Oracle® Retail Advanced Inventory Planning
Operations Guide
Release 13.1.1

August 2009

ORACLE®

Oracle® Advanced Inventory Planning Operations Guide, Release 13.1.1

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

(i) the software component known as **ACUMATE** developed and licensed by Lucent Technologies Inc. of Murray Hill, New Jersey, to Oracle and imbedded in the Oracle Retail Predictive Application Server – Enterprise Engine, Oracle Retail Category Management, Oracle Retail Item Planning, Oracle Retail Merchandise Financial Planning, Oracle Retail Advanced Inventory Planning, Oracle Retail Demand Forecasting, Oracle Retail Regular Price Optimization, Oracle Retail Size Profile Optimization, Oracle Retail Replenishment Optimization applications.

 (ii) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(iii) the **SeeBeyond** component developed and licensed by Sun MicroSystems, Inc. (Sun) of Santa Clara, California, to Oracle and imbedded in the Oracle Retail Integration Bus application.

(iv) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(v) the software component known as **Crystal Enterprise Professional and/or Crystal Reports Professional** licensed by SAP and imbedded in Oracle Retail Store Inventory Management.

(vi) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(vii) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

(viii) the software component known as **Style Report™** developed and licensed by InetSoft Technology Corp. of Piscataway, New Jersey, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

(ix) the software component known as **DataBeacon™** developed and licensed by Cognos Incorporated of Ottawa, Ontario, Canada, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Preface

Oracle Retail Operations Guides are designed so that you can view and understand the application's behind-the-scenes processing, including the following:

- Key system administration configuration settings
- Technical architecture
- Functional integration dataflow across the enterprise
- Batch processing

Audience

Anyone who has an interest in better understanding the inner workings of the AIP system can find valuable information in this guide. There are three audiences, in general, for whom this guide is written:

- System analysts and system operation personnel who:
  - Are looking for information about AIP processes internally or in relation to the systems across the enterprise
  - Operate AIP on a regular basis
- Integrators and implementation staff who have the overall responsibility for implementing AIP in their enterprise
- Business analysts who are looking for information about processes and interfaces to validate the support for business scenarios within AIP and other systems, across the enterprise

Related Documents

For more information, see the following documents in the Oracle Retail Advanced Inventory Planning Release 13.1.1 documentation set:

- Oracle Retail Advanced Inventory Planning Release Notes
- Oracle Retail Advanced Inventory Planning Patch Installation Guide
- Oracle Retail Advanced Inventory Planning Data Management Online Online Help
- Oracle Retail Advanced Inventory Planning Data Management Online User Guide
- Oracle Retail Advanced Inventory Planning Order Management Online Help
- Oracle Retail Advanced Inventory Planning Store Replenishment Planning User Guide
- Oracle Retail Advanced Inventory Planning Warehouse Replenishment Planning User Guide
- Oracle Retail Advanced Inventory Planning Implementation Guide
- Oracle Retail Advanced Inventory Planning Data Model Volume 1 Oracle Data Model
- Oracle Retail Advanced Inventory Planning Data Model Volume 2 Measure Reference Guide

Customer Support

When contacting Customer Support, please provide:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to recreate the exact error message received
- Screen shots of each step you take

Review Patch Documentation

If you are installing the application for the first time, you install either a base release (for example, 13.0) or a later patch release (for example, 13.0.2). If you are installing a software version other than the base release, be sure to read the documentation for each patch release (since the base release) before you begin installation. Patch documentation can contain critical information related to the base release and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

In addition to being packaged with each product release (on the base or patch level), all Oracle Retail documentation is available on the following Web site (with the exception of the Data Model which is only available with the release packaged code):

Documentation should be available on this Web site within a month after a product release. Note that documentation is always available with the packaged code on the release date.

Conventions

Navigate: This is a navigate statement. It tells you how to get to the start of the procedure and ends with a screen shot of the starting point and the statement "the Window Name window opens."

> **Note:** This is a note. It is used to call out information that is important, but not necessarily part of the procedure.

```
This is a code sample
    It is used to display examples of code
```

A hyperlink appears like this.

Advanced Inventory Planning

AIP Architecture

The AIP architecture consists of the AIP modules distributed across two platforms:

- The RPAS platform
- The Java/Oracle platform

External Systems

- ❑ Merchandising System (RMS)
- ❑ Forecasting System (RDF)
- ❑ Replenishment Optimization (RO)

To External Systems

Advanced Inventory Planning

RPAS Platform

AIP RPAS Client
SRP
WRP
USA

AIP RPAS Batch
AIP Binaries

RPAS Database

Java/Oracle Platform

AIP Java Client
DM
OM

AIP Java/Oracle Batch
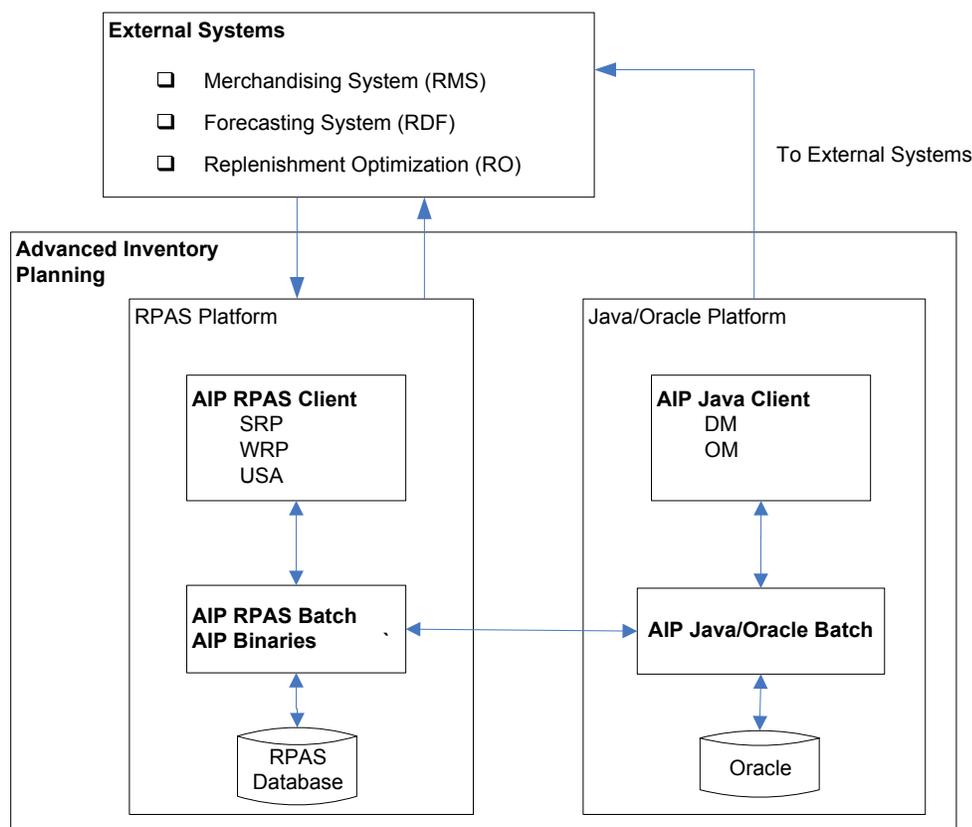
Oracle

AIP Architecture

RMS: Oracle Retail Merchandising System

RDF: Oracle Retail Demand Forecasting

RO: Oracle Retail Replenishment Optimization

OM: Order Management

RPAS: Retail Predictive Application Server

DM: Data Management

OM: Order Management

SRP: Store Replenishment Planning

WRP: Warehouse Replenishment Planning

USA: User Specified Allocation

The external merchandising system, the forecasting system, and the replenishment optimization system are integrated with AIP to provide the inventory/foundation data and the forecasting data to AIP to effectively plan the inventory flow across the retailers supply chain. AIP can integrate with any merchandising or forecasting systems. In the preceding diagram, for example, we have integrated the AIP solution with the Oracle Retail Merchandising System (RMS) and the Oracle Retail Demand Forecasting (RDF) system and Oracle Retail Replenishment Optimization (RO) system.

RPAS Platform

The replenishment and allocation calculations across the supply chain for stores and warehouses are implemented using the C++ component architecture. The calculations are provided as binaries and are integrated using rule groups and then executed using the AIP RPAS batch process.

To modify and update the parameters for store planning, or to manage the alerts raised during the batch process, AIP provides these client utilities:

- Store Replenishment Planning (SRP)
- User Specified Allocation (USA)

Similarly, to modify and update the parameters for warehouse planning, AIP provides the Warehouse Replenishment Planning (WRP) client utility.

RPAS Client Utility SRP & WRP Workbooks	Client Layer
AIP ➢ Replenishment Binaries ➢ Allocation Binaries ➢ DM & AIP Batch	Implementation Layer
RPAS Platform	
Configuration Tools	Configuration Layer
Retail Predictive Application Server Calculation Engine Alert Manager	Application Layer
RPAS Database	Database Layer
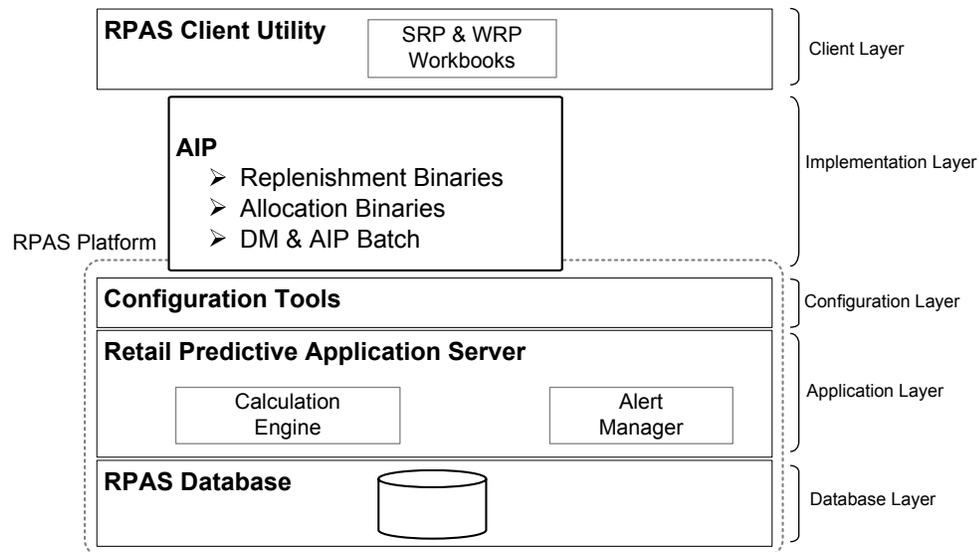
AIP RPAS Platform

The RPAS platform is used for implementing the predictive planning solutions. For AIP, the RPAS platform provides the following layers:

Client layer

RPAS provides a client utility for users to work on the worksheets within the workbooks. The users view and maintain the measures for the replenishment and allocation calculations across the supply chain. The user interfaces for SRP, WRP, and USA are provided by the client utility. Users need to install the client utility provided by the RPAS platform and configure the SRP and WRP user interfaces. Based on access rights, the AIP planner can log in to the different workbooks.

Implementation layer

The AIP solution is implemented on RPAS by using all the features and functions provided by the RPAS planning platform. AIP is implemented using the C++ component architecture, and is executed in the form of binaries during the AIP batch process.

Configuration layer

RPAS provides the Configuration Tools. The RPAS Configuration Tools are used by the AIP implementers to set up the workbooks, measures, and rules for the business needs of the retailer.

Application layer

The Alert Manager is associated with the workbooks. Alerts based on business scenarios and thresholds are designed in the alert manager. When the planners log in to the system using the client utility, the alert manager window pops up with all the alerts listed. This enables the planners to take appropriate actions.

Database layer

RPAS provides a multi-dimensional database, for AIP to support the following:

- Different hierarchies
- Flexibility to navigate to various intersections, across all the hierarchies
- Ability to easily modify values across any point in the hierarchy, and apply changes across all the levels by using the spreading and aggregating techniques

AIP Java/Oracle Platform

> **Note:** AIP Java/Oracle, AIP on Oracle, and AIP online are often used interchangeably to refer to those parts of AIP that access the Oracle relational database. This includes the Data Management and Order Management GUI components and a host of UNIX shell scripts and PL/SQL modules.

The two AIP online modules, Data Management and Order Management, are Web-based applications that interact with an Oracle database and run on Oracle application server. Using a Web browser, users can log in to the multi-user graphical user interface, an applet.

The Data Management (DM) module is implemented across both the Oracle and RPAS platforms to manage the supply chain parameters across the retailer's stores and warehouses.

The Data Management module has two parts that operate on the Oracle database:

- **DM Batch** – This is an extensive set of processes that involves extracting data for use in RPAS, loading data updated or created in RPAS and also performing a number of steps to automatically setup and maintain the supply chain.
- **DM Online Interface** – This Java applet client utility is used by the AIP planners for managing the supply chain.

The DM Batch processes execute on both the RPAS platform and the Oracle platform. It is used to sync and source data across all the AIP modules.

> **Note:** On the RPAS platform, the DM module has only the batch process. There is no user interface.

The Order Management (OM) module is used to manage the purchase orders and transfers generated by the planning calculations, and then release them to RMS. The OM module operates only on the Java/Oracle platform.

The Order Management module has two parts that operate on the Oracle platform:

- **OM Batch** – This process consists of loading the planned purchase orders and transfers from RPAS into the Oracle database and then releasing them to RMS on their lead times.

- **OM Online Interface** – This is a Java applet client utility used by AIP for manually creating purchase orders, releasing purchase orders early, and managing the purchase orders and transfers which have been released to RMS.

AIP provides a client utility for the OM module, and it also has a batch process for a daily mass release of purchase orders and transfers which have met their lead time.
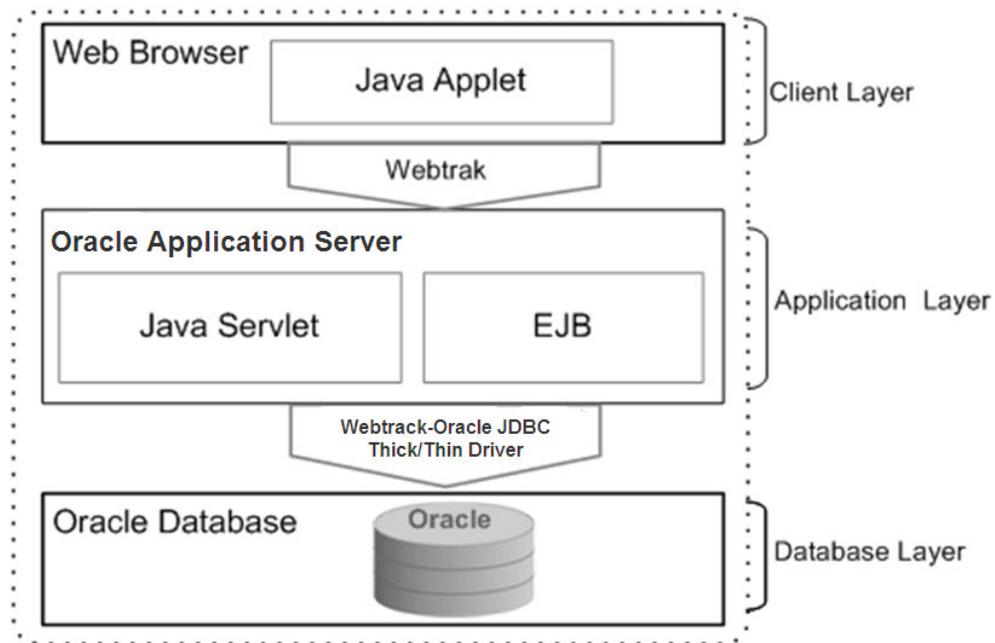
AIP Java/Oracle Platform

The AIP Java application is designed with the following layers:

Client layer

AIP Online provides a Graphical User Interface (GUI), which is a Java AWT applet. Users can log in by using a Web browser. Both the DM and OM modules are accessed through the Web interface. A common user account is created by the AIP Online administrator so that the user can access both the DM and OM modules.

Application layer

The applet communicates with Oracle application server through an Oracle Retail WebTrack communication layer. The server runs a Java servlet, which provides a data access layer.

Database layer

The server communicates through the Oracle Retail WebTrack and an Oracle JDBC driver to an Oracle database. Both thin and OCI drivers are supported.

2

AIP Integration

The AIP solution uses the following Oracle Retail integration tools to integrate within AIP and also with other Oracle Retail products:

- Oracle Retail Integration Bus (RIB)
- Oracle Retail Extract, Transform, and Load (RETL)

AIP is distributed across two platforms: the RPAS and Java/Oracle platforms. There are two types of integration in AIP: external and internal integration.

External Integration

The AIP solution integrates with the Oracle Retail Merchandising System (RMS), Oracle Retail Demand Forecasting (RDF), and Oracle Retail Replenishment Optimization products.
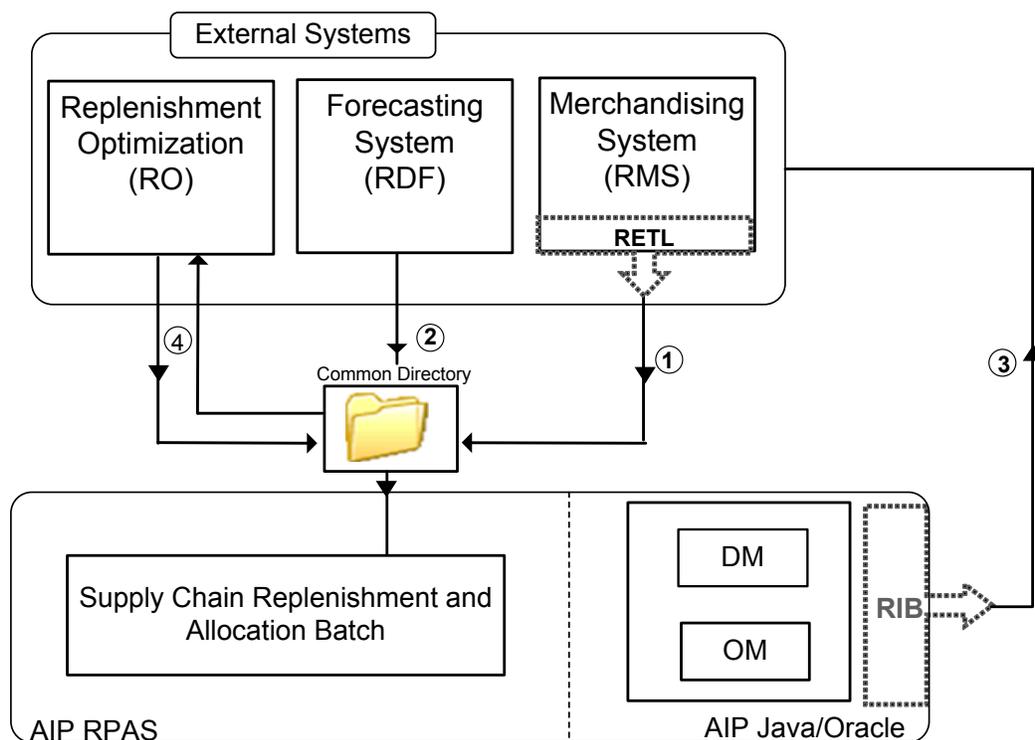

AIP External Integration

1. **Integration between RMS and AIP using RETL:** The data required by AIP from RMS is extracted. The data from RMS Oracle tables are transformed into an AIP compatible format (text files) using the custom script that calls the Oracle Retail Extract, Transform, and Load (RETL). The extracted text files are placed in the common directory for AIP batch to access these files. AIP batch scripts further processes these text files and loads them into the RPAS platform and AIP Online platform.

   > **Note:** Part of the RMS data is loaded into DM Online during the internal integration of AIP.

2. **Integration between RDF and AIP:** The retailer needs to have a custom script to extract the files from RDF for AIP. Then the AIP batch scripts load the RDF measures into the RPAS database.

3. **Integration between AIP and RMS using RIB:** After AIP has completed a replenishment plan across the retailer's supply chain the plan is passed to OM. OM determines which purchase orders and transfers have met their lead time and therefore must be passed to RMS for execution. The purchase orders that are manually created in OM Online must also be passed to RMS. All purchase orders and transfers that are to be executed in RMS are passed to RMS using the Oracle Retail Integration Bus (RIB). RIB is a near real-time data synchronization solution used by AIP for publishing orders to RMS. AIP publishes two sets of order messages to RIB: purchase orders and transfers. RMS subscribes to the RIB messages and inserts the orders into the appropriate RMS purchase order and transfer tables.

4. **Integration between AIP and RO:** RO is a replenishment optimization system and the integration of AIP and RO is an optional setup for Retailers. Lead Time, Review Time and Preferred Pack Size data is exported out of AIP in flat files for use when optimizing the replenishment settings. AIP then loads the optimized replenishment parameters from flat files provided by RO. The integration with RO occurs infrequently such as monthly or quarterly. Retailer's who do not run RO should exclude the released AIP batch scripts from their job scheduler tasks.

Internal Integration

The process of integrating the AIP Java/Oracle platform with the AIP RPAS platform is called internal integration.
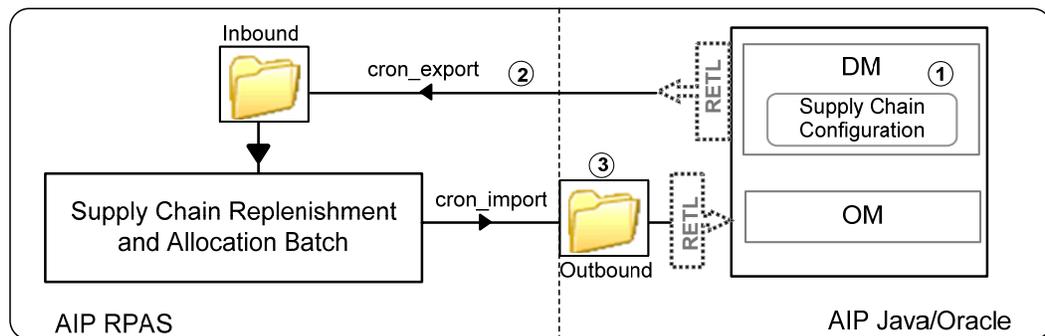
AIP Internal Integration

AIP uses RETL integration tool to integrate the AIP modules that are spread across both the platforms. RETL is used because it provides the flexibility to handle large amounts of data flow across the RPAS database and the AIP Online (Oracle) platforms.

1. Supply Chain Configuration:

   The AIP Planner uses the DM Online application to initially set up and maintain the supply chain by establishing the connectivity between the stores, warehouses, and suppliers. This is done after the initial batch of AIP is run. The DM Online application is also used for other configurations that are used during the AIP batch execution.

   For example, the retailer introduces new warehouses and stores in RMS. They need to be configured in the DM Online application to establish the sources of the warehouses and stores as well as the many other configurations that the AIP solution needs to know prior to the AIP replenishment planning calculations.

   > **Note:** The RMS information is passed into DM Online during the AIP Batch execution (cron_import).

2. The RETL extract scripts are executed during the batch process to move the DM Online parameters from the AIP Oracle database to an outbound staging directory. The job scheduler will FTP or copy the files in the directory to the inbound directory of the AIP RPAS platform.

3. The store receipt plan and the warehouse receipt plan are exported out of the AIP RPAS platform for transfer by a client's job scheduling process to the Order Management module in the Java/Oracle platform. Also, RMS data required in DM on the Java/Oracle platform are also exported out of the RPAS platform for transfer to the Oracle database.

Internal Integration Architecture

The internal integration allows the two AIP databases to communicate data that must be shared by both. The interface into the Oracle database uses the following technologies:

- Oracle Retail Extract, Transform, and Load (RETL) scripts
- Oracle PL/SQL packages
- Shell scripts
- Scheduler

The interface to the RPAS database uses the following technologies:

- Shell scripts
- RPAS utilities
- Scheduler

The diagram below shows how these components fit together.
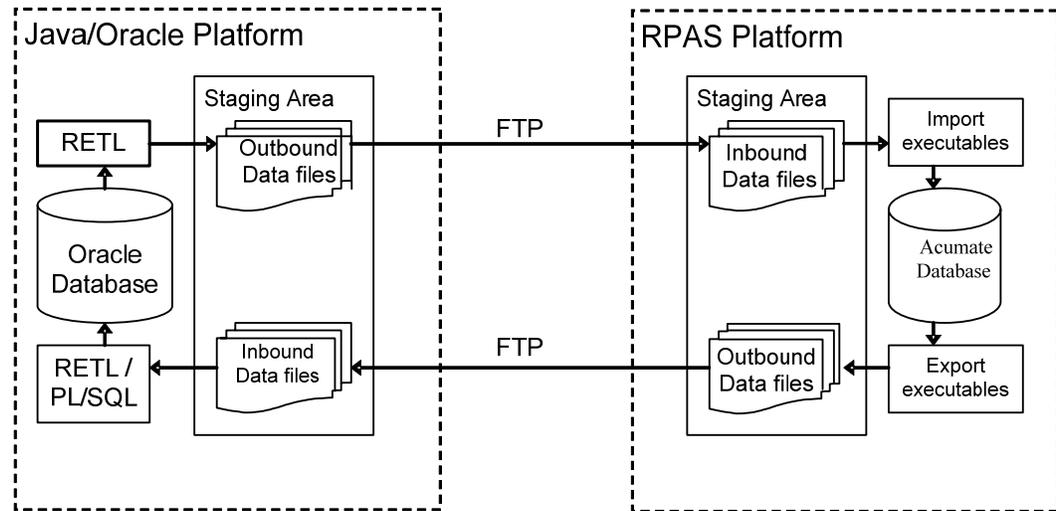
AIP Interfaces

The data is communicated between the databases on a nightly basis. Once data is exported the scheduler calls the retailer-created job to transfer the outbound files to the receiving platform/directory. Then, another scheduled job will execute the shell scripts that trigger the import processing of the files.

Exporting

The export process involves running the export RETL scripts to create the outbound data files. The process compresses them into two files, and then places the files in a staging area where they can be moved to the RPAS platform, via file copy or FTP, by a job scheduling application. RETL can export the data directly from the Oracle tables, with no additional processing.
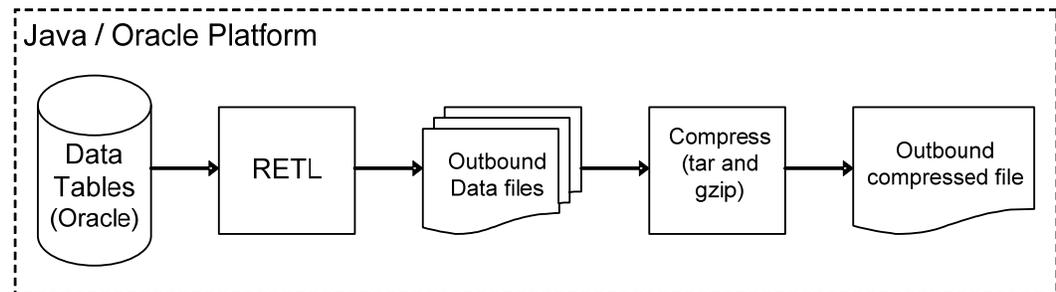
Exporting from the Oracle Database

Importing

The import process is slightly more complex than the export process. The data files must be moved via file copy or FTP by a job scheduling application from the RPAS server location. The files are then decompressed before they are processed through RETL. Due to the required additional pre-insert processing, RETL will not be able to perform a direct import to the database. For instance, duplicate entries must be checked so that updates are performed, rather than inserted. This introduces another layer of processing in the form of PL/SQL packages embedded in the database. Data is first imported to the staging tables by RETL, and then the PL/SQL logic is executed to update the master tables.
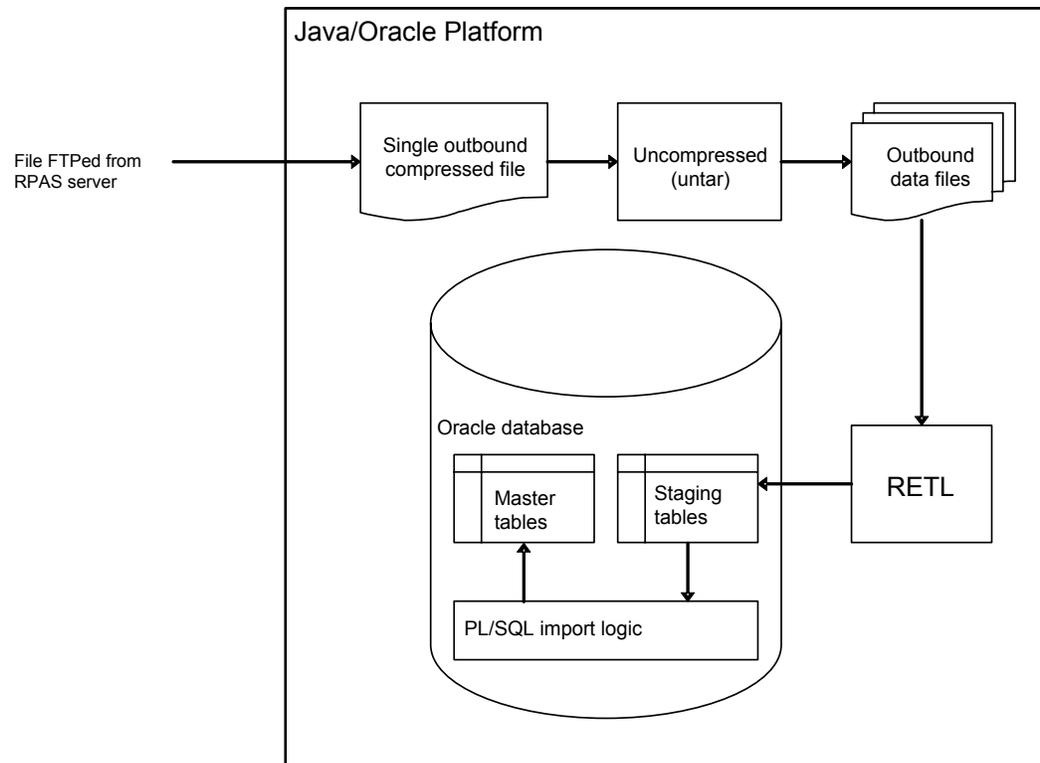
Importing into the Oracle Database

RETL Configuration

The interface process is designed to be fully automated once configured, with support from a retailer's custom process for moving the files between the AIP Oracle and AIP RPAS server locations. The config.xml needs to be configured to the specific environment. This file is located under the root integration directory (integration/).

> **Note:** Refer to the *AIP Installation Guide* for installation instructions. Refer to the *AIP Implementation Guide* for configuration details.

3

AIP Daily Process

The AIP product suite is designed with an online user interface and a batch process. In the online interfaces, the planners log in to the workbooks, DM Online, or OM online, and then set up the parameters for replenishment across the stores and warehouses. The period of time during the day that the planner can access the online interfaces is called the "online day." At the end of the business day, the user interfaces are taken offline and the batch process execution starts. The batch process performs all the replenishment plan calculations and generates alerts before the next online business day begins.

AIP Online Day

During the online day, AIP users log in to the SRP, WRP, DM, and OM client interfaces to maintain all the parameters and supply chain information for the replenishment and allocation calculations across the stores and warehouses.
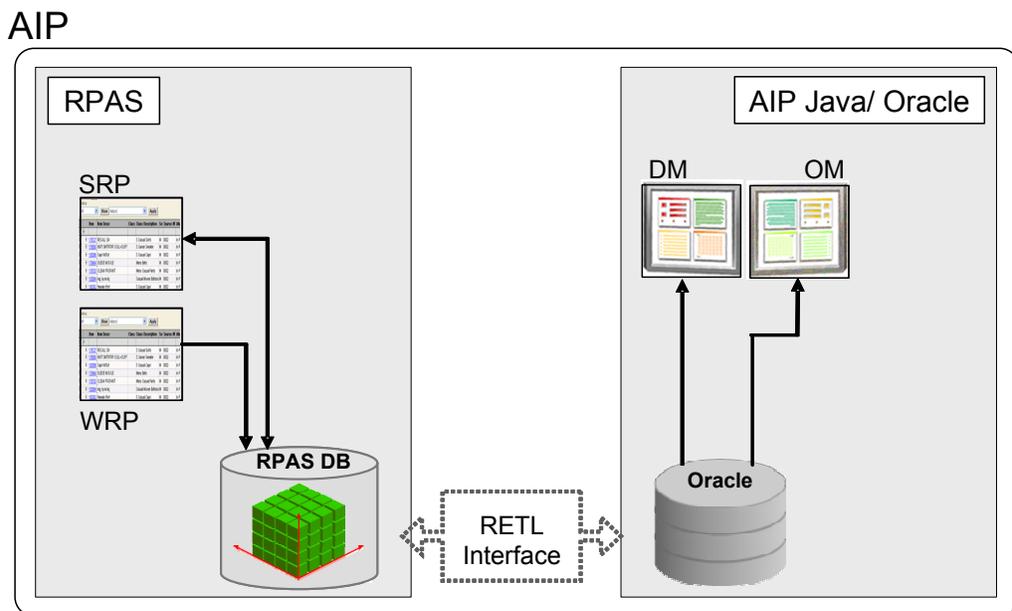
AIP Online Process

SRP: Store Replenishment Planning

WRP: Warehouse Replenishment Planning

OM: Order Management

DM: Data Management

RETL: Retail Extract, Transform, and Load

The SRP and WRP users log in to the respective workbooks to maintain the parameters for the calculations, and also to resolve the alerts that were raised during the previous nightly batch process.

Data Management users log in to maintain the supply chain across the warehouses and stores. Order Management users log in to work on the orders created during the nightly batch process. Any modifications to the orders are immediately communicated to RMS.

Once these online users are done with their activities, they can submit all the planning parameters to the Acumate or Oracle database. Later, during the batch execution process, this data is imported for the replenishment and allocation calculations.

