|  |  |
| --- | --- |
| 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 8](#_Toc508716114)

[1.1 Introduction 8](#_Toc508716115)

[1.2 Scope 8](#_Toc508716116)

[1.3 Maintenance of this Document 8](#_Toc508716117)

[1.4 Assumptions 8](#_Toc508716118)

[1.5 Minimum Specification 9](#_Toc508716119)

[1.6 Establishing a Configuration Management Process 9](#_Toc508716120)

[1.7 High Level Process 10](#_Toc508716121)

[2 Configuration Management Procedure 11](#_Toc508716122)

[2.1 Planning 11](#_Toc508716123)

[2.2 Identification 11](#_Toc508716124)

[2.2.1 Source Code 11](#_Toc508716125)

[2.2.2 Third Party Software 12](#_Toc508716126)

[2.2.3 Software Artefacts 12](#_Toc508716127)

[2.2.4 Universe Applications 12](#_Toc508716128)

[2.2.5 Configuration Data 12](#_Toc508716129)

[2.2.6 ‘Change Requests’ 12](#_Toc508716130)

[2.2.7 Integration Points 13](#_Toc508716131)

[2.2.8 Extensions 13](#_Toc508716132)

[2.2.9 Environments 13](#_Toc508716133)

[2.2.10 Tenants 13](#_Toc508716134)

[2.2.11 Builds (Releases) and Companies 13](#_Toc508716135)

[2.2.12 Service Assets 13](#_Toc508716136)

[2.2.13 Software Licenses 14](#_Toc508716137)

[2.2.14 Client Estate 14](#_Toc508716138)

[2.3 Control 14](#_Toc508716139)

[2.4 Status Accounting 14](#_Toc508716140)

[2.5 Verification of Code 14](#_Toc508716141)

[2.5.1 Unit Testing 14](#_Toc508716142)

[2.5.2 System Test 14](#_Toc508716143)

[2.5.3 System Integration Testing 14](#_Toc508716144)

[2.5.4 User Acceptance Testing / QA User testing 15](#_Toc508716145)

[2.6 Configuration Changes 15](#_Toc508716146)

[2.7 Configuration Audits 15](#_Toc508716147)

[2.7.1 Release Configuration Audits 16](#_Toc508716148)

[2.7.2 CM Repositories Audit 16](#_Toc508716149)

[2.7.3 Change Configuration Audit 16](#_Toc508716150)

[2.7.4 General CM Audit Questions 17](#_Toc508716151)

[3 Configuration Records, Types, Relationships and Authority 17](#_Toc508716152)

[3.1 Configuration Item Records. 17](#_Toc508716153)

[3.2 Configuration Authorities and Controls 18](#_Toc508716154)

[3.3 CI Relationships 18](#_Toc508716155)

[4 Issue, Defect and Change Control Procedures 19](#_Toc508716156)

[4.1 Issues 19](#_Toc508716157)

[4.2 Defects 19](#_Toc508716158)

[4.3 Changes 19](#_Toc508716159)

[5 Tools and Techniques 19](#_Toc508716160)

[5.1 Tools 19](#_Toc508716161)

[5.1.1 Configuration Management System 19](#_Toc508716162)

[6 Momentum Reporting for Configuration Management 23](#_Toc508716163)

[6.1.1 Change Management 24](#_Toc508716164)

[6.1.2 Testing 24](#_Toc508716165)

[6.1.3 Problem Management 24](#_Toc508716166)

[6.1.4 Defect Management 24](#_Toc508716167)

[6.1.5 Configuration Management Status Reporting. 24](#_Toc508716168)

[6.1.6 Release 25](#_Toc508716169)

[6.1.7 Configuration Management and Service Asset Configuration Management 25](#_Toc508716170)

[6.2 Centralised Reporting and Real Time Reporting 25](#_Toc508716171)

[7 Timing of Configuration Management and Change Control Activities 26](#_Toc508716172)

[7.1 Change Management 26](#_Toc508716173)

[7.2 Defect Management 26](#_Toc508716174)

[7.3 Environment 26](#_Toc508716175)

[7.4 Release Review 26](#_Toc508716176)

[8 Software Delivery Lifecycle 27](#_Toc508716177)

