Oracle® Retail Advanced Inventory Planning

Operations Guide

Release 12.1

January 2008

Oracle® Advanced Inventory Planning Operations Guide, Release 12.1

Value-Added Reseller (VAR) Language

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 12.1 documentation set:

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

Customer Support

- https://metalink.oracle.com

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

For a base release (".0" release, such as 12.0), Oracle Retail strongly recommends that you read all patch documentation before you begin installation procedures. Patch documentation can contain critical information related to the base release, based on new information 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:

http://www.oracle.com/technology/documentation/oracle_retail.html

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
```

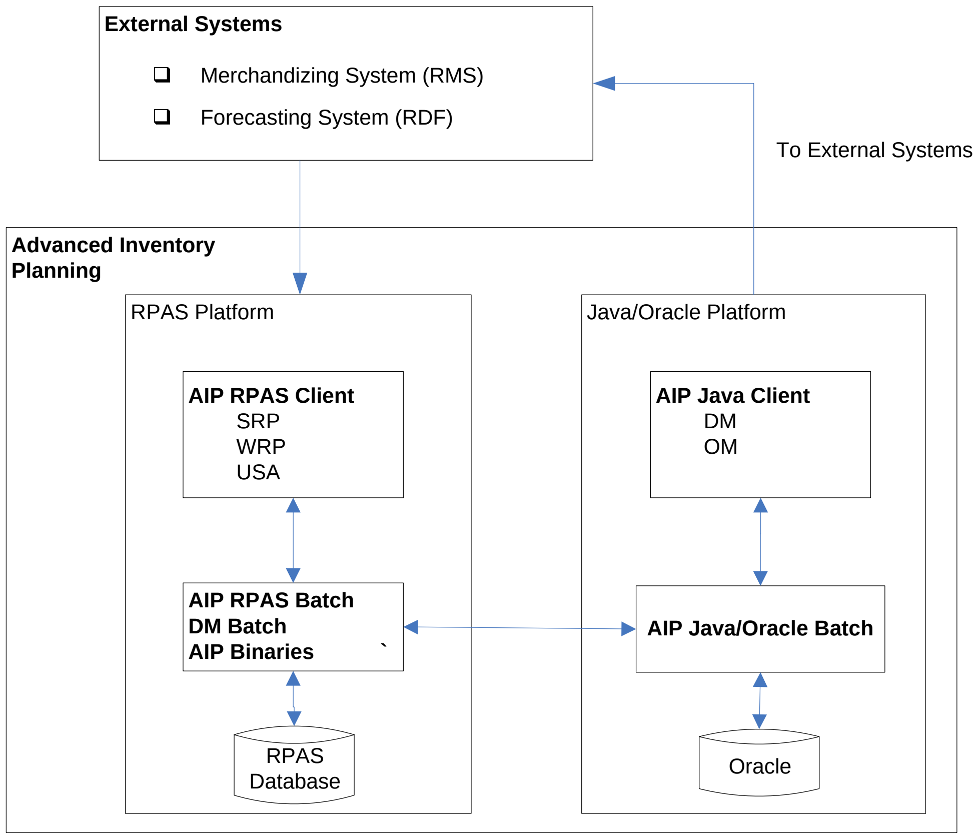
A hyperlink appears like this.

1 Advanced Inventory Planning

AIP Architecture

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

- The RPAS platform
- The Java/Oracle platform



AIP Architecture

RMS: Oracle Retail Merchandising System

RDF: Oracle Retail Demand Forecasting

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 point-of-sale 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.

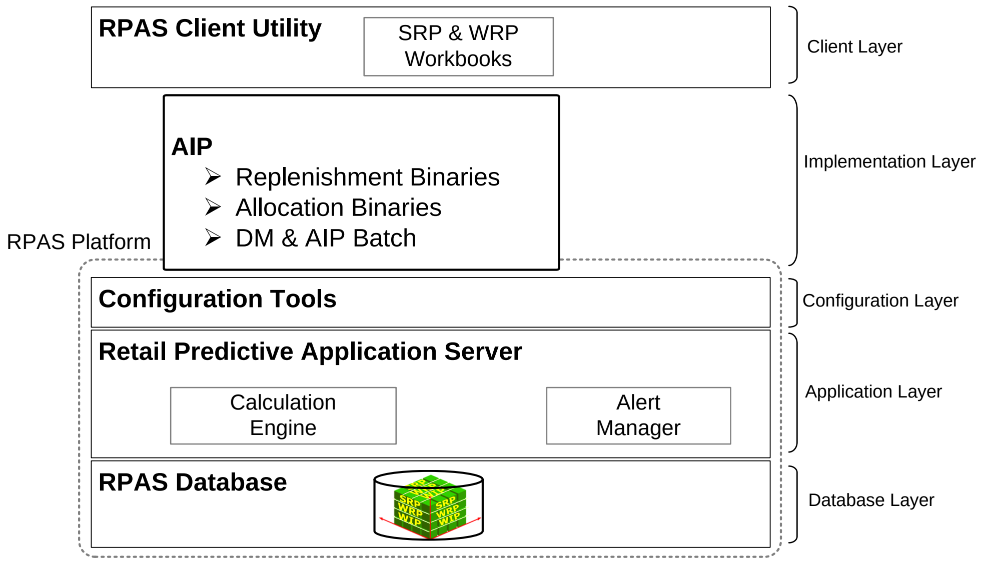
RPAS Platform

For the AIP 12.1 release, 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 integrated using rule groups and 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.



AIP RPAS Platform