AIP Batch Process

The AIP batch processing takes two paths of execution. They are as follows:

Initial Batch Run

After AIP is installed, the RPAS and the Java/Oracle databases must be populated with foundation data before the supply chain setup can be completed. A limited run of the batch scripts is used to accomplish this task on both the RPAS platform and the AIP Java/Oracle platform. Foundation data is first populated on the RPAS platform. Then, the data is copied into the Oracle database. Using this data, the DM user logs into the client application to configure the supply chain. The user must also log in to the AIP workbooks to define the replenishment parameters. Once this is done, the AIP system (across both the platforms) is ready for the daily batch run. More information on the 'first day' process can be found in the *AIP Implementation Guide.*

Daily Batch Run

This batch run is executed on a daily basis. Before executing this batch run, confirm the successful execution of the initial batch run and the DM configurations.

The daily batch run is the process that is outlined in this chapter.

After the online day ends, the daily batch process begins. The entire batch process for AIP is defined in three phases:

1. Pre AIP-Batch
2. AIP Batch
3. Post AIP-Batch

Pre AIP batch includes all processes that occur in the integrated Oracle Retail applications (i.e. RMS), as well as those processes that occur in custom integrated systems, for the purpose of providing data to AIP. These processes extract the necessary data that will be loaded into AIP.

AIP batch is executed in a series of scripted steps that occur on each of the two AIP platforms. To begin, the data required to perform AIP replenishment must be available in a loadable format. This involves extracting data from the Oracle database as well as massaging data files from RMS. Once the data has been loaded the replenishment and replenishment releated setup commences. At various points of completion data is moved/extracted from the RPAS database and readied for the Oracle database. This includes data from other systems as well as AIP data created or modified by the AIP on RPAS batch processes. The data is loaded into the Oracle database as it becomes available. Automated supply-chain setup commences in the Oracle database after the merchandise and organizational hierarchies have been loaded. Once replenishment is complete the plan is exported from the RPAS database and loaded into the Oracle database. The plan is passed through the Supplier and Container Scaling modules, when

enabled, and then POs and Tranfers that have met their lead time are released to RMS or an order execution system. Finally, post replenishment activities are performed on the RPAS platform which includes generating receipt plan alerts and automatically building workbooks.

Post AIP batch occurs as soon as the POs and Transfers have completed the AIP release process. The post AIP batch processes must retrieve and load the AIP POs and Transfers that must be executed or communicated to sending and/or receiving entities.
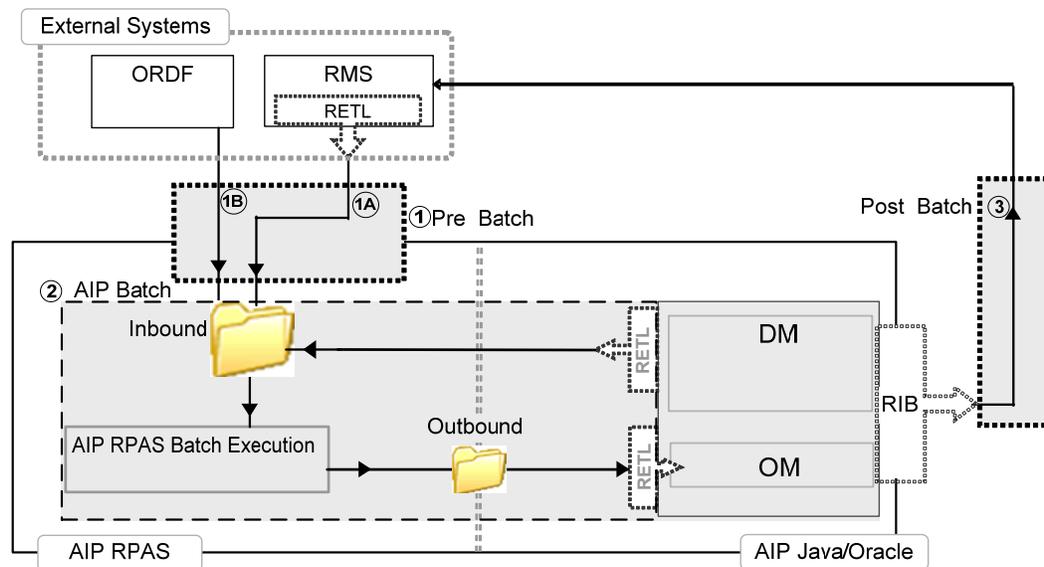
AIP Daily Batch Processing Steps

Pre AIP Batch

In the pre AIP batch process, the data for AIP is extracted from the external system and copied or transferred via FTP into the inbound AIP directory.

The Pre-AIP Batch Process in External Systems

1A. During the RMS batch process, RETL scripts extract data from the Oracle database. The files must be copied or transferred via FTP into the Inbound directory

1B. The RDF data for AIP is extracted and loaded into the common directory. During the RDF batch process, the batch scripts are executed to extract information from RDF and place it in the Inbound folder.

AIP Batch

The AIP batch process is executed on both the AIP RPAS and AIP Java/Oracle platforms.

The AIP Batch Process on the RPAS Platform

1. Transforms the RMS data files into AIP loadable format.

2. Runs the AIP batch scripts for all daily batch steps
  ▪ Formats and loads the RMS data files and other externally generated files from a directory (Inbound folder)
  ▪ Merges and loads the DM online files from a directory

- Loads the forecast data files from a directory
- Runs the replenishment and reconciliation logic
- Exports the AIP RPAS batch output data into a directory (Outbound folder)
- Loads a modified AIP-scaled order plan
- Generates alerts
- Builds SRP and WRP workbooks configured for auto-workbook builds.

The AIP Batch Process on the Java/Oracle Platform

Time wise the AIP batch processes that run on the Oracle platform straddle the AIP processes that run on the RPAS platform. Before any of the replenishment planning calculation preparation can begin in AIP RPAS the data must be extracted out of the AIP Oracle database. The cron_export script is used to accomplish this.

After the replenishment plan has been generated the plan is exported to an outbound data directory in the form of orders (purchase orders and transfers). The planned orders, along with the RMS enterprise data, and any DM data that was created or modified in the AIP RPAS batch processes will be retrieved and loaded by the AIP Oracle batch process. The cron_import and cron_import_order scripts are used to accomplish this.

An alternate approach to importing DM data into AIP Oracle can be used to maximize CPU utilization and shorten the nightly batch window. DM data can be imported into AIP Oracle immediately after it has been exported from AIP RPAS. Specifically, the 'cron_import DM' step can be executed immediately after the 'export_dm_data' step is complete. For detailed information on these steps, please refer to Chapter 6 – AIP Batch Processing. This process improves performance by executing the import of DM data at a time when the AIP Oracle server may otherwise be under little or no load.

After the necessary DM data is imported the cron_import script will also initiate a number of processes that automatically set up the supply chain for new hierarchy data. These processes also automatically default some supply chain data when missing, and replaces some supply chain data when invalid.

The planned warehouse purchase orders, once loaded, are then passed through a module which will pull-forward future POs in order to achieve supplier minimums and/or container minimums on a particular day.

After the necessary minimums have been applied to the POs all planned orders that have met their lead time will be released en masse to RMS for execution. The orders are communicated to RMS by using the RIB integration tool. The cron_release_store_order and cron_release_non_contents_order scripts perform the necessary steps to post orders to the appropriate RIB publication tables. The methods of interfacing purchase orders and transfers are dictated by separate system parameters which are stored in the database. In the event that the RIB is not used, AIP Oracle can extract purchase order and transfer data into text files. The tsf_po_export script is used to accomplish this. However, it is important to note that RMS supports RIB-based purchase order and transfer subscription only.

In addition to regular data load, order load, data export, order export, order scale and order release processes, there is a process to remove closed orders in order to keep database table space available. The Order Purge process is used to purge orders that have been closed for more than their configured purge age (as defined in order_purge_period table). This process should be executed daily considering the volume of orders loaded and processed every day. It however is a non-critical process.

Post AIP Batch

The RIB publication of the AIP purchase orders and transfers to RMS completes the batch processes related to AIP.

Environment Variables

Before any batch scripts are run, the basic environment information must be globally defined for use in all the scripts. Environment variables and implementation parameters need to be set up for both the RPAS and the AIP Java/Oracle platforms.

For information on the environment variables, refer to Chapter 4, "System Configuration" in the *Oracle Retail AIP Implementation Guide*.

For information on the implementation parameters, refer to Chapter 6, "AIP RPAS Configurations" in the *Oracle Retail AIP Implementation Guide*.

4

AIP Batch Script Architecture

The Batch Script Architecture (BSA) is designed to provide a robust, enterprise-ready architecture for parallel process control, restart control, log consolidation, and dependency checks.

Functionally, the batch operability is enhanced by making it easier to diagnose and correct problems. Errors and exceptions are isolated (contained) within the domains in which they occur. Processing is designed to easily restart and proceed with execution. Parallel tasks are managed, logged, and checked for errors, to the same standard as non-parallel tasks.

BSA provides the following features:

- Error handling and logging architecture and their implementation
- Script parallel process management (logging, error control, and waiting) framework

The Batch Script Architecture is a common component architecture that is used by AIP batch scripts for the reusing of code. The BSA has a structured approach of execution.

In addition to modularization and complete separation of the generic (not AIP-specific or dependent) functionality, additional functionality has been added, including the management of access credentials for database connections, and SQL query, move- and copy-with confirmation, and HTML- and XML-based logging. Additional component functionalities that are not independent of AIP or RPAS are similarly separated, but are placed in AIP-specific utility scripts, and not in the Batch Script Architecture.

Batch Architecture Generic Functionality

General Overview

The Batch Script Architecture provides structure and helper functionality for containment and management of shell environment settings, logging, error handling, and parallel control. The scripts include the following:

Script	Description
bsa_common.sh	A convenience script that sources bsa_env.sh, bsa_logger.sh, bsa_verify.sh, and bsa_para.sh for simplified packaging.
bsa_env.sh	Environment settings (paths, etc.) optional redefinitions that can be preset in the environment or in a local environment script to specialize for each installation. Also includes routines to programmatically create, verify, and adjust the environment settings.
bsa_logger.sh	Functionality to perform automatic and manual information and error logging.
bsa_para.sh	Functionality to perform parallel process control, error containment, and logging.
bsa_verify.sh	Utility functions to perform verifications of scripts and binary executions.
bsa_unit.sh	Functionality to standardize and streamline the building of shell script unit tests.
bsa_cred.sh	Utility functions to manage access credentials (Connect string for Oracle database).

Script	Description
bsa_sql.sh	Utility functions to fetch values from, and executed stored procedures/functions on, the Oracle database.

Logger Functionality Overview

The bsa_logger.sh script contains utility API functions to perform basic information and error logging. At its core are the _log_message and _call APIs that perform explicit information and error logging, and the wrapping of the standard out and standard error streams to the log.

In addition to explicit functionality, the logger.sh header script performs the following automatic functions:

Log path setup: A log directory, ${LOGHOME}/<date>, is created. Logs are created in subdirectories and in a folder hierarchy that exactly parallels the script call stack. Log files are named "<script base name>.log". (Later, a summary log may be implemented.) Each log directory name is constructed as follows:

<base script name>.<timestamp>.<sequence number>

The sequence number is important, as some scripts are called repeatedly in a sequence or in parallel by other scripts. Each distinct call gets its own log. For example, if, on December 27, 2005, the main.sh script calls utility_1.sh once and utility_2.sh three times, the following log directories and files result:

- • • • 2005-12-27
    - • • • • main.143719.1
        - • • • • main.log
        - • • • • utility_1.143720.1
        - •   • • • • utility_1.log
        - • • • • utility_2.143721.1
        - •   • • • • utility_2.log
        - • • • • utility_2.143721.2
            - • • • • utility_2.log

Script entry logging: The _log_message API, described below, is automatically called to log the script entry point to the script's log.

Script exit logging: Upon exit, the _log_message API is called to log the script exit to the script's log. Both normal and Ctrl-C and other hard exit-induced exits are logged and distinguished, and a descriptive error code number is included in the log message.

Read from aip_env_rpas.sh or aip_env_online.sh, and create, if absent, the environment variable, LOGLEVEL. This is the minimum level for which log content is generated. Valid values for this environment variable are PROFILE, DEBUG, INFORMATION, WARNING, and ERROR. If this variable is not already set by the environment, logger.sh defaults the value to INFORMATION.

Automatic functionality is implemented by using the public API documented below, as well as the following private APIs:

__internal_log_message

Logs the message, complete with the caller's line number

__internal_log_entry

Calls __internal_log_message to log the entry to the script's log

__internal_log_normal_exit

Called implicitly by the firing of the shell EXIT trap; calls __internal_log_message to log the exit code and message to the script's log

__internal_log_forced_exit

Called implicitly by the firing of the shell INT and CLOSE traps; calls __internal_log_message to log the exit code and message to the script's log

Logger Public API

_log_message <level> <message>

Implemented as an alias that captures the script line number and calls __internal_log_message. The level can be PROFILE, DEBUG, INFORMATION, WARNING, or ERROR. It writes the time stamp, level, script name, line number, and message to the script's log. This renders obsolete most uses of echo… to the log file.

_call <script or binary call with params>

Wraps any script or binary call so that the standard error and standard out are redirected to the script's log file. This renders obsolete most uses of >>.

Verification of Successful Completion

The verify functions provide single-line error handling after calling any script or binary. The three API functions, _verify_script, _verify_binary, and _verify_log ensure the success or report the failure and exit the batch, based on the run of a script, binary, or the identification of an error string in the script's log.

Verify the Public API

_verify_script

This is implemented as an alias that captures the script line number and calls __internal_verify_script. This function must be called immediately after the script to be verified. It checks the return value from the script; if nonzero, it posts an error code specific message to the script's log with level=ERROR, and then exits from the script with the same code. This, along with _verify_binary, renders obsolete most processing functions of $?.

_verify_binary <custom error message>

This is implemented as an alias that captures the script line number and calls __internal_verify_binary. This function must be called immediately after the binary to be verified. It checks the return value from the binary; if nonzero, it posts the passed custom error message to the script's log with level=ERROR, and then exits from the script with a standard error code (not the binary exit code). The standard error code is returned so that the calling script can automatically report the error by use of _verify_script. This, along with _verify_script, renders obsolete most processing functions of $?.

_verify_log

This is implemented as an alias that captures the script line number and calls __internal_verify_log. This function performs grep-based checking of the script's log for an easily configured list of strings, including "error," "exception," and "failure." If a configured string is found, a log entry is generated with level=ERROR, indicating the error found; and then a non-zero exit is performed to immediately halt the script. This renders obsolete most functions in the explicit grepping of log files.

> **Note:** In order for _verify_script to work correctly, all the scripts should exit ONLY with predefined exit codes. Under no circumstances should exit be called without a code, nor should any return be called.

The codes defined for use as script exits are as follows:

__CODE_SUCCESS
__CODE_NONZERO_EXIT
__CODE_FORCED_EXIT
__CODE_FILE_NOT_FOUND
__CODE_NONEXISTENT_LOG_FILE
__CODE_NONEXISTENT_ERROR_FILE
__CODE_TOO_FEW_ARGS
__CODE_TOO_MANY_ARGS
__CODE_UNSPECIFIED_ERROR
__CODE_GENERAL_SCRIPT_ERROR
__CODE_UNDEFINED_ERROR_LEVEL
__CODE_MACE_ERROR_LEVEL
__CODE_INVALID_PATH
__CODE_INVALID_DOMAIN_PATH
__CODE_PARALLEL_ERROR
__CODE_UNSUPPORTED_FUNCTIONALITY
__CODE_UNSUPPORTED_DATA_SET_KEY
__CODE_FIRST_LOG_ERROR
__CODE_LOGGED_ERROR
__CODE_LOGGED_EXCEPTION
__CODE_LOGGED_FATAL
__CODE_LOGGED_ABORT
__CODE_LAST_LOG_ERROR

Overview of the Parallelization Functionality

A framework for the parallelization of script and binary calls is introduced to replace the use of the "naked" and wait commands. The framework has these three components:

BSA_MAX_PARALLEL environment variable

_para_spawn function

_para_wait function

The BSA_MAX_PARALLEL environment variable, found in aip_env_rpas.sh or aip_env_online.sh, should hold a positive integer value. This value is interpreted by the _para_spawn function as the maximum number of processes that can be started in parallel, from any spawning process. This value is NOT the maximum total number of running processes, as spawned processes may further spawn more processes.

The _para_spawn function is a replacement for invoking a script or binary with a trailing '&'. _para_spawn <command> starts command in parallel, as soon as the number of processes spawned by the spawning process falls below BSA_MAX_PARALLEL. Scripts spawned with _para_spawn, log messages just like the non-parallel-called scripts.

_para_spawn must be used only in conjunction with _para_wait. See the API definition below.

The _para_wait function is a replacement for the wait command, and it should always be used to wait for processes spawned with _para_spawn. _para_wait handles the detailed reporting of spawned process failures and returns; either __CODE_SUCCESS (0) if all the parallel processes succeed, or __CODE_PARALLEL_ERROR if any spawned process fails. _para_wait must be used only in conjunction with _para_spawn. See the API definition below.

Para API

_para_spawn <script or binary call with params>

This function wraps the calling of a script or binary, in parallel. This renders obsolete the use of standard parallel calling "&". Scripts or binaries called using this function are started only if the number of processes that have already been spawned from the same calling script and still running do not exceed the BSA_MAX_PARALLEL environment variable. If the number of already spawned and still running jobs equals BSA_MAX_PARALLEL, then, this function waits until at least one of the jobs completes, before spawning the requested job. _para_spawn must be used in conjunction with _para_wait. _para_spawn should not be called with the _call function, although the use of _call within a script spawned by _para_spawn is acceptable.

_para_wait

This function wraps the wait function, and should be used to wait for jobs spawned using _para_spawn. It returns 0 if all the jobs spawned with _para_spawn return 0, or nonzero (actually, the resolved value of __CODE_PARALLEL _ERROR), otherwise.

SQL Query Wrapping

New API functions, _sqlplus and _sqlplus_fetch, are used to simplify the database access through batch scripts. Both functions use the BSA credential file to obtain the schema access. The credential file application can be sent to the functions. Both functions use DEFAULT_BSA_SQL_CRED_APP for credential lookup if an application is not passed.

The _sqlplus function can be sent a string of PL/SQL commands, including calls to stored procedures and functions. The function propagates connection errors, syntax errors, unhandled database errors, and handled PL/SQL logic errors. The _sqlplus function commits automatically.

The _sqlplus_fetch function can be sent a SQL query. It propagates connection errors, syntax errors, and unhandled database errors.

File Copy and Move with Verification

Simple wrappers of cp and mv are implemented, which confirm the existence of source and destination files before and after copy or move, log the errors accordingly, and cause the script to exit on any error.

BSA Setup

Prior to the sourcing of the BSA common architecture scripts, a batch script or its caller must specialize the BSA environment settings as necessary (for example, to set LOGHOME to a local, writeable directory). The BSA architecture is designed to set the environment settings only if they are not preset by the specialized environment. In other words, set up the application environment, and then source the BSA architecture scripts, which fine tune and default any necessary settings that do not already exist.

Specialized application environment setup is best done by setting up a common script that is sourced by all the specialized application environment scripts. The common script should initialize the specialized environment settings, and then source bsa_common.sh to complete the environment setup. Each specialized environment script should normally cache any input parameters (preferably saved to a local, descriptively-named script variable), and then source the common application script.

AIP Interfaces and Transformation Scripts

The AIP system interfaces with the merchandising system, the forecasting system, and any other external system that provides data in the predetermined file format. This chapter provides information on how AIP interfaces with RMS version 11+ or RMS version 10, and RDF. For a complete list of supported RMS versions, see the *AIP Installation Guide*.

RMS to AIP Interfaces and Transformation Scripts

During AIP installation, the RMS-AIP transform scripts and output schema files were installed to a directory specified by the user, as the "Dir to store RMS transform files". This location is referred to as $RMS_AIP_TRANSFORM for this section. (Refer to the *AIP Installation Guide* for more information about the installation details.)

The AIP installation includes a separate set of transformation scripts for interfacing with either RMS 10 or RMS 11+. These were installed to $RMS_AIP_TRANSFORM/k_shell_scripts. The schema files in $RMS_AIP_TRANSFORM/aip_schema_dir are used for interfacing with any version of RMS. The following documentation describes the differences in the interfacing process for both RMS 10 and RMS 11+.

The overall process of interfacing RMS with AIP begins with generation of the RMS RETL extracts. A separate set of RMS extract scripts and schema files control this generation. These scripts and schema are not included in the AIP installation.

After generating the RMS RETL extracts, the RMS-AIP transformation scripts convert the RMS RETL extracts into RPAS loadable format. There are two sets of schema files: one which matches the formats of the RMS RETL extract files, and a second which matches the formats of the RPAS-loadable files.

Transform Interface Scripts

Location: $RMS_AIP_TRANSFORM/k_shell_scripts *and* $RPAS_HOME/bin

The Transform Interface Scripts are the Korn shell scripts used to drive the process for transforming the RMS RETL extract files into AIP RPAS domain-loadable files.

The $RMS_AIP_TRANSFORM/k_shell_scripts directory's contents should have been copied to $RPAS_HOME/bin during installation. For RMS 11+, $RPAS_HOME/bin must contain the following scripts:

aip_t_master.ksh

aipt_item.ksh

aipt_orghier.ksh

aipt_sub_item.ksh

aipt_str_prd_life.ksh

aipt_future_delivery.ksh

aipt_rename.ksh

aipt_delete_input_files.ksh

aipt_move.ksh

For RMS 10, $RPAS_HOME/bin must contain the following scripts:

aip_t_master_rms10.ksh

aipt_item_rms10.ksh

aipt_orghier_rms10.ksh

aipt_sub_item_rms10.ksh

aipt_str_prd_life_rms10.ksh

aipt_future_delivery_rms10.ksh

aipt_rename_rms10.ksh

aipt_delete_input_files_rms10.ksh

aipt_move_rms10.ksh

The execution of these scripts is controlled by running the master script, aip_t_master.ksh for RMS 11+ transforms and aip_t_master_rms10.ksh for RMS 10 transforms. These scripts are run before the AIP RPAS batch is run.

Input Schema Files

Location: The schema files matching the RMS RETL extracts are provided by the RMS installation. These files are owned by RMS and are not included with the AIP Package.

Input schema files are the schema files which are required by the Transform Interface scripts to read the RETL extracts from RMS.

These files are used as output schema by the RMS RETL extract scripts.

For RMS 11+, the $RMS_ SCHEMA_DIR directory (see aip_env_rpas.sh, in "System Configuration" in the *AIP Implementation Guide*) must contain the following input schema files from the RMS installation:

rmse_aip_item_master.schema

rmse_aip_item_retail.schema

rmse_aip_item_supp_country.schema

rmse_aip_merchhier.schema

rmse_aip_purged_item.schema

rmse_aip_dmx_bndprdasc.schema

rmse_aip_store.schema

rmse_aip_orghier.schema

rmse_aip_substitute_items.schema

rmse_aip_item_loc_traits.schema

rmse_aip_future_delivery_order.schema

rmse_aip_future_delivery_tsf.schema

rmse_aip_tsf_in_well.schema

rmse_aip_future_delivery_alloc.schema

rmse_aip_alloc_in_well.schema

rmse_aip_wh_dat.schema

rmse_aip_wh.schema

For RMS 10, the $RMS10_SCHEMA_DIR directory (see aip_env_rpas.sh, in "System Configuration" in the *AIP Implementation Guide*) must contain the following input schema files from the RMS installation:

rmse_aip_item_master.schema

rmse_item_retail.schema

rmse_item_supp_country.schema

rmse_merchhier.schema

rmse_purged_item.schema

rmse_dmx_bndprdasc.schema

rmse_store.schema

rmse_orghier.schema

rmse_substitute_items.schema

rmse_item_loc_traits.schema

rmse_future_delivery_order.schema

rmse_future_delivery_tsf.schema

rmse_tsf_in_well.schema

rmse_future_delivery_alloc.schema

rmse_alloc_in_well.schema

rmse_wh.schema

rmse_whse1.schema

Output Schema Files

Location: $RMS_AIP_TRANSFORM/aip_schema_dir

Output Schema files are the schema files which are required by the Transform Interface scripts to write the AIP loads/feeds. These files are owned by AIP and should have been unpacked to $RMS_AIP_TRANSFORM/aip_schema_dir during AIP installation.

These output schema files are the same regardless of which version of RMS is interfaced with AIP.

The $AIP_SCHEMA_DIR (see aip_env_rpas.sh, in "System Configuration" in the *AIP Implementation Guide*) directory must contain the following output schema files from the AIP installation:

aipt_item.schema

dmx_dscdt_.schema

aipt_loc.schema

aipt_dm0_pmsstasrc.schema

aipt_dm0_pmsendsrc.schema

aipt_dmx_vadprdasc.schema

aipt_str_prd_life.schema

aipt_sr0_it_.schema

aipt_sr0_oo_.schema

aipt_wr1_it_.schema

aipt_wr1_oo_.schema

aipt_wr1_ow_.schema

RETL Extracts (Data Files from RMS)

Location: RMS server

The RETL Extracts from RMS are used as AIP loads/feeds. There are four categories of RETL Extracts: one, the set which requires transformation before being loaded into AIP RPAS; second, the set of extracts which does not require any transformation before being loaded into AIP RPAS; third, the set of extracts which is not used by AIP RPAS but is used by AIP Oracle; fourth, the set of extracts which is not used by either AIP RPAS or AIP Oracle. All RMS RETL extracts will be deposited to a location on the RMS server. The four sets of files must be copied from the RMS server to the AIP server, into the $RAW_RMS_DATA_DIR directory.

Following is the list of RETL Extracts that will be used by the RMS 11+-AIP Interface Transform scripts to generate a new set of AIP RPAS loads/feeds.

RMS 11+ Group 1 – Needed by AIP RPAS domain: RMS RETL Extracts which are transformed:

rmse_aip_item_master.dat

rmse_aip_item_retail.dat

rmse_aip_item_supp_country.dat

rmse_aip_merchhier.dat

rmse_aip_purged_item.dat

dmx_bndprdasc.txt

rmse_aip_store.dat

rmse_aip_orghier.dat

rmse_aip_substitute_items.dat

rmse_aip_item_loc_traits.dat

rmse_aip_future_delivery_order.dat

rmse_aip_future_delivery_tsf.dat

rmse_aip_tsf_in_well.dat

rmse_aip_future_delivery_alloc.dat

rmse_aip_alloc_in_well.dat

rmse_aip_wh.dat

rmse_aip_wh.txt

rmse_aip_wh_type.txt

> Note: dmx_bndprdasc.txt is used by the transformation process, but is also a direct feed into AIP RPAS.

In addition, the following are the RETL extracts that do not need to be transformed.

RMS 11+ Group 2 – Needed by AIP RPAS domain: RMS RETL Extracts which are not transformed:

splr.txt

dmx_dirspl.txt

dmx_prdspllks.txt

sr0_curinv_[*1…n*].txt

wr1_curinv.txt

> **Note:** As shown above, the sr0_curinv data feed may be
> partitioned. For example: sr0_curinv_1.txt, sr0_curinv_2.txt,
> etc. The data will always have at least one partition, namely
> sr0_curinv_1.txt. AIP RPAS batch steps will combine any
> partitions before loading the data into the AIP domain.

Third, there are several RMS RETL extracts which are not used by AIP RPAS batch but
are used by AIP Oracle batch.

RMS 11+ Group 3 – Needed by AIP Oracle DB: RMS RETL Extract which are not transformed:

    closed_order.txt
    received_qty.txt

Finally, there are several RMS RETL extracts which at this time are not used by AIP.

RMS 11+ Group 4 – Not used by AIP:

    dm0_onseffdt_.txt
    dm0_ofseffdt_.txt
    rmse_aip_item_supp_country_reject_ord_mult.txt
    rmse_aip_suppliers.dat

> **Note:** dm0_onseffdt_.txt and dm0_ofseffdt_.txt are produced
> by RMS but are not usable by AIP. A custom transformation
> script is required in order for AIP to load these files. At this
> time, AIP currently expects to load these files instead from a
> separate external system which produces AIP loadable file
> formats.

Following is the list of RETL Extracts that will be used by the RMS 10-AIP Interface
Transform scripts to generate a new set of AIP RPAS loads/feeds.

RMS 10 Group 1 – Needed by AIP RPAS domain: RMS 10 RETL Extracts which are transformed:

    rmse_aip_item_master.dat
    rmse_item_retail.dat
    rmse_item_supp_country.dat
    rmse_merchhier.dat
    rmse_purged_item.dat
    dmx_bndprdasc.txt
    rmse_store.dat
    rmse_orghier.dat
    rmse_substitute_items.dat
    rmse_item_loc_traits.dat
    rmse_future_delivery_order.dat
    rmse_future_delivery_tsf.dat
    rmse_tsf_in_well.dat
    rmse_future_delivery_alloc.dat
    rmse_alloc_in_well.dat
    rmse_wh.dat
    rmse_aip_wh.txt

rmse_wh_type.txt

> **Note:** dmx_bndprdasc.txt is used by the transformation
> process, but is also a direct feed into AIP RPAS.

In addition, the following are the RETL extracts that do not need to be transformed.

RMS 10 Group 2 – Needed by AIP RPAS domain: RMS 10 RETL Extracts which are not transformed:

splr.txt

dmx_dirspl.txt

dmx_prdspllks.txt

sr0_curinv_[*1…n*].txt

wr1_curinv.txt

> **Note:** As shown above, the sr0_curinv data feed may be
> partitioned. For example: sr0_curinv_1.txt, sr0_curinv_2.txt,
> etc. The data will always have at least one partition, namely
> sr0_curinv_1.txt. AIP RPAS batch steps will combine any
> partitions before loading the data into the AIP domain.

Third, there are several RMS 10 RETL extracts which are not used by AIP RPAS batch but
are used by AIP Oracle batch.

RMS 10 Group 3 – Needed by AIP Oracle DB: RMS 10 RETL Extract which are not transformed:

closed_order.txt

received_qty.txt

Finally, there are several RMS 10 RETL extracts which at this time are not used by AIP.

RMS 10 Group 4 – Not used by AIP:

dm0_onseffdt_.txt

dm0_ofseffdt_.txt

rmse_aip_item_supp_country_reject_ord_mult.txt

rmse_aip_suppliers.dat

> **Note:** dm0_onseffdt_.txt and dm0_ofseffdt_.txt are produced
> by RMS but are not usable by AIP. A custom transformation
> script is required in order for AIP to load these files. At this
> time, AIP currently expects to load these files instead from a
> separate external system which produces AIP loadable file
> formats.

Running the Transform Scripts

File Transformation and Transfer (AIP RPAS)

1. Copy Group 1 and Group 2 extracts from the RMS server to the
   $RAW_RMS_DATA_DIR directory on the AIP server. This process is not part of the
   Transformation process or AIP batch. It must be scheduled by the client.
2. For interfacing with RMS 10, copy the RMS 10 RETL schema files from the RMS 10
   installation to the directory specified by $RMS10_SCHEMA_DIR. For interfacing
   with RMS 11+, copy the RMS 11 RETL schema files from the RMS 11 installation to

the directory specified by $RMS_SCHEMA_DIR. This needs to be done only once per AIP install / patch release.

3. Copy the $RMS_AIP_TRANSFORM/aip_schema_dir files from the AIP installation into the $AIP_SCHEMA_DIR. This needs to be done only once per AIP install / patch release.

4. When extracts are from RMS 11+, execute aip_t_master.ksh. When extracts are from RMS 10, execute aip_t_master_rms10.ksh. The location of both scripts is $RPAS_HOME/bin. Either script will execute the appropriate RMS-AIP Interface Transform scripts. Once this script has successfully completed, the following new AIP loads/feeds are created:

- item.txt
- dmx_dscdt_.txt
- loc.txt
- dm0_pmsstasrc.txt
- dm0_pmsendsrc.txt
- dmx_vadprdasc.txt
- sr0_prdlfe.txt
- sr0_it_.txt
- sr0_oo_.txt
- wr1_it_.txt
- wr1_oo_.txt
- wr1_ow_.txt
- whse.txt
- wh_type.txt

After the transforms scripts create the new files, there is logic in the scripts (see aipt_move.ksh, aipt_move_rms10.ksh) to move all files required by AIP RPAS into $AIPDOMAIN/interface/rms, a location defined by $INTERFACE_RMS_DIR in aip_constants_rpas.sh. This is the location where the files are processed by AIP RPAS batch. In addition, there is logic in the scripts (see aipt_delete_input_files.ksh, aipt_delete_input_files_rms10.ksh) to remove the raw RMS data from the $RAW_RMS_DATA_DIR directory which is not needed by the AIP RPAS domain.

File Transfer for Order Management (AIP Oracle Database)

The client must schedule a batch job to copy Group 3 extracts from the RMS server to the AIP Oracle server location defined by $ONL_INBOUND_DIR in the aip_env_online.sh file. The suffixes for each of the two files (closed_order.txt and received_qty.txt) must be renamed to ".dat", to create closed_order.dat and received_qty.dat.

The cron_import.sh script will not FTP or copy these two data files to the DM Online server, nor will it uncompress, or untar any package (for example, om.tar.Z) containing these files. Clients can still use the om.tar.Z file; however, they are required to tar, compress, ftp, uncompress, and untar by using a batch scheduler.

> **Note:** AIP also accepts the OM file rmse_order_purge.txt in this location; however, this file is not critical to AIP—it can be configured as optional—and must be generated from a custom RMS script.

At this point, the RMS-AIP transformation is complete.

RDF to AIP Interfaces

AIP does not provide transformation scripts to format data files from RDF. The data files are expected to arrive in an AIP loadable format. AIP does not provide any script to transfer the RDF data into AIP. You need to have a job scheduled to transfer the files from the RDF server to the AIP RPAS server.

This batch job must copy or FTP the forecast data from the RDF server location to the AIP location, $AIPDOMAIN/interface/forecast.

RDF Extracts

The table below displays the forecast data files that are loaded from the forecasting system into AIP.

Input File Name	Description
iprdfdtdaltv.txt	WRP RDF Alert
sr0_rdfdtdmsk.txt	RDF Detailed Alert Masks
sr0_fcterrlvl1.txt	Store Weekly Demand Forecast Error
sr0_fcterrlvl2.txt	Store Weekly Demand Forecast Error
sr0_dayslsld.txt	RDF Daily Sales
sr0_rdfdtdcnt.txt	RDF Detail Alert Count
sr0_frclvl1_*.txt	Store Approved RDF Forecast Level 1
sr0_frclvl2_*.txt	Store Approved RDF Forecast Level 2

Once the files are loaded onto the AIP RPAS platform, the AIP RPAS batch script processes the files. The check_load_forecast_data.sh wrapper script is executed to verify that all required files are present before proceeding to load the data into the AIP measures.