[8.1 HLD Lifecycle Diagram 27](#_Toc508716178)

[8.2 Lifecycle RACI 27](#_Toc508716179)

[9 Roles and Responsibilities 27](#_Toc508716180)

[9.1.1 Change Manager 27](#_Toc508716181)

[9.1.2 Configuration Manager 27](#_Toc508716182)

[9.1.3 Release Manager 28](#_Toc508716183)

[9.1.4 Issue / Problem Owner 28](#_Toc508716184)

[9.1.5 Defect Manager 28](#_Toc508716185)

[9.1.6 Service Manager / Service Delivery Manager 28](#_Toc508716186)

[9.1.7 Technical Lead 29](#_Toc508716187)

[9.1.8 Developer 29](#_Toc508716188)

[9.1.9 Support Analyst 29](#_Toc508716189)

[10 Scales for Priority and Severity 29](#_Toc508716190)

[10.1 Priority 29](#_Toc508716191)

[10.1.1 Urgency 29](#_Toc508716192)

[10.1.2 Impact 30](#_Toc508716193)

[10.2 ITIL Priority Matrix 30](#_Toc508716194)

[10.3 Severity 31](#_Toc508716195)

[10.3.1 Critical 31](#_Toc508716196)

[10.3.2 Major 31](#_Toc508716197)

[10.3.3 Medium 31](#_Toc508716198)

[10.3.4 Minor 31](#_Toc508716199)

[10.3.5 Cosmetic 31](#_Toc508716200)

[Annex A. Additional Configuration Items 32](#_Toc508716201)

[Annex B. Project Deliverable Configuration Management 32](#_Toc508716202)

[Annex C. Quality Gates and CI Gates 33](#_Toc508716203)

[Annex D. Configuration Types and Subtypes. 34](#_Toc508716204)

[11 Appendix B – Quality Assurance 34](#_Toc508716205)

[Annex E. Service Support 35](#_Toc508716206)

Index of Figures and Tables

[Table 1.6-1 Establishing a Configuration Management Process 10](#_Toc508716207)

[Diagram 2-1 – ITIL Based Configuration Management Procedure 11](#_Toc508716208)

[Diagram 2.6-1 Configuration Change High Level Lifecycle 15](#_Toc508716209)

[8.1-1 Proposed HLD Lifecycle Diagram (including QA Gates) 27](#_Toc508716210)

[Table 10.2-1 ITIL Priority Matrix 30](#_Toc508716211)

[Table 10.2-2 Priority Code Target and Resolution Times 31](#_Toc508716212)

[Table 6-1 Review Gate Process 33](#_Toc508716213)

Document Control

Changes History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version Number | Date | Author | Change | Status (Draft / Authorised) |
| 0.1 | *08/03/2018* | *Jamie Martin* | Initial draft | Draft |
| 0.2 | 13/03/2018 | Jamie Martin | Updated diagrams added references and defined responsibilities | Review |

Authorised by

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

Distribution

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

References

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

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 (6) |
| Tenant | Tenant is the equivalent cloud term for ‘environment’. |
| TortoiseSVN | A popular GUI version control tool that operates on a subversion database |

Acronyms

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

# 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.

The assumption is made that any device utilising M3 will meet the minimum specification idenfied by Infor.

## Minimum Specification

In order to utilise the M3 Solution the following requirements are identified by Infor

* HTML 5 Compliant Browser
* Windows 7 (Internet Explorer 10 or later)
* Mac OS 10.6 or later
* iOS (latest Safari version)
* Network Bandwidth 40-45kbps

## Establishing a Configuration Management Process

CMMI v1.3, 2011 (v4) Identifies the following QA basis for establishing Configuration Management standards within a project.

