DaaS Inventory Configuration: Process Overview

1.0 Introduction and System Context

The Data-as-a-Service (**DaaS**) system was originally architected as a back-end data provider, functioning without a dedicated user interface. However, due to functional limitations within the downstream Master Order Administration (**MAO**) system, **DaaS** 's scope was strategically extended to compensate for these limitations by housing several critical inventory configuration features. This document provides a comprehensive overview of the current, file-based workflow established to handle these configurations.The primary function of this **DaaS** pipeline is to empower business users to perform manual inventory adjustments through a structured, file-based submission process. The system is designed to ingest and validate these user-submitted files, store the configuration data, and transmit it to the **MAO** system according to specific business rules and schedules. This overview will now detail the core architecture of this data pipeline.

2.0 Core Data Pipeline Architecture

Understanding the end-to-end data flow is crucial for appreciating the system's mechanics and dependencies. This section breaks down the current, file-based process, detailing each stage from the initial user submission in Azure Blob Storage to the final integration with downstream systems. The following steps outline the key system handoffs and actions.

1. **File Ingestion** Users from various Business Units (BUs)—including **CDS** , **CFM** , **CFR** , **CMG** , and **Super Sport** —initiate the process by placing configuration files into a designated container within **Azure Blob Storage** . The storage is segregated into folders for each respective BU.
2. **DaaS Processing Trigger** The **DaaS** system automatically polls, or "sweeps," the storage location at regular intervals. When a new file is detected, **DaaS** retrieves it to begin the processing and validation cycle.
3. **Data Validation & Storage** **DaaS** reads the contents of the submitted file. If the file is processed successfully without errors, its data is stored in the **DaaS** database, and the source file is moved to an archive location (referred to as **ACE**). If any errors are detected during processing, the entire file is moved to an **error** folder.
4. **Scheduled Data Transmission** **DaaS** does not immediately push the configuration data downstream. Instead, it acts as a scheduler. Based on the start and end dates specified within the file's data, **DaaS** sends the configured stock information to the **MAO** system at the appropriate, designated time. When the configuration's end date is reached, **DaaS** stops transmitting the override, and **MAO** reverts to sourcing stock data from the primary merchandise systems (**JDA** / **RMS**).
5. **Downstream Integration** In the final stage, the **MAO** system receives the data from **DaaS** . **MAO** may perform its own set of calculations (e.g., deducting safety stock or reservations) before transmitting the final, sellable stock quantity to the Product Master Platform (**PMP**).This pipeline architecture provides the foundation for several distinct inventory management features, each designed to address a specific business need.

3.0 Analysis of Stock Configuration Features

This section provides a deep dive into the specific business functionalities enabled by the **DaaS** inventory pipeline. Each of the following configuration types serves a unique purpose, managed through a distinct technical mechanism.

3.1 Overline & Pre-Order Stock Configuration

- **Business Purpose:** This feature is used to manually create, or "override," stock counts for items that have no official inventory recorded in the primary merchandise systems (e.g., **JDA** , **RMS**). This is essential for handling specific scenarios such as consignment goods (like fresh produce) and for managing pre-order items available in limited quantities. In effect, this configuration allows the business to 'trick' the system into recognizing and selling stock that is not formally tracked in the primary merchandise systems.
- **Technical Mechanism:** The configuration is controlled by a frequency parameter within the submitted file, leading to two distinct behaviors executed by **DaaS** .| Configuration Type | Frequency | DaaS Action || ------ | ------ | ------ || **Pre-Order** | one time | Sends the specified stock quantity to **MAO** *only once* at the start date of the configuration. The stock depletes as it's sold and is not replenished. || **On-Hand (Consignment)** | daily | Sends the specified stock quantity to **MAO** *every day* for the duration of the configuration period (from start date to end date). |

3.2 Safety Stock Configuration

- **Business Purpose:** Safety Stock serves a dual function. Its primary purpose is to act as an inventory buffer, holding back a specific quantity of an item from being sold online. This accounts for potential discrepancies like damaged goods or simultaneous in-store sales. Its secondary use case is to effectively make an item unavailable for online purchase by setting an intentionally high safety stock value (e.g., 999999) that exceeds any possible on-hand quantity. Conversely, this feature may be intentionally bypassed for items with very low stock counts in remote locations, where setting aside even a single unit as safety stock could prevent a sale entirely.
- **Technical Mechanism:** A user submits a file specifying an item, location, and the quantity to reserve as safety stock, along with start and end dates. **DaaS** passes this configuration to **MAO** . Upon the configuration's end date, the safety stock value automatically reverts to zero, releasing the previously reserved quantity and making it available for sale online.

