Release Management

Standard Practises Document

GAVS Technologies N.A., Inc

116 Village Blvd, Suite 200, Princeton,

New Jersey 08540, USA.

http://www.gavstech.com/

	Designation	Name
Prepared By	Lead Executive – Operations & Strategy Excellence	Gouri Mahendru
Reviewed By	Senior Executive- Quality	Santhiya Palanisamy
Approved By		

Change History

Version No.	Release Date	Process Improvement Proposal Reference No.	Summary of Changes	Prepared By	Approved By
2.0	3 April, 2020	Initial Document	-	Gouri Mahendru	Santhiya Palanisamy

Glossary and Acronyms

Sr. No.	Glossary / Acronyms	Description
1	CAB	Change Advisory Board
2	CD	Continual Delivery
3	CI	Continual Improvement
4	CMDB	Configuration Management Database
5	CSF	Critical Success Factors
6	DSL	Definitive Software Library
7	ERM	Enterprise Release Management
8	FRM	Functional Product Requirement
9	KPI	Key Performance Indicators
10	LOB	Line of Business

Sr. No.	Glossary / Acronyms	Description
11	OLA	Operational Level Agreement
12	PIR	Post Implementation Review
13	QA	Quality Assurance
14	RFC	Request for Change
15	SDLC	Software Development Lifecycle
16	SLA	Service Level Agreement
17	UAT	User Acceptance Testing

Contents

Section A: Introduction

1. Overview

The following Release Management Process has been designed for the GAVS IT Service Management program. It will be used as a reference for the implementation and use of the Release Management process on an ongoing basis. This process document is based on the best practices described in the Information Technology Infrastructure Library (ITIL®) Framework.

It includes Release Management goals, objectives, scope, policies, key terms, roles, responsibilities, authority, process diagrams and associated activity descriptions. This Process will have relationships with other Processes and those documents should be read and understood along with this, the primary related processes being Change, Configuration, CMDB, Service Desk, Incident, Problem, Availability, Service Continuity, Capacity, Security and Project Management.

2. Definition

ITIL® defines **Release** (also called a Release Package) as a set of authorized changes to an IT service. That means a release can include hardware and software, documentation, processes, or other components that are essential to successfully implementing an approved change to your IT services.

Release Management is the process of managing, planning, scheduling and controlling a software build through different stages and environments.

It is the responsibility of Release Management to ensure that changes get introduced into the live production environment efficiently and effectively. It includes everything from requirements gathering to managing planning, scoping, building, testing, and deploying. However, it has no responsibility over the content of the 'release' - only in the way it takes place.

Techniques like Agile development, continuous delivery, DevOps, and release automation have helped optimize release management. As a discipline, it draws from both traditional, business-focused project management and technical knowledge of the software development life cycle (SDLC) and the IT Infrastructure Library, a set of practices for IT service management.

3. Process Objective

Release Management aims to plan, schedule and control the movement of releases to test and live environments. The primary goal of this ITIL process is to ensure that the integrity of the live environment is protected and that the correct components are released.

The objectives for the Release Management process are to:

- To create, test, verify, and deploy release packages
- To drive the release strategy, which is the overarching design, plan, and approach for deployment of a change into production in collaboration with the change advisory board (CAB)
- To manage organization and stakeholder change by implementing consistent release processes and enhanced release communication
- To ensure that new or changed services can deliver the agreed utility and warranty

- To determine the readiness of each release based on release criteria (quality of release, release package and production environment readiness, training and support plans, rollout and backout plans, and risk management plan)
- To record and manage deviations, risks, and issues related to the new or changed service and take necessary corrective action
- To ensure there is knowledge transfer to enable customers and users to optimize use of services that support their business activities
- To coordinate comprehensive user acceptance testing and pilot staging that includes verification of rollout and backout procedures

4. Scope

The scope of Release and Deployment Management includes all Configuration Items (CIs) that are required to implement a release, including:

- Virtual and physical assets
- Applications and software
- Training for staff and users
- All related contracts and agreements

4.1 In Scope

The main components to be controlled are:

- Application programs developed in-house
- Externally developed software (including standard off-the-shelf software as well as customer-written software)
- Utility software
- Supplier-provided systems software
- Hardware, and hardware rollouts
- Assembly instructions and documentation, including User manuals.
- User Acceptance Testing
- Configuration Item versioning methodologies
- Procedures to maintain copies of production software in a Definitive Software Library (DSL)

4.2 Out of Scope

There is a close relationship between Release Management and Change Management. The boundaries between the responsibilities of Release and Change Management need to be clearly understood with Operational Level Agreements defining the obligations and performance expectations involved (i.e.., Release and Change Boards).

The Release process considers application builds from the moment of testing completion to "release" into the production environment. Project Management up to the point of pre-production testing is covered by Application Project Management. Of course, project administration and the end-user participation will appreciably affect the success of the release. Steps done early as part of the project plan will pay dividends during the release process.

4.3 Discretionary

Procedures associated with building and testing releases may or may not be included within the release process. If they are not, included in Release they must be completed as part of the applications overall software development lifecycle (SDLC)

5. Policies and Guidelines

Below are the major policies and guidelines to be considered in Release Management:

- Ultimate approval for any change, including releases, into the production infrastructure resides with Change Management. All releases will ultimately be described in a Request for Change (RFC) and will undergo:
  - Approval of the Change Concept in which a Release Plan and estimated release dates will be presented, and,
  - Requests for approval at each release point wherein implementation preparedness will be assessed. This stage will be accompanied by a "Release Readiness Report" which is attached to the RFC for consideration by the appropriate Change authority.
- The Change Advisory Board (CAB) should make recommendations on a case by case basis, based upon all the relevant facts, on whether the Release unit stipulated in the Release policy is appropriate or whether a Delta Release is preferable.
- All estimated release dates will be recorded in a common Change Calendar (Forward Schedule of Changes) which will constitute the authorized date for the change to be implemented. Changes to this date must be approved by the designated Change authority. The Release Manager should try to keep the Change Manager informed of changes to estimated dates as early as possible and, in any event, must secure authorization before implementing the release into the production environment.
- There will be an Emergency Release process which will parallel the Urgent Change process in Change Management which will have expedited procedures to allow a release to be placed into the production infrastructure.
- A single “Release Engineer” must be identified for every Release. The Release Engineer will be responsible for successful coordination and execution of the Release, as well as ensuring all required documentation related to the Release exists.
- Prior to release all software must be tested from CIs checked out of the DSL. Following Release to the production environment the Release Coordinator is responsible for ensuring both the

DSL and the CMDB are updated accordingly.

- A Defect List describing Known Errors with the release will be provided as part of the Request for Change and forwarded to the Service Desk prior to the release.
- Attempts will be made to secure standard software and hardware when implementing a release based upon the principle that known, standard types with a longevity in the production environment are inherently less risky than new, untested equipment and software.
- Change Management should ensure that there is a formal User acceptance and sign-off before Release Management can continue to roll out the Release.
- The production of back-out plans for each Change is the responsibility of Change Management, but Release Management has a role in ensuring that the back-out plans for each Change within a Release operate together to create a Release back-out plan.
- All releases should have an identifiable business contact who can formally sign-off that the release has been put into production successfully from the point of view of the business.