The RPAS platform is a predictive platform used for implementing the 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 either a WebSphere application server or an 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.

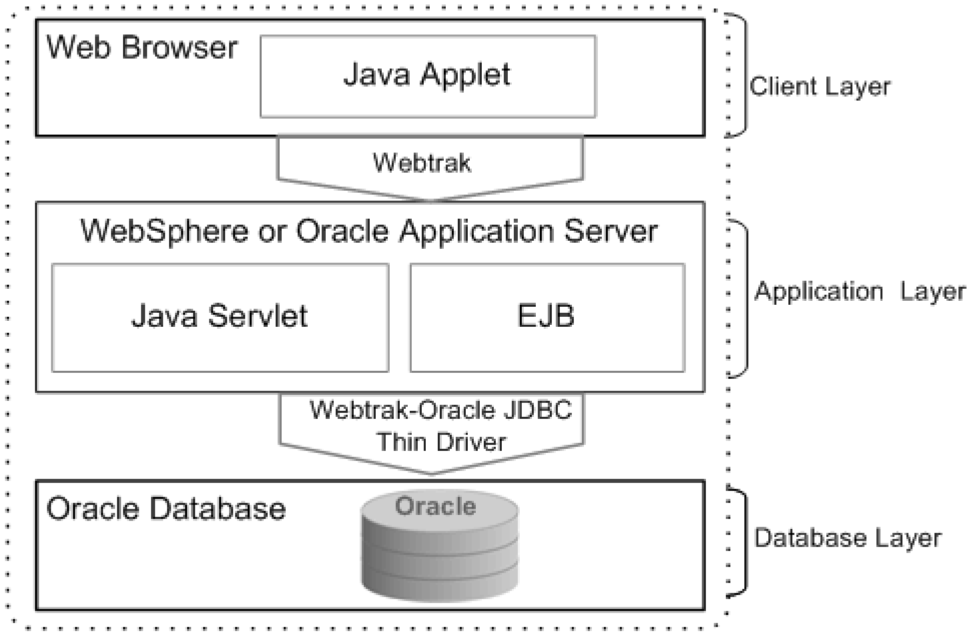
Note: On the RPAS platform, the DM module has only the batch process. There is no user interface.

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, releaseing 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 either an IBM WebSphere application server or an Oracle application server through an Oracle Retail Webtrak 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 thin driver to an Oracle database.

2

AIP Integration

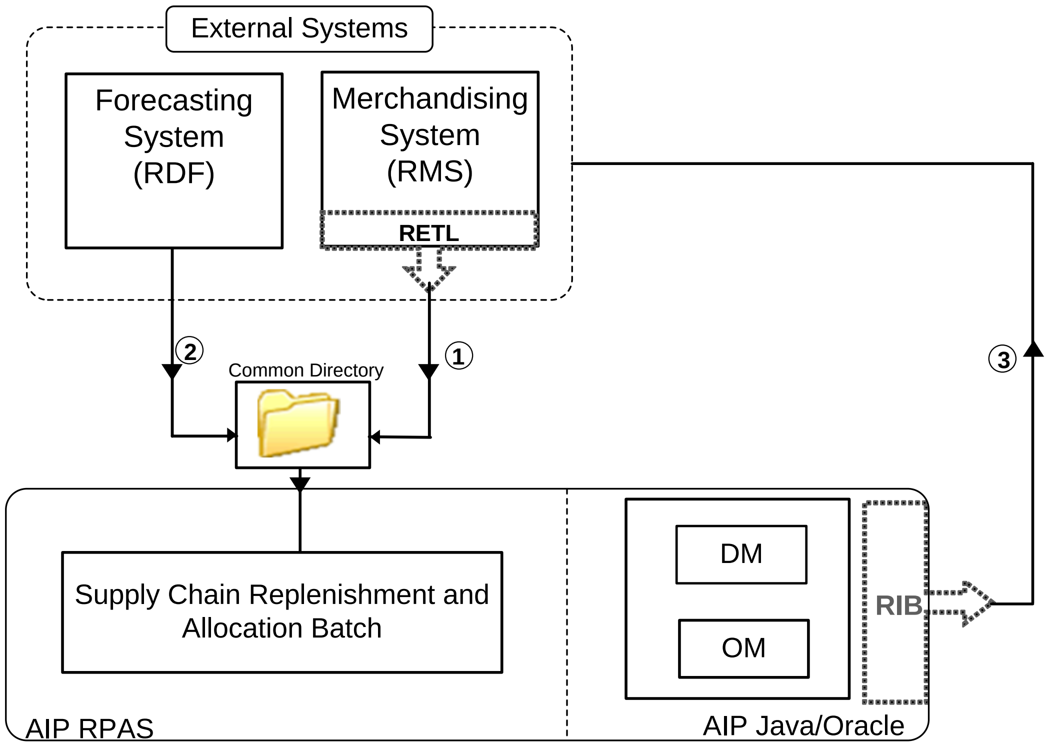
The AIP solution uses the following Oracle Retail integration tools to integrate within AIP and also with other Oracle Retail products:

- 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) and Oracle Retail Demand Forecasting (RDF) products.



AIP External Integration

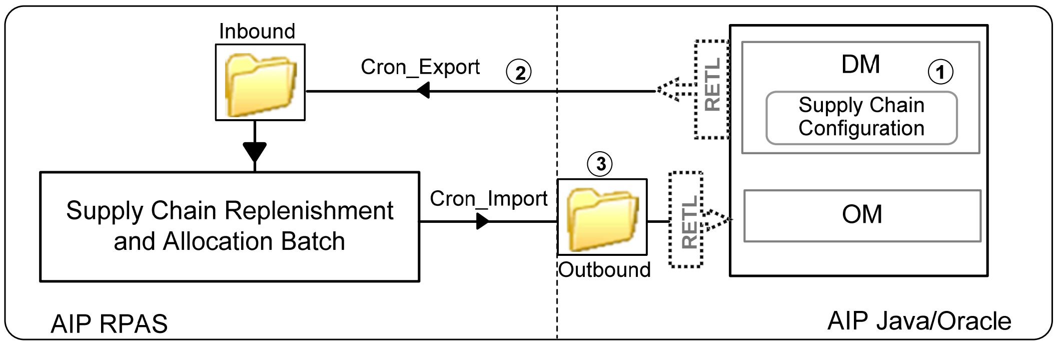
1. **Integration between RMS and AIP using RETL:** The data required by AIP from RMS is extracted. The data from RMS Oracle tables are transformed into an AIP compatiable format (text files) using the custom script that calls the Oracle Retail Extract, Transform, and Load (RETL). The extracted text files 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 customers need to have a custom script 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 chian 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.