|  |  |  |  |
| --- | --- | --- | --- |
| CMMI Stage | Description | Compliance (Yes / No / Partial) | Description |
| 1.1 | Identify Configuration Items | Y | Configuration Items identified in this document and for Infor |
| 1.2 | Establish a Configuration Management System | N | No CMS really exist. |
| 1.3 | Create / Release Baselines | P | Baseline exist for Code and Builds |
| 2.1 | Track Change Requests | P | CR are utilised and recorded in spreadsheets manually. |
| 2.2 | Control Configuration Items | P | Code build CI captured and controlled through DevOps process. |
| 2.2.0 | Establish / Identify Organisational Policy | NA |  |
| 2.2.1 | Establish CM Plan | N | Infor CM plan exists. No configuration management plan or process in place for Momentum. Process and Plans for SCAM for Travis Perkins exist. |
| 2.2.3 | Provide Resources | Y | Resources Currently available |
| 2.2.3 | Assign Responsibilities | P | Defined for Release, Test and Defect Management |
| 2.2.5 | Train People | N |  |
| 2.2.6 | Manage Configurations | N | Configurations managed via the ‘Changes in Test’ Document – Infor configuration is managed on the environment level. |
| 2.2.7 | Identify and Involve Relevant Stakeholders | Y | Stake Holders Identified |
| 2.2.8 | Monitor and Control the Process | N | Process doesn’t exist. |
| 2.2.9 | Objectively Identify Adherence | N |  |
| 2.2.10 | Review Status with Higher Management | N |  |
| 3.1 | Establish a Defined Process | N |  |
| 3.2 | Collect Improvement Information | N |  |
| 4.1 | Establish Quantitative Objectives for a process | N |  |
| 4.2 | Stabilise Subprocess Performance | N |  |
| 5.1 | Ensure Continuous Process Improvement | N |  |
| 5.2 | Correct Root Cause Analysis of problems | Y | Identified in the Defect Management Process |

Table 1.6-1 Establishing a Configuration Management Process

## 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

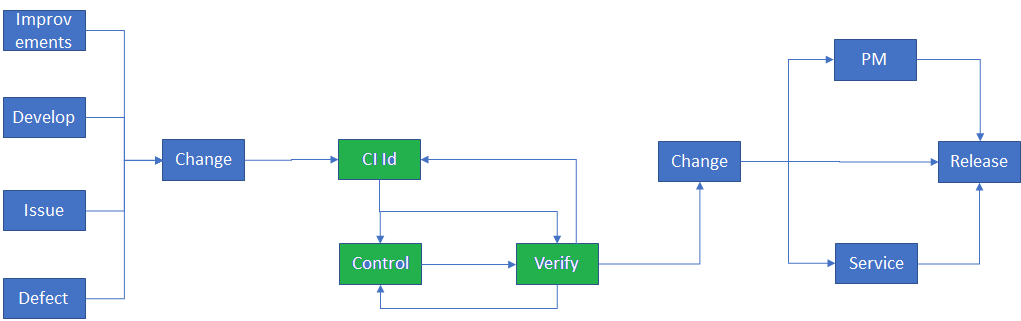


Diagram 2-1 – ITIL Based Configuration Management Procedure

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 3.1 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 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 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.

Currently all source code is managed within the DevOps SVN / GIT applications

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

Source code for Infor M3 is held by the Infor Cloud Operations team and is outside of the Momentum CM strategy for control, being managed by Infor (7).

### 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.

### Universe Applications

Where identified that a heritage or universe application is utilised with Momentum, these will be identified as a configuration item, to ensure that change applied to the heritage system to brands outside of the current, and previous, Momentum Release scopes are risk managed in their impact to Momentum deployed brands.

### 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.

No changes will be accepted beyond a development environment without a formal Change Request (11).

### 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.

### Extensions

Extensions are specific applications produced by Infor that are not part of the default M3 solution – These typically utilise the Mongoose application and reside inside of the Infor Suit – but are effectively applications in their own right, consisting of their own integrations, artefacts and code.

### 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 will be managed under change control.

### Tenants

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) and Companies

Each generated company build deployed to a tenant will be version controlled and managed through the Configuration Management and Release process. Specific attention will be applied to ‘Gold Master’ Builds aka Configuration Builds as these forms the basis on which all companies are refreshed.

Builds and Companies are defined in IT Core Release Management Strategy, 0.1, 2017 (8)

### 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.

Configuration Items will be controlled as ‘baselines’ consisting of one, or more, related CI’s that exist as a fixed point against which changes can be measured. Any change to a CI within a baseline will generate a new baseline of CI. As such any baseline is also a CI

This will include the removal of a CI from a baseline, as well as addition of new CI’s and new revisions on established CI’s within that baseline.

Baselines may also exist of a number of existing established baselines (for example a release may exist as a baseline of several existing baselines plus release deliverables).