6. Benefits

Benefits of ITIL Release Management combined with effective Configuration, Change and Operational testing are:

- Increased success rate in the Release of hardware and software and improved quality of service delivered to the business
- Consistency in the Release processes of hardware platforms and software environments
- Minimize disruption of service to the business, via synchronization of Releases within packages involving hardware and software components from different platforms and environments
- Assurance of hardware and software in live use is of known quality, as Releases built properly, components subject to quality control and effective testing, constructed under ITIL Change Management
- Stable test and live environments, as Changes normally combined into Releases with fewer individual implementations
- Efficient use of User resources as efforts combined when testing new Releases - easier to justify cost of system-wide regression testing
- Minimizes regression-testing requirements, offering greater coverage than possible with small Changes that occur frequently or are too close together
- Improved expectation setting within an organization on publication of Release schedule in advance
- Error reduction via controlled Release of hardware and software to the live environment
- Audit trail of Changes to the live environment is maintained, both for software distributions

and hardware Changes

- Control and safeguarding of hardware and software assets, on which organization are heavily dependent
- Ability to absorb high rates of Change to live systems, effectively and without adversely affecting IT service quality. Achieved by releasing large number of Changes together as a single, controlled and well-understood Release
- Ability to build and control software used at remote sites from a central location
- Reduced support costs through ability to maintain consistent software over many locations
- Reduced likelihood of illegal software copies in use at any location
- Easier detection of incorrect versions and unauthorized copies of software
- Reduced risk of unnoticed introduction of viruses and other malicious software
- Reduced time to Release with fewer delays
- Fewer Releases rolled out to Customers
- Smoother transitions of Releases from development activities (projects) to Customer's business environment

7. Inputs and Outputs

Inputs

- Project lifecycle
- Project documentation
- Service-related deliverables
- Authorized RFCs
- Release Policy
- Overview of business needs
- Constraints and dependencies

Outputs

- New Software in DSL
- Updated CMDB
- Known Errors
- Testing results
- User Training

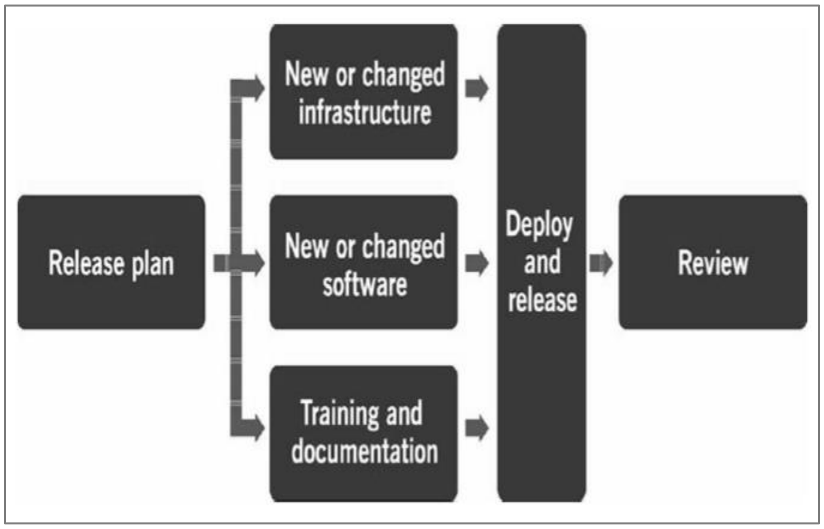
8. Release in Software Engineering

In software engineering, a release is a new or modified software and the process of its creation. A release constitutes a fully functional version of the software, and it is the climax of the software development and engineering processes. Alpha and beta versions of the software typically precede its release.

Most organizations identify releases with a unique set of numbers or letters that update sequentially. This naming process is called **Software Versioning**.

8.1 Release Management with Waterfall

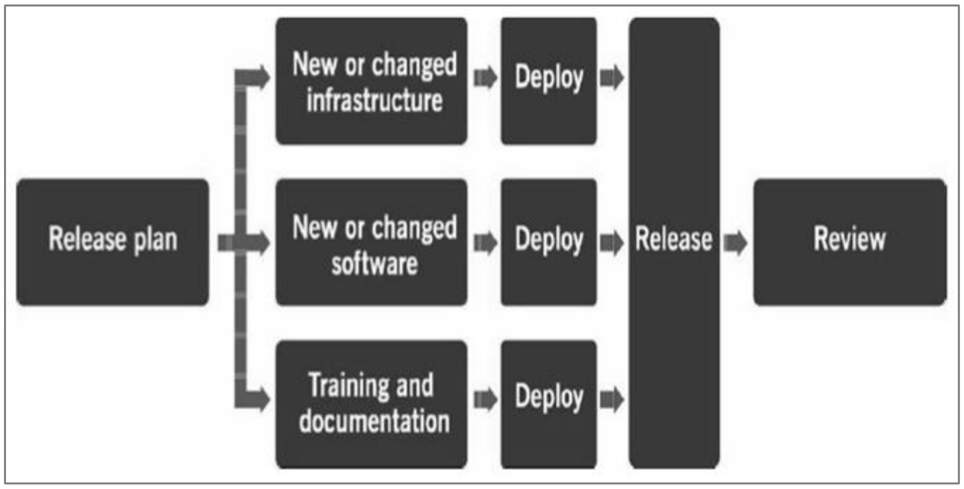
Waterfall methodologies work best in situations where software product requirements can be well-defined up-front with a high degree of certainty. In a waterfall methodology, the definition of work is generally prioritized over the speed of delivery.



- To ensure that work definitions are stringently adhered to, a CAB must be set up to approve software product changes.
- The release manager is then responsible for creating and executing a release plan.
- As the release package goes through testing, any changes that are required to meet quality standards will then be reviewed by the CAB.
- Finally, once a release has been accepted by the QA team, passed testing, or other stakeholder approvals as defined by the Release Policy, it is ready for deployment in the Production environment where end-users or customers will be able to access the new capabilities released.

8.2 Release Management with Agile

Release management can be viewed as an inherently agile process as releases are incremental improvements to a software product at a cadence. The Agile approach to software release is called **Continuous Delivery**, a method that aims to create code that is ready for deployment at any time. This method allows for requirements to evolve over time through small iteration cycles, with automating the release process on very short cycles.



- In Agile methodologies, the central role of the CAB is often replaced by a more decentralized change management process that can adapt requirements sprint by sprint.
- To effectively and efficiently release the code base to production, release managers rely on three things: automation, a DevOps mindset, and continuous integration.
  - **Automation** relates mainly to testing functions
  - **DevOps** mindset vastly improves coordination between development and operations (the delivery and infrastructure people may be the same) in order to smooth out the potentially abrupt transition from the former to the latter
  - **Continuous Integration** is the practice of developers frequently updating their own working copies of code to a mainline — usually about once a day – to weed out errors quickly and speed up the change process

8.3 Release Management in DevOps

Release management is responsible for creating the continuous flow to production that DevOps strives for while aligning work within the team and across the organization.

DevOps cultures emphasize collaboration, experimentation, and continuous improvement through tight feedback cycles. This breaks down work and information silos, creating decentralized, product-oriented teams. It aims to constantly reduce the size of software changes to reduce the risk of disruption from any release. The automation of testing, building, integration, delivery, and other processes reduce the human activities in every release as well as the amount of management and coordination required.

DevOps implement a Continuous Integration and Continuous Delivery (CI/CD) pipeline but not **Continuous Deployment** (a concept that every change made in the code base will be deployed almost immediately to production if the results of the pipeline are successful).

Not deploying Continuous Deployment allows enterprises to retain a level of centralization through release policies that include RACI requirements that incorporate leadership in deployment plans.

- The Product Owner often takes on the role of Release Manager, acting as the bridge between business, product, development, and operations.

- A centralized release management team can work with product owners to ensure that organization release policies are met and spread best practices across teams.
- CAB approves software requirements changes and inspects work after development.
- CAB verifying work performed post-development with a mix of non-functional requirement verification, such as security tests, incorporated into the development process.
- Disciplined DevOps enables automated criteria gates to verify non-functional requirements during development.