AIP to RMS Interface

The AIP to RMS process flow appears below.
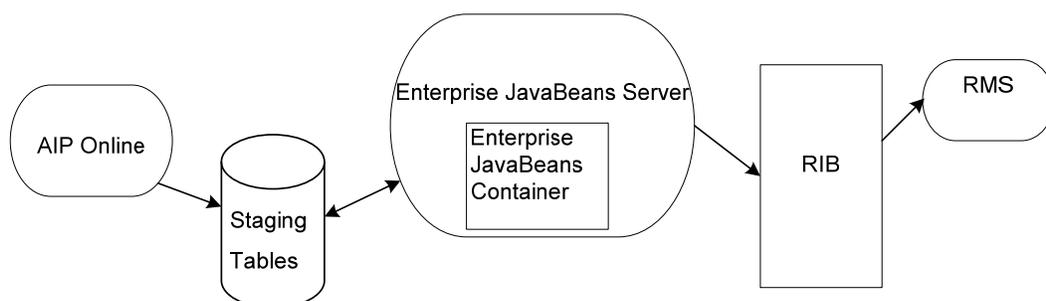
AIP to RMS Process Flow

The AIP Order Management releases data into the staging tables.

- The polling thread runs EJB at certain time intervals.
- EJB's "stateless Session Beans" polls data from the staging table and publishes messages to RIB.
- RMS uses RIB to receive messages published by AIP.

Message Types

There are five types of messages:

1. Create Purchase Order   "XORDERCRE"
2. Create Purchase Order Detail   "XORDERDTLCRE"
3. Modify Purchase Order   "XORDERMOD"
4. Modify Purchase Order Detail   "XORDERDTLMOD"
5. Create Transfer   "XTSFCRE"

RIB Order Publication

The Oracle Retail Integration Bus (RIB) is a near real-time data synchronization solution used by AIP for publishing orders to RMS. Order publication begins with the order release batch adding the affected order to the appropriate message family queue staging table, and then marking each message with a sequence number. AIP publishes two sets of order messages to the RIB; Purchase Orders, and Transfers. RMS subscribes to the RIB messages and inserts the orders into the appropriate RMS Purchase Order and Transfer tables.

AIP Message Flow

A polling operation in the servlet triggers the message creation. The polling is performed by two threads:

- One for the PO_MFQUEUE staging table
- One for the TSF_MFQUEUE staging table

The polling is controlled by the configuration settings in the main.properties file.

The order period count defines the number of time intervals that are to be used. An order period count of 0 indicates that no orders will be released. If the order period count is 0, no threads will be started.

The time interval defines the amount of time for which the threads sleep. A thread will not go to sleep until less than the maximum number of allowable messages is processed in a given call to the publisher (OrderSenderBean). Publishing less than the maximum allowable messages indicates that all orders on the staging table (at the time it was queried) have been processed. Any orders added to the staging table afterward, will be processed the next time the thread wakes up and the publisher is invoked.

For each order period count greater than zero, an order period start and order period end must be added to the properties file. When the thread wakes up and the current time falls between the start and end of any of the intervals (up to X intervals, where X is the order period count), the thread will call the publication procedure. If desired, various time intervals can be created to manage the publication of orders by forcing the threads to poll the staging tables only between certain time periods.

The publisher is an Enterprise Java Session Bean named OrderSenderBean. Its checkAndPublish method will query the staging table and the base order tables to get the message details. The publisher will also ensure that the messages are published to the RIB, in the correct order.

Once the message payload is built by the OrderSenderBean, the RIB message publisher takes the payload and wraps it with an envelope used by the RIB infrastructure.

Purchase Order Message

The purchase order publication messages are in the XOrder message family. In AIP, this message family processes the staged orders on the PO_MFQUEUE table.

There are four purchase order message types used by AIP:

XORDERCRE

XORDERDTLCRE

XORDERMOD

XORDERDTLMOD

All four message types use XOrderDesc.dtd.

XORDERCRE

This message type indicates that a brand new purchase order is being sent to RMS. The orders are sent to RMS in an 'A'pproved status. This message type is inserted into PO_MFQUEUE in three different circumstances:

The purchase order was released by the batch, or you chose to release the purchase order in the OM Order Maintenance screen.

You created a new purchase order in the OM Order Create screen.

In the OM Order Maintenance screen, you chose to move a purchase order delivery date and/or destination, and also generated a new order number.

XORDERDTLCRE

This message type indicates that a new line item is being added to the purchase order after the order was externally communicated. This message type is inserted into PO_MFQUEUE when you have moved the purchase order destination and chosen to retain the existing order number, and the destination does not already exist on the order for that item.

XORDERMOD

This message type indicates that a modification was made to the overall purchase order details (header level information). This message type is inserted into PO_MFQUEUE in the following circumstances:

You have moved the purchase order delivery date and chosen to retain the existing order number.

You have canceled all ordered quantities of all items on the purchase order. The total order quantity for the entire purchase order is zero. The purchase order is sent to RMS with a 'C'anceled status.

XORDERDTLMOD

This message type indicates that a modification was made to the purchase order line items after the order was externally communicated. This message type is inserted into PO_MFQUEUE when you perform various actions in the OM Order Maintenance screen:

You modify the order quantity of a purchase order that is not "Closed."

You chose to move a purchase order line item to a new destination, and also retain the order number. If the "move to" destination already exists on the order, a message will be written to the staging table to increase the quantity at the "move to" location.

> **Note:** Only one message can be inserted for the "move to" destination, for a particular purchase order. This will either be an XORDERDTLCRE if the destination is new or XORDERDTLMOD if the SKU is already being delivered to the "move to" destination.

The order quantity of the "move from" destination must be decremented to equal the received quantity. A message will be staged for the "move from" destination.

Transfer Message

The transfer publication messages are in the XTsf message family. In AIP, this message family processes the staged orders on the TSF_MFQUEUE table.

There is one transfer message type used by AIP. It uses XTsfDesc.dtd.

XTSFCRE

This message type indicates that a brand new transfer is being sent to RMS. The transfers are sent to RMS in an 'A'pproved status. This message type is inserted into TSF_MFQUEUE when the transfer is released by the batch.

Contingency Order Processing

The contingency orders in OM give the planners a view of the expected future orders and also serve as backup in the event that new orders cannot be produced from AIP replenishment batch. Since the contingency orders are yesterday's view of "today's" orders they are very close to the orders that would have likely been produced if "today's" run of the replenishment batch had completed.

Contingency orders may be available to be released to the RMS in the event that AIP batch cannot produce an order plan, or the data imports into AIP Oracle cannot be completed. The contingency orders, sometimes referred to as "yesterday's (forecast) orders," are orders that were generated as forecast orders during the previous batch run. During a normal batch run, yesterday's forecast orders that had not met their release lead time, and had not been released early, are re-planned to ensure the order quantities reflect the latest forecast and inventory information. At the end of the AIP RPAS batch run, the new orders to be release today along with some of the new forecast orders are exported from AIP RPAS to AIP Oracle. During the AIP RPAS replenishment batch calculation of the new order plan, AIP Oracle (OM) still has yesterday's forecast orders in the database. These orders remain in the Oracle database until the import scripts have been run to load the new orders from AIP RPAS.

It is important to understand that when following the contingency order steps that the contingency orders are a replacement for the orders that would normally be produced in the replenishment batch run for release today. When the batch run is complete the orders must not be released or executed.

Purchase Order and Transfer Files

The table below provides a list of the related Purchase Order and Transfer files and what orders are contained in each file.

Store Orders	
strsplrord.dat	This file contains store Purchase Orders to be released today. It also contains all Purchase Orders forecasted for the entire planning horizon. The release date in the file indicates whether the order is to be released today, or whether it is a forecast order. Purchase Orders with a release date greater than today are forecast orders. This file is loaded into the store_order table in the Oracle database.
strwhord.dat	This file contains store Transfers to be released today. The release date in the file will equal today. This file is loaded into the store_order table in the Oracle database.
strsplrord.dat1	This file contains the forecasted store Purchase Orders with an expected release date of tomorrow.
strwhord.dat1	This file contains the forecasted store Transfers with an expected release date of tomorrow.

Warehouse Orders (Non-contents Orders)	
vendor_to_wh_order.dat	This file contains into-warehouse Purchase Orders to be released today. It also contains all Purchase Orders forecasted for the entire planning horizon. This file is loaded into the PURCHASE partition of non_contents_order table in the Oracle database.
wh_to_wh_transfer.dat	This file contains into-warehouse transfers to be released today. It also contains all Transfers forecasted for the entire planning horizon. This file is loaded into the TRANSFER partition of non_contents_order table in the Oracle database.

Contingency Steps

Your operations to perform depend on the point of failure. For your selected failure point, be sure to read and understand all steps before taking action.

> **Note:** The steps listed below to release contingency orders include running cron_release_store_order.sh and cron_release_non_contents_order.sh from cron_release.sh. These are regularly schedule batch scripts that release the orders to be executed. They simply identify ALL into-store or into-warehouse orders in the Oracle database that have met their lead time.
>
> These scripts MUST NOT be run a second time upon successful completion of AIP replenishment batch otherwise the system is likely to double order.

Failure Point 1: No orders imported from AIP RPAS

If AIP RPAS batch fails before completion of send_scrp_measures_to_online.sh no new RPAS order extracts are available for import into AIP Oracle.

Load Contingency Into-Store Transfers

1. In the Oracle schema, navigate to $ONL_INBOUND_DIR.
2. Add an extension, such as today's date, to strwhord.dat in order to prevent the file from being re-loaded into the Oracle table.

   For example, if today's date is 18-Januaray-2015, rename the file to strsplrord.dat.20151801.
3. Rename strwhord.dat1 to strwhord.dat.
4. Navigate to $INTEGRATION_HOME/scripts/import.
5. Execute the ord_imp_store_ord_tsf_in.sh script found in this directory. This will load the contingency store transfers found in the strwhord.dat file.
6. No action is needed to load any other contingency orders. The contingency store purchase orders and contingency warehouse purchase orders and transfers were loaded during the previous run of cron_import_order.sh.

Release Remaining Orders That Have Met Their Lead Time

7. Run cron_release.sh to release today's contingency store purchase orders and transfers.
8. Run cron_release.sh to release today's contingency warehouse purchase orders and transfers.

> **Note:** If failure occurs during send_scrp_measures_to_online.sh, some of the export files may have been generated. If you can verify that some of the files (strsplrord.dat, strwhord.dat, vendor_to_wh_order.dat, wh_to_wh_transfer.dat) have been *completely* extracted successfully, they should be manually moved to the Oracle database and manually loaded. If strwhord.dat is manually loaded, skip steps 1 through 6 above and perform steps 7 and 8.

Failure Point 2: Loading strsplrord.dat

During the run of cron_import_order.sh while loading strsplrord.dat (scripts/import/ord_imp_store_ord_po_in.sh).

If the error is returned by the subscript ord_imp_store_ord_po_in.sh, the contingency purchase orders have already been deleted in order to prepare the database to import the new purchase order plan.

1. Manually delete any orders from the STORE_ORDER table partition PURCHASE that was partially loaded as a result of executing ord_imp_store_ord_po_in.sh. These orders can be identified as the orders that are in status 'U,' release_date = VDATE, order number IS NULL.

2. Navigate to $ONL_INBOUND_DIR. Make a backup of strsplrord.dat1 by renaming it strsplrord.dat1bak. This will be restored at the end of the contingency process.

3. Make a backup of strsplrord.dat by renaming it strsplrord.datbak.

4. Navigate to $ BSA_ARCHIVE_DIR. Locate the srp.tar.Z file archived with yesterday's date. Extract the strsplrord.dat1 file and copy it to $ONL_INBOUND_DIR.

5. Navigate to $ONL_INBOUND_DIR and rename strsplrord.dat1 to strsplrord.dat.

6. Navigate to $INTEGRATION_HOME/scripts/import. Execute: ord_imp_store_ord_po_in.sh.

Release Orders That Have Met Their Lead Time

7. Navigate to $INTEGRATION_HOME and execute cron_release.sh to release store purchase order.

Restore Backup Files

8. Navigate to $ONL_INBOUND_DIR. Remove strsplrord.dat. Rename/move strsplrord.datbak to strsplrord.dat. Rename strsplrord.dat1bak to strsplrord.dat1.

> **Note:** This process only reloaded one days worth of purchase orders. In the event that the Corrupt Data File process is needed in tomorrows batch run no contingency purchase orders will be available.
>
> It is also important to note that the contingency file (.dat1) will contain any forecasted purchase orders that were released early. If any purchase orders that were scheduled for release today were executed early (during the previous Online day), a duplicate order will be released today.

Failure Point 3: Loading strwhord.dat

During the run of cron_import_order.sh while loading strwhord.dat (scripts/import/ord_imp_store_ord_tsf_in.sh).

1. **Manually delete** any orders from STORE_ORDER table partition TRANSFER that was partially loaded as a result of executing ord_imp_store_ord_tsf_in.sh. These orders can be identified as the orders that are in status 'U,' release_date = VDATE, order number IS NULL.

2. Navigate to $ONL_INBOUND_DIR. Make a backup of strwhord.dat1 by renaming it strwhord.dat1bak. This will be restored at the end of the contingency process.

3. Make a backup of strwhord.dat by renaming it strwhord.datbak.

4. Navigate to $ BSA_ARCHIVE_DIR. Locate the srp.tar.Z file archived with yesterday's date. Extract the strwhord.dat1 file and copy it to $ONL_INBOUND_DIR.

5. Navigate to $ONL_INBOUND_DIR and rename strwhord.dat1 to strwhord.dat.

6. Navigate to $INTEGRATION_HOME/scripts/import. Execute ord_imp_store_ord_tsf_in.sh. This will load the contingency transfers produced yesterday.

Release Orders That Have Met Their Lead Time

7. Navigate to $INTEGRATION_HOME and execute cron_release.sh to release store transfer orders.

Restore Backup Files

8. Navigate to $ONL_INBOUND_DIR. Remove strwhord.dat. Rename/move strwhord.datbak to strwhord.dat. Rename strwhord.dat1bak to strwhord.dat1.

Failure Point 4: Loading vendor_to_wh_order.dat

During the run of cron_import_order.sh while loading vendor_to_wh_order.dat (scripts/import/ ord_imp_non_cont_ord_fcst_in.sh)

Corrupt Data File

If the error is not an error returned by the procedure delete_non_contents_order, then the data file is corrupt. The import procedure has not been executed; therefore, contingency orders are still available in the database.

Release Orders That Have Met Their Lead Time

1. Navigate to $INTEGRATION_HOME and execute cron_release.sh to release into-warehouse purchase orders.

Post Load Procedure Error

If the error is returned by import_non_contents_orders_frc, the contingency orders might have been deleted in order to prepare the database to import the new into-warehouse order plan.

If the error was technical in nature and can be fixed and the interface table i_non_contents_order_forecast still has the data, then reruning the procedure import_non_contents_order_frc should be attempted to load the orders.

No contingency file is available in the AIP release. The vendor_to_wh_order.dat file archived in yesterday's srp.tar.Z contains contingency orders; however, it also contains the orders that were released yesterday. The orders which were released yesterday must be deleted from the file before it can be reloaded (including any orders that were released early).

Failure Point 5: Loading wh_to_wh_transfer.dat

During the run of cron_import_order.sh while loading wh_to_wh_order.dat (scripts/import/ ord_imp_non_cont_tsf_in.sh)

Corrupt Data File

If the error is not an error returned by the procedure delete_non_contents_order, then the data file is corrupt. The import procedure has not been executed; therefore, contingency orders are still available in the database.

Release Orders That Have Met Their Lead Time

1. Navigate to $INTEGRATION_HOME and execute cron_release.sh to release into-warehouse purchase orders.

Post Load Procedure Error

If the error is returned by import_non_contents_transfer, the contingency orders might have been deleted in order to prepare the database to import the new into-warehouse order plan.

If the error was technical in nature and can be fixed and the interface table i_non_contents_transfer still has the data, then rerunning the procedure import_non_contents_transfer should be attempted to load the orders.

No contingency file is available in the AIP release. The wh_to_wh_transfer.dat file archived in yesterday's srp.tar.Z contains contingency orders however it also contains the orders that were released yesterday. The orders which were released yesterday must be deleted from the file before it can be reloaded (including any orders that were released early).

AIP RPAS Batch Processing

The complete AIP batch processing is comprised of the following:

- AIP RPAS batch processing
- AIP Java/Oracle batch processing

Because the AIP solution resides on two platforms and needs to exchange information across both the platforms, the batches are executed on both the platforms.

The aip_batch.sh script is provided with the AIP installation, and it is used to run the entire RPAS daily batch process from a UNIX scheduler. The scripts called by aip_batch.sh may have multi-threaded process calls; however, all the scripts called directly by aip_batch.sh are run inline. The top level control script does not run any scripts in parallel. Therefore, it should be used only as a guide for individual script calls/jobs that function similarly, but can be scheduled to run in parallel.

The AIP RPAS Batch Control Script (aip_batch.sh)

The AIP RPAS batch script, aip_batch.sh, accepts a number of arguments that allow more control over what portion of the batch scripts are run. Step names have been defined, which may also be passed into the script as arguments that define exactly which steps and corresponding scripts should be run.

Usage

The table below provides descriptions of the aip_batch.sh arguments.

Argument	Description
-f	First AIP RPAS batch. A predetermined set of steps will be run. All other flags also can be used with this flag to further limit the steps that are run. The predetermined 'skip-steps' that should not be run during the first run AIP RPAS batch (as indicated by the -f) will not be run, regardless of what arguments are passed.
-s	Indicates that a starting step is defined. The step at which the batch should start must follow this flag. This flag can be used with all the other flags; however, only one batch step can follow the flag. This flag MUST be used along with the -e flag and its associated argument.
-e	Indicates that an ending step is defined. The step after which the batch should end must follow this flag. This flag can be used with all the other flags; however, only one batch step can follow the flag. This flag MUST be used along with the -s flag and its associated parameter.

The aip_batch.sh Steps

The following is a list of valid steps that can be used with the -s and -e flags. These steps can also be passed as parameters to the aip_batch.sh script, in the form of a list of steps (not flagged with -s and -e). Each step is described in detail in either the Initial Batch Run section or the Daily Batch Steps section. Steps that are run during both the initial and daily batch runs are detailed in the Daily Batch Run section.

Execution Sequence of the AIP RPAS Batch Scripts

The table below displays the steps in the order that aip_batch.sh execute them.

Step Name	Description of Action	Initial Batch Run	Daily Batch Run
set_implementation_parameters	Sets all the AIP RPAS implementation parameters.	☑	
check_process_external_data	This is a wrapper script for _check_for_required_files.sh and process_external_data.sh. It first verifies the existence of all the required files as specified by the earlyfiles.config file, and then calls process_external_data.sh.	☑	☑
create_empty_default_files	This script creates empty hierarchy files necessary during the first run of the batch program, when the AIP Online data is not available. These empty files are later used for merging with the RMS hierarchy files during the daily batch run.	☑	
prep_onl_data	Prepares the daily export files from AIP Online for loading into the AIP RPAS platform.		☑
merge_hierarchies	This script merges the hierarchy data from AIP Online with the hierarchy data from RMS. Parameter: Pass the DM Domain path.	☑	☑
convert_hierarchies_for_loading	Converts the hierarchies from RMS merged with AIP Online, for loading into the RPAS domains. The optional -s option specifies the source of the hierarchy files; if not present, the default DOMAIN/interface directory is used.	☑	☑
reconfig_domain_partitions	This script is a wrapper script to the RPAS_HOME/bin utility reconfigGlobalDomainPartitions, to be used within AIP. It checks for the existence of specific configuration files that trigger the addition or removal of partitions to the prod hierarchy.	☑	☑
load_all_hierarchies	Loads all the hierarchies in the AIP domain. This script uses environment variables to determine the domain paths.	☑	☑
load_onl_data	Loads the data extracted from the DM Online and OM Online (AIP Oracle) databases.		☑
load_rms_dm_data	Preprocesses and loads all the DM measures that come from RMS.	☑	☑
create_empty_archive_files	This script creates new item alerts for all the items.	☑	

Step Name	Description of Action	Initial Batch Run	Daily Batch Run
create_alerts	Calls load_one_newitem_alert_measure.sh for each new item alert measure, and then saves the hierarchy load files for the next time.	☑	☑
load_non_rms_external	This script looks for *.ovr files from a non-RMS external system, in the directory specified in the input parameter. The files are moved to the $DOMAIN/input directory, where loadmeasure will be run on them.	☑	☑
auto_commit_wkbooks_batch	This script commits all the workbooks (SRP and WRP) on the AIP RPAS platform.		☑
run_partial_dm_batch	Runs the partial Data Management batch to create and assign profiles.	☑	
run_dm_batch	This script calls the scripts for the DM critical and noncritical paths of execution.		☑
export_dm_data	Extracts all the DM hierarchy and measure data required to keep DM RPAS and DM Online in sync.	☑	☑
check_load_forecast_data	This script is basically a wrapper for load_non_rms_files.sh, as it pertains to the loading of the forecast data. It first verifies the existence of all the required files as specified by the forecastdata_from_external.config file. It then calls load_non_rms_files.sh.		☑
purge_low_variability_advance	Purges and advances the "effective date" encoded measures along the time dimension, according to the command xml file, purgeLowVariabilityAdvance.xml.		☑
purge_truncate_history	Purge and Truncate History of Selected Replenishment Measures		☑
copy_sister_data	Copies the sister store and warehouse data by running the command xml files, copySisterStore.xml and copySisterWarehouse.xml.		☑
check_process_inventory_data	This script calls _check_for_required_files.sh and process_inventory_data.sh. It first verifies the existence of all the required files as specified by the latefiles.config file, and then calls process_inventory_data.sh.		☑
load_rms_replenishment_data	Preprocesses and loads all the SRP and WRP measures that come from RMS in the late data feed.		☑

Step Name	Description of Action	Initial Batch Run	Daily Batch Run
run_replenishment	Runs the replenishment and reconciliation actions, across the supply chain.		☑
export_replenishment_data	Creates the replenishment plan extracts at the local domain level.		☑
send_replenishment_to_online	Concatenates .dat files from an AIP domain's local domain output directories into combined export files that are located in the global domains.		☑
post_ocs_import	Loads plan adjustments after Scaling is complete.		☑
run_replenishment_post_ processing	Executes the replenishment post-process by sequentially running the defined rule group.		☑
run_dm_alerts	Executes the 'dmb_master20.sh "${AIPDOMAIN}"'script to create alerts, if any.	☑	☑
export_dm_alerts	Data management alerts are exported to the AIP Online platform.	☑	☑
run_srp_alerts	Executes the generation of SRP alerts by running the defined SRP alert rule group.		☑
run_wrp_item_alerts	Executes the generation of WRP alerts by running the defined WRP alert rule group.		☑
run_wrp_network_alerts	Executes the generation of WRP network alerts by running the defined WRP network alert rule group.		☑
run_replenishment_alerts_post_ processing	Finds generated WRP network alerts and runs post WRP network alerts for global history maintenance.		☑
auto_build_wkbooks_batch	Builds all the SRP and WRP workbooks that have been configured for automatic builds.		☑

Example Script Calls

The following command runs a subset of the aip_batch.sh steps. These are the steps that should be run on the very first run of the AIP RPAS batch. See Initial Batch Run for more details.

```
aip_batch.sh –f
```

The following command runs the aip_batch.sh steps, starting with check_process_external_data, and up to and including load_non_rms_external:

```
aip_batch.sh -s check_process_external_data –e load_non_rms_external
```

The following command will run only the aip_batch.sh step listed which is run_dm_alerts:

```
aip_batch.sh run_dm_alerts
```

The following command will result in the aip_batch.sh running run_dm_alerts, followed by run_srp_alerts and auto_build_wkbooks batch:

```
aip_batch.sh  run_dm_alerts  run_srp_alerts  auto_build_wkbooks_batch
```

7

AIP RPAS Daily Batch Scripts

AIP RPAS Batch Overview

Functionally the AIP batch scripts are broadly classified as mentioned below. Each of the scripts listed below its descriptive step name exists to perform that batch step.

AIP Batch Class	Step Description	Batch Script
Set Implementation Parameters	Set Implementation Parameters for DM and Replenishment	set_implementation_parms.sh
External Integration Data Processing	Verify and Process Foundation Data from External System	check_process_external_data.sh
	Create Empty Hierarchy files	create_empty_default_files.sh
Internal Integration Data Processing	Prepare Data from AIP Online platform	prep_from_aiponline.sh
Process Merchandise System and AIP Online Hierarchies	Merge Hierarchies	merge_hierarchies.sh
	Convert Hierarchies for Loading	convert_hierarchies_for_loading.sh
	Reconfigure AIP Domain Partitions	reconfigAIPDomainPartitions.ksh
Load Hierarchy and Non-Inventory Measure Data into AIP RPAS	Load All Hierarchies	load_all_hierarchies.sh
	Load AIP Online Measure Data	load_online_measures.sh
	Load External Non-Inventory Measure Data	load_rms_dm_measures.sh
Generate and Load New Item Alert Measures	Create Empty Archive Files	create_empty_archive_files.sh
	Generate Batch and Online Alerts	load_all_newitem_alert_measures.sh
Load External System Measure Data into AIP RPAS	Load Non-RMS External Files	load_non_rms_external.sh
Commit Workbooks before Batch Run	Auto Commit Workbooks	workbook_batch.sh COMMIT
Perform and Export Data Management Calculations	Run Initial Load DM Batch	run_partial_dm_batch.sh
	Run DM Batch	dmb_master.sh
	Export DM Data	export_dm_data.sh <true|false>

AIP Batch Class	Step Description	Batch Script
External Integration Forecast Data Processing and Load into AIP RPAS	Check and Load Forecast Data	check_load_forecast_data.sh
Prepare Replenishment Data and Maintain History	Purge and Advance Low-Variability Data	purge_low_variability_advance.sh
	Purge Truncate History	for_each_local_domain.sh -p purge_truncate_history.sh.sh [DOMAIN]
	Copy Sister Stores and Warehouses	for_each_local_domain.sh -p copy_sister_data_local.sh [DOMAIN]
External Integration Inventory Data Processing and Load into AIP RPAS	Verify and Process Inventory Data from External System	check_process_inventory_data.sh
	Load Replenishment Inventory Data	load_rms_replenishment_measures.sh
Calculate and Export Replenishment Plan	Run Replenishment	scrp.sh
	Export Supply Chain Replenishment Data	for_each_local_domain.sh -p export_scrp_inter_meas_local.sh [DOMAIN]
	Package Supply Chain Replenishment Data	send_scrp_measures_to_online.sh
Load the Scaled Replenishment Plan	Run Post Supplier and Container Scaling Import	post_ocs_import.sh
Perform Post-Replenishment Calculations	Run Replenishment Post-processing	for_each_local_domain.sh -p scrp_post_local.sh [DOMAIN]
Calculate and Export Data Management Alert	Run Non-critical Data Management Alerts	dmb_master_alerts.sh <true|false>
	Export DM Alerts	export_dm_alerts.sh <true|false>
Compute Replenishment Alerts	Run SRP Item Alerts	scrp_srp_alerts.sh
	Run WRP Item Alerts	wrp_item_alerts.sh
	Run WRP Network Alerts	wrp_network_alerts.sh
	Run Replenishment Alerts Post Processing	replenishment_alerts_post_processing.sh
Build Workbooks after Batch Run	Auto Build Workbooks	workbook_batch.sh BUILD

Set Implementation Parameters

Set Implementation Parameters for DM and Replenishment

Step Name

set _implementation_parameters

Script Call

`set_implementation_parms.sh`

Functional Overview

This script is called to set configurable parameters for DM and replenishment. All parameters are set in measure data and are initialized by running mace expressions using values defined in the shell script aip_env_rpas.sh.

Technical Details

The script set_implementation_parms.sh calls the scripts set_implementation_parms_dm.sh and set_implementation_parms_aip.sh. Each of the scripts is a simple list of rules to be executed as mace commands on the global $AIPDOMAIN. The values of the measures are taken from the configurable shell script aip_env_rpas.sh.

Day on Day Processing

1. Call set_implementation_parms_aip.sh to execute its list of expressions via mace.
2. Call set_implementation_parms_dm.sh to execute its list of expressions via mace.

This script calls the following scripts:

set_implementation_parms_dm.sh

set_implementation_parms_aip.sh

Appropriate Batch Run (First Time and/or Daily)

First Time or ad hoc if a desired value changes.

Prerequisites

▪ Values for all implementation parameters listed in aip_env_rpas.sh must be edited to meet the business requirements of the client.

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

External Integration Data Processing

To begin, the data must be exported from RMS or another external system. AIP is integrated with an external system by means of this series of scripts. Therefore we refer to this series of scripts as "Integration." The export of this data is not part of the AIP data export scripts; however, AIP is dependant on the data being available before AIP can proceed with data population. The foundation and inventory data from RMS or another external system are loaded into RPAS as part of the AIP data load process.

Verify and Process Foundation Data from External System

Step Name

check_process_external_data

Script Call

check_process_external_data.sh

Optional Parameters

-h: to display script usage

Functional Overview

RMS is an enterprise solution that provides merchandise hierarchy and organizational setup and maintenance. This data, together with non-RMS external system data, becomes the foundation for the AIP supply chain replenishment. The data is processed as a set of flat files that follow an agreed AIP file format. If the client does not use RMS, but rather some other external system, this data is still required in the same format. AIP validates all of the files to ensure that the required data is present, and massages the files to produce the input required for subsequent steps of the AIP RPAS batch. AIP divides the integration data into two categories: early files and late files. The batch schedule for AIP may aim to maximize the available time by moving as much workload off the critical path as possible. For this reason the static data is extracted from RMS, or similar system, before the dynamic transaction data is available. This allows the hierarchies to be loaded into AIP and the DM batch run before the dynamic data is extracted. This step processes the early, static data.

Technical Details

The static hierarchy and measure data are listed in the tables below. For AIP RPAS batch to properly process external system data, all of the flat files must follow a particular format. The flat files should have a *.txt extension.

The first table lists files that are verified and processed inside the check_process_external_data.sh script, which if the client has RMS and has installed the RMS-AIP Transformation Scripts, are outputs from that transformation process. If the client does not have RMS, then these files are still required. These data are loaded into the RPAS domain as either hierarchy or measure data.

Table 1: Early Dynamic and Reference Data

File Name	Explanation
item.txt	Product Hierarchy
loc.txt	Location Hierarchy
splr.txt	Supplier Hierarchy
whse.txt	Warehouse Hierarchy
dm0_pmsendsrc.txt	Store Promotional Substitution End Date for Warehouses
dm0_pmsstasrc.txt	Store Promotional Substitution Start Date for Warehouses
dmx_bndprdasc.txt	Banded Commodity Association
dmx_dirspl.txt	Direct Suppliers
dmx_dscdt_.txt	Corporate Discontinuation Date
dmx_prdspllks.txt	Commodity-Supplier Links
dmx_vadprdasc.txt	Value Added Commodity Association

The next table lists additional files processed by the check_process_external_data.sh script. However, these are not possible outputs from the RMS-AIP Transformation. These data are also loaded into the RPAS domain as either hierarchy or measure data.

Table 2: Load-Ready External Systems Data

File Name	Explanation
had.txt	Advertising Hierarchy
intv.txt	Interval Hierarchy
dm0_ofseffdt_.txt	Off-Sale Effective Date
dm0_onseffdt_.txt	On-Sale Effective Date
dmx_pprsts.txt	Pre-Priced Status
dmx_shpto_.txt	Receiving Supplier / Ship To
ipavgrtslsi.txt	Total Store Average Rate Of Sales
iphldbckqtyi.txt	Hold Back Quantity
ipldssi.txt	Loaded Safety Stock
ipodcmti.txt	Order Commit
iprpltcdi.txt	Replenishment Type Code
iprplstcdi.txt	Replenishment Subtype Code
ipslsi.txt	Historical Weekly Sales
sr0_ad_.txt	Store Ads
sr0_ad_go_.txt	Store Ads Grand Opening
sr0_ad_irt.txt	Store Ads Inserts
sr0_ad_oth.txt	Store Ads Others
sr0_ad_rop.txt	Store Ads Run on Press
sr0_adjsls.txt	Store Adjusted Sales
sr0_avgrosld_.txt	Store Average Weekly Rate of Sale Loaded
sr0_co_.txt	Store Customer Orders
sr0_expwrtoff.txt	Store Expected Write-off
sr0_hstls_.txt	Store Historical Lost Sales
sr0_knowndemand.txt	Store Known Demand
sr0_rplcde.txt	Store Repl Type Code
sr0_rplsubcde.txt	Store Repl Subtype Code
sr0_ss_ld_.txt	Store Loaded Safety Stock
sr0_tdgday.txt	Store Trading Days
srx_prdrpr.txt	SKU Retail Price
ipfctwkprfd.txt	Week to Day Forecast Percentage Default (Un-Normalized)

ipfctwkprfe.txt	Store Week to Day Forecast Percentage Override (Un-Normalized)
ipwhhldcpci.txt	Stocking Point Holding Capacity
srx_poidst.txt	Poisson Distribution Lookup Table
sr0_wkbsf_ld.txt	Loaded Weekly Base Sales Forecast
ipttlhlstki.txt	Total Held Stock
ipcmtmtde.txt	SPQ Commitment Type Exception
ipstrraltg.txt	Store Global Default Receipt to Availability Lead Time
ipstrraltd.txt	Store Default Receipt to Availability Lead Time
ipstrralte.txt	Store Receipt to Availability Lead Time
ipadstai.txt	Store Ad Start Date
ipadendi.txt	Store Ad End Date
sr0_dyscsls.txt	Daily Short Code Sales
sr0_invadj.txt	Inventory Adjustments
sr0_wstadj.txt	Waste Adjustments

The next table lists non-load-ready data that is received from an external (non-RMS) system, which is processed by the check_process_external_data.sh script. This data is prepared for intragration transfer to AIP Online, but some of the data is split to create load-ready measure data that is loaded into the RPAS domain.

