed

# 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 in some cases.

## 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/SCM facing Travis Perkins or Momentum Project. Systems exist for the control of some CI and are defined in the following paragraphs.

Tool use in the SACM aspect of Travis Perkins exists, and is centrally managed as a single team (Led by Nick Ingram reporting into Olly Richards).

The primary role of a CMS is provided using SVN and GitHub as a means of controlling source code control; meaning that configuration control is managed only in terms of software configuration management. However, this is strictly limited to the control of code only.

#### 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.

The degree to which this should be rectified should be considered in terms of the eventual rationalisation of the Momentum solution effectively retiring these heritage systems over time.

#### 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 suite 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 to facilitate operational management.

Discussion with Nick Ingram identified two key objectives that would benefit Travis Perkins and Momentum. The first of these is to ‘lock down’ assets in active service to prevent unauthorised changes to assets from downloading tools for use on an ad-hoc basis. The second of these is to define a Definitive Service Library (DSL) of applications to rationalise down, to provide authorised tools to be used (so rather than multiple pdf readers and converters downloaded by individuals piecemeal the DSL would provide an authorised software application to be utilised to facilitate that function). This would essentially rationalise software usage and licensing into a more controlled and auditable function in terms of both client builds and licence management.

Additionally, by creating a DSL there is a potential cost saving to be made in terms of licensing, as centralised management limited to specific approved software would allow greater capacity in Travis Perkins in negotiation of licence costs and support costs.

##### 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.

Momentum will need to utilise these builds to form a reasonable baseline for testing.

##### 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.

M3 and related licensing will need to be captured within the SACM solution for licensing.

##### 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.

#### 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.

# Reporting Presently

This section of the strategy will deal with reporting currently within Travis Perkins and Momentum and present a list of recommended reporting based on best practice within disciplines (2 and 3). Agile recommends that where possible reporting is rationalised and centralised – Typically driven by tools, such as Jira or Slack (4).

## Reported and Baseline Requirement for Reporting

### Heritage and Development

All Heritage reports are produced and controlled through the use of Jira, which forms the Kanban lists (4) for the integration and deployment (authorised under change control via ServiceNow).

### Infor (CM)

Infor produce a weekly status report of configuration items (3.10.1 above)

### Integration

The utilisation of Jira and Confluence allow the integration team and the Platform DevOps function to produce audited end to end reporting on every stage of the lifecycle that falls within their domain.

### Testing

Testing produce regular reports of defect and daily testing status updates.

### 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.

### Deployment and Environments

TBC

### Software Asset

A monthly report of software licence assets is produced and published for Travis Perkins.

### Client Management

### Hardware Management

### CMDB

## Best Practice Recommended Reporting (KPI’s)

The following section identifies the basic key performance indicators from ITIL V3 – These should for the very basic output of all reports.

### Change Management

ITIL (3) identifies the follow Key Performance Indicators for Change Management

* Number of major changes
* Number of CAB meetings
* Time for Change Approval / Rejections
* Change Approval vs Rejections
* Number of Emergency Changes

### Testing

ITIL identifies the following KPI’s for Verification

* Percentage of failed Release Component Acceptance Tests
* Number of identified Errors
* Time for Error Fixing
* Incidents caused by New Releases
* Percentage of failed Service Acceptance Tests

### Problem Management

* Number of Problems
* Problem Resolution Time
* Number of unresolved Problem
* Number of Incidents per Known Problem
* Time until Problem Identification
* Problem Resolution Effort

### Configuration Management Status Reporting.

Configuration Management Status Reporting should identify;

* Unique Reference
* Description
* Version
* Status
* Relationships (Annex F)
* Type and Subtype (Annex D)
* Authorities (Annex E)

### Release

ITIL identifies the following KPI’s for Release Management

* Number of Releases
* Duration of Major Deployments
* Number of Release Backouts
* Number of releases which had to be reversed

### Configuration Management and Service Asset Configuration Management

ITIL identifies the following KPIs for Configuration Management.