8.4 Enterprise Release Management (ERM)

Enterprise release management (ERM) manages the big picture of releases at the organizational level of entities with large or complex software, such as healthcare systems, big companies, universities, and the like. ERM deals with the coordination of individual software releases and how they fit into the organization’s larger plans, strategies, and calendar.

ERM enables organizations to roll out large software products that work well as an integrated whole, and it helps them to do this efficiently. ERM efforts typically call for many release managers to work in tandem, so they can synchronize their respective releases.

8.5 Release Management and PMI Principles

The processes of Release Management correspond quite closely to those of Project Management and an effective release manager needs to possess the same skills as a project manager. Release management rests in part on applying project management principles to software releases.

These include the main project management process areas: Initiating, Planning, Executing, Controlling, and Closing.

- **Functional Product Request (FPR)**: This is the mechanism for a “functional group” to formalize its request for new features or functions to be added to software.
- **Documentation Process**: The team puts information-sharing protocols for the upcoming release into place.
- **Release Packaging Process**: During this process, the team plans about what goes into the final release, based on factors such as market pressure, cost, revenue, development time, and integration.
- **Development Process/Change Control Process**: These two steps run in tandem, and they’re self-explanatory. During development, quality gates may be used to indicate milestones in the development cycle. Change control continues into lab and field tests, which check both whether new functionality is working as expected and whether it has affected existing features.
- **Training Process**: This involves training technical support staff, direct sales and marketing personnel, and telemarketing staff.
- **Customer Testing**: This step happens via beta releases to customers.
- **Customer Notification Process**: The last stage prior to deployment involves informing

customers that a release is forthcoming.

- **Deployment**: Deployment itself is a multi-stage process that includes pre-deployment, actual deployment, and post-deployment.

9. Key Concepts

Following are the most commonly used terms and process components in Release Management:

9.1 Release Unit

'Release unit' describes the portion of the IT infrastructure (new, changed, or unchanged Configuration Item) that is normally released together. The unit may vary, depending on the type(s) or item(s) of software and hardware.

It is a set of configuration items that a team simultaneously tests and releases into the live environment to implement approved changes.

9.2 Release Package

A Package contains an initial version of a new service, several new versions of batch programs, several new and initial versions of individual modules, together with the Release of a completely new desktop system (both hardware and software).

Release Package is a combination of one or more release units deployed together as a single release due to interdependencies, scheduling, or business priorities.

Types of Release Package are:

9.2.1 Major Releases

Infrequent release packages that include often include many release units that have a high or critical business impact.

9.2.2 Minor Releases

More frequent release packages with fewer release units that do not include mission critical components.

9.3 Release Pipeline

The Release Pipeline is a specific release process from feature planning to delivery.

9.4 Release Value Stream

The release processes that add or create value across the release pipeline.

9.5 Release Plan

Release plan is an instance of a release template developed for a specific release.

9.6 Release Record

A release record documents the history of a release, from the planning to the closure of the development process.

9.7 Release Template

The release template is a single, repeatable workflow process for release pipeline that includes human and automated activities and follows an organization's release policies.

9.8 Deployment Plan

Deployment plan contains the activities to deploy a release to the production environment.

9.9 Deployment Work Order

This is a work order for the development or modification of a software application or system.

9.10 Installation Work Order

Like a development work order, an installation work order is for the installation of a software application, system, or infrastructure component.

9.11 Release Policy

Release Policy is a set of rules for deploying releases into the live operational environment, defining different approaches for releases depending on their urgency and impact. This typically contains the definition of release types, standards and governance requirements for an organization.

9.12 Release Types

9.12.1 Full Release

A release that replaces all components of a release unit, regardless of whether they have changed since the last version of software.

The major advantage of Full Releases is that all components of the Release unit are built, tested, distributed and implemented together. There is no danger that obsolete versions of CIs that are incorrectly assumed to be unchanged will be used within the Release.

The disadvantage is that the amount of time, effort and computing resources needed to build, test, distribute and implement the Release will increase.

An example of a Full Release could consist of the complete Release of a new version of client desktop software, or client desktop hardware, or both.

9.12.2 Delta Release

A delta, or partial, release is one that includes only those CIs within the Release unit that have changed or are new since the last full or delta Release.

It is recommended that delta Releases be allowed, with the decision being taken case by case. In each case the Change Advisory Board (CAB) should make a recommendation, based upon all the relevant facts, on whether the Release unit stipulated in the Release policy is appropriate or whether a delta Release is preferable.

For example, if the Release unit is the program, a delta Release contains only those modules that have changed, or are new, since the last full Release of the program or the last delta Release of the modules.

9.13 Release Risk Category

9.13.1 High Risk

- Any Production Support project that has a history of instability or negative business impact as a result of similar releases done in the past.
- Any project that is not frequently repeated and has any of the following characteristics:
    - Involves a change to any part of the system considered to be "business critical"
    - Impacts at least one other system
    - Impacts more than one department
    - Involves formal training (i.e., classroom, workshop, etc.)
    - Impacts customers external to the organization

9.13.2 Low Risk

- Risk Type is assessed in order to determine a Proposed Release Date.
- A Low Risk Type is a project that is frequently repeated and does not include:
    - A history of instability or negative business impact as a result of similar releases done in the past.
    - Impacts to at least one other system
    - Impacts to more than one department
    - Formal training (i.e. classroom, workshop, etc.)
    - Impacts to customers external to the enterprise

9.14 Definitive Software Library (DSL)

This is the term used to describe a secure area in which the definitive authorized versions of all software Configuration Items are stored and protected. This one storage area may consist of one or more software libraries or file-storage areas that should be separate from development, test or live file-store areas. It contains the master copies of all controlled software in an organization. The DSL should include definitive copies of purchased software (along with license documents or information), as well as software developed on site. Master copies of controlled documentation for a system will also be stored in the DSL in electronic form.

9.15 Release Build

"Release build" refers to the task of building the software release from the application's source code that is stored in the DSL. Release management builds all releases for implementation in both pre-production and live environments. A release built for the pre-production environment may malfunction due to environmental dissimilarities if it is implemented in the live environment. Consequently, the release should be built in both environments from applications stored in the DSL.

At a minimum, four types of testing are performed prior to releasing an application:

9.15.1 User Acceptance Testing (UAT)

UAT is designed to ensure that the system operates in the manner expected, and any supporting material such as procedures, forms etc. are accurate and suitable for the purpose intended. The testing checks for compliance with statutory regulations or organizational business processes and rules, and to observe how it will behave under operational conditions.

9.15.2 Load and Stress Testing

The emphasis here is to simulate the conditions of the real environment as closely as possible. However, because equipment and conditions are expensive and difficult to duplicate exactly, measures are frequently taken to accept some risk by utilizing less expensive duplicate equipment or by simulating (rather than duplicating) the conditions which might be expected to occur in the live environment.

9.15.3 Implementation Testing

All major changes to the Production environment should be preceded by implementation tests to ensure that the change can be correctly placed into the Production environment. A completely successful build of the release should occur prior to any final approval to release new configuration items into the live business environment. Implementation Testing applies to all kinds of Major changes and is usually considered part of Change Management.

9.15.4 Back-Out Testing

If an implementation encounters problem within the timeframe established for the implementation there should be a plan to restore the environment to its' pre-implementation condition so that business can resume without interruption. The back-out test provides insurance that the 'known and trusted environment' can be re-created in the event of an implementation failure.