Momentum and Travis Perkins CI Control is limited specifically to deployable objects only, and not to ‘the means, by which, they are generated, 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 at present for TP and Momentum.

## Status Accounting

There is no reliable Best Practice method for Status Accountancy Reporting available within Momentum. Status Accounting is managed locally utilising spreadsheets (see section 5 below). This falls beneath the general acceptance of audit capacity and best practices (3; 4; 5 and 6).

## Verification of Code

Formal Verification is defined in TES-010 Momentum Test Strategy, 1.2, 2017 (10).

Verification phases can be simplified for the process of Configuration Management is as follows;

### Unit Testing

Successful system test will be conducted on all changes prior to promotion into the System Test environment. Unit testing is conducted within the development teams. (10). This defines the principle critical test phases for acceptance below.

An exception to this process will need to be defined for Configuration Changes to M3 parameters (these will need to be dealt with as an exception see 2.6 below).

### System Test

Successful system test will be conducted on all changes prior to promotion into the System Integration Test environment. (10).

### System Integration Testing

Successful system integration test will be conducted on all changes prior to promotion into the pre-production environment (10).

### User Acceptance Testing / QA User testing

Successful User Acceptance test will be conducted on all changes prior to promotion into the production environment. (10).

## Configuration Parameter Changes

Changes to parameter configurations to M3 will go through a specific process of CI capture, change and verification, as these will represent changes to settings within the system, as opposed to the issue of code-based changes.

All configuration changes will need to go through a process of impact – identification – verification, within the frame work of the Change Management process.

This will occur against a verified, approved baseline of configuration. Upon deployment a new baseline of verified approved configuration will be created.

Configuration Changes will not be conducted in the Production environment without first establishing

* An Approved Change Request
* Proof of successful verification of the Configuration Change
* Configuration Change made in a representative test environment
* Establishing that the configuration change does not create regression issues
* An updated baseline of Configuration Parameters
* Completed Change Request

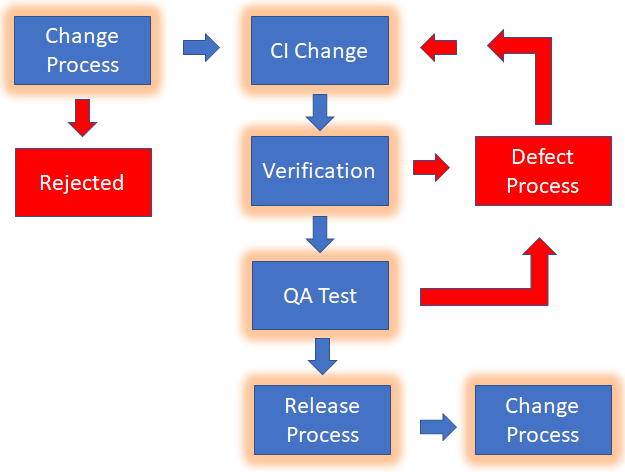


Diagram 2.6-1 Configuration Change High Level Lifecycle

## Configuration Audits

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.

Were viable the use of applications will reduce the physical work required for Configuration Audits.

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 / Configuration 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?

# Configuration Records, Types, Relationships and Authority

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).

## 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 (3.3 below)
* Licensing information
* Document references

## 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

## CI Relationships

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

* "Is a component of"
* "Is associated with"
* "Uses"
* "Supercedes"
* "Superceded by”

# 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.

Defect Management for Momentum is defined in Momentum Defect Management Process, 2.0, 2018 (9).

## 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 for the 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). This needs to be determined how it will work with Momentum.

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.

Any CMS will need to make use of the existing DevOps repositories where existing code baselines are stored or be populated from this source[[4]](#footnote-4). This will necessitate reference directly to baselines identified in SVN from GitHub revisions.

It is recommended that the Momentum project adopts the use of Jira and Confluence across the end to end Release Management lifecycle of the project. This would allow for far greater integrity of service with the existing TP integration and DevOps approach. It would also present a far more rigorous and controlled approach to process modelling that represents best practice (2; 3; 5). It is recommended that access to Jira and Confluence also be rolled out to Infor.