3.3 Stock Reservation (*CMG Specific*)

- **Business Purpose:** This feature is designed to reserve specific quantities of stock for dedicated sales campaigns hosted on various external platforms, such as Shopee or Lazada. It allows the business to allocate inventory for a specific event without making it unavailable across all channels.
- **Technical Mechanism:** In the current architecture, **DaaS** 's role is strictly that of a pass-through. It ingests the reservation request file—containing campaign details, platform, item, location, quantity, and dates—and transmits it directly to **MAO** . The core

logic for processing and enforcing the stock reservation currently resides entirely within the **MAO** system.

3.4 Stock Resync (*CMG Specific*)

- **Business Purpose:** This is an on-demand corrective feature used to manually trigger a stock resynchronization from **MAO** to **PMP** . It is invoked when data between systems is inconsistent or when an immediate update is required outside of the standard, scheduled data flow.
- **Technical Mechanism:** The process is initiated by a user providing a file that specifies the item and location needing a resync. **DaaS** passes this simple request to **MAO** . In response, **MAO** re-transmits the current, accurate stock count for the specified item to **PMP** , ensuring data consistency across platforms.The effectiveness of these features is heavily dependent on how the system manages failures and communicates them back to the user.

4.0 Error Handling and Reporting

In a file-based integration system, robust error handling is critical for operational efficiency and user trust. The **DaaS** pipeline has evolved in its approach to error reporting, with a clear distinction between legacy and modern implementations. The following table contrasts these two models.| System Version | Error Handling Logic | User Impact || ------ | ------ | ------ || **Legacy (non-CMG BUs)** | If a single row in a submitted file contains an error, the *entire file* is rejected and moved to the **error** folder. | The user receives no specific feedback, forcing them to manually diagnose the entire file to find the unknown error. This inefficient, all-or-nothing approach requires resubmission of the entire batch, even for a single incorrect row. || **Modern (CMG)** | The system processes the file line-by-line. It successfully processes all valid rows and generates a detailed report that identifies exactly which rows failed and provides a specific reason (e.g., "Item not found"). | The user can quickly isolate and fix only the erroneous rows for re-submission, resulting in a much more efficient and less frustrating workflow. |
Based on the operational challenges of the legacy model, a key requirement for future development is the universal implementation of a clear, accessible reporting mechanism. Users need a system that definitively identifies which submissions succeeded and which failed, supplemented with actionable error messages for every failure.

5.0 System Variations by Business Unit

While the core **DaaS** data pipeline is leveraged across the organization, its implementation varies, giving certain business units access to more advanced functionalities and improved user experiences.

- **Core Features (Super Sport , CDS , CFR , CFM):** These BUs utilize the foundational feature set, which includes Overline/Pre-Order Stock and Safety Stock. Critically, these units currently operate with the legacy error handling model, where a single error results in the rejection of the entire file.
- **Advanced Features (CMG):** This BU has access to all core features in addition to the more advanced Stock Reservation and Stock Resync capabilities. Furthermore,

CMG benefits from the modern, line-by-line error reporting mechanism, which provides detailed feedback on processing failures and successes.These variations highlight a clear trajectory toward a more sophisticated and user-friendly system, which informs the vision for its future state.

6.0 Future State and Desired Enhancements

The current file-based system, while functional, is recognized as a workaround for limitations in other systems. To improve efficiency, usability, and reliability, significant enhancements are desired. The future state of this system is envisioned to include the following key capabilities:

1. **Dedicated User Interface (UI):** The primary goal is to replace the manual **Azure Blob Storage** file-drop process with a formal, web-based user interface for managing all inventory configurations.
2. **Real-time Validation:** The future UI should incorporate front-end validation to provide users with immediate feedback on errors such as missing mandatory fields or incorrect data formats. This would prevent submissions that are destined to fail in back-end processing.
3. **Enhanced Reporting:** A user-accessible dashboard or report center is required to view the status of all submissions. This interface must provide detailed error logs with clear, actionable messages, empowering users to resolve issues independently.
4. **Centralization of Logic:** There is an ongoing strategic discussion about migrating key business logic—such as the final available stock calculation (deducting safety stock, reservations, etc.) and reservation management—from **MAO** into **DaaS** . This would position **DaaS** as the central source of truth for stock calculations within the future "New OMS" architecture, streamlining the entire data ecosystem.