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 the AIP is run. The DM Online applicaiton 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 applicaiton to establish the sources to the warehouses and stores, also 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 DM module on the AIP Java/Oracle platform maintains the supply chain parameters used by the batch process for calculating the into-store and into-warehouse replenishment plans on the RPAS platform. The RETLextract scripts are executed during the batch process to move the DM Online parameters from the AIP Oracle database to the AIP RPAS platform.
3. FTP and RETL are used to transfer the store receipt plan, and the warehouse receipt plan, from RPAS to the Order Management module in the Java/Oracle platform. Also, RMS measures specific to DM on the Java/Oracle platform are also passed from the RPAS platform to the Java/Oracle platform.

AIP Interfaces

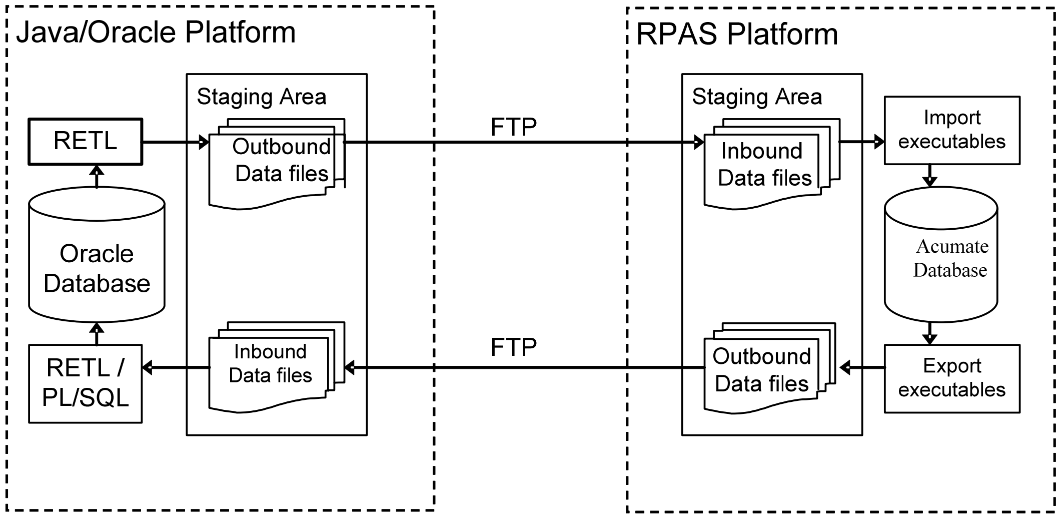
The Java/Oracle portion of AIP is essentially a reviewing and manipulation tool for the data that is ultimately stored and processed in the RPAS platform. There are a substantial number of files to communicate the data between these two systems, and this section describes the interface that handles this transfer.

> **Note:** The data in RPAS at the time of the replenishment planning calculation is considered to be the most accurate, up to date version.

The Java/Oracle side of the interface between the two platforms uses the following technologies:

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

The diagram below shows how these components fit together.

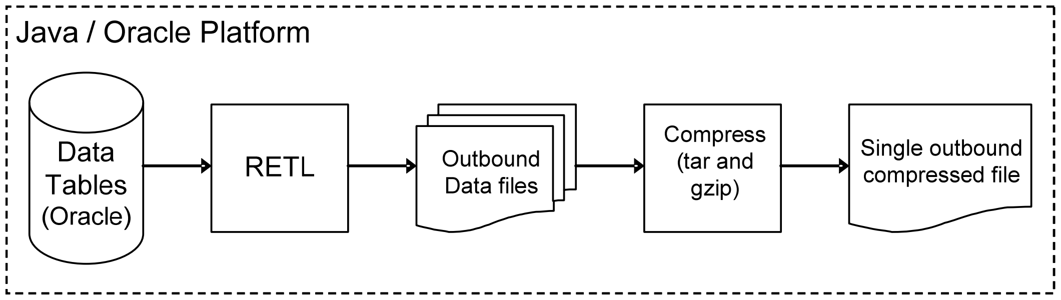


AIP Interfaces