Table 3: Non-Load-Ready External Systems Data

File Name	Explanation
default_wh.txt	Default Warehouse. This data file is split into two load-ready data files: ▪ dmx_defwh_.txt / Default Warehouse ▪ dmx_defwh_csc.txt. / Default Warehouse Customer Service Center
direct_store_format_pack_size.txt	Direct Store Format Pack Size
direct_store_pack_size.txt	Direct Store Pack Size
store_format_pack_size.txt	Store Format Pack Size
store_pack_size.txt	Store Pack Size
item_attribute.txt	Item Attribute
item_attribute_type.txt	Item Attribute Type
sister_store.txt	Sister Store Information. This data file is split into two load-ready data files: ▪ dmx_sst.txt / Sister Store ▪ dmx_stropndt_.txt. / Store Open Date

File Name	Explanation
sister_wh.txt	Sister Warehouse. This data file is split into two load-ready data files: ▪ dmx_wh_.txt / Sister Warehouse ▪ dmx_wh_opndt_.txt / Warehouse Open Date
wh_type.txt	Warehouse Type

Day on Day Processing:

1. Verify that all required files have been downloaded. Failure to download any of the required files will result in an error and termination of the batch.

2. Split the sister_store.txt, sister_wh.txt and default_wh.txt files into load-ready data files as described in the tables above.

3. Prefix the following measure data with the stocking point prefixes, using construct_ntier_measuredata.ksh:
   ▪ dm0_pmsstasrc.txt
   ▪ dm0_pmsendsrc.txt
   ▪ dmx_prdspllks.txt
   ▪ dmx_dirspl.txt
   ▪ wh_type.txt

4. Using interutil binary, $AIPDOMAIN/interface/config/rms/hier/item.config, and the $AIPDOMAIN/interface/config/rms/hier/prod.format file created during RPAS domain build, the following hierarchy and measure files are created from item.txt.

File Name	Label
prod.dat	RMS Product Hierarchy
dmx_rmsskumap.ovr	RMS SKU Map
dmx_pszmap.ovr	Pack size Map

   In addition, the interutil program will convert from RMS SKU to AIP SKU all RMS-sourced inventory measure data that has been configured to arrive "early," and therefore is present to be processed by this script (as opposed to arriving late, and therefore is processed by check_process_inventory_data.sh). See following script and the *AIP Implementation Guide* for details on this configuring.

   Note that dmx_rmsskumap measure data is not loaded into RPAS. The later mapping of RMS SKU to AIP SKU is done without use of this loaded data. However, this data is still required as it is delivered to the AIP Oracle part of the application.

5. Rename prod.dat to prod.txt.

6. Call construct_ntier_hierarchies.sh to add stocking point prefixes to whse.txt, loc.txt, splr.txt, and generate the ssp and dsp hierarchy .dat files.

7. Using run_interutil.sh and *.config files from $AIPDOMAIN/interface/config/hier, selected RMS/external hierarchies are formatted with flags that will later be used to distinguish the external hierarchy data with the AIP Online data during the merge operation of the two sets of hierarchies. In addition the run will convert the hierarchy positions from RMS SKU to AIP SKU. The following table lists the inputs to and outputs from this call and follow-up shell commands. All locations referenced are

inside $AIPDOMAIN. Supplier and location hierarchies which do not require field separator flags are copied to $AIPDOMAIN/interface/import.

Input Files from interface/rms	Config File from interface/config/hier	Output Files written to interface/import
prod.txt	prod.config	prod.txt
whse.txt	whse.config	whse.txt

8. Move non-RMS, external, measure data listed in $AIPDOMAIN/interface/config/external/measdata_from_external.config from the $AIPDOMAIN/interface/rms directory to the $AIPDOMAIN/interface/external directory where they will be loaded into the AIP domain via a later script, load_non_rms_files.sh.

9. Move non-RMS, external, non-measure data listed in $AIPDOMAIN/interface/config/external/aiponlinedata_from_external.config to $AIPDOMAIN/interface/export so it can be exported to AIP online via a later script. During this step, prior to moving wh_type.txt, make a copy of wh_type.txt as $AIPDOMAIN/interface/external/dmx_wh_typ.ovr so it can be loaded into the $AIPDOMAIN.

10. Create a copy of hspl.txt named prof.def. In a later step, when Automatic Warehouse Profile Creation is enabled, this file will be formatted to match the warehouse profiles created for Suppliers and merged with the existing DM Online Warehouse Profiles.

11. Remove the marker $AIPDOMAIN/interface/export_dm_data, set by send_dm_measures_to_online.sh. This marker indicates a successful export in the export_dm_data.sh call from the previous run of AIP RPAS batch. When marker is not present, the export process will export deltas, and will set the marker. If the marker is present, then export_dm_data.sh will preserve a backup of the previous export before generating a new one, in order that a restart/recovery of this step can be implemented.

This script calls the following scripts:

_check_for_required_files (a function defined in bsa_check_for_required_files.sh)

process_external_data.sh

Appropriate Batch Run (First Time and/or Daily)

First Time and Daily

Prerequisites

This step must not be initiated until the following is successfully completed:

Before this script runs, all data files from Table 1, Table 2, and Table 3 should be copied by the client's batch scheduler into the $AIPDOMAIN/interface/rms directory. Some of the data is required, and some is optional. Reference the $AIPDOMAIN/interface/config/external/earlyfiles.config for requirements.

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues.

3. Move prod.txt, whse.txt, splr.txt, and loc.txt to the $AIPDOMAIN/interface/import directory. The purpose of this is to have hierarchies ready in AIP Online format, and they should be merged with the actual AIP Online hierarchies at a later step.

4. Make sure that all required RMS *.txt files have been FTPed correctly into the $AIPDOMAIN/interface/rms directory. If the files were not FTPed correctly, rerun the client-owned batch process which performs the FTP operation, but ensure that the required RMS files are present in the source location before restarting.

5. Ensure that the width and offsets of all positions match what is in the prod.format file. The labels may differ due to having executed the sed and awk commands to produce the final format.

6. Restart the batch from this step.

> **Note:** This step can be run in parallel with "Export DM and OM Online Data" and "Retrieve data from AIP Online."

Create Empty Hierarchy Files

Step Name

create_empty_default_files

Script Call

create_empty_default_files.sh

Functional Overview

When AIP Batch is run for the first time, there is no AIP Online data that can be imported into AIP RPAS. Therefore there are no product and warehouse hierarchy files provided by AIP Online to be merged with the product and warehouse hierarchy files provided by an inventory management system. This step creates empty files that act as placeholders for the merge operation (see script "merge_hierarchies.sh" below).

Technical Details

The script create_default_positions.sh is used to create 0-byte files for prod.dat and whse.dat.

Day on Day Processing

1. Remove $AIPDOMAIN/interface/import/*.dat. This clears out any unwanted, residual data that exist prior to the first-time run.
2. Use UNIX "touch" command to create $AIPDOMAIN/interface/import/prod.dat and $AIPDOMAIN/interface/import/whse.dat.

Appropriate Batch Run (First Time and/or Daily)

First Time only

Prerequisites

None

Restart/Recovery

If create_empty_default_files.sh fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Internal Integration Data Processing

AIP RPAS is integrated with AIP Online by means of exporting data from AIP Online into AIP RPAS, and at the end of the AIP RPAS batch process, by exporting data from AIP RPAS into AIP Online. Therefore we refer to these two exchanges inside AIP as "Intragration." The first export occurs early in AIP RPAS batch to retrieve the AIP Online data into the AIP RPAS domain.

Prepare Data from AIP Online Platform

Step Name

prep_onl_data

Script Call

prep_from_aiponline.sh

Functional Overview

You will setup and maintain the supply chain and many replenishment parameters in DM Online. All of this data will be used by AIP RPAS batch. For AIP RPAS batch to use this data, it must be retrieved from the Oracle database and loaded into the RPAS domain.

In addition to supply chain data, DM Oracle batch maintains the virtual date used by AIP RPAS batch. This value is loaded into RPAS along with the rest of the DM Online data.

Technical Details

AIP RPAS processes data from AIP Online. AIP Online exports hierarchy information and DM measures which are placed by the user into the AIP RPAS domain, in the $AIPDOMAIN/interface/import directory.

The prep_from_aiponline.sh script is called to process the data files created by the AIP Online export. The data files should be transferred from AIP Online to the export directory by a job scheduling application.

Day on Day Processing

1. prep_files.sh is called two times. The first will process the AIP Online hierarchy export. The second will process the AIP Online measure export. The prep_files.sh script is invoked with two keys that are listed inside $AIPDOMAIN/interface/config/bsa_prep_files.config:

   - **DMo_hier_export** is keyed to process the file hierarchy.tar.Z and unpack the data contained within to the $AIPDOMAIN/interface/import/hier directory.
   - **DMo_meas_export** is keyed to process the file aip.tar.Z and unpack the data contained within to the $AIPDOMAIN/interface/import/meas directory.
   - The data unpacked will overwrite any existing files in these two directories.
   - Before each archive is unpacked, a backup of the file will be created in the $BSA_ARCHIVE_DIR directory. The backup filenames will be aip.tar.Z.<timestamp> and hierarchy.tar.Z.<timestamp>.
   - After each archive is unpacked, the archive file is removed from $AIPDOMAIN/interface/import.

2. Two data files are moved from $AIPDOMAIN/interface/import/hier to $AIPDOMAIN/interface/import: prod.dat and whse.dat. This is to prepare for the hierarchy merging.

3. Boolean TRUE values are appended to the data that is currently exported from AIP Online. The files are overwritten in the $AIPDOMAIN/interface/import/meas directory. Files currently exported in this way that have Boolean TRUE values appended are currently the following files:

   ▪ dm1_prflks

This script calls the following scripts:

prep_files.sh

Appropriate Batch Run (First Time and/or Daily)

Daily only

Prerequisites

This step must not be initiated until successful completion of the following:

The cron_export.sh script runs in the AIP Oracle schema.

Restart/Recovery

If this step fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues.

3. Ensure that cron_export.sh ran correctly on the AIP Oracle schema. If it did not, perform the restart/recovery steps for cron_export.sh.

4. Ensure that *.Z files were correctly FTPed to RPAS. If not, archive or remove any RPAS zip files under $AIPDOMAIN/interface/import and rerun the client owned batch process to FTP the data from AIP Oracle side to AIP RPAS side.

   > **Note:** Both the hierarchy.tar.z and aip.tar.z files need to exist in the [domain]/interface/rms directory.

5. Restart the batch from this step.

Notes

This step can be run in parallel with check_process_external_data.

Process Merchandising System and AIP Online Hierarchies

Before AIP RPAS can manipulate the measure data, first the new hierarchy data must be loaded. New hierarchy positions can potentially arrive from both the merchandising system (such as RMS) as well as from AIP Online.

Merge Hierarchies

Step Name

merge_hierarchies

Script Call

merge_hierarchies.sh

Functional Overview

After the initial run, AIP RPAS batch will load data from RMS and AIP Online. Various AIP specific attributes of the RMS foundation data are maintained in DM Online. These attributes must be merged with the data that was loaded from RMS before being loaded into the RPAS domains.

> **Note:** AIP Online does not allow the modification of any attributes mastered in RMS; such as warehouse name, SKU name, or department.

Technical Details

This step merges the AIP Online hierarchies with the external system hierarchies (RMS). At a later time, the merged hierarchies are loaded into the AIP RPAS. The merge_hierarchies.sh script is used to perform the merge.

Day on Day Processing

The prod.dat and whse.dat files have default positions, which are later assigned through AIP Online. Therefore, these files need to be merged with the RMS hierarchies. AIP Online does not affect the location and supplier hierarchy files. Therefore, the RMS contributions of loc.txt and hspl.txt are moved to be loaded directly into AIP RPAS domains.

1. The hierarchies are merged using run_interutil.sh. The run_interutil.sh uses the interutil binary and hierarchy configuration files to merge the online hierarchy with the RMS hierarchy and to generate the final hierarchy file.

   - prod.dat + prod.def using prod.config generates prod.dat
   - whse.dat + whse.def using whse.config generates whse.dat

   All of these outputs are in AIP Online format and need to be converted to AIP RPAS format before getting loaded into RPAS.

2. $AIPDOMAIN/interface/import/hspl.txt and splr.txt are moved to $AIPDOMAIN/interface/import/hier/loc.dat and hspl.dat respectively.

3. Call merge_prof.sh to merge prof.def from External Source and prof.dat from AIP Online.

4. prof.def, which was created in a previous AIP Batch step, is formatted and merged with prof.dat from AIP Online. This occurs inside merge_prof.sh when Automatic Warehouse Profile Creation is enabled.

This script calls the following scripts:

run_interutil.sh

merge_prof.sh

Appropriate Batch Run (First Time and/or Daily)

First Time

Daily

Prerequisites

All steps documented above in this Operations Guide with the exception of check_process_inventory_data.

Restart/Recovery

If this step fails, perform the following:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Ensure that the RMS data files (prod.def and whse.def) are created under the $AIPDOMAIN/interface/import directory with the correct format.
4. Ensure that the AIP Online files (prod.dat and whse.dat) are created correctly under the $AIPDOMAIN/interface/import directory with the correct format.
5. If files are not present, re-run the previous steps to re-generate the required files
6. Restart the batch from this step.

Convert Hierarchies for Loading

Step Name

convert_hierarchies_for_loading

Script Call

convert_hierarchies_for_loading.sh

Functional Overview

This script converts hierarchies from RMS merged with AIP Online for loading into RPAS domains.

Technical Details

This script takes the merged RMS/AIP Online hierarchy files found in the $AIPDOMAIN/interface directory, and uses the interutil executable to convert them to RPAS-loadable format. The converted hierarchy data files are destined for the $AIPDOMAIN/input directory.

Day on Day Processing

1. Remove any existing hierarchy load files, $AIPDOMAIN/input/*.dat.
2. For each RMS/AIP Oracle hierarchy file (matching the pattern *.dat) found in $AIPDOMAIN/interface/import/hier/, except for Profile, Location and Supplier, spawn a parallel instance of run_interutil.sh to execute the interutil binary with the REMOVE_FIELD_SEPARATORS option. For each file, this creates an output file in $AIPDOMAIN/input/, formatted according to the corresponding configuration file

in $AIPDOMAIN/interface/config/hier/. All field separators in the processed files are removed. The unprocessed data files are copied to $AIPDOMAIN/input as well.

3. Make a copy of the product, location, supplier and warehouse hierarchy files for processing by the new hierarchy alert logic later in the AIP RPAS batch.

4. Move the interval, advertising, source and destination stocking point hierarchy files into $AIPDOMAIN/input.

5. Merge the position name into the label for the product, location, warehouse, supplier, source and destination stocking point hierarchies, so that when viewed through the RPAS Client, the position name will be displayed in the label.

This script calls the following scripts:

run_interutil.sh

mergeProdPositions.awk

mergeStrPositions.awk

mergeWhPositions.awk

mergeSplrPositions.awk

mergeDspSspPositions.awk

Appropriate Batch Run (First Time and/or Daily)

First Time

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

▪ merge_hierarchies.sh

Restart/Recovery

If this step fails, perform the following:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues.

3. Ensure that directory $AIPDOMAIN/interface contains the RMS/AIP Online hierarchy data files indicated in the configuration file $AIPDOMAIN/interface/config/hier/${hier}.config.

> **Note:** Both the had.txt and intv.txt files need to exist in the [domain]/interface/rms directory

4. If files are not present, re-run the previous steps to re-generate the required files

5. Restart the batch from this step.

Reconfigure AIP Domain Partitions

Step Name

reconfig_domain_partitions

Script Call

reconfigAIPDomainPartitions.ksh

Functional Overview

This step automates the addition of new partitioning dimension positions to the AIP RPAS domain. When adding these subclasses, it ensures that no local domain is overloaded and spreads the incoming subclasses as evenly as possible. It also ensures that related items, in term of merchandise hierarchy, are clustered together in local domains.

Technical Details

This script is a wrapper script to the script generateReconfigPartDimXml.ksh and the RPAS_HOME/bin utility reconfigGlobalDomainPartitions to be used within AIP. The script generateReconfigPartDimXml.ksh creates reconfigpartdim.xml if any new subclass is coming from RMS in the prod.dat file.

The xml file reconfigpartdim.xml may be created during this process. This file is created if a new subclass is found in the prod.dat which is not part of the existing domain structure. The RPAS utility reconfigGlobalDomainPartitions will use the auto-created reconfigpartdim.xml to add the new subclass position(s) to the domain.

To remove positions along the partitioning dimension, use the AIP RPAS Purging functionality described in the AIP Interval Batch Scripts chapter of this guide.

Day on Day Processing

1. Call generateReconfigPartDimXml.ksh to search $AIPDOMAIN/input/prod.dat, looking for new partitioning dimension positions. If any is found, generate $AIPDOMAIN/config/reconfigpartdim.xml.
2. If both of the following files exist:
   - $AIPDOMAIN/config/reconfigpartdim.xml
   - $AIPDOMAIN/input/prod.dat

   Then run dimension addition by:
   - Calling reconfigGlobalDomainPartitions to add the positions indicated in reconfigpartdim.xml.
   - Move reconfigpartdim.xml to the $AIPDOMAIN/input/processed directory and append a timestamp to the file.

Appropriate Batch Run (First Time and/or Daily)

First Time

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

convert_hierarchies_for_loading.sh

Restart/Recovery

If this step fails, perform the following:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Load Hierarchy and Non-Inventory Measure Data into AIP RPAS

Load All Hierarchies

Step Name

load_all_hierarchies

Script Call

load_all_hierarchies.sh

Functional Overview

This script preprocesses a subset of the hierarchy load files, prepending position names to the existing label field text, and then loads all hierarchy load files into the global and local domains.

Technical Details

The following hierarchy load files are preprocessed to prepend the hierarchy position names to the position labels:

- prod
- loc
- whse
- hspl
- ssp
- dsp

The prepending process is handled in a set of awk scripts.

After preprocessing of the select hierarchy load files, all hierarchy load files are loaded into AIPDOMAIN and its local domains by standard RPAS functionality.

Day on Day Processing

1. Preprocess the following hierarchy load files in the $AIPDOMAIN/input directory, by passing to the indicated awk scripts (internally, each of these files is first renamed to differentiate input and allow writing of output to the original file name):
   - prod.dat: mergeProdPositions.awk
   - loc.dat: mergStrPositions.awk
   - whse.dat: mergeWhPositions.awk
   - hspl.dat: mergeSplrPositions.awk
   - ssp.dat: mergeDspSspPositions.awk
   - dsp.dat: mergeDspSspPositions.awk
2. Call loadHier to load all hierarchy load files found in the $AIPDOMAIN/input directory.

This script calls the following scripts:

mergeDspSspPositions.awk

mergeProdPositions.awk

mergeSplrPositions.awk

mergStrPositions.awk

mergeWhPositions.awk

Appropriate Batch Run (First Time and/or Daily)

First Time

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

reconfigAIPDomainPartitions.ksh

Restart/Recovery

If this step fails, perform the following:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. If the script failed to load specific positions, identify from where these positions are coming.
4. If new positions are coming from RMS, identify them in item.txt. If the data coming from RMS is incomplete, remove that line and raise a defect to the RMS team.
5. If new positions are coming from DM Online with erroneous data or bad formatting, remove data from the load file and raise a defect for DM Online.
6. If there are no new positions coming from either system, revert to the previous successful hierarchy loads.
7. Restart the batch from this step.

Notes

If the files are in the correct format, you should be able to run the `loadHier` RPAS utility on individual files:

```
loadHier  -d <Global Domain> -loadAll -checkParents -forceInputRollups
```

Load AIP Online Measure Data

Step Name

load_onl_data

Script Call

load_online_measures.sh

Functional Overview

Data files from the AIP Online part of AIP are loaded into the AIP RPAS domain.

Technical Details

The data files from AIP Online are listed in two configuration files: $AIPDOMAIN/interface/config/meas contains compressed.config and uncompressed.config. The compressed.config file lists the measure data files that are compressed by time, in that consecutive (by day) equivalent values are not contained in the data for the same intersection (e.g. sku/store). The uncompressed.config lists the measure data files that are not compressed by time.

Each measure listed in these two configuration files is contained in the aip.tar.Z file which is transferred from the AIP Online server to the AIP RPAS server, and unpacked in the $AIPDOMAIN/interface/import/meas directory, in the process described above.

The compressed measure data follows this technical process: the data is loaded into a compressed staging measure, then uncompressed into the actual measure array. The measure data is uncompressed to a different target end date depending on the planning horizon and other functional considerations. The uncompressed measure data is loaded into the measure array without need for staging measure load and uncompressing. In both cases, the measure data is always loaded as an overlay (a load into currently populated data locations will overwrite, and all other data remains intact); however, the measure array may be partially or a fully cleared prior to this overlay load, depending on the functional requirements. Neither the compression/uncompression flag, nor the end date for uncompressing, nor the clearing prior to load are configurable by the client.

Day on Day Processing

1. All measure data files listed in the uncompressed.config and compressed.config are copied from $AIPOMAIN/interface/import/meas/*.dat to $AIPOMAIN/input/*.ovr.

2. The measure data "deletion flags", which indicate values have been deleted in AIP Online, are converted into actual NA values.

3. The planning horizon measures are loaded.

4. All measure data files listed in the uncompressed.config are cleared (if scheduled for clear operation) and loaded into the measure arrays in $AIPDOMAIN.

5. All measure data files listed in the compressed.config are cleared (if scheduled for clear operation) and uncompressed into the measure arrays in $AIPDOMAIN.

6. All measures with a day dimension in their base intersection, but which are not scheduled for future-clear operation, are extended by one day to accommodate the rollover of the actual day.

This script or sub-scripts call the following programs:

- clearArray (AIP binary)
- compressValues (AIP binary)
- loadmeasure (RPAS binary)
- run_interutil.sh (script wrapper for interutil AIP binary)
- xmace (AIP script wrapper for RPAS mace binary)

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

All prior steps in this AIP batch process must be completed with the exception of check_process_inventory_data. The data load is dependent on the successful loading of the new hierarchy information from AIP Online merged with External Hierarchy data.

Restart/Recovery

If this step failed, check the following:

1. Examine the log generated to understand why it failed.

2. If the failure was for a specific measure(s), check which position(s) within that measure(s) is failing.

3. Ensure that those positions were successfully loaded during the "Load All Hierarchies" step.

4. If all positions are correctly loaded, check to ensure that the data formats in the *.dat files from Online are correct.

5. If positions were incorrectly loaded during the "Load All Hierarchies" step, fix it, and re-run `loadHier` with the correct values for the domain where the failure occurred.

6. Restart the batch from this step.

Load External Non-Inventory Measure Data

Step Name

load_rms_dm_data

Script Call

load_rms_dm_measures.sh

Functional Overview

This step preprocesses and loads into the RPAS domain all Data Management measure data that comes from RMS or another external inventory system.

Technical Details

This script loads into $AIPDOMAIN all Data Management measure data in $AIPDOMAIN/interface/rms.

Day on Day Processing

1. Call loadmeasure.sh (a script wrapper for loadmeasure RPAS binary) on the measure data files corresponding to the Data Management measure data received from the inventory system (e.g., RMS). The measures loaded are listed in the following configuration files:

   - $AIPDOMAIN/interface/config/rms/meas/rms_sku_map.config
   - $AIPDOMAIN/interface/config/rms/meas/dm_rms_measures.config

   Note some measures are loaded as overlays (.ovr) and some are loaded as full replacement (.rpl). This is not intended to be configurable due to functional requirements.

This script calls the following scripts:

loadmeasure.sh

Appropriate Batch Run (First Time and/or Daily)

First Time

Daily

Prerequisites

This step is dependent on successful completion of the following:

All steps through the load_all_hierarchies.sh script.

Restart/Recovery

If this step fails, you need to check the following:

1. Ensure that the *.txt files are transferred via FTP by client scheduled process from RMS to the correct location.

2. Ensure all RMS file are present.

3. Ensure all files are formatted correctly.

4. Ensure that the "Load All Hierarchies" step ran correctly, and all of the hierarchies' values received from RMS are loaded correctly.

5. If any of the above did not run correctly, re-run the corresponding step.

6. Restart the batch from this step.

Notes

This step can be run in parallel with "Load Measures with AIP Online Data."

Also, when this script is run for the first time, these measures (dm0_pmsendsrc.txt and dm0_pmsstasrc.txt) are not populated yet so the output for these measures will be empty. During the next daily batch run, the RMS sends this information again and updates the measure correctly.

Generate and Load New Item Alert Measures

Create Empty Archive Files

Step Name

create_empty_archive_files

Script Call

create_empty_archive_files.sh

Functional Overview

When AIP Batch is run for the first time, there are no hierarchy alert archive files present from a previous run, which prevents the subsequent step, load_all_newitem_alert_measures.sh, from completing. This step creates empty files that act as placeholders for the archive files.

Technical Details

The script creates the following zero-byte files needed for the hierarchy alerts:

- $AIPDOMAIN/interface/prod.dat.last
- $AIPDOMAIN/interface/whse.dat.last
- $AIPDOMAIN/interface/loc.dat.last
- $AIPDOMAIN/interface/hspl.dat.last

> **Note:** Other 0-byte alert archive files are created; however, only the files listed above are required.

Day on Day Processing

1. Remove $AIPDOMAIN/interface/*.last

2. Call UNIX touch to create:

      a. $AIPDOMAIN/interface/prod.dat.last

      b. $AIPDOMAIN/interface/whse.dat.last

      c. $AIPDOMAIN/interface/loc.dat.last

      d. $AIPDOMAIN/interface/hspl.dat.last

Appropriate Batch Run (First Time and/or Daily)

First Time

Prerequisites

None

Restart/Recovery

If create_empty_archive_files.sh fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Generate Batch and Online Alerts

Step Name

create_alerts

Script Call

load_all_newitem_alert_measures.sh

Functional Overview

Each day, alerts must be generated to inform you that new foundation data was loaded into AIP. Once visible in DM Online, the alert will prompt you to set up the supply chain for the new data.

The alerts are also used by AIP RPAS batch and DM Oracle batch to perform new hierarchy specific processing; such as Demand Group assignment, Profile assignment, and Store Source assignment.

Technical Details

This step generates AIP Online and AIP RPAS batch alerts. This step compares the new hierarchy data files with the previous set of hierarchy data files (stored as <hierarchy>.dat.last). If it is a first time aip_batch.sh run, the comparison is against empty <hierarchy>.dat.last files created in a previous first-day step, create_empty_archive_files.sh.

Five batch alerts are generated.

Alert Measure	Label
dm0_new	New Stores
dm0_newspl	New Supplier Alert
dm1_new	New Warehouses
dmx_newprd	New Commodities

dmx_newpsz	New Commodity-Pack Sizes

Day on Day Processing

The script generates RPAS batch alerts by using load_one_newitem_alert_measure.sh script.

load_one_newitem_alert_measure.sh compares the old hierarchy, which was saved as hierarchy.dat.last with the new hierarchy that was loaded earlier to obtain new alerts. If one of the files does not exist for comparison, an empty file is generated and then loaded.

The script uses printArray binary to obtain dimensions, hierarchy, starting position, and the width of the alert. It also compares the hierarchy.last.dat with hierarchy.dat file. The difference is saved as measurename.rpl in the input folder of the DM domain. The generated alert measures are then loaded using the loadmeasure binary.

Once the comparison is complete and the alerts are created, the current $AIPDOMAIN/interface/<hierarchy>.dat files are renamed to $AIPDOMAIN/interface/<hierarchy>.dat.last for the next day's alert generation.

This script calls the following scripts:

load_one_newitem_alert_measure.sh

Appropriate Batch Run (First Time and/or Daily)

- First Time
- Daily

Prerequisites

This step can only run if the load_all_hierarchie.sh script ran successfully.

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Check to see if the following measures are flagged correctly based on the new data passed from RMS or AIP Online:
   - dm0_new
   - dm0_newspl
   - dm1_new
   - dmx_newprd
   - dmx_newpsz
4. If any of the above measures were not flagged correctly, manually update the measure(s) with the correct value:
   - 1 = True
   - 0 = False
5. If necessary, ftp the *.txt files from the external system to $AIPDOMAIN/interface/external.Restart the batch from this step.

Load External System Measure Data into AIP RPAS

Load Non-RMS External Files

Step Name
load_non_rms_external

Script Call
load_non_rms_external.sh

Functional Overview
This step is used to load non-forecast measures from external sources other than RMS.

Technical Details
This script is a wrapper for load_non_rms_files.sh passed with a command line argument of "external." This subscript will load the early arriving non-RMS external system data into the RPAS domain. See Chapter 7, section "Load Replenishment Inventory Data" for information about late arriving non-RMS external system data.

Day on Day Processing
1. Create a list of <measure>.ovr file names by reading $AIPDOMAIN/interface/config/external/measdata_from_external.config, and converting all suffixes to "ovr".
2. For each of the <measure>.ovr file names in the above list that refer to existing files in directory $AIPDOMAIN/interface/external:
   a. Move the <measure>.ovr file to directory $AIPDOMAIN/input
   b. Call, in parallel, loadmeasure.sh to load measure <measure> with data from <measure>.ovr.

This script calls the following scripts:
- load_non_rms_files.sh
- loadmeasure.sh (called by load_non_rms_files.sh)

Appropriate Batch Run (First Time and/or Daily)
- First Time
- Daily

Prerequisites
This step must not be initiated until successful completion of the following:
- check_process_external_data.sh
- load_all_hierarchies.sh

Restart/Recovery
If this script fails, perform the following steps:
1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. If necessary, restart the batch from this step.

Notes

This script may be run in parallel with load_rms_dm_measures.sh and load_onl_data.sh.

Commit Workbooks before Batch Run

Auto Commit Workbooks

Step Name

auto_commit_wkbooks_batch

Script Call

workbook_batch.sh COMMIT

Functional Overview

The user-modified workbooks may be configured to automatically commit during the batch. All SRP and WRP workbooks configured for commit at both global and local domain level will be committed in this step. This script may also be run from the command line ad hoc throughout the day by administrators or users.

Technical Details

The workbook_batch.sh script is called to process the SRP and WRP workbooks scheduled for automatic commit. The script uses the RPAS utility wbbatch to automatically commit all SRP and WRP workbooks on the workbook commit queue for each global and local domain. The COMMIT action defined by the single parameter is first performed on the global domain workbooks, then in parallel for each of the local domains.

Day on Day Processing

1. Call wbbatch to commit all workbooks on the $AIPDOMAIN global domain commit queue.
2. For each local domain, call wbbatch in parallel to commit all workbooks on each local domain commit queue.

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

The workday for AIP workbook users must be complete.

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Perform and Export Data Management Calculations

Run Initial Load DM Batch

Step Name
run_partial_dm_batch

Script Call
run_partial_dm_batch.sh

Functional Overview
During a first-time run, execute a partial Data Management RPAS Batch process to establish the planning horizons used by AIP RPAS batch, to create and assign warehouse profiles, and to set store source values.

Technical Details
The Data Management binaries dmcreateprf, dmplanhrzn, dmassignprf, and dmcatchup are executed in their entirety.

Day on Day Processing
1. Runs dmcreateprf.sh shell script wrapper to the dmcreateprf AIP binary on the $AIPDOMAIN. This assigns the default Store Order Cycle and all warehouses to automatically created Warehouse Profiles.
2. For each local domain of the RPAS global domain $AIPDOMAIN, runs the dmplanhrzn.sh and dmassignprf.sh shell script wrappers to the dmplanhrzn and dmassignprf AIP binaries. The two scripts are run in series, in the background, for each local domain, such that each local domain is processed in parallel. After all instances are started, the script waits for all instances to complete.
3. For each local domain of the RPAS global domain $AIPDOMAIN, runs the dmcatchup.sh shell script wrapper to the dmcatchup AIP binary. The script is run in the background for each local domain such that each local domain is processed in parallel. After all instances are started, the script waits for all instances to complete.

This script calls the following scripts:
- dmcreateprf.sh
- dmplanhrzn.sh
- dmassignprf.sh
- dmcatchup.sh

Appropriate Batch Run (First Time and/or Daily)
First Time

Prerequisites
This step must not be initiated until successful completion of the following:
- Set Implementation Parameters
- Verify and Process Data from External Systems
- Load All Hierarchies
- Load Measure Data

- Load Non-RMS External Data
- Generate Batch and Online Alerts

Restart/Recovery

If this step fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues.

3. If there is a problem with the dm1_prfhme measure content, the clear the measure so that the next attempt at overlay load will not have any leftover data from the previous erroneous load attempt.

4. Restart the batch from this step.

Run DM Batch

Step Name

run_dm_batch

Script Call

dmb_master.sh

Functional Overview

DM RPAS batch is a precursor to the Supply Chain Replenishment Planning (SCRP) calculations. It performs a number of setup and maintenance steps that prepare the data for the rest of the batch processes.

The data setup primarily relates to the technical needs of SRP and WRP, known together as SCRP. It also relates to creating various masks or flag measures to ease the processing. Setting up a mask usually involves combining two or more values (for instance; store planning horizon, on-supply dates, and off-supply dates) along with business validation rules in order to produce a single value that can be repeatedly referenced. This keeps the calculation/validation centralized, and it prevents the need for repeating time consuming logic each time the resultant value is needed.

The most notable data setup calculations that are performed include the:

- warehouse schedule
- store release schedule
- store placement schedule

DM RPAS also performs a number of maintenance and default procedures. These include:

- Warehouse Profile Creation
- New SKU assignment to a warehouse Profile
- Store Source assignment--a.k.a 'Catch up'
- Setting warehouse ranging statuses
- Pre-priced status change

Finally, DM RPAS will generate alerts that are not related to new hierarchy values. These alerts are exported to DM Online to inform you that various pieces of the supply chain are incomplete or could not be set to the appropriate default settings by DM RPAS. This

generation occurs later in the AIP Batch process, however, and not in this step. See the "Run Data Management Alerts" section, below.

Technical Details

At this stage all of the data is loaded into the RPAS global domain $AIPDOMAIN. Now the dmb_master.sh script is called to execute the DM RPAS batch. DM RPAS batch consists of several modules comprised of Korn shell scripts, RPAS rule groups, and C++ binaries. Each module uses some integration (e.g. RMS, RDF) and intragration (AIP Oracle/Online) inputs and creates outputs, and many times uses as inputs previous DM RPAS batch module outputs. dmb_master.sh is responsible for beginning the critical path processing of DM RPAS batch, so it is called first.