It is recommended that all problem and issue management outside of Testing utilise a shared tool basis. Currently, there is no centralised, reliable means of reporting on all existing issues in Momentum. This is something that should be fairly easily accessable.

Use of Jira and Confluence would allow for effective real time and centralised reporting on any lifecycle defined process, that can be made accessible to any user.

Jira and Confluence have been singled out specifically, as these are currently in use within the DevOps platform, Heritage and Integration teams within Travis Perkins.

From a professional perspective, Jira and Confluence are very useful tools for the management of most project based activities and serves as a means of escaping reliance on emails (not auditable), spreadsheets (not auditable) and reduces the impact of ‘Project Silo’ mentality.

Jira itself is a very easy tool to learn how to use, and relatively easy to manage and maintain.

#### 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.

Issues in terms of how dependencies are established in relation to Momentum and Infor integration need to be captured properly. Development teams need to centrally establish requirements and how requirements will be fulfilled.

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 (5) 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.

#### 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.

All CR’s for deployment must be raised in ServiceNow. Use of tools such as Jira can be used to manage the processes leading into the ‘Request for Change’ and the processes authorised by the approval of a ‘Request for Change’.

Without an approved change, no CI may be updated beyond a development environment.

#### Issue / Problem 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.

Defects within Momentum are managed under an approved process (9) and managed within HP ALM under full defined control.

#### 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.

Momentum will need to establish how the Momentum Solution will be identified for Licencing, Hardware and the CMDB discovery collection.

##### 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, and establish a minimum specification for Momentum for client build.

##### 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

There should be no impact for Momentum Releases in relation to hardware. However, as the ION /WS02 system feeds data to and from the Heritage applications, and some of these reside on physical assets, it will be important to consider hardware changes in relation to impact these may have on Momentum.

##### Configuration Management Database (CMDB)

The CMDB operating within Travis Perkins is based in the ServiceNow application. This system operates based on 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 to produce environments, to create an independence from the Integration / DevOps platform team. This allows duplication of the DevOps Platform processes, utilising specific identified and approved CI’s.

# Momentum Reporting for Configuration Management

This section of the strategy will deal with reporting for Momentum

Best Practice Recommended Reporting (KPI’s)

The following section identifies the basic key performance indicators from ITIL v3, 2011 (3) – These should for the very basic output of all reports produced for Configuration Management.

### 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

Testing is defined in TES-010 Momentum Test Strategy, 1.2, 2017 (10)

ITIL identifies the following KPI’s for Verification (3).

* 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

ITIL identifies the following KPI’s for Problem Management (3).

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

### Defect Management

Defect Management is defined in Momentum Defect Management Process, 2.0, 2018 (9).

KPI for Defect Management are as identified in 6.1.3 above.

### Configuration Management Status Reporting.

ITIL identifies the following KPI’s for Configuration Management Status Reporting (3).

* Unique Reference
* Description
* Version
* Status
* Relationships (3.3)
* Type and Subtype (Annex D)
* Authorities (3.2)

### Release

Release Management is defined in IT Core Release Management Strategy, 0.1, 2017 (8)

ITIL identifies the following KPI’s for Release (3).

* 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 KPI’s for SACM (3).

* 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

## Change Management

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.

All changes for Momentum will need to go through the existing TP Change Management team.

## Defect Management

Full definition of the Defect Process can be found in the Momentum Defect Management Process, 2.0, 2018 (9).

Defect Meetings are held daily (9).

A list of open defects will be distributed daily before 15:00:00.

Defect Triage Meetings will be held daily (9).

## Environment

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

Notification of deployment and shakedown testing is reported by email.