The import/export process is triggered by a job scheduler on a nightly basis. The scheduler calls the import and export shell scripts that perform the necessary file transfers, and also run the RETL scripts.

Exporting

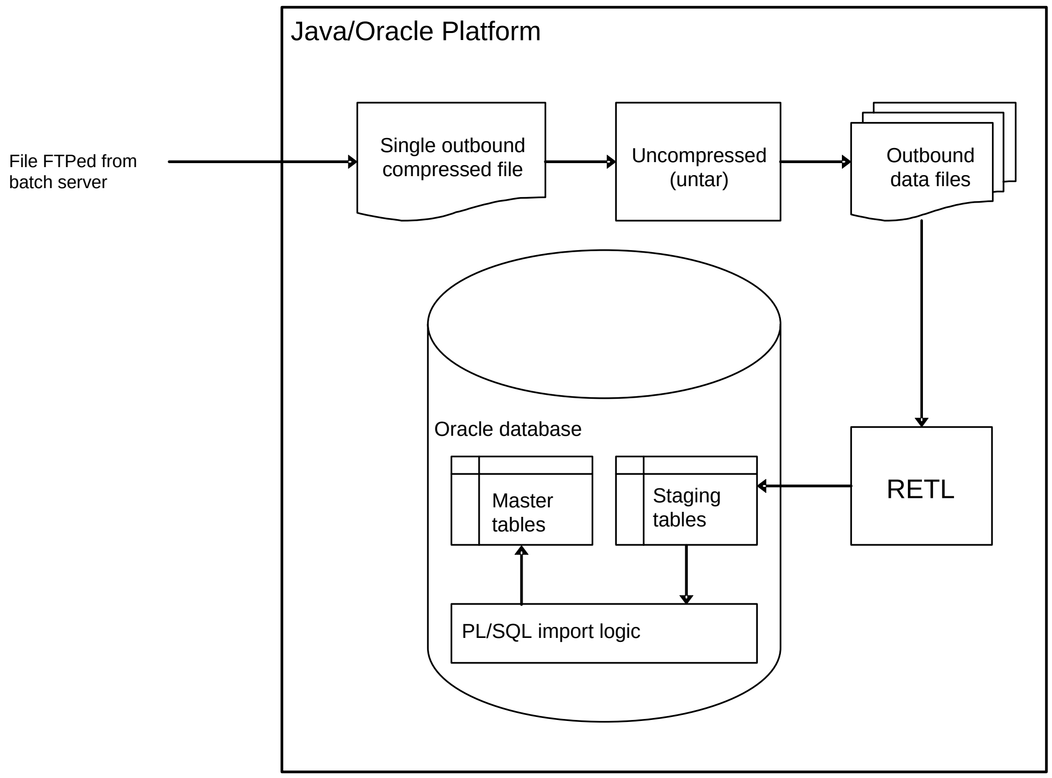
The export process involves running the export RETL scripts to create the outbound data files. The process compresses them into two files, and then places the files in a staging area, from where the RPAS batch processes can retrieve it via file copy or FTP. RETL can export the data directly from the 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 fetched from the RPAS server location and 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

Configuration

The interface process was designed to be fully automated once configured. 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.

Configuring config.xml

This configuration file contains the database connect information for RETL, for the import and export operations. See the RETL documentation for detailed descriptions of the element definitions. The “oraread” section describes the database for the export function, and the “orawrite” section for the import function. Both are normally the same. Databases can be local or remote, but if they are remote, they must be reachable by normal means. In other words, they should be described in tnsnames if they are Oracle databases, and they should be reachable by SQL*Plus.

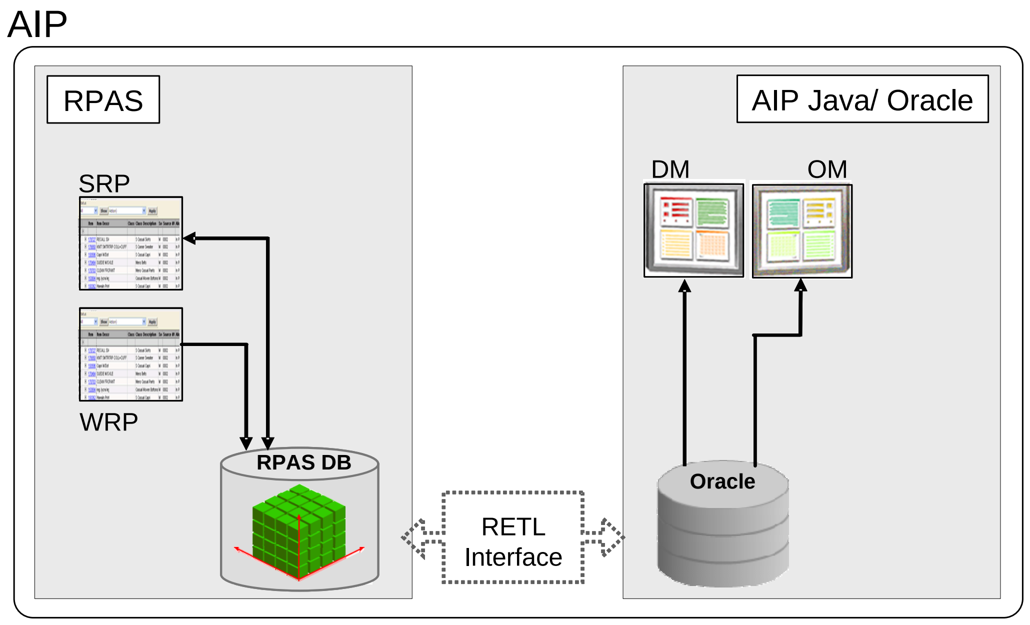
3

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 client utilities to work on the respective workbooks, to maintain the parameters for the calculations, as 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 fetched for the replenishment and allocation calculations.