Section B: Roles and Responsibilities

1. User Roles and Functions

The responsibilities of various user roles in Release Management are listed as follows:

Roles	Responsibilities
Release Manager	• The Release Manager is responsible for planning and controlling the movement of Releases to test and live environments. • Their primary objective is to ensure that the integrity of the live environment is protected and that the correct components are released. • All these items do not necessarily have to originate from the same product. • For example, this manager must also coordinate work on any other products that are designed to integrate with the new release.
Project Manager	• The Project Manager is responsible for planning and coordinating the resources to deploy a major Release within the predicted cost, time and quality estimates. • They're responsible for defining the product roadmap and vision and for negotiating deliverables.
Service Owner	• A service owner takes on high-level accountability for a specific IT service within agreed service levels. • They're less concerned with the daily operations needed to deliver the service. • Typically, they act as the counterpart of the Service Level Manager when negotiating Operational Level Agreements (OLAs). • Often, the Service Owner will lead a team of technical specialists or an internal support unit.
Product Owner	• The product owner on a development project is the main stakeholder. • They usually represent the business or the product's (eventual) users, and they define the vision for a product.
Product Manager	• A product manager takes charge of the direction for a single product. • The Product Manager is responsible for both product planning and product marketing. • This includes managing the product throughout the Product Lifecycle, gathering and prioritizing product and customer requirements, defining the product vision, and working closely with engineering, to deliver winning products.

Roles	Responsibilities
	• Typically, a product manager's core responsibilities do not extend beyond a single product or closely related family of products.
DevOps Team	• Since the whole point of DevOps is to increase coordination between the development and operations functions, creating a separate team makes the term a misnomer. • Despite the name, the term usually refers to key team members on both the development and operations sides who work to coordinate the two functions.
Quality Manager	• A quality manager ensures that a release meets stipulated standards. • They may have release managers reporting to them.

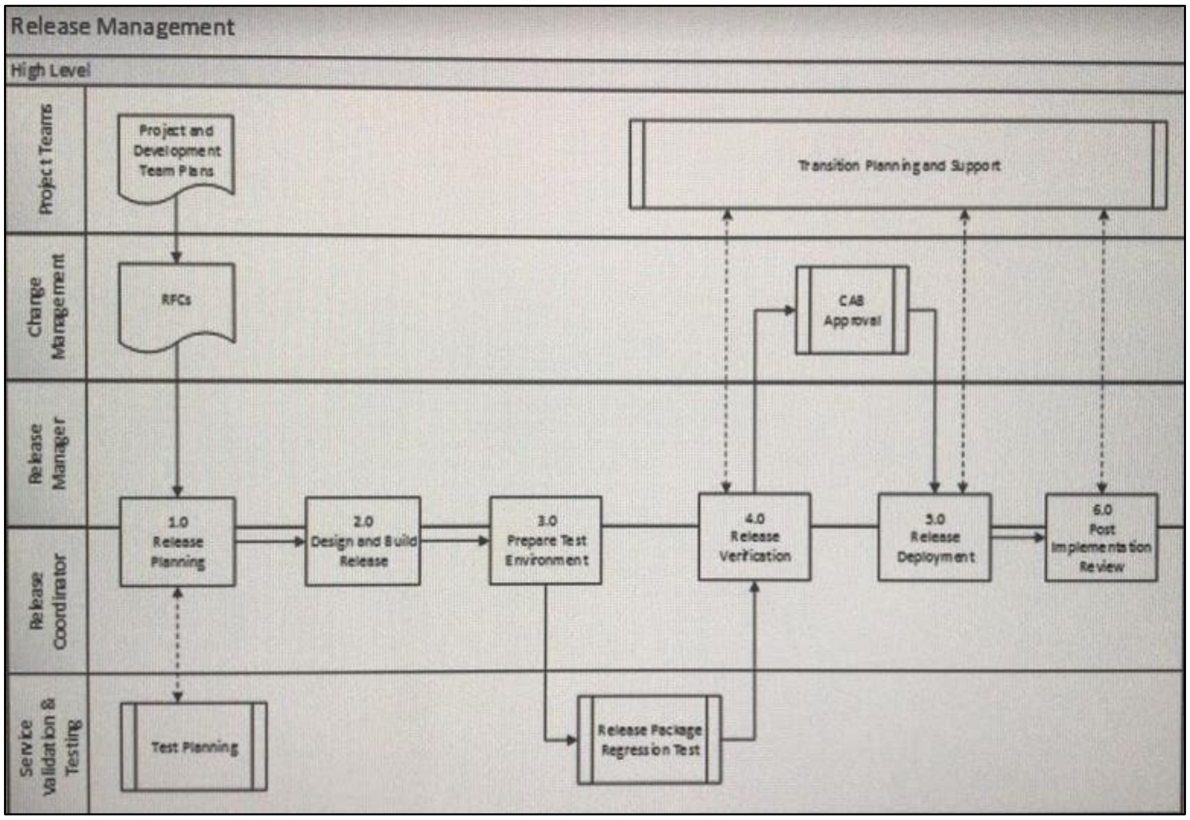
2. RACI Matrix

The following RACI chart outlines which positions are Responsible, Accountable, Consulted, and Informed for each service desk process.

Sr. No.	Activity Description	Release Manager	Project Manager	Service Owners	Other Roles Involved
1	Release Planning	AR	R	I	I
2	Design and Build Release	AR	C	I	I
3	Prepare Test Environment	A	R	C	I
4	Release Verification	AR	I	R	R
4	Release Deployment	AR	C	R	R
5	Post Implementation Review	AR	I	R	R
6	Release Closure	AR	I	C	I

Section C: Process Flow

1. High-level Release Management Process


Release Management
High Level
Project Teams
Change Management
Release Manager
Release Coordinator
Service Validation & Testing
Project and Development Team Plans
Transition Planning and Support
RFCs
CAB Approval
1.0 Release Planning
2.0 Design and Build Release
3.0 Prepare Test Environment
4.0 Release Verification
5.0 Release Deployment
6.0 Post Implementation Review
Test Planning
Release Package Regression Test

1.1 High-level Release Management Process Flow Description

The Release Management process has six basic activities:

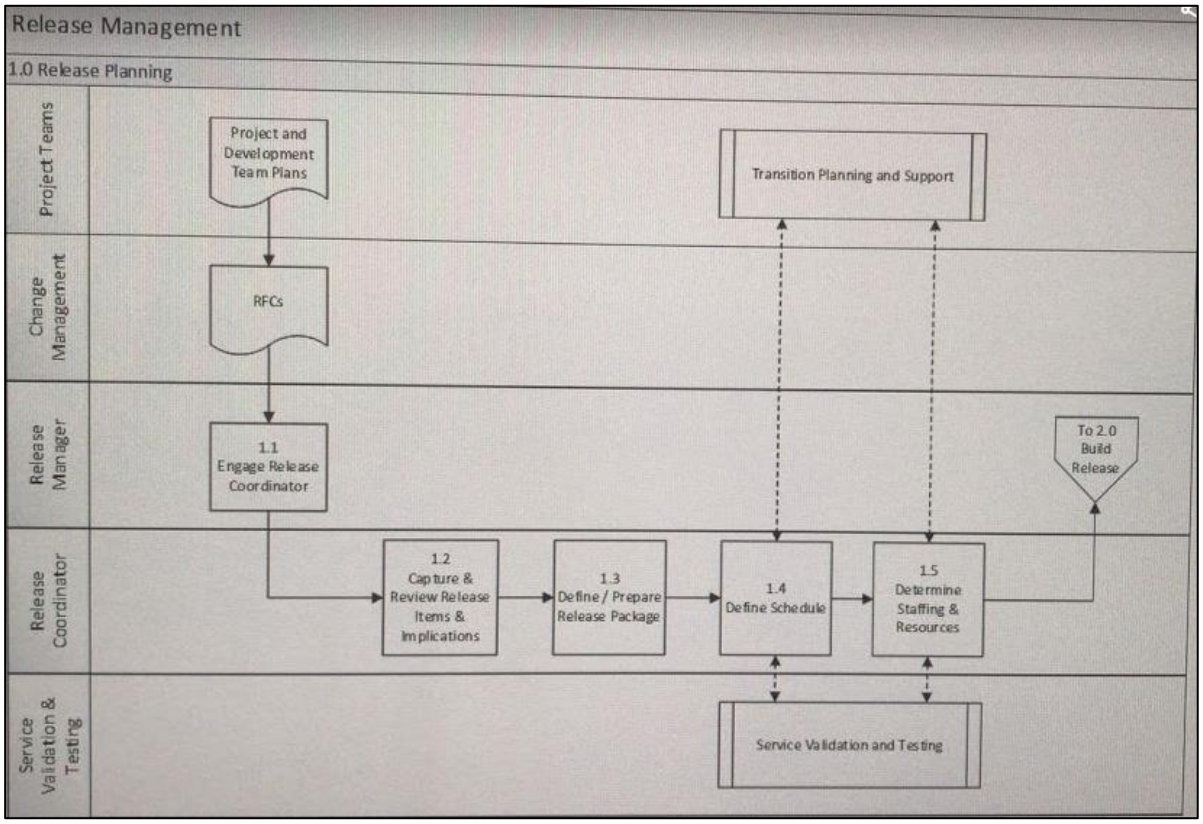
- Release Planning
- Design and Build Release
- Prepare Test Environment
- Release Verification
- Release Deployment
- Post Implementation Review