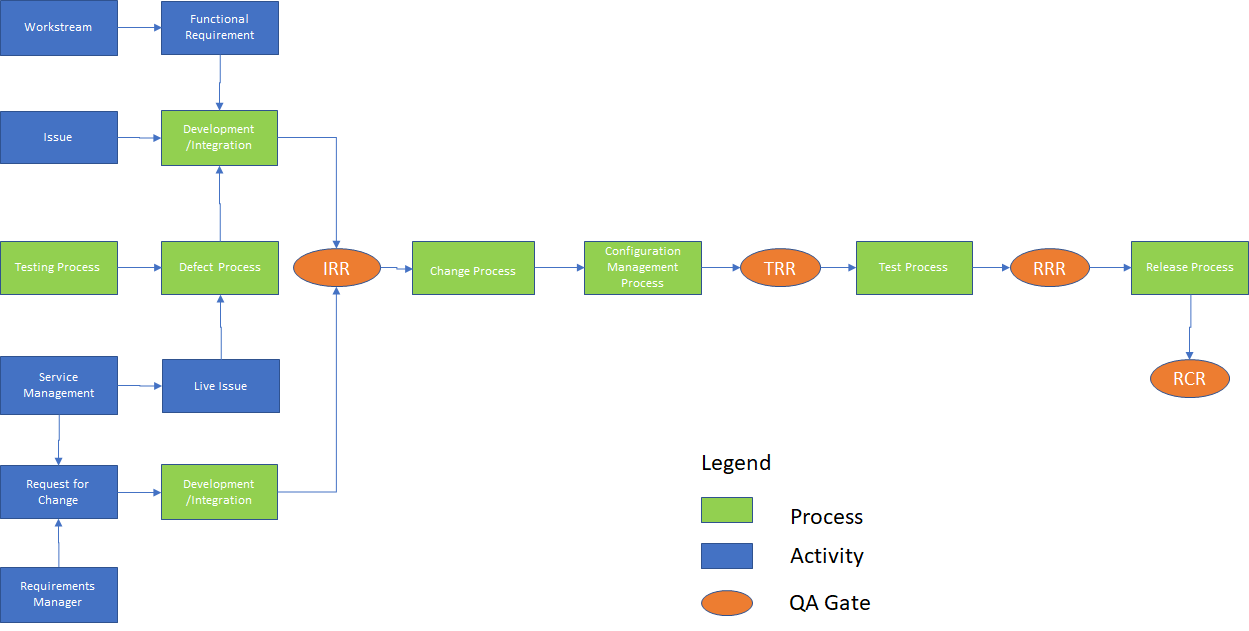
## Release Review

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.

# Software Delivery Lifecycle

## HLD Lifecycle Diagram



8.1-1 Proposed HLD Lifecycle Diagram (including QA Gates)

## Lifecycle RACI

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Stage** | **Description** | **Change Manager** | **Configuration Manager** | **Release Manager** | **Defect Manager** | **Test Manager** | **Service Manager** | **Issue / Problem Owner** | **Technical Lead** | **Developer** | **Support Analyst** | **Requirements Manager** |
| Workstream | Change developed by workstream | - | I | - | - | - | - | - | R | C | - | A |
| Issue | Existing Issue discovered in Development or integration | - | I | - | - | - | - | R | I | C | - | - |
| Functional Requirement | Delivery of a Functionally defined content | - | I | - | - | - | - | - | A | R | - | C |
| Development / Integration | Production of the change or functional code in a Dev environment | - | C | I | I | I | I | C | A | R | - | I |
| Testing Process | Production of a defect discovered in Formal Testing | - | I | I | R | A | - | C | C | - | - | C |
| Defect Process | Management of Change to resolve a defect | - | I | I | R | A | I | I | C | - | - | - |
| Requirements Management | Identification of a required change to a Requirement | I | I | I | I | I | I | C | C | C | I | AR |
| Service Management | Identification of a potential Live Defect | - | I | - | - | - | A | - | C | - | R | C |
| Request for Change | A formal request to apply a change to the environment, either by new / changed requirement or Service Management identified Client Need | R | I | I | I | I | R | I | C | C | I | AR |
| Live Issue | Live issue is recorded as a potential defect | - | I | I | C | I | I | C | C | A | R | C |
| IRR | Quality Review Gate to establish progression of CI changes into Baseline | C | AR | C | I | I | I | C | C | C | I | I |
| Change Process | Process of Reviewing, Impacting and Approving the request for change | A | C | C | C | C | C | C | R | - | - | I |
| Configuration Management Process | Process of affecting the change to the CMS and estabishing new 'candidate release' for Testing | I | A | C | I | C | I | I | R | R | - | I |
| TRR | Quality Review Gate to establish new baselibe is ready for Test | I | AR | C | C | C | I | C | C | I | - | I |
| Testing Process | Application of ST, SIT and QA / UAT as appropriate | I | I | I | R | A | - | C | C | - | - | C |
| RRR | A Quality Review Gate to establish the promotion of candidate release to a release to be deployed | I | AR | C | C | C | C | I | C | - | - | I |
| Release Process | Process that defines progression of 'candidate release' into Pre-Prod and Production | I | C | AR | I | I | C | I | I | I | - | I |
| RCR | Quality Review Gate to establish completion of the Release into Production | I | AR | C | I | I | I | I | I | I | I | I |