* Verification Frequency
* Number of Incidents owing to inaccurate CMS Information
* Effort for CMS Verifications
* CMS Coverage
* Number of unauthorized Changes detected automatically
* Number of CMS Errors

## 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.

The use of Jira presents excellent reporting mechanics and dashboard facilities that can produce real time reports across complex criteria relatively simply. Additionally, where external reports distributed by Spreadsheets is desired, Jira can be easily connected to a master excel document via an internet link that allows for reports to be generated automatically (once written). Confluence builds on this extensively (Confluence pages can contain Jira reports and dashboards that can be fed into documentation and processes).

This creates a more efficient means of reporting and communication across a project, moving reliance away from shared spreadsheets and documents, towards a ‘centralised single point of the truth’ that is updated in real time, with regulated status audit reporting. This reduces a reliance on email and meeting based confirmation of events towards a continual basis.

# 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 and Platform DevOPs

## 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

Licencing utilises the Snow application which produces a real time account of licensing within the Travis Perkins estate. SACM publishes a monthly licence report of all assets reported by Snow.

## 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

Defect triage meetings are arranged on a regular basis and owned by the Test Defect Manager.

## 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

To be determined.

# Scales for Priority and Severity

Based on Prince2 Methodology and ITIL definitions the following section provides a basis for establishing priority and severity of incidents; this should form a basis on which priority and severity are defined in relation to issues, defects, changes, improvements and requests for change.

## Priority

ITIL defines priority as being defined by the combination of urgency and impact, to create a scale of 1 to 5 (with one being the highest, and five the lowest priority). These are defined in section 9.1.1 and 9.1.2 below) to produce a priority matrix

### Urgency

#### High

The damage caused by the Incident increases rapidly, or work that cannot be completed by staff is highly time sensitive. Minor incidents that need to be acted on to prevent them becoming a major severity or where the issue is high profile and impact is likely to reflect poorly on the company. A memory leak might be medium severity, but high priority as over time the risk of increased severity creates a priority to fix.

#### Medium

The impact caused by the Incident increases considerably over time. Or there is a ‘high profile’ impact on the company.

#### Low

The damage caused by the Incident marginally increases over time or is stable. Work cannot be completed that is not time sensitive.

### Impact

#### High

A large number of staff are affected and/or not able to do their job. A large number of customers are affected and/or acutely disadvantaged in some way. The financial impact of the Incident is (for example) likely to exceed $10,000 or the damage to the reputation of the business is likely to be high.

#### Medium

A moderate number of staff are affected and/or not able to do their job properly. A moderate number of customers are affected and/or inconvenienced in some way. The financial impact of the Incident is (for example) likely to exceed $1,000 but will not be more than $10,000. The damage to the reputation of the business is likely to be moderate.

#### Normal

A minimal number of staff are affected and/or able to deliver an acceptable service, but this requires extra effort. A minimal number of customers are affected and/or inconvenienced but not in a significant way. The financial impact of the Incident is (for example) likely to be less than $1,000 or the damage to the reputation of the business is likely to be minimal.

## ITIL Priority Matrix

Cross reference of the urgency with the impact provides a priority code. This is used to then determine the priority (critical to very low), a target response time and target resolution time.

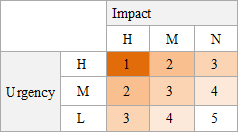


Table 9.2-1 ITIL Priority Matrix

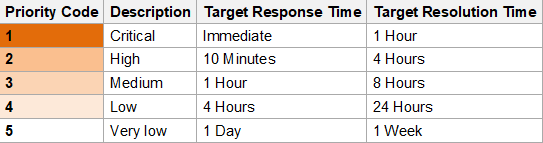


Table 9.2-2 Priority Code Target and Resolution Times

## Severity

Severity defines the impact of an issue in terms of its effect on requirement functionality. The purpose of this strategy a central requirement is defined as a unit of functionality, and non-central represents a process within a unit of functionality.

### Critical

An issue that affects a central requirement for which there is no workaround. It prevents either use or testing of the system.

### Major

An issue that affects a central requirement for which there is a workaround. Use or testing of the system can proceed in a degraded mode.