2. Release Management Sub-process

This section explains the sub-processes defined under Release Management, as:

2.1 Release Planning

Objective: To assign authorized Changes to Release Packages and to define the scope and content of Releases. Based on this information, the Release Planning process develops a schedule for building, testing and deploying the Release.


Release Management
1.0 Release Planning
Project Teams
Change Management
Release Manager
Release Coordinator
Service Validation & Testing
Project and Development Team Plans
Transition Planning and Support
RFCs
1.1 Engage Release Coordinator
To 2.0 Build Release
1.2 Capture & Review Release Items & Implications
1.3 Define / Prepare Release Package
1.4 Define Schedule
1.5 Determine Staffing & Resources
Service Validation and Testing

2.2.1 Gain approval for Release Objectives

Objective: Obtain approval for Release Objectives and approval to proceed with Release planning without expending significant resources in case of a release conflict of conceptual omission or inaccuracy.

The Release Manager coordinates the articulation of the objectives of the release by:

- Determining line of business (LOB) requirements for the release
- Referencing these requirements in view of other release events maintained on the Change Calendar
- Assessing the availability of pre-production test beds and detailing risks associated with the release
- Estimating financial and resources required for release
- Getting the approval to proceed to develop the Release Plan

2.2.2 Develop high-level Release Schedule

Objective: Schedule releases in order to assess their impact upon the infrastructure in relation to other change events.

If approval to proceed is granted by the Release Board then the Release Manager details based on best available estimates, the timing for the rollout. This is done in consultation with line of business representatives and using the Change Calendar as a reference to determine points of conflict and assessing the CMDB for potential points of infrastructure impact. This schedule forms part of the Request for Proposal to be submitted to the Change authority.

2.2.3 Develop Rollout Timetable

Objectives: Better implementation through planned actions

- Produce a timetable of events and document an action plan by site, as well as who will do what (i.e. a resource plan)
- Listing CIs to install and decommission, with details on the method of disposal for any redundant equipment and software

2.2.4 Assemble Release Resources

Objective: Cost-effective provision of resources available to develop design, build and configuration of release. Ensure smooth project execution by ensuring all necessary conditions are in place.

- Using Build Book, obtain any additional IT staff and purchase or otherwise acquire hardware and software components.
- Then identify other resources needed both for reproducing the conditions set out in the Release Plan and for ongoing service delivery.
- All software, parameters, test data, run-time software and any other software that is required for the Release should be under Configuration Management control.
- Quality control checks should be performed on this software before the application is built.
- A complete record of the build results should be logged in the CMDB.

2.2.5 Develop Testing Strategy and Plan

Objective: Ensure that all tests to be performed have been identified, and formal processes and procedures have been developed to encapsulate all test scenarios.

- Establish criteria to determine when and if the application is Fit for Release. This assessment forms part of a Testing Strategy to be developed by the Release Manager.
- The Test Coordinator is then responsible for developing a comprehensive test plan based on the Testing Strategy that includes scope and objectives, schedule, risks, test scripts, documentation requirements, resource requirements, and problem resolution.

- The Testing Coordinator will have the Release Manager approve the Plan. The Plan will be part of the documentation to be placed in the DSL and recorded as a CI in the CMDB.

2.2.6 Develop Site Success Criteria

Objective: Establish site installation performance criteria to assess success of distribution and installation.

Establish Site Performance Goals and Metrics:

- Maximum Number of Users; Web Page Load Time; Percentage Compliance, etc.
- Simulated User Bandwidth (%) LIKE 28.8k, 56k, 128k, 256k, etc.
- Identify Cases for Testing like making a purchase and how the user logs in, selects a product, and enters credit card information

2.2.7 Develop Site and Send Out Communications

Objective: Ensure affected parties are kept informed and establish contingency communications to mitigate effects of any issues encountered.

- Produce Release notes and communications to end Users at specific sites
- Plan communication templates and protocols to Sites
- Establish contingency communications in the event of the need for back-out, incidents, problems, etc.
- Schedule meetings for managing staff and groups involved in the Release

2.2.8 Gain internal approval for Release Plan

Objective: Obtain wide acceptance and acknowledgement for release plan.

Assess magnitude of changes involved and what Release/Change forums need to be approached for approval by one or more of:

- Executive CAB
- Corporate CAB
- Local CAB
- Corporate Change Manager

Change is pre-authorized (ensure adherence to pre-authorized procedures)