The Korn shell scripts call a number of sub-scripts, which call a number of binaries and execute a number of rule groups in a particular order to calculate new measures. The order of the execution is important because some of the calculated measures are dependent on previously calculated measures. The name of the binary called from these scripts is the same as the name of the script. For example, dmplanhrzn.sh calls the binary dmplanhrzn.

In a later step, the script dmb_master20.sh, which performs non-critical path processing, is executed.

Day on Day Processing

1. dmb_master.sh calls the dmb_master10.sh script, which contains calls to several other DM batch scripts and DM rule groups that perform the critical path processing of Data Management batch in the RPAS domain $AIPDOMAIN. The following table presents a high level summary of the Data Management modules that are executed along with their functionally descriptive name.

Order of Execution	Script / Rule Group	Description
1	dmcreateprf.sh	Assigns Store Order Cycle and Warehouses to automatically created Warehouse Profiles.
2	rulegroup dm_batch_global	Calculate Delivery Pattern Default, Non-release Day Default, Non-order Day Default, Non-delivery Day Default
3	dmplanhrzn.sh	Calculate Planning Horizons
4	dmwhsrc.sh	Calculate Warehouse Source
5	dmassignprf.sh	Attempts to assign new SKUs to warehouse profiles.
6	dmdirsppflg.sh	Direct Supply Point Flag
7	dmcatchup.sh	Store Source Assignment
8	horizonFilter.sh dm1_prdalwsts	Filter SKU-Warehouse Allowable Status down to Planning Horizon
9	dmcmwhal.sh	Ranging Warehouses to SKU-pack sizes
10	dmstrordpks.sh	Calculate Store Ordering Pack Size
11	dmstrordunt.sh	Calculate Store Ordering Unit
12	dmconwhpalmul.sh	Convert Warehouse Pallet Multiple

Order of Execution	Script / Rule Group	Description
13	dmsmlordpksz.sh	Calculate Smallest Ordering Pack Size
14	dmdgrspec.sh	Calculate Demand Group Specifier
15	dmpreprcstchg.sh	Identify Pre-Priced Status Change
16	dminheritattr.sh	Inherit Attributes
17	rulegroup dm_batch_local	Build Supply Chain Schedule
18	rulegroup dm_batch_local_p	Calculate all schedule dependent measures.
19	rulegroup scrp_pre1	Calculate all supply-chain parameters

This script calls the following scripts:

dmb_master10.sh

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- load_non_rms_external.sh
- load_all_newitem_alert_measures.sh
- load_rms_dm_measures.sh
- load_online_measures.sh

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure. This step will generate many log files. Navigate through the BSA logging directory structure to find the log file corresponding to the DM batch script or binary which reported an error.

2. Correct any identified setup or environment issues.

3. Restart the batch from this step.

Export DM Data

Step Name

export_dm_data

Script Call

export_dm_data.sh

Required Parameters

firstTime (true | false)

Functional Overview

Certain hierarchy and measure data are required by AIP Online for the users to set up the supply chain. The following data must be made available in a set of flat files that follow an agreed AIP RPAS/AIP Online file format.

- the hierarchy data
- the measure data calculated in Data Management during the AIP RPAS batch

These are exported from the AIP RPAS domain and bundled for pickup by the client's custom processes to transfer the data to the AIP Online server, for processing by AIP Online's initial batch processes.

Technical Details

The export_dm_data.sh script is called. This script handles exporting the necessary AIP RPAS hierarchy and measure data into flat files formatted for AIP Online.

Day on Day Processing

1. Call the script export_aip_hiers.sh.
   a. This script processes the location, supplier, product, warehouse and profile hierarchies for export to AIP Online and copies them to $AIPDOMAIN/interface/export. The product and warehouse hierarchies copied contain the needed field separators from a previous processing step. The profile hierarchy does not need field separators. The location and supplier hierarchies are processed by adding field separators to them, using the "ADD_FIELD_SEPARATORS" option to run_interutil.sh.

2. Call the script send_dm_measures_to_online.sh.
   a. This script exports the Data Management measures required for export to AIP Online and copies them to $AIPDOMAIN/interface/export.
   b. It then packages the exported hierarchy and measure data files into the file $AIPDOMAIN/interface/export/dm.tar.Z, a compressed archive that will be processed by AIP Online's initial batch step after the client transfers the data from AIP RPAS to AIP Online using a non-AIP process.
   c. Finally this script touches the marker $AIPDOMAIN/interface/export_dm_data indicating that the export has completed. This marker is used by the next day's execution of process_external_data.sh, to determine if the previous day's export was successful.

This script calls the following scripts:

- export_aip_hiers.sh
- send_dm_measures_to_online.sh

Appropriate Batch Run (First Time and/or Daily)

- First Time
- Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- run_partial_dm_batch.sh (First Time run only)
- dmb_master.sh (Daily run only)

Restart/Recovery

If this step fails, perform the following:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Remove all *.dat and *.ref files from the $AIPDOMAIN/interface/export directory.
4. Remove all dm_* files from the $AIPDOMAIN/interface/export directory.
5. Restart the batch from this step.

External Integration Forecast Data Processing and Load into AIP RPAS

Check and Load Forecast Data

Step Name

check_load_forecast_data

Script Call

check_load_forecast_data.sh

Optional Parameters

 -h: to display script usage

Functional Overview

AIP manages pulling stock through the supply chain just in time to meet the expected demand. It does not calculate the forecasted store demand, so it must load the calculated demand forecasts for the stores from a demand forecasting system, usually RDF.

Technical Details

This script first checks for files listed as "required" in the configuration file $AIPDOMAIN/interface/config/forecast/forecastdata_from_external.config. These files must exist in directory $AIPDOMAIN/interface/forecast. Then the script moves both the required and optional files listed in the configuration file to the $AIPDOMAIN/input directory before calling loadmeasure to load each.

Day on Day Processing

1. Verify that all files listed as "required" in $AIPDOMAIN/interface/config/forecast/forecastdata_from_external.config exist in $AIPDOMAIN/interface/forecast.
2. Create a list of <measure>.ovr file names by reading $AIPDOMAIN/interface/config/forecast/forecastdata_from_external.config, and converting all suffixes to "ovr".
3. For each of the <measure>.ovr file names in the above list that refer to existing files in directory $AIPDOMAIN/interface/external:
   a. Move the <measure>.ovr file to directory $AIPDOMAIN/input
   b. Call, in parallel, loadmeasure to load measure <measure> with data from <measure>.ovr.

This script calls the following scripts:

_check_for_required_files (function in bsa_check_for_required_files.sh)

load_non_rms_files.sh

loadmeasure.sh (called by load_non_rms_files.sh)

Appropriate Batch Run (First Time and/or Daily)

Daily only

Prerequisites

This step must not be initiated until successful completion of the following:

Forecast data is available from the external system (RDF).

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Ftp the *.txt files from the forecasting system to $AIPDOMAIN/interface/forecast.
4. Restart the batch from this step.

Prepare Replenishment Data and Maintain History

Purge and Advance Low-Variability Data

Step Name

purge_low_variability_advance

Script Call

purge_low_variability_advance.sh

Functional Overview

Time-series data measures that hold values that change infrequently in time are stored in a very efficient, specialized encoding that takes advantage of this structure. In order for the AIP calculations to make use of these measures, however, they must be adjusted for the current date and relevant period of history for which AIP calculations are to be performed. This adjustment is a combination of purging information in the past that is "too old," while simultaneously advancing key encoded values to the beginning of the relevant time window so that they are not lost. This adjustment must be performed at least once on the day of, and as a precursor to, the running of replenishment calculations.

Technical Details

Some of the data affected by this script is stored at the master level of the domain. Some is stored at the subdomain level. At both the global and the local domain level, the purge process is driven by the aipcmd binary, which processes the command files purgeLowVariabilityAdvanceGlobal.xml and purgeLowVariabilityAdvanceLocal.xml. These XML resource files contain the list of measures on which the purge and advance operation applies at each level. Please see the two XML files, located in $RPAS_HOME/applib/resources, for the lists of measures to which this action applies.

Day on Day Processing

This script performs the following steps:

1. Determine the list of local domains under $AIPDOMAIN.
2. Call the aipcmd binary with the purgeLowVariabilityAdvanceGlobal.xml resource file to purge and advance the data stored at the master level.
3. For each local domain, call in parallel the binary aipcmd with the purgeLowVariabilityAdvanceLocal.xml resource file to purge and advance the data stored at the subdomain level.

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following steps (script calls).

▪   workbook_batch.sh COMMIT

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Purge Truncate History

Step Name

purge_truncate_history

Script Call

for_each_local_domain.sh -p purge_truncate_history.sh [DOMAIN]

> **Note:** This complete script call must be made verbatim, including the substring "[DOMAIN]". This does not indicate the variable $AIPDOMAIN. Instead, this special string is a token interpreted by for_each_local_domain.sh script as a placeholder for the various local domain paths. for_each_local_domain.sh replaces this token with local domain paths as it calls the purge_truncate_history.sh script for each local domain.

Functional Overview

AIP retains historical data for a few time-series measures that hold value for display on Company Level Inventory Analysis worksheet found on WRP Interactive Evaluation/SRP Interactive Evaluation workbooks. This worksheet displays a configurable length of historical data. The XML Resource file purgeTruncateHistory.xml, located in $RPAS_HOME/applib/resources contains list of base measures used by the above worksheet along with required historical age in days. The daily AIP batch truncates any excess historical data found in the measures listed, thus maintaining adequate data in the system.

Technical Details

This step's command line script call is a composite of two scripts, for_each_local_domain.sh and purge_truncate_history.sh, which cause the purge truncate history process to be run in parallel across all local domains. At the local domain level, the process is driven by the aipcmd binary, which processes the command file purgeTruncateHistory.xml. This XML resource file contains the list of measures on which the purge truncate history operation applies. Please see file purgeTruncateHistory.xml, located in $RPAS_HOME/applib/resources for the list of measures to which this action applies.

Day on Day Processing

Script for_each_local_domain.sh performs the following steps:

1. Determine the list of local domains under $AIPDOMAIN.
2. For each local domain, call in parallel the script purge_truncate_history.sh and pass it the local domain path. That path is substituted automatically for the [DOMAIN] token by for_each_local_domain.sh.

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following steps (script calls).

workbook_batch.sh COMMIT

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Copy Sister Stores and Warehouses

Step Name

copy_sister_data

Script Call

for_each_local_domain.sh -p copy_sister_data_local.sh [DOMAIN]

> **Note:** See note for Purge and Advance Low-Variability Data regarding the use of for_each_local_domain.sh and its interpretation of "[DOMAIN]."

Functional Overview

Data to populate measure positions for newly introduced stores and warehouses is often copied to new positions from the positions of so-called "sister" stores and warehouses. Sister stores and warehouses have similar characteristics and therefore similar position data to the newly introduced positions. Such copying is declaratively configured for a list of affected measures, based on sister-to-new-position map and copy date. When "today"

equals a store or warehouse's copy date, then for all affected measures, data for each new position that corresponds to the copy date is copied from its defined sister position corresponding to the copy date.

Technical Details

This step's command line script call is a composite of two scripts, for_each_local_domain.sh and copy_sister_data_local.sh, which cause the copying of sister store and warehouse data to be performed in parallel across all local domains. At the local domain level, the process is driven by the aipcmd binary, which processes the command files copySisterStore.xml and copySisterWarehouse.xml, both of which use the CopyLikePositions command. The XML resource files contain the lists of measures on which the sister-copying operations apply. Please see files copySisterStore.xml and copySisterWarehouse.xml, located in $RPAS_HOME/applib/resources, for the lists of measures to which this action applies.

Day on Day Processing

Script for_each_local_domain.sh performs the following steps:

1. Determine the list of local domains under $AIPDOMAIN.
2. For each local domain, call in parallel the script copy_sister_data_local.sh and pass it the local domain path. That path is substituted automatically for the [DOMAIN] token by for_each_local_domain.sh.

Appropriate Batch Run (First Time and/or Daily)

Daily only

Prerequisites

This step must not be initiated until successful completion of the following:

- workbook_batch.sh COMMIT
- load_onl_data.sh

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

External Integration Inventory Data Processing and Load into AIP RPAS

Verify and Process Inventory Data from External System

Step Name

check_process_inventory_data

Script Call

check_process_inventory_data.sh

Optional Parameters

-h: to display script usage

Functional Overview

RMS is an enterprise solution that provides merchandise hierarchy and organizational setup and maintenance. This data, together with non-RMS external system data, becomes the foundation for the AIP supply chain replenishment. The data is processed as a set of flat files that follow an agreed AIP file format. If the client does not use RMS, but rather some other external system, this data is still required in the same format. AIP validates all of the files to ensure that the required data is present, and massages the files to produce the input required for subsequent steps of the AIP RPAS batch. AIP divides the integration data into two categories: early files and late files. The batch schedule for AIP may aim to maximize the available time by moving as much workload off the critical path as possible. For this reason the static data is extracted from RMS, or similar system, before the dynamic transaction data is available. This allows the hierarchies to be loaded into AIP and the DM batch run before the dynamic data is extracted. This step processes the early, static data.

Technical Details

The dynamic measure data are listed in the table below. For AIP RPAS batch to properly process external system data, all of the flat files must follow a particular format, which is explained in the *AIP Implementation Guide*. The flat files must have a *.txt extension.

File Name	Description
sr0_curinv_[*1..n*].txt	Store Current Inventory
sr0_it_.txt	Store In Transits
sr0_oo_.txt	Store On Orders
sr0_prdlfe.txt	Store Product Life
wr1_curinv.txt	Warehouse Current Inventory
wr1_it_.txt	Warehouse In Transits
wr1_oo_.txt	Warehouse On Orders
wr1_ow_.txt	WH Orders In The Well

Day on Day Processing

1. Verify that all required files have been downloaded. Failure to download any of the required files will result in an error and termination of the batch.
2. If store current inventory sr0_curinv data contains more than one partition, like sr0_curinv_1.txt, sr0_curinv_2.txt., …, sr0_curinv_*n*.txt, merge all of the partitioned files. Rename single or consolidated data file to sr0_curinvinit.txt.
3. Rename wr1_curinv.txt to wr1_curinvinit.txt.

4. Prefix the following measure data with the stocking point prefixes, using construct_ntier_measuredata.ksh:
   - sr0_curinvinit.txt
   - sr0_it_.txt
   - sr0_oo_.txt
   - wr1_curinvinit.txt
   - wr1_it_.txt
   - wr1_oo_.txt
   - wr1_ow_.txt
5. Using interutil binary, convert from RMS SKU to AIP SKU all RMS-sourced inventory measure data that has been configured to arrive "late," and therefore is present to be processed by this script (as opposed to arriving early, and therefore is processed by check_process_external_data.sh). See earlier script and the *AIP Implementation Guide* for details on this configuring. Note the AIP configuration is to have all dynamic data listed in the above table arrive as "late."

This script calls the following scripts:

_check_for_required_files (defined in bsa_check_for_required_files.sh)

process_inventory_data.sh

Appropriate Batch Run (First Time and/or Daily)

Daily only

Prerequisites

This step must not be initiated until successful completion of the following:

- All data files from the above table should be copied by the client's batch scheduler into the $AIPDOMAIN/interface/rms directory. Some of the data is required, and some is optional. Reference the $AIPDOMAIN/interface/config/external/latefiles.config for requirements.

Restart/Recovery

If the **process_inventory_data.sh** script failed, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Make sure that all required *.txt files have been transferred correctly via FTP into the $AIPDOMAIN/interface/rms directory.
4. Restart the batch from this step.

Notes

This step can be run in parallel with Export DM and OM Online and all aip_batch.sh steps up to scrp.sh.

Load Replenishment Inventory Data

Step Name

load_rms_replenishment_data

Script Call

load_rms_replenishment_measures.sh

Functional Overview

This script preprocesses and loads into the RPAS domain all store and warehouse replenishment measure data that come from RMS or other external inventory system.

Technical Details

This script loads into $AIPDOMAIN all Store and Warehouse measure data in $AIPDOMAIN/interface/rms. This step also loads the late arriving non-RMS external data.

Day on Day Processing

1. Call loadmeasure.sh (a script wrapper for loadmeasure RPAS binary) on the measure data files corresponding to the Replenishment measure data received from the inventory system (e.g., RMS). The measures loaded are listed in the following configuration files:

   - $AIPDOMAIN/interface/config/rms/srp_rms_measures.config
   - $AIPDOMAIN/interface/config/rms/wrp_rms_measures.config

   Note all measures are loaded as full replacement (.rpl). This is not intended to be configurable due to functional requirements.

2. Call load_non_rms_files.sh on the data files corresponding to the late arriving non-RMS external measure data files in the $AIPDOMAIN/interface/external.

This script calls the following scripts:

- loadmeasure.sh
- load_non_rms_files.sh

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- All steps through the load_all_hierarchies.sh script.
- This step can be run in parallel with "Load Measures with AIP Online Data."

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Ensure that the *.txt files are transferred via FTP by client scheduled process from RMS to the correct location.
4. Ensure all RMS file are present.
5. Ensure all files are formatted correctly.
6. Ensure that the "Load All Hierarchies" step ran correctly, and all of the hierarchies' values received from RMS are loaded correctly.
7. Restart the batch from this step.

Calculate and Export Replenishment Plan

Run Replenishment

Step Name

run_replenishment

Script Call

scrp.sh

Functional Overview

Replenishment and Reconciliation is part of the AIP batch run which generates a Receipt Plan for Warehouses and Stores across the length of the horizon. The final output from this batch process is 1) Constrained Receipt Plan for Warehouses and Stores across the Fixed Period. 2) Unconstrained Receipt Plan for Warehouses and Stores across the post Fixed period up to the end of the horizon.

Technical Details

This process is run by running the master script scrp.sh. This script initiates the batch run on global and local domains by first calling the scrp_global.sh on the global domain and then simultaneously calling the scrp_local.sh on all local domains. The global and local scripts internally invoke a set of rule groups according to the sequence described below. Each rule group is responsible for performing a specific step in the supply chain replenishment planning process.

The following scripts are used for the Replenishment/Reconciliation batch run.

Script Name	Description
scrp.sh	This script calls scrp_global.sh first on the global domain and then calls scrp_local.sh on all local domains
scrp_global.sh	Runs the batch on the global domain
scrp_local.sh	Runs the batch on local domains

The following table gives a list of all the rule groups involved in the AIP Replenishment & Reconciliation batch process.

Rule Groups	Description
Global Rule Groups	This rule group is called by the scrp_global.sh script.
scrp_pre_glb	Generates Forecast percentages, and sets store singles flag.
Local Rule Groups	The following rule groups are called by scrp_local.sh script
scrp_pre2	Runs all the pre replenishment steps for AIP.
scrp_replFixed	Runs the replenishment batch run within the fixed period and generates an unconstrained receipt plan.
scrp_invCpFxd	Restricts the unconstrained plan of stores within the fixed period to inventory caps set by user on workbook. The demand forecasted on the warehouse remains uncapped or the "True Demand."
scrp_prReconcile	Runs pre reconciliation steps for AIP.
scrp_reconcile	Runs the shortfall, SPQ and stockless to warehouse reconciliation when the demand is greater than the available inventory at the source and generates a constrained receipt plan.
scrp_invCpRec	Checks if the capped plan was reduced further due to shortfall reconciliation and adjusts the warehouse demand accordingly.
scrp_reconcSub	Runs the substitution reconciliation and substitutes alternate allowed pack sizes when there is still some unmet demand from destinations.
scrp_pushToStore	Runs the stockless reconciliation at store destinations for stockless items and pushes the excess quantity to the stores.
scrp_replPstFxd	Runs the replenishment batch run post fixed period and generates an unconstrained receipt plan for post fixed period.
scrp_invCpPstFxd	Restricts the post fixed period unconstrained plan of stores to inventory caps set by user on workbook.
WipConvOrders	Used to map warehouse to warehouse chamber.

Day on Day Processing

scrp.sh script is the master script to be called once on the global domain. It calls scrp_global.sh on the global domain first and then calls scrp_local.sh on all local domains.

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- load_rms_replenishment_measures.sh
- check_load_forecast_data.sh
- dmb_master.sh
- for_each_local_domain.sh -p copy_sister_data_local.sh [DOMAIN]

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Run dmb_master.sh
4. Restart the batch from this step.

Export Supply Chain Replenishment Data

Step Name

export_replenishment_data

Script Call

for_each_local_domain.sh -p export_scrp_inter_meas_local.sh [DOMAIN]

> **Note:** See note for Purge and Advance Low-Variability Data regarding the use of for_each_local_domain.sh and its interpretation of "[DOMAIN]."

Functional Overview

The replenishment plan's planned orders from suppliers and transfers from warehouses must be exported from populated measures into files suitable for interfacing with AIP Online. In this batch step, the exports are performed independently in each local domain, resulting in each having its own copy of the following export files placed in the local domain output directory:

strsplrord.dat	Supplier to store orders for release today through the planning horizon.
strsplrord.dat1	Contingency supplier to store orders for release tomorrow only.
strwhord.dat	Warehouse to store transfers for release today.
strwhord.dat1	Contingency warehouse to store orders for release tomorrow only.
vendor_to_wh_order.dat	Supplier to warehouse orders for release today through the planning horizon.
wh_to_wh_transfer.dat	Warehouse to warehouse transfers for release today through the planning horizon.

Technical Details

This step's command line script call is a composite of two scripts, for_each_local_domain.sh and export_scrp_inter_meas_local.sh, which together cause the exporting of order and transfer data to be performed in parallel across all local domains. At the local domain level, the process is driven by the aipcmd binary, which processes the XML resource command files to produce the parenthesized output files:

- exportPlanStrSplrOrd.xml (strsplrord.dat)
- exportPlanStrSplrOrd1.xml (strsplrord.dat1)
- exportPlanStrWhOrd.xml (strwhord.dat)
- exportPlanStrWhOrd1.xml (strwhord.dat1)
- exportPlanWhOrd.xml (vendor_to_wh_order.dat)
- exportPlanWhXfer.xml (wh_to_wh_transfer.dat)

These XML resource files all make use of the ExportPlan command to perform the specialized data exports.

Day on Day Processing

Script for_each_local_domain.sh performs the following steps:

1. Determine the list of local domains under $AIPDOMAIN.
2. For each local domain, call in parallel the script export_scrp_inter_meas_local.sh and pass it the local domain path. That path is substituted automatically for the [DOMAIN] token by for_each_local_domain.sh.

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

▪ scrp.sh

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Package Supply Chain Replenishment Data

Step Name

send_replenishment_to_online

Script Call

send_scrp_measures_to_online.sh

Functional Overview

Planned orders and transfers, exported separately per local domain in the Export Replenishment Data step, are combined, tarred, and compressed for interfacing with AIP Online.

Technical Details

In this batch step, the filenames listed in the global domain's interface/config/meas/scrp_export.config file are located in each local domain's output directory and concatenated into like-named files in the global domain's interface/export directory.

Day on Day Processing

1. Verify the existence and writeability of the global domain's interface/export directory, the destination of the concatenated export files.
2. For each of the files listed in the global domain's interface/config/meas/scrp_export.config file, delete any existing instance in the global domain's interface/export directory, then iterate over the list of local domains, concatenating each instance of the file found in the local domain's output directory

into the global interface file of the same name in global domain's interface/export directory.

3. Compress each global domain export file, package them all into a single tar file, then compress the tar file into a single file: $AIPDOMAINinterface/export/srp.tar.Z.

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

▪   for_each_local_domain.sh -p export_scrp_inter_meas_local.sh [DOMAIN]

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Load the Scaled Replenishment Plan

Run Post Supplier and Container Scaling Import

Step Name

post_ocs_import

Script Call

post_ocs_import.sh

Functional Overview

Modifications made to the warehouse plan by the Scaling module are imported so that the future projections of inventory and ordering, which are displayed in workbooks and reflected in alert calculations, are based on the final planned order quantities.

This step can be excluded from the batch if the business will not use the Scaling module within AIP. This step must not run if the corresponding AIP Oracle database extract is not run to generate the data files.

Technical Details

Day on Day Processing

1. prep_files.sh is called to process the AIP Oracle scaling export. The prep_files.sh script is invoked with the key **OCS_meas_export** listed inside $AIPDOMAIN/interface/config/bsa_prep_files.config. This processes the file ocs.tar.Z and unpacks the data contained within to the $AIPDOMAIN/interface/import/meas directory.
    ▪   Before the archive is unpacked, a backup of the file will be created in the $BSA_ARCHIVE_DIR directory. The backup filename will be ocs.tar.Z.<timestamp>.

- After the archive is unpacked, the archive file is removed from $AIPDOMAIN/interface/import.
- The data unpacked will overwrite existing files in the directory.

2. Verify that all files have been downloaded and unpacked. Failure to download any of the files will result in an error and termination of the batch. Required files are found in **post_ocs_import.config**.

3. If the SPQ Date Type measure holds the value for 'Ship Date' then the delivery date in the file containing Executed SPQ (SPEQ) quantities is converted to TODAY, the ship date. The file extension is then changed to .inc (incremental) to cause the values loaded into the same postion to be added together. This need can arise when the SPQs are applied to the Ship week and multiple delivery dates have the same release (ship) date.

4. The measures to be loaded are cleared and then loaded with data from the files.

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful ftp or copy of the ocs.tar.Z file.

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues. If necessary, copy and rename the archived version of the ocs.tar.Z file from the archive directory into the $AIPDOMAIN/interface/import directory.

3. Restart the batch from this step.

Perform Post-Replenishment Calculations

Run Replenishment Post-Processing

Step Name

run_replenishment_post_processing

Script Call

for_each_local_domain.sh -p scrp_post_local.sh [DOMAIN]

> **Note:** See note for Purge and Advance Low-Variability Data regarding the use of for_each_local_domain.sh and its interpretation of "[DOMAIN]."

Functional Overview

Replenishment post processing generates Safety Stock, Projected Inventory, Receipt Point, and Receive Up To Level values across the horizon. These values are calculated on the fly during the Replenishment and Reconciliation stage but are not written to a measure as the demand and inventory picture changes across the Fixed Period. Once the

constrained receipt plan is generated, the correct demand and inventory levels are known for the post processing to generate the above values across the horizon.

Also, when order and container scaling is implemented, it updates Receipt Plan, Forecasted Receipts, Demand Output, and Total Forecast Demand.

Technical Details

The following scripts are used for the Replenishment Post Processing batch run.

Script Name	Description
scrp_post_local.sh	This script calls scrp_post rule group on the local domains which actually does the post processing.

The following table gives a list of all the rule groups involved in the AIP Post Replenishment & Reconciliation batch process.

Rule Groups	Description
Local Rule Groups	The following rule groups are called by scrp_post_local.sh script.
scrp_post_ocs1	Copies replenishment data to the pre-scaling measures.
scrp_post_ocs2	Adds Scaling adjustments to the replenishment data.
scrp_post	Performs the post processing on local domains and generates inputs for calculating historical measures and alerts.
scrp_histmaint	Preserves data for certain measures for historical purposes which are especially useful for alerts.
scrp_cleartemp	Clears some of the stored inputs calculated in scrp_post now that its data was preserved in scrp_histmaint.

Day on Day Processing

Script for_each_local_domain.sh performs the following steps:

1. Determine the list of local domains under $AIPDOMAIN.
2. For each local domain, call in parallel the script scrp_post_local.sh script sh and pass it the local domain path. That path is substituted automatically for the [DOMAIN] token by for_each_local_domain.sh.

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- scrp.sh
- post_ocs_import.sh (if implementing order and container scaling)

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. If order and container scaling is not implmented: Restart the batch from this step.

4. If order and container scaling is implemented:
   - Determine the local domains that had an issue.
   - Determine which of the rule groups above ran successfully.
   - If none ran successfully: Restart the batch from this step.
   - If some ran sucessfaully: Re-run manually the rule groups that did not run successfully. Then restart the batch from the next step.

Calculate and Export Data Management Alerts

Run Non-critical Data Management Alerts

Step Name
run_dm_alerts

Script Call
dmb_master_alerts.sh

Required Parameters
firstTime (true | false)

Functional Overview
DM RPAS will generate alerts that are not related to new hierarchy values. These alerts are exported to DM Online to inform you that various pieces of the supply chain are incomplete or could not be defaulted by DM RPAS.

Technical Details
Now the dmb_master_alerts.sh script is called to execute the DM RPAS batch, non-critical path, including the alerts generation. As before, this process is comprised of several modules that use Korn shell scripts and C++ binaries in a particular order to calculate new measures. The order of the execution is important because some of the calculated measures are dependent on previously calculated measures.

Day on Day Processing
dmb_master_alerts.sh calls the dmb_master20.sh script, which calls dmb_master21.sh on all local domains, which contains calls (via subscripts) to several other DM batch scripts that perform the non-critical path processing of Data Management batch in the RPAS domain $AIPDOMAIN. The following table presents a high level summary of the Data Management modules that are executed along with their functionally descriptive name.

Order of Execution	Script / Operation	Description
1	mace call	Calculate Warehouse Demand
2	dmcmdstralerts.sh	Commodity/Store Alerts
3	dmmsgdatalert.sh	Missing Data Alert
4	dmnasprfalert.sh	Not Assigned to Profile Alert
5	dmdscalert.sh	Discontinuation Alert

This script calls the following scripts:

- xmace
- dmb_master20.sh

Appropriate Batch Run (First Time and/or Daily)
- First Time
- Daily

Prerequisites
This step must not be initiated until successful completion of the following:
- run_partial_dm_batch.sh (First Time run only)
- dmb_master.sh (Daily run only)

Restart/Recovery
If this script fails, perform the following steps:
1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Export DM Alerts

Step Name
export_dm_alerts

Script Call
export_dm_alerts.sh

Required Parameters
FirstTime (true | false)

Functional Overview
The Data Management Alert measures calculated by the processes of dmb_master20.sh are required for import into AIP Online. These are exported from the AIP RPAS domain and bundled for pickup by AIP Online's initial batch processes.

Technical Details
The export_dm_alerts.sh script is called. This script handles exporting the necessary AIP RPAS alert data into flat files formatted for AIP Online.

Day on Day Processing
1. Call the script export_dm_inter_meas.sh to export the intragration alert data from the calculated alert measures.
2. Package the exported alert measure data files into the file $AIPDOMAIN/interface/export/dm_alerts.tar.Z, a compressed archive that will be imported by AIP Online's initial batch step.

This script calls the following scripts:
export_dm_inter_meas.sh

Appropriate Batch Run (First Time and/or Daily)

- First Time
- Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- dmb_master_alerts.sh

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Compute Replenishment Alerts

Run SRP Item Alerts

Step Name

run_srp_alerts

Script Call

scrp_srp_alerts.sh

Functional Overview

All of the SRP alerts for out-of-stocks and excessive planned orders must be generated by AIP RPAS batch. These alerts allow the retailer to identify potential supply chain problems before they happen so that potential stock-outs and excess inventory problems can be prevented or reduced.

Technical Details

This step runs the SRP Alert calculations after the batch jobs have finished. It is recommended that this part of the system run off the critical path.

During the domain build the following SRP alerts measures are registered in the AIP domain:

Alert Name	Description
sr0_art__1	Store Alert 1: Large Consecutive Out of Stocks
sr0_art__2	Store Alert 2: Large Out of Stocks Last Night
sr0_art__3	Store Alert 3: Single Store Availability Problems
sr0_art__4	Store Alert 4: High Projected Out of Stock
sr0_art__5	Store Alert 5: Large Non-Consecutive Out of Stocks
sr0_art__6	Store Alert 6: Day on Day Repeat Out of Stocks
sr0_art__7	Store Alert 7: High Projected Low Stocks
sr0_art__8	Store Alert 8: High Planned Orders

Alert Name	Description
sr0_art__9	Store Alert 9: RDF Detail Alert
sr0_art__10	Store Alert 10: No Like SKU Found
sr0_arthstsum	Historic Availability Alert
sr0_artprjsum	Projected Availability Alerts
sr0_ovrstkalt	Overstock Alert
sr0_hrdart	High Repeated Dissipation
sr0_hdyart	High Dissipation Yesterday

Day on Day Processing

1. Determine the list of local domains under $AIPDOMAIN.
2. For each local domain, run in parallel the scrp_srpAlerts rule group.
3. Execute the scrp_preRegAlert on the global domain $AIPDOMAIN.

This script calls the following scripts:

xmace

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- scrp.sh
- for_each_local_domain.sh -p scrp_post_local.sh [DOMAIN]

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Notes

This step calculates the basics on which SRP alerts are based. Later, the replenishment_alerts_post_processing.sh will run the RPAS utility alertmgr to find all alerts registered in the $AIPDOMAIN domain.

Run WRP Item Alerts

Step Name

run_wrp_item_alerts

Script Call

wrp_item_alerts.sh

Functional Overview

The WRP alerts for unmet demand and warehouse capacity issues are generated by AIP RPAS batch. These alerts allow the retailer to identify potential supply chain problems before they happen so that potential stock-outs and excess inventory problems can be prevented or reduced.

Technical Details

This step runs the WRP Alerts after the batch jobs have finished. It recommended that this part of the system run off the critical path.

During the domain build the following WRP Item alerts measures are registered in the AIP domain:

Alert Name	Description
IpSlsCrdAltV	Sales Credit Alert
IpDmdCrdAltV	Demand Credit Alert
IpOvrStkAltV	Overstock Alert
IpRdfAltV	RDF Alert

Day on Day Processing

1. Determine the list of local domains under $AIPDOMAIN.
2. For each local domain, run in parallel the WrpItem_alert rule group.

This script calls the following scripts:

xmace

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- scrp.sh
- for_each_local_domain.sh -p scrp_post_local.sh [DOMAIN]

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Notes

This step calculates the basics on which WRP item alerts are based. Later, the replenishment_alerts_post_processing.sh will run the RPAS utility alertmgr to find all alerts registered in the $AIPDOMAIN domain.

Run WRP Network Alerts

Step Name

run_wrp_network_alerts

Script Call

wrp_network_alerts.sh

Functional Overview

The WRP Network alerts notify the user of various product flow scenarios across user-defined network of SKUS. These alerts highlight potential supply chain problems before they happen, so that potential stock-outs and excess inventory problems can be prevented or reduced.