### Medium

An issue that affects a non-central requirement for which there is no workaround. The feature cannot be used. Or a major issue, for which there is a reasonable work around.

### Minor

An issue that affects a non-central requirement for which there is a workaround.

### Cosmetic

Information is correctly shown but the appearance is wrong, such as misspelled words, wrong font, wrong indentation, etc.

1. Additional Configuration Items

Over the lifecycle of large project it is advisable to treat certain key deliverables as Configuration Items, and place these under controls to produce baselined, controlled objects.

Some examples of these are;

* Requirements
* Milestone Deliverables
* Test Exit Reports
* Test Cases (each Test phase should capture all test cases used as a baseline)
* Requirements
* Training materials
* Process and Procedure documents
* Known Issues (Defects accepted into live service)

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’s 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 C below.

* 1. Document Management (Centralised Document Library).

The production of a centralised library of documentation, under revision control is recommended as a methodology for controlling key project documents. This should include, at the least, contracts, designs requirements, specifications, processes, procedures, strategies, plans, templates, analysis, guides, training documents, work arounds / Instructions.

Currently this is managed using google docs and related drives, which provides some revision control over documents. However, there is a reasonable case that some documents should be considered CI’s due to their universal impact across the lifecycle of a project and changes should be controlled and reviewed accordingly. For example, where changes to a document could have impact elsewhere, such as a design specification which would have potential impact on the developed solution, defined test cases and support of the live service.

This would arguably be a process typically associated with Quality Management but also touches on the basis principles of configuration management.

1. Milestone Lifecycle and CI Gates

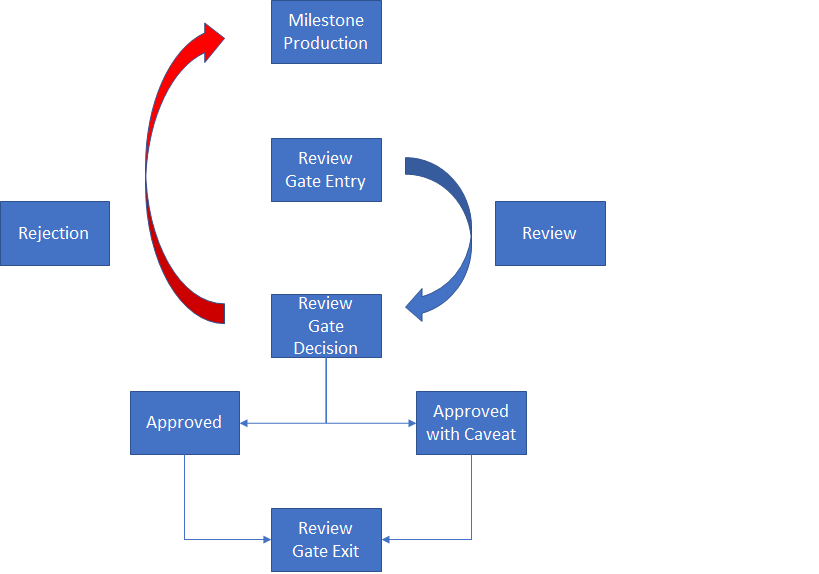


Table 6-1 Review Gate Process

Review Gates serve as a process validation point. These are utilised at key points within a project, typically where handover of ownership occurs or at milestones. This assumes a set of key deliverables, or activities and confirmations that must occur for progress through to the next lifecycle stage.

The review is conducted against an approved criteria checklist, typically based on a template controlled under the auspices of Configuration Management. This checklist forms the basis of a short meeting in which team managers confirm or present evidence that key activities have begun, occurred or been completed.

Where activities have not occurred or are incomplete then actions will be raised against the owning parties to produce these. Dependent on the severity of the non-compliance a caveat may be issued or that rejection of progress occurs, pending a revisit of the review gate based on the completion of assigned actions.

Approval is granted automatically, where all criteria have been met, and all questions raised by the reviewing board are approved. Caveated approvals can be granted, where in actions are raised to be completed before the next review gate (and these become criteria to be revisited at the next review gate).