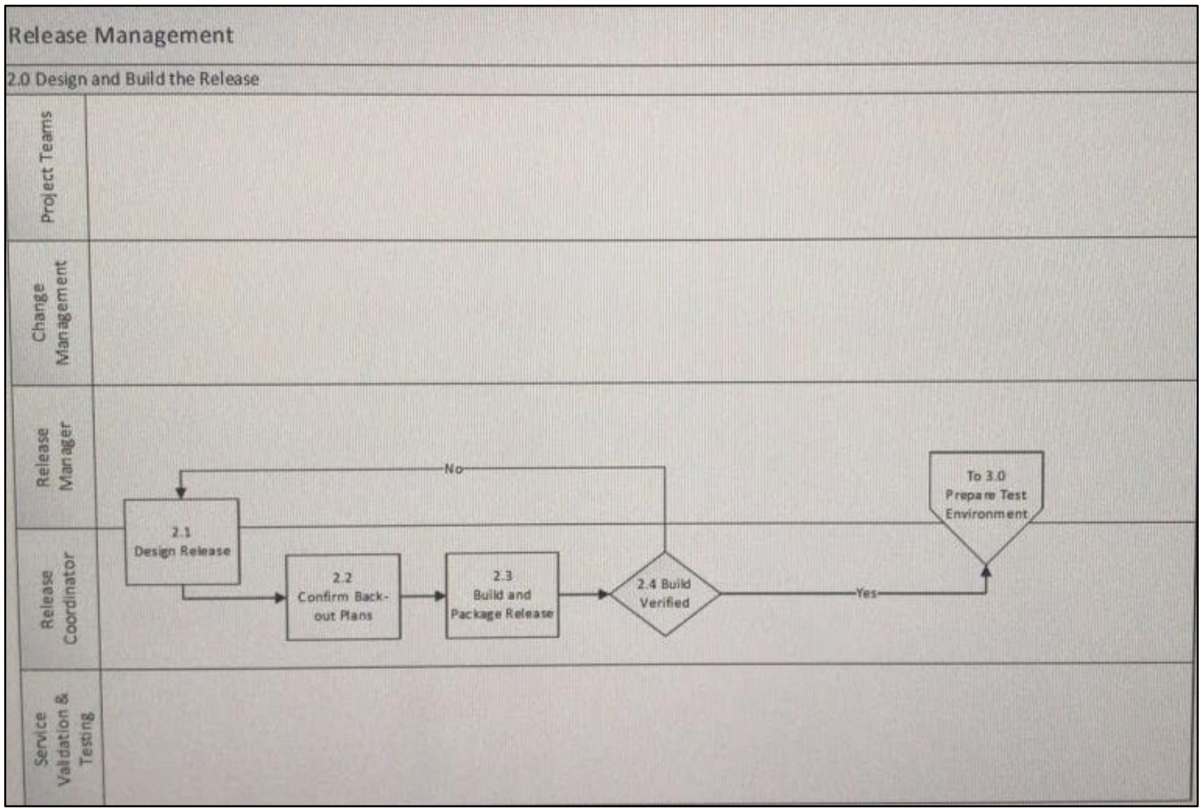
2.2.9 RFC Filtering and Approval

Objective: Gaining approval from the authorized source which is best aware of all possible changes occurring and which best represents the total interests of the entire infrastructure.

The Release Manager develops the Request for Proposal to be submitted to Change Management to gain overall approval to proceed with the Release schedule.

2.3 Design and Build Release

Objective: To issue all necessary Work Orders and Purchase Requests so that Release components are either bought from outside vendors or developed/ customized in-house. At the end of this process, all required Release components are ready to enter the testing phase.


Release Management
2.0 Design and Build the Release
Project Teams
Change Management
Release Manager
Release Coordinator
Service Validation & Testing
2.1 Design Release
2.2 Confirm Back-out Plans
2.3 Build and Package Release
2.4 Build Verified
No
Yes
To 3.0 Prepare Test Environment

2.3.1 Procure / Assemble Hardware

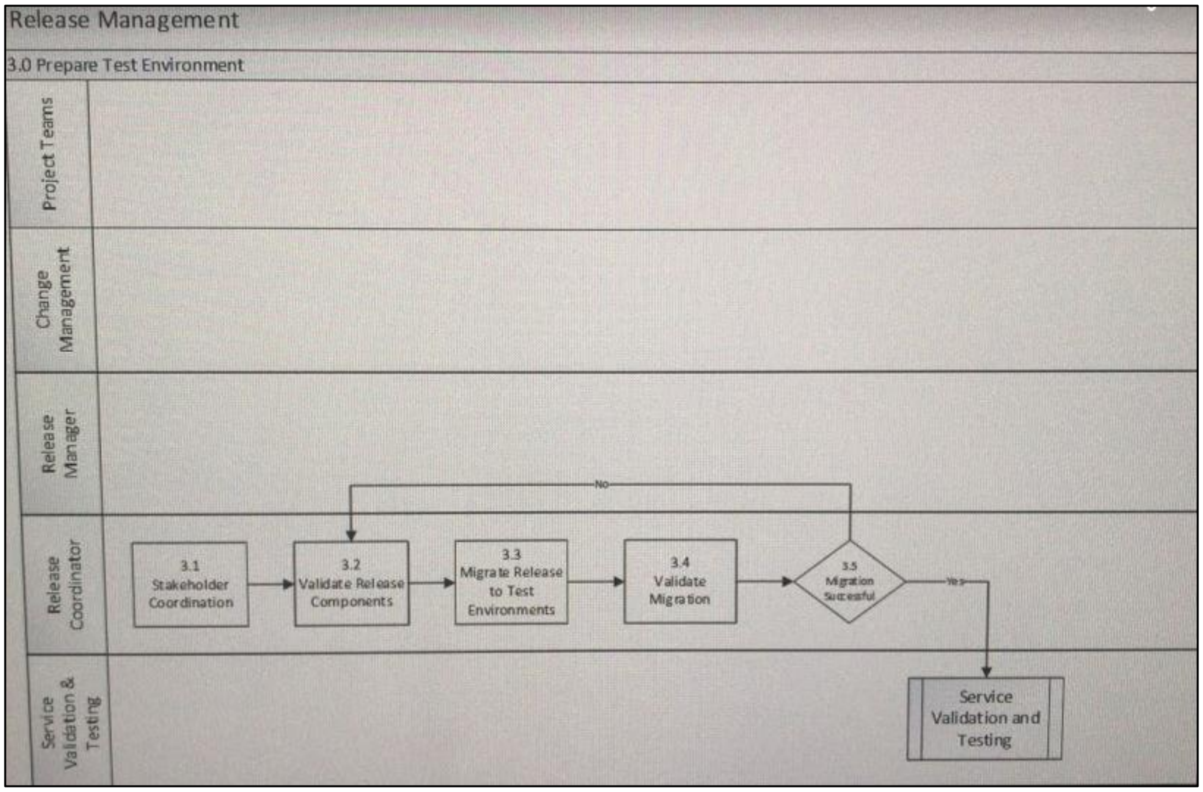
- Using Build Book, reproduce the components required to deliver the service function or end-to-end service.
- Do the same for the components required to provide service support and control.
- Move the components to designated target sites for final assembly, integration, and implementation.

2.3.2 Build / Assemble Hardware

- Using Release plan, perform the final installation and integration of all required management hardware and management application software at the designated target sites.
- Install any software not addressed by the initial component assembly and integration using available software control and distribution tools and methods.

2.4 Prepare Test Environment

Objective: Test environment simulates a production environment or a real environment that enables the testers to influence their testing activities with the conditions and factors of the real environment, to evaluate the working of the software product in the real world.


Release Management
3.0 Prepare Test Environment
Project Teams
Change Management
Release Manager
Release Coordinator
Service Validation & Testing
3.1 Stakeholder Coordination
3.2 Validate Release Components
3.3 Migrate Release to Test Environments
3.4 Validate Migration
3.5 Migration Successful
No
Yes
Service Validation and Testing

2.4.1 Migrate Configuration Items (CI) from DSL to UAT

Objective: Migrate CIs to User Acceptance Testing Environment

The Testing Coordinator checks the CIs designated in the Release and Test Plans from the DSL into the Pre-production (testing) environment. Any additional check-outs are noted by ensuring modification to the Release and Test Plan documents prior to exit.

2.4.2 Conduct Acceptance Testing

Objective: Provide assurance that all system changes and performance issues have been identified during previous test stages, and that the design changes meet original system / user specifications.