Technical Details

This step runs the WRP Network Alerts after the batch jobs have finished. It recommended that this part of the system run after the critical path.

During the domain build the following WRP Network alerts measures are registered in the AIP domain:

Alert Name	Description
IpStkCvrAltV	Stk Cvr Alert (Act In vs. Act Out)
IpIbDODAltV	Inbound Day-on-Day Change Alert
IpDstObCpcAltV	Outbound Distribution Capacity Alert
IpObDODAltV	Outbound Day-on-Day Change Alert
IpScDODAltV	Stock Cover Day-on-Day Change Alert
IpHldCpcAltV	Warehouse Holding Capacity Alert

Day on Day Processing

1. Clear all network alert measures in $AIPDOMAIN.
2. Determine the list of local domains under $AIPDOMAIN.
3. For each local domain, run in parallel the WrpNwAlertLocal rule group.
4. Execute the WrpNwAlertGlobal on the global domain $AIPDOMAIN.

This script calls the following scripts:

xmace

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- scrp_srp_alerts.sh
- wrp_item_alerts.sh

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Notes

This script should not be run consecutively on the same day as it will affect the day-on-day change alerts.

Replenishment Alerts Post Processing

Step Name

run_replenishment_alerts_post_processing

Script Call

replenishment_alerts_post_processing.sh

Functional Overview

The WRP Network alerts notify the user of various product flow scenarios across user-defined network of SKUS. These alerts highlight potential supply chain problems before they happen, so that potential stock-outs and excess inventory problems can be prevented or reduced.

In this step, the history is updated to contain the current day's run.

Technical Details

This step finds the WRP Network Alerts after the batch jobs have finished. It recommended that this part of the system run after the critical path.

Day on Day Processing

1. Call alertmgr to find all generated alerts on $AIPDOMAIN.
2. Execute the WrpNwAlertGlobal2 on the global domain $AIPDOMAIN.

This script calls the following scripts:

xmace

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until successful completion of the following:

- wrp_network_alerts.sh

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step.

Notes

This script should not be run consecutively on the same day as it will affect the day-on-day change alerts.

Building Workbooks after Batch Run

Auto Build Workbooks

Step Name

auto_build_wkbooks_batch

Script Call

workbook_batch.sh BUILD

Functional Overview

Many of the SRP and WRP workbooks can be configured by the system administrator to be automatically built each night as part of the AIP RPAS batch run. This allows you to enter the worksheets directly without going through the workbook creation wizard. All SRP and WRP workbooks configured for commit at both global and local domain level will be committed in this step. This script may also be run from the command line ad hoc throughout the day by administrators or users.

Technical Details

This step uses the RPAS utility `wbbatch` to automatically build SRP and WRP workbooks. In order to successfully auto-build workbooks, they must be configured via the RPAS client. Refer to the *RPAS Administration Guide* for information on automatically building workbooks.

Day on Day Processing

1. Call `wbbatch` to build all workbooks on the $AIPDOMAIN global domain build queue.
2. For each local domain, call wbbatch in parallel to build all workbooks on each local domain build queue.

Appropriate Batch Run (First Time and/or Daily)

Daily

Prerequisites

This step must not be initiated until the following steps (scripts) are successful completed.

- scrp_srp_alerts.sh
- wrp_item_alerts.sh
- wrp_network_alerts.sh
- replenishment_alerts_post_processing.sh

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues.

3. Restart the batch from this step.

8

AIP Java/Oracle Batch Process

The top level control scripts (cron_export.sh, cron_import.sh, cron_import_order.sh, pre_scale.sh, scale_order.sh, post_scale.sh, cron_release.sh, and cron_purge.sh) are provided with the AIP installation. They may be used to run the AIP Oracle batch process from a UNIX scheduler. There are many other internal scripts called by these scripts that may have multi-threaded process calls; however, all the scripts called directly by cron_export.sh, cron_import.sh, cron_import_order.sh, cron_release.sh, and cron_purge.sh are run inline.

The top level control scripts like cron_export.sh, and cron_import.sh, do not run any scripts in parallel. Therefore, they should be used only as a guide for individual script calls/jobs that function similarly, but can be scheduled to run in parallel. Keep in mind, however, the existing dependencies and limitations on what can be run in parallel.

In addition to the daily batch processes which include a small amount of order purging, there is also a periodic batch purge. The data purging process is intented to release occupied tablespace by purging hierarchy and historical data that has become invalid or ineffective based upon their configured purge age. While the order purging process is scheduled to be run daily, the data purging process can be run less frequently like once a week or once a month.

Execution Sequence of AIP Java/Oracle Batch Scripts

The table below provides information about the AIP Java/Oracle batch scripts.

Script Call	Description of Action	Initial Batch Run	Daily Batch Run	Interval Batch Run
vdate.sh	Sets a virtual date that is exported for the AIP RPAS batch process	☑	☑	
cron_export.sh	Exports all the DM and hierarchy information from AIP Online to a common folder		☑	
cron_import.sh dm	Imports all the DM and hierarchy information into the Oralcle database	☑	☑	
cron_import.sh sku-attributes	Imports the SKU dimension data used for Supplier Scaling and Container Scaling.	☑	☑	
cron_import.sh om	Imports OM data for updating order details.in the Oracle database.		☑	
pre_scale.sh	Prepares database for scaling the warehouse POs that will be loaded from AIP RPAS.		☑	
cron_import_order.sh order_type_transfer dest_type_all	Imports RPAS generated transfers into AIP Oracle tables.		☑	
cron_release.sh order_type_transfer dest_type_all	Release transfers to the tsf_mfqueue table. This is a wrapper script that calls cron_release_store_order.sh and/or cron_release_non_contents_order.sh depending upon the destination parameter passed.		☑	
Cron_import_order.sh order_type_purchase dest_type_all	Imports RPAS generated purchase orders into AIP Oracle tables.		☑	
scale_order.sh	Performs Supplier Scaling and/or Container Scaling.		☑	
merge_order.sh	Moves imported warehouse POs to the final database table for Release and OM user maintenance.		☑	
Cron_release.sh order_type_purchase dest_type_all	Release purchase orders to the po_mfqueue table. This is a wrapper script that calls cron_release_store_order.sh and/or cron_release_non_contents_order.sh depending upon the destination parameter passed.		☑	
post_scale.sh	Exports modified warehouse POs and executed SPQ quantities to .dat files.		☑	
cron_post_release.sh	Gathers the table statistics on STORE_ORDER and NON_CONTENTS_ORDER table. This is a non critical script and can be run anytime after release and before daily online user activity		☑	
cron_purge.sh	Deletes closed purchase orders and/or transfers that have exceeded their defined purge age. This is a wrapper script that calls		☑	

Script Call	Description of Action	Initial Batch Run	Daily Batch Run	Interval Batch Run
	cron_purge_store_order.sh and/or cron_purge_non_contents_order.sh depending upon the order type and destination parameter passed. This script can be run in non-critical window.			
cron_import.sh alerts	Imports DM alerts generated by AIP RPAS batch.	☑	☑	
aip_purge.sh	Depending upon purge age parameters, this script purges the invalid or ineffective hierarchy and measure data from Oracle tables in order to maintain database tablespace. This can be scheduled to run less frequently like once a week or once a month etc			☑

Directory Structure

The directory structure and control scripts for the files and folders present in the AIP Java/ Oracle platform is as mentioned below.

Files and Directories	Explanation
config.xml	The RETL configuration file containing database connection information.
cron_import.sh	Executable automated import script for loading DM data, OM data, DM alerts, etc. into AIP online.
cron_import_order.sh	Executable automated import script for loading RPAS generated orders and/or transfers into AIP online depending upon order type and destination parameters passed.
cron_export.sh	Executable automated export script for exporting DM and hierarchy data from AIP online tables into files.
pre_scale.sh	Executable script for performing Scaling releated prep logic.
scale_order.sh	Executable script for performing Supplier Scaling and Container Scaling logic.
merge_order.sh	Executable script to manage the loading of warehouse POs into the permanent table. .
post_scale.sh	Executable script for performing an export of the Scaled plan from the AIP Oracle database into files.

Files and Directories	Explanation
cron_release.sh	Executable wrapper script that calls cron_release_non_contents_order.sh and/or cron_release_store_order.sh depending upon the order type and destination parameters passed.
cron_release_non_contents_order.sh	Release into-warehouse purchase orders and/or transfers from the non_contents_order table to the po_/tsf_mfqueue tables depending upon the order type parameter passed.
cron_release_store_order.sh	Release into-store purchase orders and/or transfers from the store_order table to the po_/tsf_mfqueue tables depending upon the order type parameter passed.
cron_purge.sh	Executable wrapper script that calls cron_purge_non_contents_order.sh and/or cron_purge_store_order.sh depending upon the order type and destination parameters passed.
cron_purge_non_contents_order.sh	Depending upon the order type parameter passed, this script deletes the closed into-warehouse purchase orders and/or transfers from the non_contents_order table that have exceeded their defined purge age.
cron_purge_store_order.sh	Depending upon the order type parameter passed, this script deletes the closed into-store purchase orders and/or transfers from the store_order table that have exceeded their defined purge age.
cron_post_release.sh	This script performs the post release tasks in AIP Online. The script can be run at any non critical time after orders are released and before order management online users start their activities.
tsf_po_export.sh	Exports the released transfers and purchase orders from AIP online tables into flat files.
data/	The directory that is used to store the tarred and compressed inbound data files for loading into AIP online by cron_import.sh and cron_import_order.sh scripts.
schema/	The directory of schema (xml) files used by RETL to load into and extract data from AIP online tables.
scripts/	The location of automation scripts for data processing.
scripts/aip_purge.sh	Executable table that purges invalid or ineffective supply chain data like hierarchies, historical data etc from Oracle table.
temp/	The temporary folder required for Batch Script Architecture (BSA).
config/	The location of configuration files, such as import_dm.config.
logs/	The location of log files.
archive/	The location of the archived input data files.

Prerequisites for AIP Java/ Oracle Platform

The machine running interface scripts must have the ability to schedule jobs to execute scripts on a timed schedule.

RETL version 11.2 or later must be installed and available in the PATH.

Oracle 9i or higher database schema must be installed and built with all interface procedures compiled and available.

AIP Java/Oracle Batch Process Flow

There are two phases for the AIP Java/Oracle batch process flow.
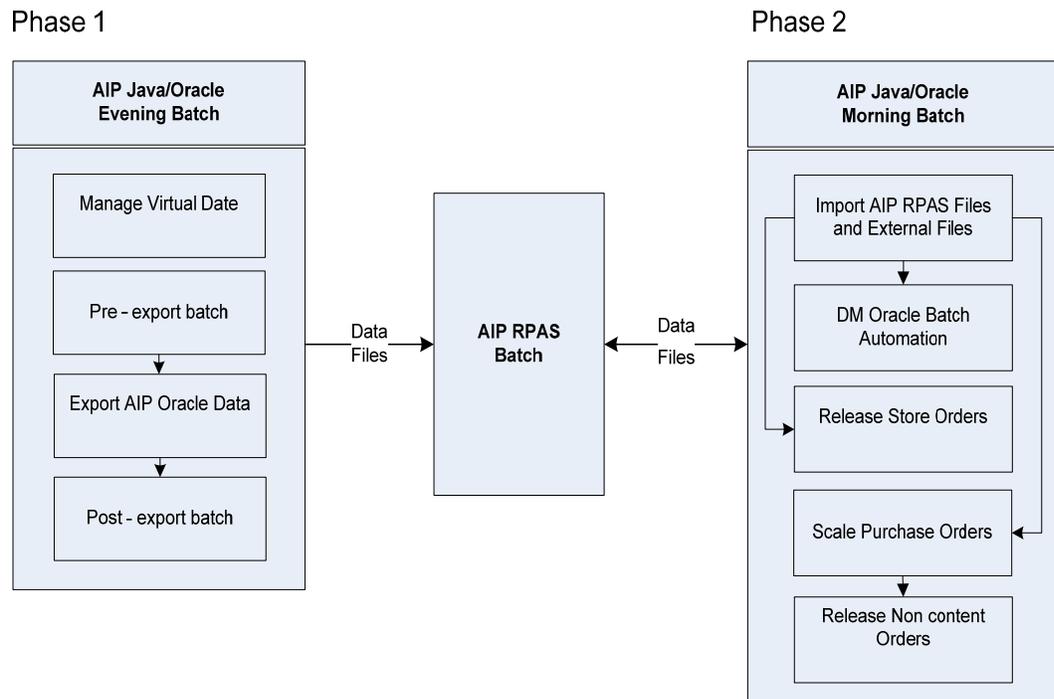
1. Phase 1 is an evening batch that is executed in AIP Java/Oracle platform before the RPAS batch run. The data files created during this batch process are moved to the AIP RPAS platform.

2. Phase 2 is a morning batch executed in the AIP Java/Oracle platform after the AIP RPAS replenishment plan is calculated and exported. During this batch process, the data files are loaded from the RPAS platform, processed and finally the orders are released to the external system. Once orders are released the Scaled purchase order quantities and executed SPQ quantities are exported for update in the RPAS database.

Export from AIP Online to RPAS

1. At the end of the business day, user-entered data from the AIP Online application must be transferred to RPAS before AIP RPAS batch begins.

2. The cron_export.sh process begins by running a pre-export, pre_aiponline_extract.sh, script. This script runs clean-up in the outbound directory for any data files from a previously failed export run. If cron_export.sh completes successfully, there should

be no data files left in the outbound directory. The presence of these files can end up creating incorrect *.tar.Z files.

3. Step 2 above is followed by a SQL query that resets required parameters in the Oracle database, including date and horizon information.

4. Data is then extracted from the appropriate tables and stored in .dat files. When the export is complete, the .dat files are compressed and will wait for retrieval by RPAS interface scripts.

5. After export is complete, the 'last_export' parameter is reset in the Oracle database.

6. A log file is generated at ${INTEGRATION_HOME}/logs

Import from RPAS and External to AIP Online

1. After AIP RPAS batch has successfully completed, the AIP RPAS interface processes generate files for loading into the Oracle-based AIP Online application. By default, these files will be compressed and placed in the AIP RPAS domain. Specifically, the files will be placed in the $AIPDOMAIN/interface/export directory. The names of the files are dm.tar.Z, dm_alerts.tar.Z, and srp.tar.Z.

2. The inbound files from the RPAS server should be retrieved by FTP using a job scheduling application.

3. Once files are successfully transferred and the scheduled job begins the cron_import.sh scripts, the three files are copied to $BSA_ARCHIVE_DIR with new file names dm.tar.Z.<timestamp>, dm_alerts.tar.Z.<timestamp>, and srp.tar.Z.<timestamp>.

4. The three files are unpacked into $ONL_INBOUND_DIR and the original archives dm.tar.Z, dm_alerts.tar.Z, and srp.tar.Z are removed.

5. Once the data files are extracted from dm.tar.Z, dm_alerts.tar.Z, and srp.tar.Z, the loading scripts cron_import.sh and cron_import_order.sh load the data from .dat files into the Oracle tables.

6. A log file is generated at ${INTEGRATION_HOME}/logs.

7. External systems (om.tar.Z, sku-attributes.tar.Z) will also provide data to be loaded directly into the Oracle database. These files must be archived, unpacked and placed in the ONL_INBOUND_DIR by a job scheduling application task or the sending application.

8. The loading script cron_import.sh, called with the appropriate parameter, will load the data from the .dat files into the Oracle tables.

Scale Purchase Orders

The scaling module will perform the necessary calculations to meet supplier minimums and/or container minimums. This module operates on the warehouse POs that are output from the AIP RPAS replenishment process.

Before scaling occurs pre-scaling preperations will ready the database for scaling prior to entering the critical batch window. Once the warehouse POs are loaded, the scaling module will pull forward orders as needed to meet the supplier minimums or container minimums. At the completion of scaling the orders which have been moved or modified by scaling will be exported out of the oracle database and archived in the ocs.tar.Z file. This file must be moved to the AIP RPAS inbound directory for the final load step.

Purchase Order/Transfer Release from AIP Online to Merchandising System

The into-store and into-warehouse PO/TSF release processes are both preceded by validation to ensure that order definitions exist, order purge periods exist, etc. Also, overdue PO/TSFs are identified. Both processes populate the PO_MFQUEUE and TSF_MFQUEUE tables, which are queried by the OrderSenderBean.

9

AIP Java/Oracle Daily Batch Scripts

AIP Java/Oracle Batch Script Overview

Functionally, the AIP batch scripts are broadly classified as mentioned below.

- Virtual Date Maintenance
- Pre-Export Batch
  * Export Planning Horizon
  * Walking Order Cycles
-

Export DM and OM Data for AIP RPAS Processing

Import Data into AIP Oracle Database

- Import DM Data and OM data
- Post DM Import Automation
- Import Purchase Orders and Transfers from AIP RPAS

Supplier Scaling and Container Scaling

Release Orders to RMS

- into-Store Purchase Orders and Transfers
- into-Warehouse Purchase Orders and Transfers

Export Order Data

- If not using RIB, export released orders into flat files.
- Export post scaled plans to AIP RPAS

Purge closed orders (daily)

The batch scripts are executed on the AIP Oracle database before and after the AIP batch execution on the AIP RPAS platform.

The batch scripts will be initiating a number of processes serially. It is accepted that they will not provide optimum parallelization and restart-ability/recovery. The restart/recovery of the batch from the highest level scripts will require a significant amount of the batch to be rerun. The operator has the option of manually running the lower level scripts from the failing script onward, however that is generally not desirable in production. For this reason, the second level scripts are modularized such that they can be executed by the job scheduler, with the appropriate dependencies, and run individually (followed by the scheduled dependent process(es)).

Virtual Date Maintenance

The entire batch process will be run for the next business day. To guarantee that the whole batch process uses the same date, and is not affected by a potential change in calendar days on the server, a virtual date will be used.

The virtual date, also referred to as the batch run date or vdate, will be set immediately prior to the commencement of the batch. If the scheduled start time is prior to midnight the virtual date will be one day ahead of the system date. If the scheduled start time is after midnight the virtual date will match the system date. On normal, day-to-day batch

runs the virtual date can simply be incremented by one day to achieve the proper batch run date (regardless of scheduled start time).

A script is used to maintain the vdate. The maintenance of the vdate can be easily accomplished as a scheduled daily task and as a manual task.

The system operator will schedule the increment of the vdate on a daily basis.

In the event of a batch failure all or portions of the batch may be re-run without changing the virtual date. No other batch process will attempt to update or maintain the vdate. The scheduler and system operator are solely responsible for this maintenance.

The system operator will also be able to manually set the vdate. Any attempt to set an invalid date will result in an error being logged; the date will not be updated. If the operator updates the date directly rather than via the vdate.sh script provided any attempt to retrieve or use the invalid date will result in an error that halts the calling process and all dependant processes.

All batch processing dependant on dates (effective dates or end dates) will consider the vdate as the current effective date. The export planning horizon and any other calculated time periods (i.e. sister store/warehouse copy, walking order cycles, etc.) will be calculated based on the vdate as the current effective date.

Script Name: vdate.sh

This script is a wrapper to the vdate functions, the get_vdate, set_vdate and inc_vdate from the package aip_util.

> **Note:** The vdate.sh script should not be used to set or increment the vdate more than once in a single, complete batch run.

Input Parameters

This script will accept the following commands:

1. Action: [set | get | inc]
2. Export: [noexport | export]
3. Date: [YYYYMMDD] (only accepted for the 'set' action)

set [date in YYYYMMDD]
> This function performs the following:
> - calls the SET_VDATE function (using call_bsa_sql.sh)
> - If export= true, write vdate value to text file.

get
> This function calls the GET_VDATE function (using call_bsa_sql.sh).

inc
> This function performs the following:
> - calls the INC_VDATE function using call_bsa_sql.sh)
> - If export=true, write vdate value to text file using GET_VDATE function output.

Example:

Script Call	Explanation
vdate.sh	retrieves the vdate

Script Call	Explanation
vdate.sh get	retrieves the vdate
vdate inc	increments the vdate, without transferring it
vdate inc noexport	increments the vdate, without exporting it
vdate inc export	increments the vdate and exports it
vdate set noexport 20001130	sets the vdate, without exporting it
vdate set export 20001130	set the vdate and exports it

Pre-Export Batch

Prior to physically exporting the DM Oracle data out of the tables the calculation of the walking order cycles lead time will be performed along with the export planning horizon start and end date update.

Export Planning Horizon

The export planning horizon is a calculated planning horizon value that is used to limit the amount of data exported to AIP RPAS. The export planning horizon start date is equal to the virtual batch run date (vdate). The export planning horizon end date is equal to the start date plus the calculated maximum export planning horizon value. These values will be used throughout the extract logic to limit the extracted data to that which falls within the export planning horizon start and end dates. In addition to calculating maximum planning horizon, individual SKU planning horizon is also calculated in this step. Currently SKU planning horion is used only for store source export. The horizon itself is exported to AIP-RPAS for use in AIP-RPAS batch processes.

It is important to note that this export planning horizon does not control replenishment. However the data that is exported directly effects the replenishment plan batch calculates. Therefore, it is pertinent that all data is exported for the duration of its corresponding planning horizon.

Walking Order Cycles

Walking order cycles are used to drive stock into a new store prior to its opening. A profile/store release schedule exception is created with a special empty order cycle to wipe out all lead times for the store. Then, a second calculated release schedule lead time exception is created. This is the 'walking order cycle lead time' which is recalculated each day.

Because the user will have the opportunity to create new Profiles during the online day the profile/store release schedule exception must be created for the new profiles after the online day. The profile/store release schedule exception will be created for all new profile/store combinations prior to the export of DM Online data to AIP RPAS.

Prior to exporting the DM Online data for running the AIP RPAS batch calculations the 'walking order cycle lead time' will be calculated for the current batch run date (the next business day). This lead time is saved as a SKU/store release schedule exception.

Build Scalable Assignments

If scaling functionality is turned ON then scalable assignments are identified and set up in preparation for future scaling run. One of two global scaling flags needs to be turned

ON. Active scaling assignment are considered scalable depending upon the local (scaling group level) flags, scaling horizons and scaling constraints.

Scalable assignments from last export run are deleted and new scalable assignments for upcoming scaling run are set in 'scalable_assignment' table.

During export, a list of distinct vendor/sku from scalable assignments is exported to AIP-RPAS so that AIP-RPAS can return warehouse release schedule and other required information only for the vendor-sku listed in the file.

Script Call

These steps are performed just prior to the physical export of data but are executed by the same control script which triggers the exports. The control script is described in detail in the next section.

Export DM and OM Data (cron_export)

Functional Overview

The DM Online and Order Management applications are used to enter and maintain AIP supply chain configurations, replenishment parameters, and purchase orders. While all of the user entered data is mastered in the Oracle database, it must ultimately be used to generate the desired replenishment plans.

At the end of each business day (or online day), the DM and OM application server is brought down to prevent user modifications during the batch run. Once the user modifications are halted, all of the new and modified data entered on that day must be extracted to a set of flat files that follow an agreed AIP RPAS and AIP Java/Oracle file format.

Technical Details

cron_export is the wrapper script executedto export all the data from DM for the AIP RPAS platform. It performs the following activities:

    a.   Calls the pre_aiponline_extract.sh script to do the following activities.

           Remove all the .dat & .int files in the ONL_OUTBOUND_DIR directory.

           Call the pre_extract_wrapper function from the aip_util package.

    b.   Processing and archiving hierarchy and DM data

           The script loops through the following configuration files and executes each export script listed in the file:

           $INTEGRATION_HOME\config\export_hierarchy.config

           $INTEGRATION_HOME\config\export_dm.config

The following files are needed to export data out of the AIP Oracle schema:

    Export Script ($INTEGRATION_HOME/scripts/export) – This folder contains the export SQL query.

    Schema File ($INTEGRATION_HOME/schema) - This defines the .dat file format using xml tags.

    Config File ($INTEGRATION_HOME/config) - This lists the .sh export scripts that can be executed in parallel.

Two key tables are used to extract data from the Oracle tables:

INTERFACE_PARAMETER

The INTERFACE_PARAMETER table holds the timestamp of the last successful run of the cron_export.sh.

This is how RETL knows what was added or modified since the last extract. In many cases, only modified data is extracted.

INTERFACE_HORIZON_DATE_LOOKUP

The INTERFACE_HORIZON_DATE_LOOKUP table has one row for each day starting today through the entire planning horizon (as calculated in the pre-export logic).

This table is used as a join in the SQL statement within the export script.

It is used to blow out the data to the daily level or to ensure that the data being exported falls within the planning horizon.

c. Updating Interface Parameters

The update_interface_param.sh script updates the INTERFACE_PARAMETERS table to reflect whether WIP was activated as of this export and what the timestamp was at the end of this export.
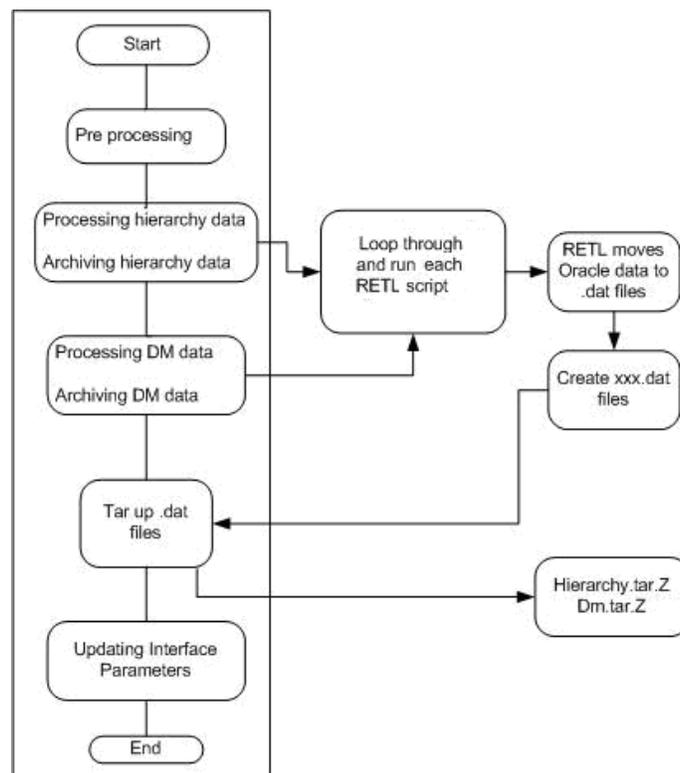
Export Flow Diagram

Script Name: cron_export.sh

Input Parameters

noTimestamp – An optional parameter that indicates whether to update the INTERFACE_PARAMETER table or not. By default, the table will be updated unless the value noTimestamp is entered to prevent the LAST_EXPORT value from being updated.

Requirements

This script requires a configuration file containing a list of export scripts.

Day on Day Processing

The script exports data from the Oracle database to AIP RPAS. The script is designed to execute automatically by a scheduled CRON job every night after the online day and before AIP RPAS Batch is run.

The script will first call pre_aiponline_extract.sh to perform pre-export updates. Next, each export script is invoked. After the data is successfully extracted, the script updates the timestamp on the INTERFACE_PARAMETERS table.

Processing Steps

    a. Execute pre-extract setup and maintenance

The pre_aiponline_extract.sh script sets up the planning horizon days which are used throughout the extract logic and maintains/decrements the walking order cycle exceptions.

    b. Processing Hierarchy Data

Use the process_aiponline_data.sh script with the input parameter as export_ hierarchy.config to process the hierarchy data.

Input Parameters

export_ hierarchy.config

Day on Day Processing

Each export script within the configuration file makes a single call to RETL with a database connection parameter. The export scripts will contain the export SQL query. When more than one file is being generated, the export script will contain and one query for each unique file name being generated.

Hierarchy data included in the export_hierarchy.config file are

    delivery_groups

    network_groups

    order_cycle

    order_groups

    profile

    sku_pack

    store_order_cycle

    warehouse

    c. Archive Hierarchy output files

        After processing the hierarchy data using the previous step, the data is archived by converting the hierarchy (.dat) files into tar files.
        For example, hierarchy.tar.Z file.

    d. Build scalable assignments

Use PL/SQL stored procedure to build a new set of scalable assignments if scaling is turned ON.

    e. Processing DM Data

        Use the process_aiponline_data.sh script with the input parameter as export_ dm.config to process the DM data.

Input Parameters

export_dm.config

Day on Day Processing

Each export script within the configuration file makes a single call to RETL with a database connection parameter. The export scripts will contain the export SQL query. When more than one file is being generated, the export script will contain one query for each unique file name being generated.

DM data included in the **export_dm.config file are**

  assigned_commodity
  chamber_product_type
  commodity_order_cycle_exceptions
  default_order_cycle
  delivery_group_assignment
  delivery_pattern
  demand_group_data
  department_profile
  direct_store_format_pack_size
  direct_store_pack_size
  direct_suppliers
  keep_together
  non_delivery_date
  non_delivery_date_exceptions
  non_order_date
  non_order_date_exceptions
  non_receipt_date
  non_release_date
  non_release_date_exceptions
  on_supply_off_supply
  order_cycle
  order_group_assignment
  order_multiple
  orderable_unit
  pallet_multiple
  placement_schedule_exceptions
  planning_horizon
  profile_network_groups
  profile_order_cycle
  profile_order_cycle_exceptions
  profile_placement_schedule_exceptions
  profile_release_schedule_exceptions
  ranged_commodity_pack_size
  rdc_reconciliation
  receipt_to_availability

release_schedule_exceptions

singles_enabled_warehouse

sister_store

sister_wh

source_split_tb

stockless_indicator

store_calendar

store_format_pack_size

store_order_cycle

store_order_cycle_exceptions

store_pack_size

store_planning_horizon

store_priority

store_source

supplier_order_cycle_assignment

valid_warehouse

vendor_lock

vendor_sku_to_scale

warehouse_coupled_flag

warehouse_promotional_dates

warehouse_rollout_status

warehouse_schedule

f. Archive DM output files

After processing the DM data using the previous step, the data is archived by converting the DM (.dat) files into tar files. For example, dm.tar.Z file.

g. Update the Interface Parameters

The update_interface_param.sh script updates the INTERFACE_PARAMETERS table to reflect whether WIP was activated as of this export and what the timestamp was at the end of this export.

Due to the fact that some data is extracted and/or maintained at high volumes that data will only be exported when changes occur. In order to identify changes that occur since the last export a last export timestamp is maintained. This timestamp is updated immediately after all export data files have been successfully created and compressed.

All subsequent updates made in the DM Online application, or as part of the AIP Oracle Morning Batch will have a later timestamp and will therefore be picked up in the next export.

Day on Day Processing

When no parameter is passed into cron_export.sh the last_export timestamp is updated to the current date and time. When the noTimestamp parameter is passed to cron_export.sh the last_export timestamp is not updated.

The WIP indicator is also updated to reflect the value of the HAVE_WIP environment variable during the export. This value must always be false.

Prerequisites

This step should be run after the DM and OM Online application is brought down for the day. This will prevent any opportunity for you to add or modify data between the time that data is extracted from a table and the last export timestamp is updated. Such an occurrence could result in AIP Oracle and AIP RPAS getting out of sync.

The data extracted also depends on the INTERFACE_HORIZON_DATE _LOOKUP table and the INTERFACE_PARAMETER table. If data is not extracted as expected, ensure that the days_in_export_horizon value in the Config package is set correctly to cover the longest planning horizon in the AIP system.

The data files that do not contain a full refresh of the data are controlled by the last_export timestamp value. The timestamp on the row being extracted is compared to the last_export timestamp. If the timestamp on the row is later than the last_export value, it was added or modified since the last extract occurred.

This step must run before the AIP RPAS batch step **prep_onl_data**.

Restart/Recovery

If this step fails, perform the following:

1. Review the log file, which is located in the 'logs' folder of the directory that is specified for the Oracle LOGHOME environment variable.

2. Perform the necessary corrective actions.

3. Re-run cron_export.sh.

Import Data into AIP Oracle Database

Load Data

Much of the AIP supply chain management is performed in Data Management Online, which is built on an Oracle database. The RMS and RPAS data is imported into DM Oracle as part of the final steps of the AIP nightly batch.

The import includes foundation data and the orders that were produced by AIP RPAS batch. Once imported, today's orders are Scaled and immediately released to RMS by OM batch. The release to RMS serves to execute the orders or to communicate them to the source warehouses and suppliers.

Import Overview

Upon completion of the batch jobs, RPAS generates flat files (.dat data files) that contain the hierarchy and measure data that is required in DM and OM Oracle Online. Data files exported from RPAS should then be retrieved by copy or FTP using a job scheduling application. Once the Oracle import process begins, the data is loaded into the Oracle database by using RETL (Oracle Retail Extract, Transformation, and Load). For this step, only the Load part of RETL is used.

Importing Flow

Staging tables, prefaced with "INTERFACE_" or "i_", temporarily store the records from the .dat file in the Oracle database. RETL then invokes a stored procedure to move the data from the staging table to the base table.

The following two files are needed to move the flat file to the staging table using RETL:

Import Script ($INTEGRATION_HOME/scripts/import)

Schema File ($INTEGRATION_HOME/schema)
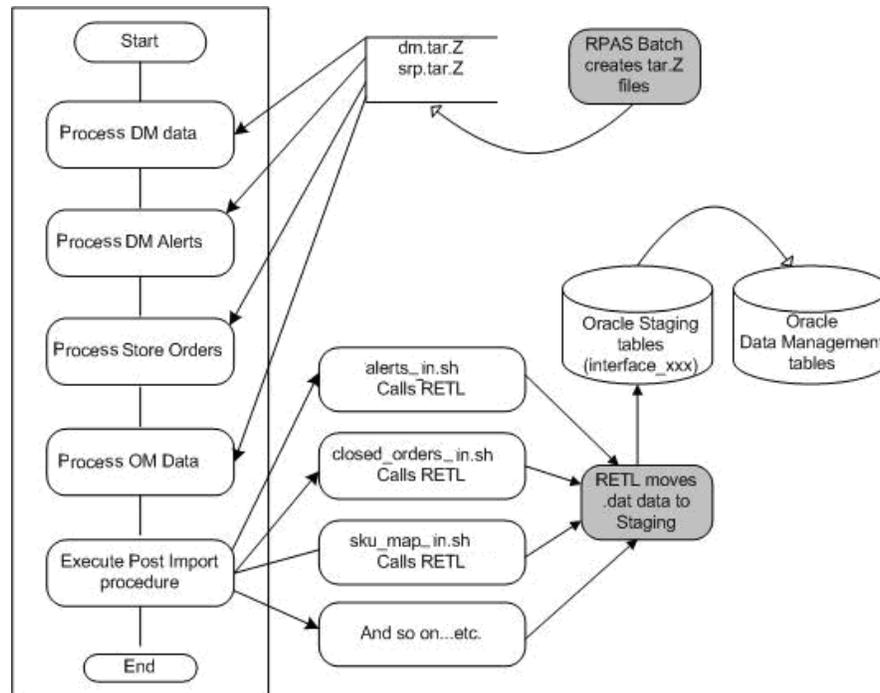
Technical Details

The cron_import.sh script currently performs two main functions:

1. Import Hierarchy, DM, SKU attributes, DM Alerts, and OM data into the AIP Online database (process_aiponline_data.sh).