Table 8.2-1 CM Lifecycle RACI

# Roles and Responsibilities

### Change Manager

This is the Travis Perkins Change Manager. The Change Manager is responsible for ensuring that all changes are recorded and reported on. The change manager will own the process by which all changes are reviewed, impacted and approved.

* All changes are recorded and reported on
* Ownership and enforcement of the Change Management Process
* Provide guidance and support on Change Management Process
* Ownership of all CI’s held within the ServiceNow Change Management Application
* Publishing the Forward Schedule of Change
* Ownership of the Change Control Board

### Configuration Manager

Configuration Management is defined within this document. The Configuration Manager is responsible for

* Implementation and management of the Configuration Management System
* Ensuring the completion of QA Gates (see Annex C)
* Conducting Configuration Audits
* Identification of Configuration Items
* Enforcing the Configuration Management Process
* Validation and recording changes to CI Records, CI Relationships etc
* QA Gate Reviews at IRR, TRR, RRR and RCR
* Establishing the criteria for QA Gates
* Reporting on the outcome of QA Gates
* Assigning and managing actions from QA Gates
* Configuration Status Accounting Reporting

### Release Manager

Release Management is defined in IT Core Release Management Strategy, 0.1, 2017 (8)

* Manages the overall release process.
* Coordinates release sequences in release slots.
* Coordinates all project teams associated with the release.
* Acts as a liaison with regards to Release in terms of Configuration Management
* Manages the evaluation process upon completion of the project.
* Forms a release team to manage the many required activities (selects team members and assigns team roles and responsibilities).
* Facilitates team communication to ensure that releases are implemented according to schedule.
* Ownership of the Forward Schedule of Release for the Project
* Chairing Release Management Boards and Reviews.

### Issue / Problem Owner

For, the purpose of, this document and process, the issue / problem manager will be individuals within Momentum who have been assigned an issue or problem.

* Issue / Problem Owners are responsible for communication of status and progress of issues that are assigned to them.
* Issue / Problem Owners are responsible for updating CRs relating to progress made.
* Provide regular updates to the technical lead

### Defect Manager

Defect Management is defined in Momentum Defect Management Process, 2.0, 2018 (9).

The Defect Manager is responsible for

* Recording and Management of the Defect Process
* Recording Defects discovered in Testing
* Assigning defects to Technical Leads
* Management of defects through the defect process lifecycle
* Establishing and chairing Defect Triage Meetings (9)
* Confirming Live Reported Issues as New Defects or rejecting them

### Service Manager / Service Delivery Manager

This role has not yet been defined for Momentum.

* Ownership of Live Issues generated by the Support Analyst
* Communication and management of Live Issues between the Service and the Defect Manager
* Ownership of Request for Change from TP Brands

### Technical Lead

The role of the technical lead will be defined as those responsible for the management of the development of a change.

This can be a leader of a workstream, an architect or a project manager. Their responsibility is to manage the production defined by a CR or Functional Requirement, assure standards and methodology are adhered to and serve as the point of communication for the status and issues relating to requests within the process.

* Management of development to resolve the CR
* Ownership of the process by which CI are updated in development
* Assigning CR’s to Developers
* Consulting with stakeholders in throughout the lifecycle
* Ownership of the issues and assigned defects for resolution.
* Raising the Change Management Request

### Developer

For, the purpose of this document, the developer refers to someone who will engineer the change whether this is a configuration change or code change, new functionality or development of a defect resolution.

* Production of the requested change, be it code change, configuration change, fulfillment of a requirement, issue or defect resolution.
* Updating CR records with information regarding to status, resolution
* Production of CI changes to existing baselines to produce a candidate resolution
* Undertaking Unit Testing
* Reporting progress to the Technical Lead.

### Support Analyst

Support analysts refers to those who handle and raise issues reported in the live service.