These tests should include:

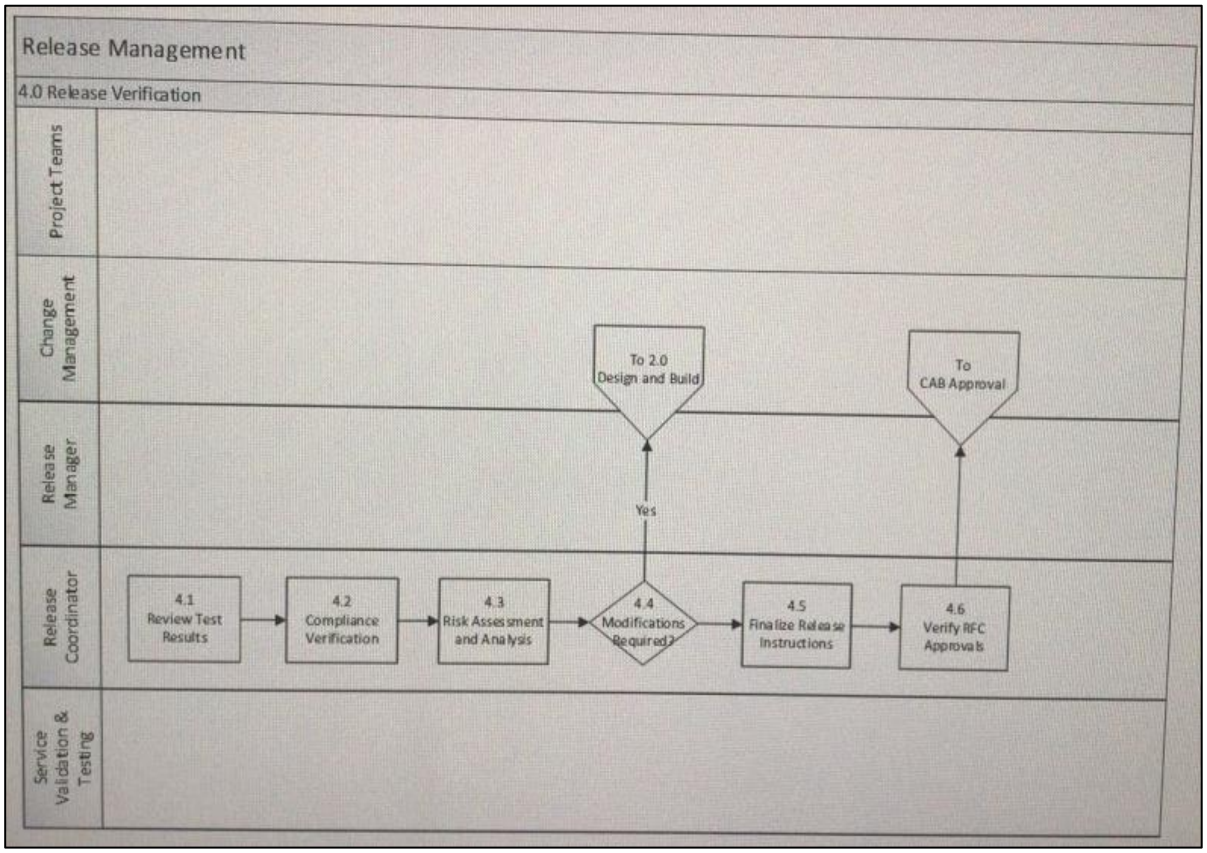
Load / Stress Testing - Creation of production environment (for data and Users), where every piece of system should be tested and monitored. Additionally, performance and response times of all hardware and software components should be analysed. It is important that transaction volumes placed on the system should be of a similar extent to those that could be reasonably expected during system operation in the live environment.

User acceptance Testing - Should be performed by Users for final approval. Business Users should generally have been involved in the testing plan creation from an early stage. Their involvement in the development and evaluation of test plans and results is crucial to achieving a successful User Acceptance function.

Testing should confirm acceptable system results and environment performance and The Testing Coordinator should confirm that test failure points have been resolved and re-tested and the entire process documented. Testing should confirm success of user acceptance shakedown and resolution of identified issues.

2.5 Release Verification

Objective: Release Verification assesses the results of the final test and primarily verify that the newly deployed applications operate as expected in the production environment.



2.5.1 Prepare Site for Release

Objective: Minimize surprises caused by changes to installation sites. Ensure all elements required for the release are in place and operational in order to increase probability of successful release.

For certain releases, it may be appropriate to perform a survey of the production environment to ensure that the rollout plan contains all the activities required to deploy the release into that environment. The survey might also uncover issues with the production environment that could significantly delay or even prevent the deployment of the release. The Release Manager will need to escalate these issues to change management.

2.5.2 Execute Release

Objective: Update the infrastructure without causing failures to any components.

- The Release Manager should be provided with status reports, as changes are made to IT components during deployment, corresponding changes must be made to the configuration items and relationships modelling them in the configuration management database (CMDB).
- Once the release is deployed, the Release Manager and technical support staff needs to confirm that it is working correctly before proceeding with any further deployments.
- If the release fails to meet expectations or serious problems are encountered during deployment, problem management may be needed to help identify and diagnose the root cause of the problem. If a suitable fix or workaround can be found, this should be documented and a request for change (RFC) created to deploy it into the production environment. If not, it may be appropriate to use the rollback procedures to remove the release from that environment.
- Once the deployment phase is complete, the release tracking system should be used to record the results and information about any workarounds or request for change (RFC) raised in support of the release.

2.5.3 Perform Back-out

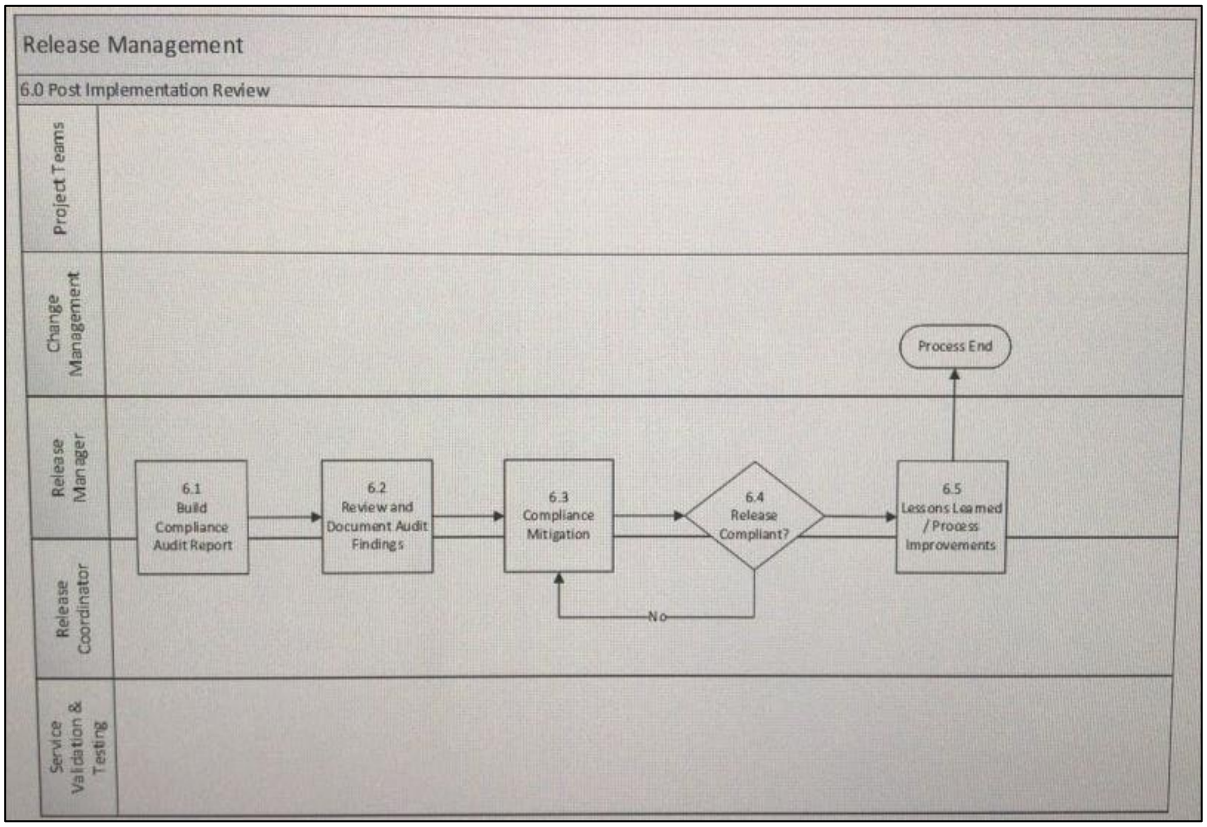
Objective: restore state of known and trusted production environment in order to avoid sources of possible incidents.