2. Call any post-load processing if necessary (post_import_wrapper.sh).

This script is a wrapper for a series of scripts.

The importing of data into AIP Online can actually run as five independent processes, as follows:

Import Data Management files

Retrieve the file containing DM data

Call the import processing script with the "hierarchy" config file

Call the import processing script with the "dm" config file

Call the post-load SQL processing script to execute the Automated Data Maintenance batch scripts.

Import SKU Attribute data

Retreive the file container SKU Attributes from the import directory

Call the import processing script with the "sku-attributes" config file.

Import Order Management data

Retrieve the file containing OM data

Call the import processing script with "om" config file

Import Data Management Alerts

Retrieve the file containing DM Alerts data

Call the import processing script with "dm alerts" config file

Script Name: cron_import.sh

The script cron_import.sh is executed to process the files by passing the appropriate parameters.

Input Parameters

Module name: [dm|alerts|om|sku-attributes]

Restart / Recovery

The import process will execute many import scripts in parallel. If this fails, it will be because one or more of the import scripts failed. After examining the logs to determine which scripts failed, and after correcting the errors, this process can be completed in a number of ways:

1. Simply restart the import processing. Since successful import scripts will delete their input files, the entire import process can be run again. Missing input files will only generate warnings and continue.

2. Execute individual import scripts from the command line.

3. Create a config file containing the list of only the import scripts that have not completed successfully. Execute the import process using this config file.

If an error occurs in the post-load SQL processing script, then once the errors have been corrected that script can simply be re-executed.

DM Post-Load Batch

Script: post_import_wrapper.sh

The DM scripts that are executed by cron_import.sh after the AIP Oracle load scripts are called DM Post-Load batch. It is also referred to as Automated Data Maintenance Batch.

The following descriptions of each process provide an overview. Additional validation and restrictions may be applied that are not detailed in the overview.

Create Delivery Groups

The automatic creation of delivery groups is triggered by a new supplier arriving in the AIP system. When the new supplier has a 'ship-to' value defined it will be compared against a mapping table. The mapping table maps 'ship-to' values to sources/source warehouse types and destination warehouse types. One delivery group will be created for each source that matches the source type. The chambers assigned to warehouses that match the 'ship-to' destination warehouse types will be assigned as the delivery group scheduling locations.

If the new supplier supplies SKU-packs that already exist in the system (SKU-packs are multi-supplied by another existing supplier) the demand group(s) of the SKU-packs will be immediately assigned to the delivery group.

Create Order Groups

The automatic creation of order groups is triggered by a new supplier arriving in the AIP system. When the new supplier has a 'ship-to' value defined it will be compared against a mapping table. The mapping table maps 'ship-to' values to sources/source warehouse types and destination warehouse types. One order group will be created for each source and destination warehouse-chamber combination assigned to the 'ship-to' source type and destination warehouse-type. The chambers assigned to warehouses that match the 'ship-to' destination warehouse type(s) will be assigned as the order group automation scheduling locations.

If the new supplier supplies SKU-packs that already exist in the system (SKU-packs are multi-supplied by another existing supplier) the demand group(s) of the SKU-packs will be immediately assigned to the order group.

Create Profiles

The automatic creation of profiles is triggered by a new supplier arriving in the AIP system. The new supplier will trigger the creation of a new direct profile by DM Oracle batch. The new direct profile will be assigned a specific default order cycle which is created at implementation time.

> **Note:** The Automatic creation of warehouse profiles will occur in the AIP RPAS batch.

Assign Profiles

The assignment of SKUs to profiles must be performed after the automatic creation of profiles. The automatic assignment of SKUs to profiles is triggered by a new Supplier/SKU-pack size combination. The SKUs which are identified as newly supplied by a supplier will be assigned to that supplier's direct profile. If the supplier does not have a direct profile, and the SKU is not assigned to any profiles, the SKU will be assigned to the Default (Class) Profile where defined.

> **Note:** The automatic assignment of SKUs to warehouse profiles is performed in AIP RPAS batch and is triggered by a new SKU.

Demand Group and SKU Group Maintenance

New SKU-pack sizes

All new SKU-pack sizes that arrive in the AIP system are assigned to a single default demand group and SKU group. These SKU-pack sizes cannot be replenished in the warehouse until they are assigned to the proper demand group and SKU group.

A new SKU group will be created for each new SKU.

The assignment of new SKU-packsizes to demand groups is determined by a configuration option. This option, set at implementation time, determines if the new packsizes of a SKU should be in a single demand group for that SKU, or each packsize should have its own demand group.

The default configuration is to assign all of a SKU's pack sizes to the same demand group.

An additional parameter – the Inventory Tracking Level – can override the demand group assignment configuration option. The Inventory Tracking Level is a global string value set at implementation time. When it is set to "Eaches" the demand group assignment configuration option mentioned above is over-ruled, and all new packsizes of a SKU must be assigned to a single demand group for the SKU.

Pre-priced

All SKU-pack sizes that are pre-priced/value-added will have a parent SKU defined. The pre-priced/value-added SKUs must be assigned to the same SKU group and demand group as their parent SKU.

Any pre-priced/value-added SKU which is assigned to a SKU group other than its parent's SKU group will be updated to reflect a SKU group assignment equal to the parent's SKU group.

Any pre-priced/value-added SKU-pack size which is assigned to a demand group other than its parent's demand group will be updated to reflect a demand group assignment equal to the parent's demand group. Because a SKU's pack sizes may be assigned to multiple demand groups the first available parent demand group (when ordered by the demand group code) will be used for all of the pre-priced/value-added SKU-pack size assignments.

Standard SKUs

All SKUs that have changed status from pre-priced/value-added to a standard SKU must be assigned to their own demand group and SKU group.

A new SKU group will be created for each SKU which is now a standard SKU.

A new demand group will be created for each new standard SKU or SKU-pack size (depending on the configuration setting)

Clean-up

When SKU-packs become pre-priced/value-added, they are moved into their parent's demand group and SKU group. Moving SKU-packs out of an existing demand group may result in an empty demand group. All demand groups which have no SKU-packs assigned to them, with the exception of the default demand group, will be deleted.

Range SKU-pack sizes

Automatic ranging is triggered by a new SKU-pack size. Any SKU-pack size that was not in the AIP system in the previous batch run will be considered new. The new SKU-packs are ranged to warehouses based on a configurable system parameter setting that allows ranging based on the SKU's supplier's "ship-to" value and delivery group and order group destinations, or all valid warehouses. The ranging rules applied in the DM Online application are also applied to the batch process.

Delivery Group Assignment

Automatic delivery group assignment is triggered by a new SKU-pack size. The assignment of demand groups to delivery groups must be performed after the automatic creation of delivery groups, after SKU group and demand group maintenance, and after automatic ranging.

New SKU-pack sizes' demand groups are assigned to delivery groups via their supplier's "ship-to" value. All delivery groups with sources that match the supplier's "ship-to" source value will be retrieved. The new SKU-pack sizes' demand groups will be assigned to all delivery group source/scheduling location combinations which are valid based on supplier/SKU-pack size links and ranging.

Order Group Assignment

Automatic order group assignment is triggered by a new SKU-pack size. The assignment of demand groups to order groups must be performed after the automatic creation of order groups, after SKU group and demand group maintenance, and after automatic ranging.

New SKU-pack sizes' demand groups are assigned to order groups via their supplier's "ship-to" value. All order groups with sources that match the supplier's "ship-to" source value will be retrieved. The new SKU-pack sizes' demand groups will be assigned to all order group source/scheduling location combinations which are valid based on supplier/SKU-pack size links and ranging.

Reset Store Format Pack Size (WH and Direct to Store Format)

This process should be run after Range SKU-pack sizes to ensure all available pack sizes are considered for replacing the existing pack size.

The store format ordering pack size default is reset when the current pack size has a discontinuation date equal to the vdate. The new pack size is selected in a pre-determined order based on the SKU-pack size pack type. The first valid available pack type that is not discontinued will become the new ordering pack size default for the store format. If no alternative pack size is found the ordering pack size is not reset.

Reset Store Pack Size (WH and Direct to Store)

This process should be run after Range SKU-pack sizes to ensure all available pack sizes are considered for replacing the existing pack size.

The store ordering pack size exception is reset when the current pack size has a discontinuation date equal to the vdate. The new pack size is selected in a pre-determined order based on the SKU-pack size pack type. The first valid, available pack type that is not discontinued will become the new ordering pack size exception for the store. If no alternative pack size is found the ordering pack size is not reset.

Set Missing Store Format Pack Size (WH and Direct to Store Format)

This process should be run after Range SKU-pack sizes to avoid a day lag in creating store format pack sizes for new SKU-pack sizes.

The default store format pack size will be automatically selected for all valid source/SKU/store format combinations. The pack size is selected in a pre-determined order based on SKU-pack size pack types. The first valid, available pack type that is not discontinued will become the ordering pack size default for the store format.

This process runs nightly to pick up all valid source/SKU/store format combinations that do not currently have a default ordering pack size defined.

Reset Warehouse Orderable Unit

This process should be run after Range SKU-pack sizes and Demand Group and SKU Group Maintenance to ensure all available pack sizes are considered for the new orderable unit.

The warehouse orderable unit is reset when the current pack size has a discontinuation date equal to or before to the vdate. The new pack size (orderable unit) is selected in a pre-determined order based on the SKU-pack size pack type. The first valid available pack type that is not discontinued will become the new orderable pack size default for the source/demand group/SKU/warehouse. If no alternative pack size is found the orderable pack size is not reset.

Set Missing Warehouse Orderable Unit

This process should be run after Range SKU-pack sizes and Demand Group and SKU Group Maintenance to avoid a day lag in creating warehouse orderable units for new SKU-pack sizes.

The warehouse orderable unit will be automatically selected for all valid source/demand group/SKU/warehouse combinations. The pack size (orderable unit) is selected in a pre-determined order based on SKU-pack size pack types. The first valid, available pack type that is not discontinued will become the orderable pack size for the source/demand group/SKU/warehouse.

This process runs nightly to pick up all valid source/demand group/SKU/warehouse combinations that do not currently have a warehouse orderable unit defined.

Set Order Multiple

This process should be run after Range SKU-pack sizes to avoid a day lag in creating order multiples for new SKU-pack sizes.

The order multiple will be automatically selected for all valid source/SKU/pack size/warehouse-chamber combinations. The order multiple is an integer value selected from the available pack sizes for the SKU. The pack size value that will be used as the order multiple is selected in a pre-determined order which is based on pack types. The fist valid, available pack type that is not discontinued will become the order multiple for the source/SKU/pack size/warehouse-chamber.

This process runs nightly to pick up all valid source/SKU/pack size/warehouse-chamber combinations that do not currently have an order multiple defined.

Set Stacking Flag

This process should be run after Range SKU-pack sizes to avoid a day lag in setting stacking flag values for new SKU-pack sizes.

The stacking flag will be automatically set for all valid source/SKU/pack size/warehouse-chamber combinations. The default stacking flag value is configurable at implementation time. This process runs nightly to pick up all valid source/SKU/pack size/warehouse-chamber combinations that do not currently have a stacking flag value defined.

Because the stacking flag is only used for truck building purposes this automation can be turned off is WIP is not enabled. Note that because this process will pick up all missing values suddenly enabling this logic on a full production set of data will cause a significant increase in the batch run time.

Set Case Weight

This process should be run after Range SKU-pack sizes to avoid a day lag in setting case weight values for new SKU-pack sizes.

The case weight will be automatically set for all valid source/SKU/pack size/warehouse-chamber combinations. The default case weight value is configurable at implementation time. This process runs nightly to pick up all valid source/SKU/pack sizes/warehouse-chamber combinations that do not currently have case weight value defined.

Because the case weight is only used for truck building purposes this automation can be turned off is WIP is not enabled. Note that because this process will pick up all missing values suddenly enabling this logic on a full production set of data will cause a significant increase in the batch run time.

Set Pallet Multiple

This process should run after order multiples are set to prevent a day lag in setting the pallet multiple.

A pallet multiple will be automatically set for all source/SKU/pack size/warehouse-chamber combinations that have an effective order multiple on the batch run date and which do not have an effective pallet multiple defined.

Create Time Balanced Source Splits

This process should run after delivery group assignment and order group assignment to prevent a day lag in creating the time-balanced source splits.

Time balanced source splits will be created for every demand group/warehouse combination that has a supplier sourced delivery group and order group assignment. The first supplier will receive 100% of the source split percent. If more than one supplier supplies SKU-pack sizes in the demand group an alert is created to indicate that the user should review the accuracy of the source split created.

Copy Sister Store

The sister store copy logic uses a combination of the new store open date and a configurable system parameter to determine when the sister store copy will occur. When a new sister store open date is received an alert is generated to indicate that a new store open date was received. This alert will also include an expected copy date. A copy will only occur once for a particular new store. Sister store copies will occur as soon as their calculated copy date is reached, and will result in the supply chain of the 'copy to' store being a replica of the "copy from" store.

> **Note:** Users and custom processes must NOT set up the supply chain in the DM Online application for a new store with a pending sister store copy. Doing so will result in a failure of the copy. The only recourse is for the batch operator to manually delete the data pertaining to the new store or the user must manually complete the entire supply-chain setup.
>
> Because the only recourse for a failed copy is manual correction or setup it is highly recommended that the introduction of new stores with a sister store copy be a tightly controlled process. To control the process the introduction of the new stores should not occur until the store open date falls within the copy timeframe. To additionally safeguard this process the "sister store offset weeks" system parameter can be set to an arbitrarily large number that is less than the maximum export horizon (converted to weeks).

Copy Sister Warehouse

The sister warehouse copy logic uses a combination of the new warehouse open date and a configurable system parameter to determine when the sister warehouse copy will occur. When a new sister warehouse open date is received an alert is generated to indicate that a new warehouse open date was received. This alert will also include an expected copy date. A copy will only occur once for a particular new warehouse. Sister warehouse copies will occur as soon as their calculated copy date is reached, and will result in the supply chain of the 'copy to' warehouse being a replica of the 'copy from' warehouse.

> **Note:** Users must NOT set up the supply chain in the DM Online application for a warehouse with a pending sister warehouse copy. Doing so will result in a failure of the copy. The ONLY recourse is for the batch operator to manually delete the data, pertaining to the new warehouse, from the database or the user must manually complete the entire supply chain setup.
>
> No custom processes can be created that create chambers or set ranging status for warehouses with a pending sister warehouse copy. Doing so will result in a failure of the copy. All data that is created as a result of these actions must be manually deleted from the database in order for a successful copy to occur. Otherwise the user must manually complete the supply chain setup.
>
> Because the only recourse for a failed copy is manual correction or setup it is highly recommended that the introduction of new warehouses with a sister warehouse copy be a tightly controlled process. To control the process the introduction of the new warehouse should not occur until the warehouse open date falls within the copy timeframe. To additionally safeguard this process the 'sister warehouse offset weeks' system parameter can be set to an arbitrarily large number that is less than the maximum export horizon system parameter (converted to weeks).

Import Orders

The receipt plan that is calculated by AIP RPAS will be Scaled by the AIP Oracle batch and then made visible in the AIP Order Management (OM) Online application. The plan is exported from AIP RPAS upon completion of the replenishment calculations. The export configuration controls how much of the plan is exported and is therefore available to load into the Oracle database. The exorted plan is translated into Purchase Orders (POs) and Transfers that are sent to RMS and can be viewed and maintained.

Technical Details

The cron_import_order.sh script loads the RPAS generated orders from .dat files into AIP Online tables.

This script is parameterized so that critical orders can be loaded and released first. The non-critical orders can be loaded afterwards by passing in different parameters.

This script can be used to load the following combinations of orders:

1. Into-Store Purchase Orders **only**

2. Into-Store Transfers **only**

3. Into-Warehouse Purchase Orders **only**

4. Into-Warehouse Transfers **only**

5. Into-Store Purchase Orders **and** Into-Store Transfers (i.e. **all** Into-Store Orders)

6. Into-Warehouse Purchase Orders **and** Into-Warehouse Transfers (i.e. ALL into-Warehouse Orders)

7. Into-Store Purchase Orders **and** Into-Warehouse Purchase Orders. (i.e. **all** Purchase Orders)

8. Into-Store Transfers **and** Into-Warehouse Transfers (i.e. **all** Transfers)

9. **All** orders (this includes Into-Store Purchase Orders **and** Into-Store Transfers **and** Into-Warehouse Purchase Orders **and** Into-Warehouse Transfers)

Before loading new into store orders or into store transfers or into warehouse transfers, the existing forecast orders/transfers are removed from the main order tables (store_order and non_contents_order). Before loading new into warehouse purchase orders, all existing orders are removed from warehouse_purchase_order table.

When loading into-store purchase orders or into-store transfers, this script calls a subscript pre_aiponline_store_orders.sh that deletes the forecasts from previous load and disables the referential integrity constraints. This step is performed to improve the load performance. The constraints are later enabled after the load by calling post_aiponline_store_orders.sh script.

This script begins with fetching and extracting the order files from the compressed tar file if the tar file is available in the "/data" folder. Then, depending upon the order_type and dest_type parameters passed into this script, it loads the appropriate order files.

Script Name: cron_import_order.sh

Input Parameters

1. order_type: [order_type_transfer | order_type_purchase | order_type_all]. This is a required parameter.

   order_type_transfer: If only transfers are to be loaded.

   order_type_purchase: If only purchase orders are to be loaded.

   order_type_all: If both transfers and purchase orders are to be loaded.

2. dest_type: [dest_type_store | dest_type_warehouse | dest_type_all]. This is also a required parameter.

   dest_type_store: If only into-store orders are to be loaded.

   dest_type_warehouse: If only into-warehouse orders are to be loaded.

   dest_type_all: If both into-store orders and into-warehouse orders are to be loaded.

Using a combination of order_type and dest_type parameters, the desired load and its priority in the critical batch window can be easily configured.

Examples:

- cron_import_order.sh order_type_transfer dest_type_all
  This will load into-store transfers and into-warehouse transfers

- cron_import_order.sh order_type_purchase dest_type_store
  This will load into-store purchase orders only

- cron_import_order.sh order_type_all dest_type_all
  This will load into-store purchase orders, into-store transfers, into-warehouse purchase orders, and into-warehouse transfer

Restart / Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step. Since successful import scripts will delete their input files, the entire import process can be run again. Missing input files will only generate warnings and continue.

Scale Purchase Orders

As soon as warehouse purchase orders have been loaded the scaling logic can be executed. Scaling is not required by all retailers. This logic is optional depending on the business need. The execution of all or portions of this logic can also be controlled by global system parameters that disable Supplier Scaling or Container Scaling.

Pre-Scaling

Technical Details

Prior to scaling some system parameters are set that aid scaling. This script first gathers table statistics on important tables used in scaling. It then sets a system parameter to indicate whether or not any release date has multiple possible delivery dates. If quantity constraints are specified in terms of pallets, it then checks for the existence of pallet multiples and remove the invalid assignments from the list of scalable assignments. Lastly it sets a system parameter to indicate whether or not a scaled Supplier/SKU/warehouse has an Order Multiple that changes in the future.

Script Name: pre_scale.sh

Input Parameters None

Restart / Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the script.

Supplier and Container Scaling

Supplier Scaling

Supplier scaling occurs on a daily basis for today upto a user specified scaling horizon. Those days on which the user has specified supplier minimums are a candidate for supplier scaling.

Supplier scaling begins by summing the planned warehouse POs for a release date (also called the order date) and grouping of Suppliers, SKUs, and Warehouses called a Scaling Group. If the summed PO totals are at least as large as the defined minimum(s) the minmums are considered statisfied for the release date. If container scaling constraints have been entered the orders are then passed through the container scaling module.

If the summed PO totals are less than some of the minimums the system will attempt to move POs planned for future release dates to the current release date until the minimums are met.

This process is repeated sequentially for each release date.

<u>Container Scaling</u>

Container scaling occurs on a daily basis for today upto a user specified scaling horizon. Those days on which the user has specified container dimension constraints are a candidate for container scaling.

Container scaling will first place order quantities into containers according to fit; at the same time attempting to group orders for a particular warehouse into the same container(s). An order quantity does not fit in a container if adding the quantity to the existing total would exceed a container maximum constraint. When all orders are place in a container those containers whose totals do not meet a user specified minimum will be subject to scaling. The system will attempt to move PO quantities planned for future release dates to the current release until a container minimum has been met.

This process is repeated sequentially for each release date.

Technical Details

The scaling logic will first retrieve the supplier scaling and container scaling system parameters to determine whether either module is enabled at the global level. If not, the script will complete successfully having done nothing.

Next, data is pulled together into the driving table, SCALING_GROUP_ORDER_MAP. Yesterday's data is deleted from the table before rebuilding the table for the current run.

Finally, scaling is performed. The process will spawn multiple threads to perform the operation over the SCALING_GROUP_ORDER_MAP table partitions. A record is logged in the RESTART_SCALING table each time scaling is completed for a Scaling Group, Release Date, and scaling module (supplier scaling or container scaling). The record will aid restart recovery and prevent re-scaling orders which, in certain circumstances, could produce inaccurate results.

Script Name: scale_order.sh

Input Parameters

Period to Scale: [releasing_today | releasing_in_future | releasing_any_day].

> releasing_today: Scale only today and exit. This allows the critical release process to execute before scaling subsequent forecast days.

> releasing_in_future: Scale all forecast days starting from tomorrow. This can be used AFTER 'releasing_today'.

> releasing_any_date: Scale all days starting from today through the scaling horizon.

> > **Note:** When separating the scaling execution by time period using the 'releasing_today' and 'releasing_in_future' parameters it is critical that the releasing_today script call is made before 'releasing_in_future'.

Restart/Recovery

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the batch from this step. Recovery is built into the scaling logic such that scaling will not be repeated for a completed Scaling Group/release date/scaling module (supplier scaling or container scaling).

Post-Scaling

Technical Details

When scaling is completed for all days (today and future release days) the modified order quantities and executed Supplier Purchase Quantities (SPQ) will be sent to the AIP RPAS platform to be reflected in the WRP workbooks and WRP alert calculations

> **Note:** The modified order quantities are not re-reconciled.

The RETL exports to be executed are contained in the export_scale.config file. Once the exports are complete they are archived and readied for moving to AIP RPAS.

Script Name: post_scale.sh

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the script.

Merge Purchase Orders

Technical Details

Prior to releasing warehouse purchase orders they must be moved from an intermediate table to the final warehouse order table. A parameter passed into the executing script will control which day or days' orders are copied. This step will first clear out unreleased warehouse purchase orders from the NON_CONTENTS_ORDER table. These are yesterday's forecast orders. Only orders that fall between the specified time period are deleted. Then, the new purchase orders are copied from the WAREHOUSE_PURCHASE_ORDER table to the NON_CONTENTS_ORDER_TABLE. Only orders with an order quantity greater than 0 are copied.

Script Name: merge_order.sh

Input Parameter

Release Period: [releasing_today | releasing_in_future | releasing_any_day]. This is a required parameter.

releasing_today: Clears out unreleased orders with a release date of today and copies new orders with a release date of today to the NON_CONTENTS_ORDER table.

releasing_in_future: Clears out unreleased orders with a release date greater than today and copies new orders, with a release date greater than today, to the NON_CONTENTS_ORDER table.

releasing_any_day: Clears out all unreleased orders and copies the new orders, having any release date, to the NON_CONTENTS_ORDER table.

This script is used only for warehouse purchase orders. All other orders and transfers do not require this script to be run before releasing.

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues.

3. Restart the script.

Release Orders to RMS

Order Release Overview

After the nightly RPAS batch run, the planned store and warehouse orders are extracted to a set of flat files (.dat data files) that are loaded into AIP Oracle. Order Release batch determines which orders must be released to the order execution system (RMS). Orders that are a candidate for release must have a release date of today. The release date is determined by the source lead time. Additionally, the destination of a transfer must have a warehouse-chamber status of either "Release" or "Closing Down." Once these conditions are met, the order is assigned either a purchase order number or a transfer number. The purchase order or transfer is then released to RMS where the SKU, destination, and order quantity will be communicated to the supplier or warehouse source.

The released orders and forecast purchase orders are also visible from the OM Online screens. You are able to perform an early release or modify the order quantity, the delivery date, and the destination of a purchase order. These user-entered modifications must also be communicated to RMS in order to be executed by the sender of the order.
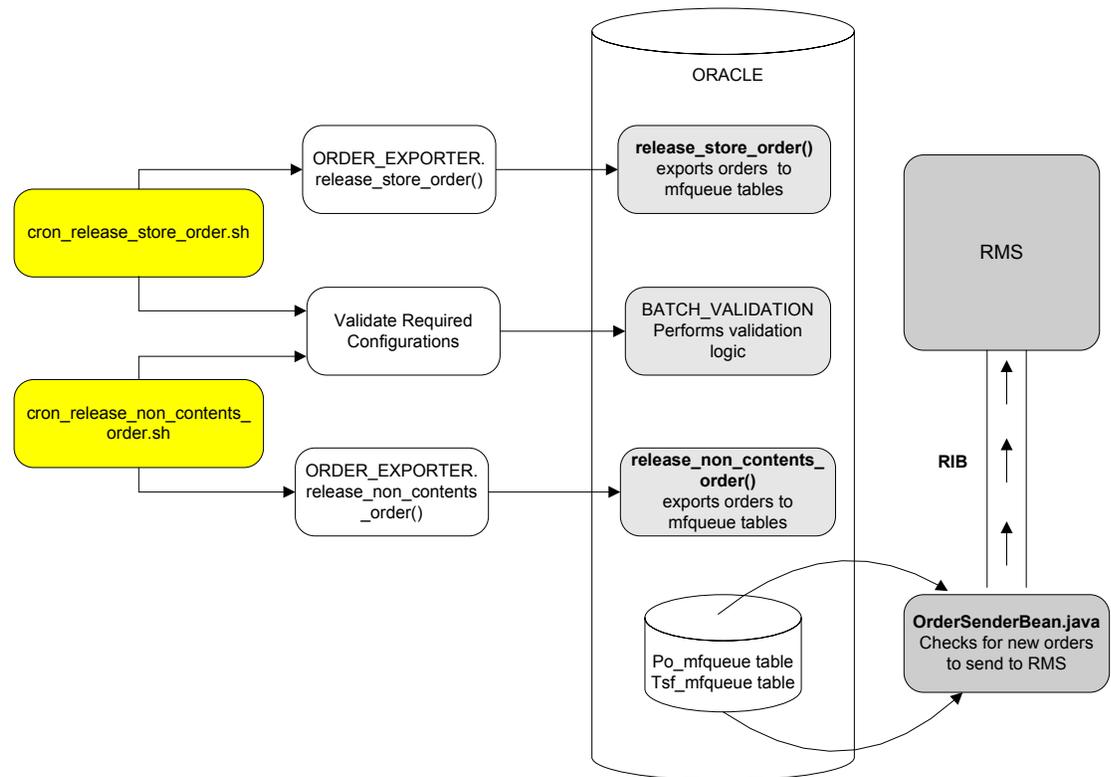
Order Release Flow

Release Orders

Functional Overview

A batch release of AIP RPAS generated orders and transfers is performed once in the morning before the online application is made available. All orders and transfers scheduled for release today are communicated to RMS. The orders are assigned a purchase order number or transfer number, and the order status will then be set to 'Open.'

Technical Details

This process releases store purchase orders, store transfers, warehouse transfers, and warehouse purchase orders to the merchandising system. The order information is written to a set of staging tables. The records on the staging tables provide detail about the order items and the type of action that must be performed by the RMS message subscription logic so that the RMS order is in sync with the AIP order. An Enterprise Java Bean called OrderSenderBean constantly polls the table to find new records. It then reads the details from the staging tables and the base order tables to generate the messages sent to RMS via the RIB.

Script Name: cron_release.sh

Input Parameters

1. Order Type: [order_type_transfer | order_type_purchase | order_type_all]. This is a required parameter.

   order_type_transfer: This will release transfers only.

   order_type_purchase: This will release purchase orders only.

   order_type_all: This will release both transfers AND purchase orders.

2. Destination Type: [dest_type_store | dest_type_warehouse | dest_type_all]. This is a required parameter.

   dest_type_store: This dest_type will release into-store orders only.

   dest_type_warehouse: This will release into-warehouse orders ONLY.

   dest_type_all: This will release both into-store AND into-warehouse orders.

Depending upon the "order type" and "destination type" parameter values passed into this script the desired combination of orders can be released. This script together with the cron_import_order.sh script can be repeated with different parameters to load and release the all orders as dictated by time sensitivity sequentially by time precedence.

Example

For many retailers releasing transfers is more time critical than releasing purchase orders. The following example shows an execution order of scripts that can be followed to load and release transfers before purchase orders.

1. Load and release all transfers first.
   - cron_import_order.sh   order_type_transfer dest_type_all
   - cron_release.sh   order_type_transfer dest_type_all
2. Next, load and release all Pos.
   - cron_import_order.sh   order_type_purchase dest_type_all
   - scale_order.sh
   - merge_order.sh

- cron_release.sh   order_type_purchase dest_type_all

3. Execute post release and post-critical processing

Processing Steps:

This script calls the following scripts depending on the value of the destination type parameter passed into it:

> cron_release_store_order.sh
>
> cron_release_non_contents_order.sh

Release Store Orders

This process is executed by cron_release.sh when the dest_type parameter is 'dest_type_store' or 'dest_type_all'. This process releases store transfers and purchase orders to the merchandising system. Order line items are grouped together and assigned a purchase order number or transfer number, and the order status is then set to 'Open'. The order information is written to a set of staging tables. The records on the staging tables provide detail about the order items and the type of action that must be performed by the RMS message subscription logic so that the RMS order is in sync with the AIP order. An Enterprise Java Bean called OrderSenderBean constantly polls the table to find new records. It then reads the details from the staging tables and the base order tables in order to generate the messages sent to RMS via the RIB.

Script Name: cron_release_store_order.sh

Input Parameter

Order Type: [order_type_transfer | order_type_purchase]. This is a required parameter.

> order_type_transfer: This order_type will release into-store transfers only.
>
> order_type_purchase: This order_type will release into-store purchase orders only.

Day on Day Processing

The script calls the following PL/SQL procedure in order to perform the release of store purchase orders and/or transfers:

Order	Procedure of package:	Script
1	BATCH_VALIDATION	order_definitions_exist
2	AIP_UTIL	gen_table_stats
3	BATCH_VALIDATION	do_validate
4	ORDER_EXPORTER	release_store_order

BATCH_VALIDATION. order_definitions_exist

This function performs the following:

Checks to see if order definitions exist.

This function causes a batch failure if order definitions are missing. The order definitions are used when assigning order numbers during order release.

AIP_UTIL. gen_table_stats

This function performs the following:

Depending upon the value of system_parameter

USE_DBMS_STATS_AUTO_SAMPLE_SIZE, this function will either use DBMS_STATS.AUTO_SAMPLE_SIZE or NULL as estimate_percent for gathering table/partition statistics. A value of NULL in estimate_percent has the effect of computing statistics which is not recommended. Oracle recommendation is to use DBMS_STATS.AUTO_SAMPLE_SIZE and so we recommend using the default value (Y) of USE_DBMS_STATS_AUTO_SAMPLE_SIZE flag in system_parameters table.

BATCH_VALIDATION.do_validate

This function performs the following:

1. Checks to see if RMS/AIP SKU mappings exist for all SKU/pack sizes.

    Orders for a particular SKU/pack size cannot be released to RMS if the mapping is missing.

    The validation failure indicator is set to 'Y' on the STORE_ORDER table and an alert is created for the user to view in DM Online.

    This does not cause the release process to halt.

    The remaining SKU/pack sizes with valid mappings are released.

2. Sets the status of any overdue orders (orders where the delivery date has passed and the order is not fully received or closed) to a status of 'V.'

3. Sets VALIDATION_FAILURE_IND to 'Y' if order quantity is 0.

    This prevents the order from being released to RMS.

ORDER_EXPORTER.release_store_order

This function performs the following:

1. If the source_type parameter is 'V', this function assigns a unique purchase order (PO) number to each unique PO as defined by the ORDER_DEFINITION table where ORDER_TYPE = 'P' and DEST_TYPE = 'S' where the release date is equal to today.

    An 'XORDERCRE' message is inserted into the PO_MFQUEUE table for each purchase order being released.

2. If the source_type parameter is 'W', this function assigns a unique transfer (TSF) number to each unique transfer as defined by the ORDER_DEFINITION table where ORDER_TYPE = 'T' and DEST_TYPE = 'S' where the release date is equal to today.

    An 'XTSFCRE' message is inserted into the TSF_MFQUEUE table for each transfer being released.

3. The status of the released POs and TSFs are updated to 'O' – open.

Restart/Recovery

In case of failure/errors while running cron_release_store_order.sh script, perform the following:

Review the log file, which is located in the 'logs' folder of the directory specified for the Oracle LOGHOME environment variable.

If you receive an error indicating that the maximum transfer or purchase order number was reached, perform the following steps.

> **Note:** This error means that the system ran out of available order numbers to assign to the orders that are ready to be released. Purchase Order number recycling will happen automatically if a custom RMS purge script captures the purged purchase orders and sends them to AIP.

1. Ensure that the custom rmse_order_purge.dat file is being received and processed once the AIP purchase order is purged from RMS.