Each review board / meeting will be minute’d and distributed to all attendees and parties identified on the process stage RACI (3).

1. Configuration Types and Subtypes.

CI’s should be managed by type and subtype where ever possible. This relates more to SACM (ITIL v3, 2011). However, its recognised that a type and subtype methodology applies to SCM content, where a ownership relationship can exist (for example source code is a subtype to an artefact which is a subtype to an application etc). This creates a hierarchy of CI’s that associates relationship allowing CI’s to be identified in relationship to any related component.

This allows a separation of Configuration Status Accounting within the hierarchy and for different attributes to be managed and different Item records to be utilised).

1. Configuration Authorities and Controls

ITIL v3, 2011 (3) Identifies the following Authorities and Controls for each CI type and subtype

* CI type owner
* Authorities for
* creating,
* authorizing,
* modifying
* deleting the CIs of this type
* Applicable processes, controls, guidelines and policies
* Applicable guidelines and policies
* Reporting, auditing and verification requirements

1. CI Relationships

CI’s should exist in relationship to other CI’s (3). For example;

* "Is a component of"
* "Is associated with"
* "Uses"
* "Is a new version of"
* "Will be replaced by”

1. Configuration Item Records.

ITIL v3, 2011 (ITIL v3, 2011) Identifies that a Configuration Record should detail the following information (if applicable[[3]](#footnote-3));

* Unique Identifier
* Name
* Description
* CI owner
* CI type
* Manufacturer information
* Manufacturer name
* Serial number
* License number/ reference to license contract
* Version information
* Location
* Physical location, if applicable
* Logical location, if applicable (e.g. URL or directory on a fileserver)
* Modification history of the CI Record
* Date of CI Record creation
* Modifications
* Date
* Person in charge
* Description of modification
* Status history
* Present status and version
* Status and version history
* Status change
* Description
* Time and date of the Change in status
* Relationships to other CI’s (Annex F above)
* Licensing information
* Document references

# Appendix B – Quality Assurance

A gap in operational service that is directly akin to CM within TP and Momentum is that of a Quality Manager (QM), and processes utilised in QA to ensure agreed and project standards are being achieved (Such as corporate policy, ISO standards, software coding standards etc). Whilst this would be more applicable where the project is providing services into another client, the role of quality management still is necessary to assure standards meet the standards required of Travis Perkins plc. Traditionally, CM and QA work closely together, with procedures for SCM and SACM being tied to QA requirements at key stages (such as defining criteria for Milestone Lifecycle and CI Gates Annex C above).

1. Service Support

Whilst Travis Perkins has an extensive capacity for the support of services, there needs to be some significant attention paid to support and configuration management of Release 1 post-cut over.

Management of configuration changes after ‘cut over’ is currently within the remit of Amanda Key. However, attention needs to be paid to ensuring familiarity with the M3 solution and ensuring that resourcing is sufficient.

Additionally, going into Release 2 there needs to be provision for ensuring that the additional brands outside of BSS can utilise Release 1 via formal verification. The assumption that Release 1 will work with other brands based on its successful implementation in BSS cannot be assured due to the complexity of the integration into Heritage systems, and how the Heritage systems are entwined and share data across brands.

* 1. Decommissioning Workstream

The act of taking heritage applications and systems out of service needs to be closely managed and reviewed. As there is little formal brand separation in terms of how heritage systems have been deployed, there is a high level of cross pollination of dependency of multiple brands to single applications that may not be clearly identified. As such, it would be advisable for obsolescence to be conducted as a formal project, with close attention paid to verification of safe obsolescence and strict change control being conducted.

1. For the purposes of this audit phrasing Change relates to any kind of driver of change in a CI – Defect, Issue, Request for Change, Change Request. [↑](#footnote-ref-1)
2. The term CR for this purpose is expanded to include any driver of change in a CI – Defect, Issue, Request for Change etc. [↑](#footnote-ref-2)
3. As CI’s exist in relationship to other CI’s at some point each field is required. For example, code and applications on a VM don’t have a physical location, however at some point in the hierarchy of relationship, this will almost always be relevant at some point. [↑](#footnote-ref-3)