- If the Change was not successfully implemented into the production environment the Change needs to be blacked-out with following checkpoints:
- Research other changes that may have occurred between the time of implementation and the completion of the back-out plan.
- Get assurance that the production environment has been restored to its' previous known state.
- Update the log with all information related the decision for backing out the change.
- When back-out is completed, change the status of the RFC to "Completed" with a completion code of Backed Out (or use whatever other method has been agree to within Change management for denoting incomplete changes).

2.6 Post Implementation Release Review

Objective: Codify lesson learned from this release for improvement of future releases

A Change Review is conducted for all major changes as part of the Change Management process: Post-Implementation Review (PIR). In a similar fashion, a Post-implementation Release Review should be conducted which assesses the successes and problems encountered during the entire release process in order to codify the lessons for future release ventures.


Release Management
6.0 Post Implementation Review
Project Teams
Change Management
Release Manager
Release Coordinator
Service Validation & Testing
6.1 Build Compliance Audit Report
6.2 Review and Document Audit Findings
6.3 Compliance Mitigation
6.4 Release Compliant?
No
6.5 Lessons Learned / Process Improvements
Process End

Section D: Governance and Process Controls

1. Critical Success Factors (CSF)

The main Critical Success Factors are:

- Change management to approve changes and track them throughout the release process.
- Configuration management information to build and verify valid test environments in the development phase of the new release, and, to assess the impact of changes to the IT environment and to provide a definitive store for the release package.
- The existence of pre-production facilities for testing releases in an environment which simulates the production environment as closely as possible.
- Load testing procedures and applications to emulate the production environment under different stress conditions.
- Established procedures governing User Acceptance testing

2. Key Performance Indicators (KPI)

The following table lists the key performance indicators (KPIs) for Release Management process. These KPIs will be measured and calculated as a percentage and reflected in the monthly SLA reports.

Sl. No.	KPI	Definition
1.	Number of Releases	Number of releases rolled out into the productive environment, grouped into Major and Minor Releases
2.	Duration of Major Deployments	Average duration of major deployments from clearance until completion
3.	Number of Release Backouts	Number of releases which had to be reversed
4.	Proportion of automatic Release Distribution	Proportion of new releases distributed automatically

3. Reports and Metrics

The following release management metrics were measured continually to tune the release management process.

- Number of changes pending future system releases (backlog)
- Number of successful changes in a release
- Number of failed changes in a release (percentage of failed changes)

- Number of outages caused by a release
- Number of incidents caused by the release
- Percentage of releases delivered on time for QA testing
- Percentage of releases delivered on time for production
- Percentage of releases by priority or type
- Total release downtime, in hours

4. Controls

- Release Policy and Guidelines
- Change Guidelines

Section E: ITIL Inter-relationships and Best Practices

1. ITIL Inter-relationships with other Processes

1.1. Change Management

Once a Change Management process has been defined this imposes restraints around the release of software and hardware products. Because all changes must now be tested and have associated back-out plans there is the need for efficiency in releasing versions. For this reason, explicit policies are devised to "bundle" changes (e.g. Bug/fixes, patches, etc.) into packaged releases which can be put through change management as a single change.

The CAB, as defined in the Change Management process, with advice from Release Management, is responsible for recommending the content and scheduling of Releases. Release Management is then responsible for implementing the agreed Releases. Release Management is normally represented on the CAB and is involved in establishing the organization's Release policy.

Although Release Management oversees the details of the roll out of a Change, it is under the control and authority of Change Management.

1.2. Configuration Management

Whenever new versions of software are added to the DSL, its details should be simultaneously included in the CMDB. Similarly, whenever new or changed hardware is rolled out, the CMDB should be updated.

Release Management may use various services of Configuration Management during the implementation of a Release (e.g. configuration audit to ensure that the target environment is as expected).

1.3. Service Desk and Incident Management

The service desk provides efficient and quality user support and incident management. The service desk should provide support during the pilot program and immediately following full release implementation. Defect Lists and Known Errors should be made available to the Service Desk as these are a prime source of possible failures and, hence, reported incidents. If adequate forewarning is not provided users frustration will be aggravated.

1.4. Problem Management

At the end of the successful distribution and installation of a new Release, various records in the Problem Management system need updating as follows:

- Any related Problems or enhancement requests should be closed
- Problem Management staff should be informed of the new Release so that they can support its use in the live environment. Such staff should receive training in any new or revised support procedures.

Problem Management is also often involved in identifying faults that will lead to Requests for Change (RFCs) and so eventually lead to new Releases.

1.5. Availability Management

Availability management focuses on the availability and reliability of the live computing environment. Implementation of the release in the live environment should be managed by the availability manager in order to minimize disruptions to system availability during the pilot and full releases.

1.6. Service Continuity Management

Normal change, configuration and release procedures address risk mitigation from incidents causing short-term outages. However, a prolonged outage may have repercussions involving risks to the competitiveness or even survivability of the enterprise, there is a need to develop contingencies to cover these events. A Disaster Recovery Plan deals with these kinds of situations. It is closely allied with a Business Continuity Plan, which covers how the organization will continue to operate in the event of a prolonged outage.

1.7. Capacity Management

Capacity management ensures that appropriate IT resources are available to meet business requirements. Capacity management plans for additional resources as use of current system resources increases and nears the point of full capacity. Based on the release scope, the capacity manager determines the infrastructure modifications that are required to maintain system performance and availability after the release is implemented. This individual coordinate all modifications with other release team members.

1.8. Project Management

For software developed under the control of a project, Release authorization requires Operations Acceptance and User Acceptance letters.

Release Management should assist project management in planning and implementing a Release, but it does not take control.

2. Challenges

- Initial staff resistance to change from old procedures
- Teams that need the help of Release Management the most will often have the least time to adopt it.
- Circumvention of Release Management procedures may be attempted, deal with firmly, particularly if it involves installation of unauthorized software as most likely cause of viruses. Untested software makes support very difficult and costly
- Staff tempted to bypass standard procedures to install an emergency fix
- Reluctance to carry out controlled builds in a test environment
- Distributed system difficulties may arise if new versions of software / hardware are not installed or activated on time at remote locations
- Release Management procedures viewed as cumbersome and expensive
- Unclear ownership and responsibilities between operational groups and development teams (project teams) may exist

- Insufficient resources available for adequate testing will reduce effectiveness of procedures, also high number of variants in live environment may limit complete testing
- Insufficient machine and network resources available. May be impossible to build and test new Releases and equipment adequately in a timely fashion.
- Lack of understanding of Release contents, build and installation components, can lead to mistakes
- Testing in one area may be acceptable but may fail in another area
- Staff reluctant to back out a Release, due to pressure to transfer inadequately tested software / hardware into the live or production environment
- Poor, restricted or non-representative testing environments and procedures may exist

3. Best Practices for Implementing Release Management

The key points to be kept in mind while designing Release are:

- Define criteria for success
- Constantly strive for minimal user impact
- Get the most from your staging environment
- Streamlined CI/CD and QA
- Use automation to your advantage
- Make things immutable when possible

Section F: Appendix

1. Templates and Checklists

a. Release Policy Checklist

b. Release Plan Checklist

c. Testing Plan Checklist