* Recording Live Issues
* Reporting Live Issues to the Service Manager
* Reporting Live Issue resolutions back to the reporter

# 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 10.1.1 and 10.1.2 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[[5]](#footnote-5) 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.

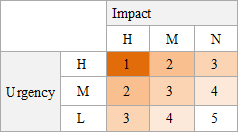


Table 10.2-1 ITIL Priority Matrix

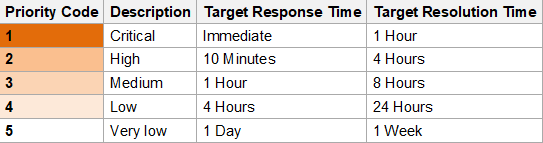


Table 10.2-2 Priority Code Target and Resolution Times

## 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. Quality Gates and CI Gates

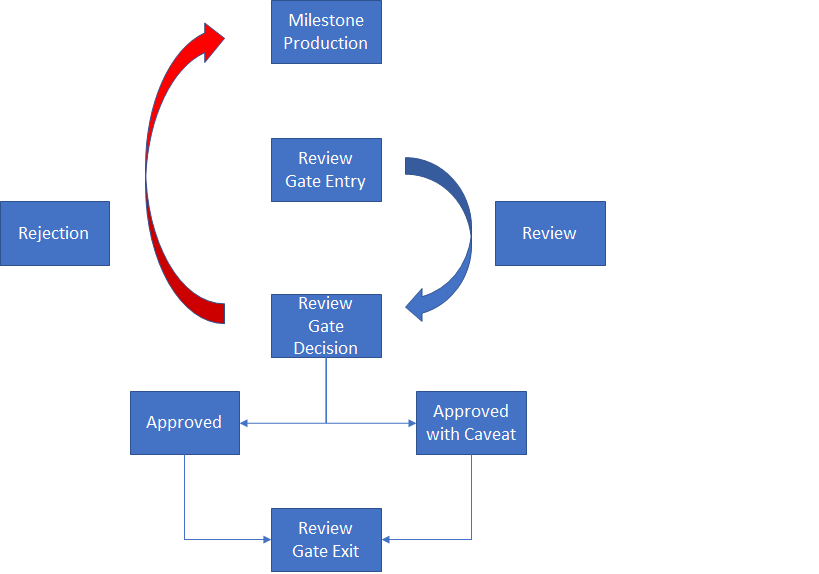


Table 6-1 Review Gate Process

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.

The use of Configuration Types and Subtypes forms the basis of a hierarchy of ownership of CI’s

CI types and sub-types managed in the CMS, usually in the form of a tree structure

This does not have to be limited to physical – Source Code, Artefact, Production scr

**Attributes**

Metadata held against a CI

**Status values**

Lifecycle Status for CI

**Relationships between CI types**

The relation between types and subtypes allows for the identification where ‘lower ranked’ CI’s can be associated into a hierarchy, across baselines (for example an artefact uses source code but it also associated with a release).

The use of relationships also creates a series of relationships in order to identify code dependencies, allowing for accurate identification of impacts related to changes, as well as identifying owning requirements. Accurate use of relationships allows for tracking of relationships throughout the CMS at from any point of reference.

Relationships are identified in 3.3 above

**CI type owner**

Authorities for creating, authorizing, modifying or deleting the CIs of this type, as appropriate, normally assigned as access rights in the CMS and relevant sub-systems

Applicable controls, guidelines and policies (e.g. mechanisms to ensure that only authorized personnel are able to apply changes to the CIs and modify the related CI records in the CMS, or controls/ procedures to ensure the configuration data remain consistent when CIs are added or removed)

# 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 Quality Gates and CI Gates Annex C above).

Of concern, there seems to be a void in terms of identifying methodology and standards that are utilised across Momentum.

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. Presently, the support teams have insufficient experience in M3.

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.

Because there is no CMS in place managing the existing estate and relationships, decommissioning should be approached as a significant risk, due to the cross pollination of applications and data sharing within Universe and Heritage estates.

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)
4. Such replication ideally should be automated as part of the release process – establishing a release baseline rather than a source code baseline. [↑](#footnote-ref-4)
5. Presented as an example [↑](#footnote-ref-5)