**1. Executive Summary**

Market Risk IT aims to implement a test environment to support the validation of MRV (Market Risk Valuation) changes before they are promoted to production. The proposed solution, the MRV Staging Environment, will enable users and IT to compare newly defined measures against actual production data in a controlled environment. This initiative will streamline development, reduce time spent on manual validations, and increase confidence in the accuracy of changes before deployment.

The staging environment will also empower business users to evaluate the impact of new measures independently and validate their accuracy prior to formal release.

**1.1 Background**

Currently, MRV changes can only be tested within the main Power BI Strategic Model. Verification in lower environments is limited due to insufficient data availability, and the QA process is tied to formal release cycles. As a result, MRV changes cannot be reliably evaluated until production deployment.

The MRV Staging environment resolves this gap by enabling earlier validation with full data context, allowing for informed approvals and reducing unnecessary production releases.

**1.2 Scope & Purpose**

The purpose of the MRV Staging Environment is to mitigate technical and operational risks associated with decentralized MRV update workflows. It provides the Reporting and Business Teams with a self-service interface to update, validate, and approve MRV definitions efficiently.

The benefits of automating this process include:

* Improve performance by minimizing communication delays and manual effort.
* Minimize errors through structured validation and impact analysis.
* Simplify governance through clear ownership and centralized workflows.

**2. High-Level Architecture**

**2.1 MRV Definitions Update Process**

The architecture separates production and staging workflows while ensuring synchronization of metadata:

* **Production Path**: The approved MRV definitions are saved in the Postgres MRV Schema, built using the MRV Builder, and applied to the ARC Risk Model (Main).
* **Staging Path**: The MRV metadata is copied to a staging schema, where an independent MRV Builder generates a staging model. Reconciliation is performed using an on-prem IceCube to validate results.

An API interface will replace the existing metadata file load process, enabling Marvel UI to interact directly with the backend.

**2.2 Technical Architecture**

The technical flow includes:

* Updated MRV definitions are submitted via the API or Arc-JDBC-Loader.
* Definitions are stored in a Staging Table for validation.
* Reconciliation processes compare main and staging models to detect data impact.
* Upon approval, definitions are promoted to the ARC Model (Main).

The update cycle is controlled and repeatable, supporting asynchronous creation and reconciliation workflows.

**3. Functional Requirements**

**3.1 Load Metadata to Staging**

* The Arc-JDBC-Loader runs multiple times daily to load metadata from the Oracle control source into the staging schema.
* If MRV changes are detected, they are copied into Postgres staging tables.

**3.2 Run MRV Builder (Staging)**

* A dedicated MRV Builder service uses the staged metadata to build the Staging ARC Risk Model.

**3.3 Reconciliation Process**

* The Recon service automatically executes to compare outputs between staging and production models.

**3.4 MRV Definition Approval**

* Users review and approve MRV definitions through a UI, which displays metadata and change summaries.
* Approved versions are logged with details such as who approved, timestamp, and version number.

**3.5 MRV Definitions Release**

* Releases occur every Friday at EOD.
* Upon release, the system applies the approved version to the production model.
* The staging table is updated, older versions archived, and schema is cleaned to retain only active definitions.

Release logic includes:

* Updating the status of the approved version to 'LIVE'
* Replacing older versions in staging
* Archiving metadata with version control for audit

Here is the revised email, updated to align with the latest version of your document:

**Subject:** Functional Requirements for MRV Definitions Update Process (Staging Environment)

**Hi [Manager’s Name],**

As discussed, please find attached the updated functional requirements document for the MRV Definitions Update Process.

This document outlines the design and implementation of a dedicated MRV Staging Environment aimed at streamlining the validation and release of MRV (Market Risk Valuation) changes. The goal is to provide a controlled and traceable process that enables teams to evaluate the impact of changes against production data before deployment.

### **Summary of the Proposed Solution**

* **Purpose**: Mitigate technical and operational risks by enabling earlier validation of MRV definitions through a dedicated staging environment.
* **Process**: The architecture includes a structured pipeline to load metadata, run model builds, reconcile outputs, and allow for user approval before release to production.
* **Release**: Approved definitions are promoted every Friday at EOD, with complete audit tracking and schema cleanup to maintain consistency.

Key benefits include:

* Improved performance by reducing delays and manual intervention.
* Minimized risk through structured reconciliation and impact analysis.
* Simplified governance via centralized workflows and clear ownership.

Please review the attached document for complete architecture diagrams, technical flow, and detailed functional requirements. Let me know if you’d like to schedule a review session or need further clarifications.

Best regards,

**[Your Name]**

**[Your Title / Team]**

**[Contact Info]**

Let me know if you'd like a PDF version of the document or a PowerPoint summary as well.