2. To immediately address the issue, the purchase order numbers and transfer numbers in RMS need to be compared against the range of numbers specified for AIP in the ORDER_NUMBER table.

   - Order_type 'P' identifies the AIP number range for purchase orders.
   - Order_type 'T' identifies the AIP transfer number range.

3. If a sufficient range of values can be found, the ORDER_NUMBER current_value for the order type can be updated to match the smallest available number. Restart cron_release_store_order.sh from cron_release.sh.

> **Note:** Purchase order numbers and transfer number ranges may be allocated to many different systems. All of these systems feed orders into RMS, but RMS will not accept duplicate order numbers. Therefore, it is not recommended that you update the ORDER_NUMBER low_value or high_value without a thorough analysis of the entire system.

If you receive an error indicating that the order definitions check failed, insert the missing rows and restart cron_release_store_order.sh from cron_release.sh.

> **Note:** This error means that rows were removed from the ORDER_DEFINITION table. This table must contain four rows, one for each destination type and order type combination.

The following tables show the only two order definition configurations supported in AIP.

DEST_TYPE	ORDER_ TYPE	USE_ SOURCE	USE_ COMMODITY	USE_ PACK_SIZE	USE_DEST	USE_DELIVERY_ DATE
S	T	Y	N	N	Y	Y
W	T	Y	N	N	Y	Y
S	P	Y	Y	Y	Y	Y
W	P	Y	Y	Y	Y	Y

DEST_TYPE	ORDER_ TYPE	USE_ SOURCE	USE_ COMMODITY	USE_ PACK_SIZE	USE_DEST	USE_DELIVERY_ DATE
S	T	Y	N	N	Y	Y
W	T	Y	N	N	Y	Y
S	P	Y	N	N	Y	Y
W	P	Y	N	N	Y	Y

An error from the orderExporter package indicates that there is an error in the release of orders. The following functions are called by release_store_order in the order listed:

Function	Parameters	Return Value
AssignPONumToStoreOrder	O_errorMessage	0 for success; 1 for failure
AssignTsfNumToStoreOrder	O_errorMessage	0 for success; 1 for failure
ExportStoreOrder	O_errorMessage	0 for success; 1 for failure

Dependencies

The ability to release orders to RMS depends upon having the AIP RPAS Replenishment Planning batch generated orders loaded in the Oracle database. Therefore, the script should not run until the corresponding import is complete. This script must run prior to the Online day.

Release Warehouse Orders

This process is executed by cron_release.sh when the dest_type parameter is 'dest_type_warehouse' or 'dest_type_all'. A batch release of warehouse orders are performed once in the morning before the online application is made available. Order line items are grouped together and assigned a purchase order number or transfer number, and the order status is then set to 'Open'. All warehouse orders scheduled for release today are communicated to RMS.'

The order information is written to a set of staging tables. The records on the staging tables provide detail about the order items and the type of action that must be performed by the RMS message subscription logic so that the RMS order is in sync with the AIP order. An Enterprise Java Bean called OrderSenderBean constantly polls the table to find new records. It then reads the details from the staging tables and the base order tables to generate the messages sent to RMS via the RIB.

Script Name: cron_release_non_contents_order.sh script

Input Parameter

Order Type: [order_type_transfer | order_type_purchase]. This is a required parameter.

order_type_transfer: This will release into-warehouse transfers only.

order_type_purchase:This will release into-warehouse purchase orders only.

Day on Day Processing

The script calls the following PL/SQL procedure in order to perform the release of warehouse purchase orders and/or transfers:

Order	Procedure of package:	Script
1	BATCH_VALIDATION	order_definitions_exist
2	AIP_UTIL	gen_table_stats
3	BATCH_VALIDATION	do_validate
4	ORDER_EXPORTER	release_store_order

1. order_definitions_exist procedure of package batch_validation

2. gen_table_stats procedure of package aip_util
3. do_validate procedure of package batch_validation
4. release_non_contents_order procedure of package order_exporter

BATCH_VALIDATION. order_definitions_exist

This function performs the following:

Checks to see if order definitions exist.

> This function causes a batch failure if order definitions are missing. The order definitions are used when assigning order numbers during order release.

AIP_UTIL. gen_table_stats

This function performs the following:

Depending upon the value of system_parameter

> USE_DBMS_STATS_AUTO_SAMPLE_SIZE, this function will either use DBMS_STATS.AUTO_SAMPLE_SIZE or NULL as estimate_percent for gathering table/partition statistics. A value of NULL in estimate_percent has the effect of computing statistics which is not recommended. The Oracle recommendation is to use DBMS_STATS.AUTO_SAMPLE_SIZE with the default value (Y) of USE_DBMS_STATS_AUTO_SAMPLE_SIZE flag in system_parameters table.

BATCH_VALIDATION.do_validate

This function performs the following:

1. Checks to see if RMS/AIP SKU mappings exist for all SKU/pack sizes.

   > Orders for a particular SKU/pack size cannot be released to RMS if the mapping is missing.

   > The validation failure indicator is set to 'Y' on the NON_CONTENTS_ORDER table and an alert is created for the user to view in DM Online.

   > This does not cause the release process to halt.

   > The remaining SKU/pack sizes with valid mappings are released.

2. Sets the status of any overdue orders (orders where the delivery date has passed and the order is not fully received or closed) to a status of 'V.'

3. Sets VALIDATION_FAILURE_IND to 'Y' if order quantity is 0 or if the destination chamber does not have the status of 'Release' or 'Closing Down.' This prevents the order from being released to RMS.

ORDER_EXPORTER.release_non_contents_order

This function performs the following:

1. If the source_type parameter is 'V,' this function assigns a unique purchase order (PO) number to each unique PO as defined by the ORDER_DEFINITION table where ORDER_TYPE = 'P' and DEST_TYPE = 'W' where the release date is equal to today. An 'XORDERCRE' message is inserted into the PO_MFQUEUE table for each purchase order being released.

2. If the source_type is 'W,' this function assigns a unique transfer (TSF) number to each unique transfer as defined by the ORDER_DEFINITION table where ORDER_TYPE = 'T' and DEST_TYPE = 'W' where the release date is equal to today. An 'XTSFCRE' message is inserted into the TSF_MFQUEUE table for each transfer being released.

3. The status of the released POs and TSFs are updated to 'O' – open.

Restart/Recovery

In case of failure/errors while running cron_release_non_contents_order.sh script, perform the following:

Review the log file, which is located in the "logs" folder of the directory specified for the Oracle LOGHOME environment variable.

If you receive an error indicating that the maximum transfer or purchase order number was reached, perform the following:

> **Note:** This error means that the system ran out of available order numbers to assign to the orders that are ready to be released. Purchase Order number recycling will happen automatically if a custom RMS purge script captures the purchase orders and sends them to AIP.

1. Ensure that the custom rmse_order_purge.dat file is being received and processed once the AIP purchase order is purged from RMS.

2. To immediately address the issue, the purchase order numbers and transfer numbers in RMS need to be compared against the range of numbers specified for AIP in the ORDER_NUMBER table.
    - Order_type 'P' identifies the AIP number range for purchase orders.
    - Order_type 'T' identifies the AIP transfer number range.

3. If a sufficient range of values can be found, the ORDER_NUMBER current_value for the order type can be updated to match the smallest available number. Restart cron_release_non_contents_order.sh from cron_release.sh.

> **Note:** Purchase order numbers and transfer number ranges may be allocated to many different systems. All of these systems feed orders into RMS, which will not accept duplicate order numbers. Therefore, it is not recommended that you update the ORDER_NUMBER low_value or high_value without a thorough analysis of the entire system.

If you receive an error indicating that the order definitions check failed, insert the missing rows and restart cron_release_store_order.sh from cron_release.sh.

> **Note:** This error means that rows were removed from the ORDER_DEFINITION table. This table must contain four rows, one for each destination type and order type combination.

The following tables show the only two order definition configurations supported in AIP.

DEST_TYPE	ORDER_ TYPE	USE_ SOURCE	USE_ COMMODITY	USE_ PACK_SIZE	USE_DEST	USE_DELIVERY_ DATE
S	T	Y	N	N	Y	Y
W	T	Y	N	N	Y	Y
S	P	Y	Y	Y	Y	Y
W	P	Y	Y	Y	Y	Y

DEST_TYPE	ORDER_ TYPE	USE_ SOURCE	USE_ COMMODITY	USE_ PACK_SIZE	USE_DEST	USE_DELIVERY_ DATE
S	T	Y	N	N	Y	Y
W	T	Y	N	N	Y	Y
S	P	Y	N	N	Y	Y
W	P	Y	N	N	Y	Y

An error from the orderExporter package indicates an error in the release of orders. The following functions are called by the function release_non_contents_order in the order listed:

Function	Parameters	Return Value
assignPONumToNonContentsOrder	O_errorMessage	0 for success; 1 for failure
assignTsfNumToNonContentsOrder	O_errorMessage	0 for success; 1 for failure
exportNonContentsOrder	O_errorMessage	0 for success; 1 for failure

Dependencies

The ability to release orders to RMS depends on having the OM generated orders loaded in the Oracle database. Therefore, the script should not run until the corresponding import is completed.

This script must run prior to the Online day.

Post Release

After all the desired orders are released, the post release process can be executed. It is recommended that this script be executed anytime after the release but before OM online users start their daily activities. This step gathers the table statistics on the STORE_ORDER and NON_CONTENTS_ORDER tables to improve the performance of SQL queries in OM online screen and other processes like the OrderSenderBean etc.

Script Name: cron_post_release.sh

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the script.

Purge Closed Orders in AIP Online

Order Purge Overview

Closed orders are purged on a daily basis to maximize response time of database accesses. The closed order status must be provided to AIP from RMS or the external order management system. Closed orders are purged after a period of time which is configurable by the system administrators.

Technical Details

The structure of order purging is very similar to the structure of the cron_release.sh script. The scripts take 2 input parameters, order_type and dest_type, to control the order type and destination types that are processed in any given execution.

Purging of orders is not a critical process and so it can be scheduled to run in any non-critical time window of the day. However it is recommended that cron_purge is scheduled to run daily to prevent a build up of closed orders in the tables.

Script Name: cron_purge.sh

Input Parameters

1. Order Type: [order_type_transfer | order_type_purchase | order_type_all] This is a required parameter.
   - order_type_transfer: This will purge transfers only.
   - order_type_purchase: This will purge purchase orders only.
   - order_type_all: This will pruge both purchase orders and transfers.
2. Destination Type: [dest_type_warehouse | dest_type_store | dest_type_all]. This is a required parameter.
   - dest_type_warehouse: This will purge into-warehouse orders only.
   - dest_type_store: This will purge into-store orders only.
   - dest_type_all: THis will purge both into-warehouse and into-store orders.

Processing Steps:

This script calls the following scripts depending on the value of the destination type parameter passed into it:

1. cron_purge_store_order.sh
2. cron_purge_non_contents_order.sh

Purge Store Orders

Technical Details

This logic purges the closed orders from the STORE_ORDER table that have exceeded their purge age. The purge age is the PURGE_PERIOD (in days) set in the ORDER_PURGE_PERIOD table. The order age is the difference between the timestamp when the order was closed and today.

Script Name: cron_purge_store_order.sh

Input Parameters

Order Type: [order_type_transfer | order_type_purchase]

   order_type_transfer: This will purge into-store transfers only.

   order_type_purchase: This will purge into-store purchase orders only.

Restart/Recovery

The purging of closed orders is a maintenance task that is not essential to the successful completion of the batch; though overtime it is essential for maintaining manageable table sizes.

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues.

3. Restart the script.

Purge Warehouse Orders

Technical Details

This logic purges the closed orders from the NON_CONTENTS_ORDER table that have exceeded their purge age. The purge age is the PURGE_PERIOD (in days) set in the ORDER_PURGE_PERIOD table. The order age is the difference between the timestamp when the order was closed and today.

Script Name: cron_purge_non_contents_order.sh

Input Parameters

Order Type: [order_type_transfer | order_type_purchase]

   order_type_transfer: This will purge into-warehouse transfers only.

   order_type_purchase: This will purge into-warehouse purchase orders only.

Restart/Recovery

The purging of closed orders is a maintenance task that is not essential to the successful completion of the batch; though overtime it is essential for maintaining manageable table sizes.

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues.

3. Restart the script.

AIP Interval Batch Scripts

Integration with Replenishment Optimization

Replenishment Optimization (RO) suggests the most appropriate replenishment method and thresholds to use based on the business goals set in RO, and the supply-chain. The supply-chain information is provided to RO by AIP. The optimized settings should then be used in AIP when replenishment planning.

AIP provides a number of control parameters to allow flexible control over the integration.

1. The frequency of import and export from AIP is controlled by the frequency of scheduled tasks. Depending on the Retailer's business strategy, new sku introduction, etc. integration updates could occur weekly, montly, quarterly, or less as needed.

2. The **Replenishment Optimization Location Types** value can be used to limit the locations processed to only stores or only warehouses.

3. The **Replenishment Optimization Update for Stores/Warehouses** measures can be used to limit the data set that is sent to RO. The measures are editable within AIP as well as loadable from RO.

4. The **Replenishment Optimization Start Date** measure will be available for specifying a date to begin pulling data for extract. This helps to eliminate the time-phased disparity between AIP and and an optimization system. The Retailer can run the optimization for the supply-chain ahead of when the supply-chain changes. In the workbooks the standard hierarchy rollups are available for mass selecting SKUs and locations to set the date. The 'Replicate' spread method maybe be used to set the same date (i.e. start of quarter) at an aggregate level. This value is optional. The batch run date will be used when no value is specified.

The following diagram illustrates the flow of data between AIP and RO.
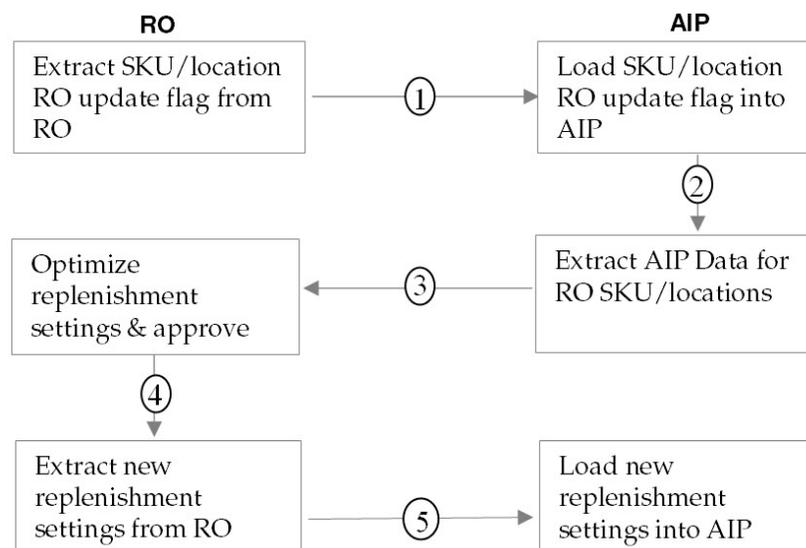
Flow of data between AIP and RO

AIP Import From RO

AIP will import the new, optimized replenishment settings to be used for planning. Because of the critical nature of the data being loaded from RO, AIP will ensure the integrity of the data by validating that every SKU/location replenishment method has corresponding thresholds and vice versa. AIP will not load a SKU/location replenishment method which is missing 1 or more of the needed thresholds. Similarly, AIP will not load SKU/location parameters for which no replenishment method has been provided. If one or more values are missing for a SKU/location the load will log an error message and continue loading subsequent SKU/location values.

This batch step is not required to operate AIP replenishment. This step is only for those Retailers integrating with RO or who wish to import the replenishment method data from an external system. Alertnativley, users can specify replenishment method data in the SRP and WRP workbooks.

Script Call

aip_import_from_ro.sh

Functional Overview

This script imports optimized replenishment data provided by RO into AIP. The approved, or working, settings are extracted from RO for loading into AIP. The replenishment method and corresponding thresholds from RO will be loaded at a SKU/destination/effective date level.

Technical Details

The following files are generated by RO and are optional. The file contents are designed in a way that each row contains all required and necessary data elements. Each file is first validated and then loaded into AIP measures.

File Name	Description
STR_MINMAX	Contains Store minmax replenishment parameters.
STR_DYNAMIC	Contains Store dynamic replenishment parameters.
STR_TIMESUPPLY	Contains Store Timesupply replenishment parameters.
STR_HYBRID	Contains Store Hybrid replenishment parameters.
STR_POISSON	Contains Store Poisson replenishment parameters.
STR_MINSS	Contains Store Minimum safety stock data.
STR_ROUPDATE	Contains information of intersections for which export/ import need to occur for stores.
WH_MINMAX	Contains Warehouse minmax replenishment parameters.
WH_DYNAMIC	Contains Warehouse dynamic replenishment parameters.
WH_TIMESUPPLY	Contains Warehouse Timesupply replenishment parameters.
WH_HYBRID	Contains Warehouse Hybrid replenishment parameters.
WH_POISSON	Contains Warehouse Poisson replenishment parameters.
WH_MINSS	Contains Warehouse Minimum safety stock data.
WH_ROUPDATE	Contains information of intersections for which export/ import need to occur for warehouses.

The details of data elements and their formats have been detailed in the *Advanced Inventory Planning Implementation Guide*.

The batch creates a <filename>.error file for bad records found in each of the above mentioned files. The import batch processes one file at a time for available files present in the $RO_INPUT directory.

Each file is validated and then loaded in the system. Standard AIP logs are created through BSA.

Prerequisites

Files must be copied or FTP'ed to the $RO_INPUT directory and unpacked by a Retailer-created scheduled task before executing this script.

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the import of data by executing aip_import_from_ro.sh. The files already processed by batch need not be reloaded.

AIP Export to RO

Script Call

aip_export_to_ro.sh

Functional Overview

This script exports lead time, review time and pack size information from AIP into pre-formatted/csv flat files for RO. This batch step is not required to operate AIP replenishment. This step is only for those Retailers integrating with RO or who wish to export the data to an external system.

Lead Time

The lead time (order cycle) pattern will generally always contain the same lead time on all days that have a lead time, or the lead time will increase for the weekend. Therefore it is expected that the most common lead time will be found during the business week.

Review Time

Review Time is the number of days until the next possible receipt + availability lead time. The review time is a key factor in determining the minimum amount of projected stock that should be available until the point of ordering. The projected available inventory needs to cover the projected need until the next earliest possible receipt. Because review time can change daily or cyclically, in order to avoid stock outs, the minimum available inventory must cover the longest review time.

Ordering Pack Size

The Ordering pack size is the preferred pack size of a SKU that should be ordered from a source to destination.

Technical Details

1. The batch script runs the rule group RoExtract on each local domain and populates temporary measures.
2. The data in the temporary measures is extracted in the flat files STR_AIP_DATA and WH_AIP_DATA. The file contents are designed in a way such that each row within the file contains all required data elements.

File Name	Description
STR_AIP_DATA	Contains Store lead time, review time and pack size data.
WH_AIP_DATA	Contains warehouse lead time, review time and pack size data.

The details of data elements and their formats have been detailed in the *Advanced Inventory Planning Implementation Guide*.

The data extracted to the files is limited by:

- The Replenishment Optimization Location Types setting which indicates the location types—stores and/or warehouses—which are are integrated with RO.
- The replenishment optimization update flag for stores or warehouses that is set at a SKU/location intersection and indicates which SKU/stores and/or SKU/warehouses should be extracted for updating and optimization in RO.

> **Note:** For each location type that is specified in the
> **Replenishment Optimization Location Types** measure the
> RO update flag is retrieved. If no SKU/location
> combinations are flagged for update of a particular location
> type then ALL SKU/location combinations of that location
> type are extracted.

3. The batch script clears all the temporary measures.
4. Files generated will be placed at the location configured in the environment variable
   $RO_OUTPUT. The outbound files from the AIP RPAS server should be retrieved by
   FTP using a job scheduling application.

Prerequisites

When running a stores extract the STR_ROUPDATE file should be loaded from RO first
to indicate which SKU/stores are being optimized in the next Optimization run.
Optionally, AIP users may manage this setting in an AIP Workbook.

When running a warehouse extract the WH_ROUPDATE file must be loaded from RO
first to indicate which SKU/warehouses are being optimized in the next optimization
run.

Restart/Recovery

If this script fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the export of data from AIP by restarting the script aip_export_to_ro.sh.

AIP Data Purge

Hierarchy and historical data is purged from Oracle database tables using aip_purge.sh
script. Hierarchy data that is not loaded into Oracle table for the specified number of
days or historical data where the end date is in the past is deleted from the tables. In
addition to this, data that has become invalid due to invalid or missing relationships is
also deleted from tables. Hierarchy data left after purging is exported to AIP-RPAS so
that AIP-RPAS can also perform the necessary purging/cleanup.

AIP-RPAS loads the hierarchy data received from AIP-Oracle during a purging interface
with a purge age set to 0, indicating that only the data in the load files remain in the AIP
domain's hierarchy arrays at the completion of the load. In addition to the hierarchy
data purging, measures whose values are positions along hierarchy dimensions are
considered to ensure that no purged hierarchy positions remain as values.

The AIP-Oracle and AIP-RPAS procedures described in this section are not a part of the
regular AIP batch processes. They are initiated separately and are controlled by different
scripts.

AIP Oracle Data Purge

Script Call: aip_purge.sh

Input Parameters None

Technical Details

This is not a critical process and so it can be scheduled to run in any non-critical time window of the day. Also it does not need to run daily. It can be scheduled to run once a week or once a month or less frequently. Due to the functional dependencies involved in purging data, certain tables must be purged first before other tables. Also due to technical dependencies (like referential key constraints) child tables must be purged before parent tables. Table purge_parameters is used to configure the purge age of hierarchical data tables.

The script purges data in groups of parent tables. Each purge group includes several parent tables that are purged in parallel. Groups themselves are purged in sequence. For example, first group to be purged includes following parent tables: supplier, warehouse, store, commodity_pack_size, chamber, historic data and on_supply_off_supply. At the end, of purging invalid data in parent tables, the remaining hierarchy positions are exported to AIP on RPAS in a compressed tarred file called 'purge_hierarchy.tar.Z'

Restart/Recovery

Before any parent table is purged, a log is inserted in a temporary log file which is removed at the end if all tables were successfully purged. If aip_purge.sh script fails in the middle, then following actions can be taken

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the script. Parent tables that were successfully purged will not be attempted again.

AIP RPAS Data Purge

The AIP RPAS Data Purging master script, purging_aip_batch.sh, accepts a number of arguments that allow more control over what portion of the batch scripts are run. Step names have been defined, which may also be passed into the script as arguments that define exactly which steps and corresponding scripts should be run.

Usage

The table below provides descriptions of the purging_aip_batch.sh arguments.

Argument	Description
-s	Indicates that a starting step is defined. The step at which the purging batch should start must follow this flag. Only one batch step can follow the flag. This flag MUST be used along with the -e flag and its associated argument.
-e	Indicates that an ending step is defined. The step after which the purging batch should end must follow this flag. Only one batch step can follow the flag. This flag MUST be used along with the -s flag and its associated parameter.

The purging_aip_batch.sh Steps

The following is a list of valid steps that can be used with the -s and -e flags. These steps can also be passed as parameters to the purging_aip_batch.sh script, in the form of a list of steps (not flagged with -s and -e). Each step is described in detail in the following section.

Execution Sequence of the AIP RPAS Purging Batch Scripts

The table below displays the steps in the order that purging_aip_batch.sh execute them.

Step Name	Script Name	Technical Details	Purging Batch Run
purging_prep_oracle_data	purging_prep_from_aiporacle.sh	Prepares the export files from AIP Oracle for loading into the AIP RPAS platform.	☑
purging_process_hierarchies	purging_process_hierarchies.sh	Pre-processes the hierarchy data exported from AIP Oracle before it is loaded into RPAS domain by inserting positions into labels, removing field separators, and creating stocking point hierarchy load files.	☑
purging_load_hierarchies	purging_load_hierarchies.sh	Purges and load all hierarchies in the AIP domain.	☑
purging_process_measures	purging_process_measures.sh	Deletes positional measure values from the domain's measure arrays.	☑

The command line interface of purging_aip_batch.sh is identical to that of aip_batch.sh, described above, except that there is no first-day flag (-f). All the steps of the purging_aip_batch.sh script are intended to be run in the same window of time, in order that the purging is synchronized across both hierarchy and measure data. However the interface to this script is modeled after aip_batch.sh for consistency.

Prepare Purging Data from AIP Oracle Platform

Step Name

purging_prep_oracle_data

Script Call

purging_prep_from_aiporacle.sh

Technical Details

AIP RPAS processes data from AIP Online. AIP Online exports hierarchy information and DM measures which are placed by the user into the AIP RPAS domain, in the $AIPDOMAIN/interface/import directory.

The prep_from_aiponline.sh script is called to process the data files created by the AIP Online export. The data files should be transferred from AIP Online to the export directory by a job scheduling application.

Interval Processing

1. Using prep_files.sh, the archive file purge_hierarchy.tar.Z is unpacked from$AIPDOMAIN/interface/purge/import into $AIPDOMAIN/interface/purge/import/hier.

2. The unpacked files are verified to ensure all the data from AIP Oracle Purging batch is present, as listed in the $AIPDOMAIN/interface/purge/config/purge_hier_import.config file. The following hierarchies mastered by AIP Oracle Purging are expected to be in purge_hierarchy.tar.Z.

File Name	Explanation
hdgr.dat	Delivery Group Hierarchy
hspl.dat	Supplier Hierarchy
loc.dat	Location Hierarchy
ntwg.dat	Network Group Hierarchy
oltc.dat	Order Lead Time Cycle Hierarchy
ordg.dat	Order Group Hierarchy
proc.dat	Profile Order Cycle Hierarchy
prod.dat	Product Hierarchy
prof.dat	Profile Hierarchy
whse.txt	Warehouse Hierarchy

Restart/Recovery

If this step fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.

2. Correct any identified setup or environment issues.

3. Ensure that the exported data from AIP Oracle exists inside the purge_hierarchy.tar.Z file, and this file is located in $AIPDOMAIN/interface/purge/import.

4. Ensure the contents of purge_hierarchy.tar.Z match those in the $AIPDOMAIN/interface/purge/config/purge_hier_import.config configuration file. All files are required.

5. Restart the purging batch from this step.

Process AIP Oracle Purged Hierarchies

Step Name

purging_process_hierarchies

Script Call

purging_process_hierarchies.sh

Technical Details

This step pre-processes the AIP Oracle purging exported hierarchy data before they are loaded into RPAS domain. The processing consists of inserting positions into labels, removing field separators, and creating stocking point hierarchy data.

Interval Processing

1. Using the interutil binary, the field separators are removed from all the hierarchy data processed in the previous step except the Profile hierarchy data file, which does not contain field separators. Output written to $AIPDOMAIN/input.

2. The Source Stocking Point and Destination Stocking Point hierarchies are created in $AIPDOMAIN/input using elements of the Warehouse, Supplier and Location hierarchies.

3. The positions are prepended to the position labels in the whse, hspl, ssp, and dsp hierarchy data files.

4. Export the product and location hierarchies from the AIP RPAS domain, and reduce them by the SKPS (from product) and STR (from location) keys from the AIP Oracle purging export. Write the outputs to the $AIPDOMAIN/input directory.

Load All Purged Hierarchies

Step Name

purging_load_hierarchies

Script Name

purging_load_hierarchies.sh

Technical Details

This step loads the purged hierarchy data processed in the previous steps, using a purge age of 0. The end result is that only the hierarchy data positions in the load files will survive the load process. Those RPAS measures whose base intersections include dimensions that have been purged of unneeded positions will be re-sized to not include those positions.

Interval Processing

Call loadHier RPAS utility with –loadAll and –purgeAge 0 options to load all the processed hierarchy data files into the AIP RPAS domain.

Restart/Recovery

If this step fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Ensure that the individual hierarchy data files to be loaded exist in the $AIPDOMAIN/input directory.
4. Restart the purging batch from this step.

Process AIP RPAS Positional Measures

Step Name

purging_process_measures

Script Call

purging_process_measures.sh

Technical Details

Some AIP RPAS measures contain values corresponding to positions along a dimension. This guide refers to these measures as "positional measures" and their data as "positional values." During the previous load hierarchy step, unused positions will be removed from the hierarchy arrays. If any of these measures' positional values are purged positions, the loadHier step will not clean these up. Those measures that have been predetermined to be positional measures are listed in the configuration file $AIPDOMAIN/interface/purge/config/purge_meas_values.config. The configuration file also lists the dimensions in which the measure's value can be a position. This step

iterates through each measure in this configuration file and verifies that their values still exist in the hierarchy. If a purged value is detected, it is replaced with the measure's NA Value to keep the hierarchy and the positional values in synchronization.

Interval Processing

Iterate over $AIPDOMAIN/interface/purge/config/purge_meas_values.config and execute in parallel the purgeMeasureValues binary on each measure.

Restart/Recovery

If this step fails, perform the following steps:

1. Examine the log files to determine the cause of the failure.
2. Correct any identified setup or environment issues.
3. Restart the purging batch from this step.

AIP Batch Environment Maintenance

The maintenance of servers and applications is an ongoing necessity. Occasionally, it may be necessary to perform maintenance on the existing AIP RPAS global domain.

Refer to the *AIP Implementation Guide* for information on configuring and partitioning AIP RPAS domains at build time.

Notes on AIP Domain Builds

There are three directory path references inside of an AIP global domain that must be mentioned before discussing domain relocation.

globaldomainconfig.xml	The AIP Implementation Guide provides instructions regarding customizing the globaldomainconfig.xml file required for an RPAS domain build using the build_aip_domains.ksh script. This XML file contains the absolute path to all domain components, namely, the master domain path and all local domain paths.
configmeasdata.db	The master domain contains the directory configmeasdata.db, located in $AIPDOMAIN/data directory. This directory contains an array named r_subdomainpath%1 which contains a map from all partitioning dimension positions to the local domains which contain each position.
admin.db	Each local domain contains the directory admin.db, located in each local domain's data directory. This directory contains an array named domain_properties. This array contains the path to the master domain.

It is important that these references to the master domain's path and local domains' paths be synchronized at all times to ensure the global domain's integrity. Following the procedures in this chapter will ensure the pointers are correct. Using UNIX mv or cp on the domain build directory (master domain) or sub-directories (local domains) will result in a corrupted set of internal links.

In addition, the shell script $RPAS_HOME/bin/aip_env_rpas.sh contains the assignment of the $AIPDOMAIN variable. If the master domain is moved, then the $AIPDOMAIN variable must also be updated to reflect the new path.

The standard domain build procedure described in the *AIP Installation Guide* and *AIP Implementation Guide* specifies that the absolute path to each domain component be specified in the globaldomainconfig.xml file prior to running the build_aip_domains.ksh script. As a result, the three references listed above will contain absolute paths at the conclusion of the domain build process.

Note that the globaldomainconfig.xml can be configured such that the master domain and local domains do not reside in the same directory. For example, the local domains can, but do not have to, be subdirectories of the master domain. This is configurable according to the needs of the client to accommodate disk space.

Domain Relocation

There are two supported utilities that can be run on the AIP domain to relocate all or part of the domain from one directory to another directory. Operations that are supported are those which maintain the integrity and correctness of the three domain references listed in the previous section, "Notes on AIP Domain Builds."

Moving a Local Domain or a Master Domain

If required, a local domain or the master domain may be moved from one directory to another. The RPAS utility, `moveDomain`, can be used to accomplish this task. See the RPAS documentation for usage information on the `moveDomain` utility.

Consolidating the Master and Local Domains

The global domain can be consolidated into a directory structure where all local domains are subdirectories of the master domain. For example, given master domain path:

/files1/AIPm

and local domain paths

/files2/AIPl-a

/files3/AIPl-b

using the command:

`copyDomain –d /files1/AIPm`

would result in the following paths:

/files1/AIPm/AIPl-a

/files1/AIPm/AIPl-b

By not specifying a target domain destination, the `copyDomain` utility will update the domain to have relative paths. Copies of "spread out" local domains are made as subdirectories of the master domain, and the references between the master and local domains are updated.

> **Note:** This operation leaves behind orphan local domains /files2/AIPl-a and /files3/AIPl-b, which are no longer attached to a master domain. These domains should be deleted.

Other Uses of copyDomain

While not elaborated in this Operations guide, `copyDomain` has other command line options which can be used to copy part of a local domain, translate from UNIX to Windows, and compress the copied domain. See the usage information for the utility (`copyDomain -help`) and the RPAS documentation for full usage of this utility.

Position Reclassification Process

On occasion it is understood that clients may wish to reclassify positions. Reclassification is changing a dimension position's rollup position from one position to another. For example, SKU-pack size 30_1 might roll up to SKU 30, but as a result of a reclassification, the SKU-pack size 30_1 would instead roll up to SKU 40. AIP supports reclassification at hierarchy load time by loading a hierarchy data file that contains the new rollup. This can happen during calls to the loadHier and reconfigGlobalDomainPartitions RPAS utilities embedded inside the AIP batch scripts.

In order for reclassification through loadHier or reconfigGlobalDomainPartitions to be successful, the client should note the old and new rollup positions in relationship to the local domains of the RPAS global domain. If, by reclassifying a position, the position would roll up into a dimension position located in a different local domain than the former rollup dimension position, then the dimension must be "buffered" in order for the reclassification to be successful. In addition, all dimensions, below the dimension whose position is reclassified, must be "buffered."

The default configuration shipped with AIP (as listed in the hierarchy.xml) contains buffering percentages set to 0% low and 0% high for all dimensions. As a result, buffering is disabled. The affected dimension's buffering percentages must be non-zero for the reclassification from one local domain to another. Therefore, before trying to load hierarchy data that has reclassified positions where the local domain rollups change, please read the RPAS documentation on how to use the dimensionMgr RPAS utility to manually increase buffering percentages on the dimensions whose positions will be reclassified. This operation will take place outside of AIP batch.

To continue the example: if the domain is partitioned across subclass, SKU 30 rolls up to subclass 1 in ldom0, SKU 40 rolls up to subclass 2 in ldom1, and the low and high buffering percentages for SKPS are 0%, then the SKPS dimension must be rebuffered in order for SKPS 30_1 to move from local domain ldom0 to the local domain ldom1 during the reclassification hierarchy load.