AIP Batch Process

After the online day ends, the batch process begins. The AIP batch scripts are executed in sequence. Most of the scripts call the binaries that are provided by AIP for the replenishment and allocation calculations. Also, some of the AIP scripts use the binaries/utilities provided by the underlying RPAS platform.

The AIP binaries are of two types:

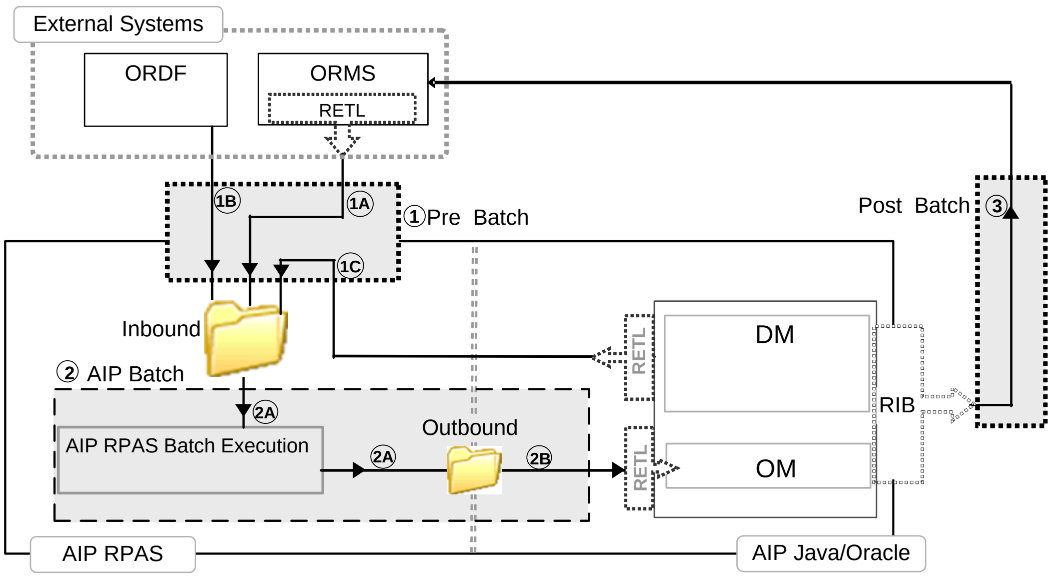
- AIP provides the binaries that are used for the replenishment and allocation calculations. The AIP binaries are C++ programs that are called by the batch scripts during their execution.
- AIP uses the binaries provided by the RPAS platform. These binaries are used for formatting and merging the data that is loaded from the external system and other modules.

The AIP batch process loads foundation, inventory and demand forecast data from external systems. This information is used in the AIP planning calculations.

For example, RMS provides the foundation and inventory data, and RDF provides the demand forecasting data, to AIP.

The AIP batch process is defined in three phases:

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



AIP 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 dabase. 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 the 28 steps
   - Formats and loads the RMS data files from a directory (Inbound folder)
   - Fetches, merges, and loads the DM online files from a directory
   - Loads the RDF files from a directory
   - Runs the replenishment and reconciliation logic
   - Exports the AIP RPAS batch output data into a directory (Outbound folder)
   - Generates alerts

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 script is 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 7 – 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.

After the necessary OM data is imported the 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.

Post AIP Batch

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

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.

AIP 12.1 provides the following features:

- A new error handling and logging architecture and its implementation; a minor code restructuring to classify the major tasks of batch processing, and a concurrent documentation effort
- Script parallel process management (logging, error control, and waiting) framework; the implementation of a new script architecture across the remaining scripts, and the refactoring of select scripts to reduce the critical path span

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 computer and database servers, simplified wrapping of file transport (FTP and copy) 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:

- 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 (ID/password lookup by server name).
- bsa_sql.sh – Utility functions to fetch values from, and executed stored procedures/functions on, the Oracle database.
- bsa_file_transfer.sh – Wrappers for file transfer primitives; ftp, mv, and cp, which use credential lookup as required.

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 parallely by other scripts. Each distinct call gets its own log. For example, if the main.sh script calls utility_1.sh once and utility_2.sh three times, the following log directories and files result:

```
••••2005-Dec-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 Succesful 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_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_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 three components:

- The BSA_MAX_PARALLEL environment variable
- The _para_spawn function
- The _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.

Access Credential Containment

Access credentials for all the physical computer servers and all the logical database servers used by the batch system are isolated to a single file defined in bsa_env.sh as ${CREDENTIALFILE}, which is by default named, bsa_cred.config. In this file, each record is keyed by an application. The application is simply a string key that must be coordinated between ${CREDENTIALFILE} and the bsa-enabled scripting application. It is suggested that this file be access protected, so that only the account under which the bsa-enabled scripts are run, may read it. The clear text file is located in the search path and formatted as follows:

```
<application>   <server>    <user>    “<password>”
<application>   <server>    <user>    “<password>”
<application>   <server>    <user>    “<password>”
. . .
```

Note that application field must begin in the first column without any preceding blanks. Note also that the password field must be enclosed in double quotes (“ ”), because passwords can contain blanks.

File Transfer Wrapping

New API functions, _file_transfer_single and _file_transfer_multiple, are constructed to supplant the direct use of FTP in a batch code. The signatures are as follows:

```
_file_transfer_single
    source_app,
    source_dir,
    source_filename,
    dest_app,
    dest_dir,
    dest filename
    mode (bin or ascii)

_file_transfer_multiple
    source_app,
    source_dir,
    source_filename_wildcard
    dest_server,
    dest_dir
    mode (bin or ascii)
```

Note that no server access credentials are passed to this API. Server credentials, if required, are looked up from ${CREDENTIALFILE}, defined by default in bsa_env.sh as server_cred.cfg for both *source_server* and *dest_server.* The _file_transfer_*xxx* functions determine whether it is necessary to perform FTP or simple cp, based on the server names. Also, if the batch server differs from both the source and destination servers, a remote shell is initiated to the destination server prior to the transfer. All uses of FTP (or optional FTP/local copy) are changed to use this API. Note that only the following cases are implemented at this time. Other cases will cause the error ${__CODE_UNSUPPORTED_FUNCTIONALITY}.

- Batch, source, and destination servers; all are the same
- Batch and source servers are the same, destination is different
- Batch and destination servers are the same, source is different

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.

5 AIP System Configurations

Environment Variables

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

AIP RPAS Environment Setup

The aip_common_rpas.sh script is included in all the AIP RPAS batch scripts. This script uses aip_env_rpas.sh to provide configurable environment information needed by the batch scripts. The variables defined in aip_env_rpas.sh are detailed in the table below. These variables must not be removed or have their names altered. The customizations should be made only on the right hand sides of the variable assignment statements. They should not be altered with typeset, or exported.

Variable Name	Description
AIPDOMAIN	Fully qualified path of the AIP RPAS global domain.
RPAS_INTEGRATION_HOME	Fully qualified path of the readable/writeable directory that serves mainly as a base for the other path definitions, later in aip_env_rpas.sh. Commonly set equal to AIPDOMAIN.
BSA_TEMP_DIR	Fully qualified path of the readable/writeable directory, where scripts can store temporary files. A valid definition of this variable is required by the BSA common scripts.
BSA_LOG_LEVEL	Script logging threshold severity. Only log entries at this or higher severity level will be written to the script logs. Must be one of `{PROFILE | DEBUG | INFORMATION | WARNING | ERROR}. A` valid definition of this variable is required by the BSA common scripts.
BSA_MAX_PARALLEL	Script parallel process fan-out maximum. The number of processes that any given process (script instance) can spawn. A valid definition of this variable is required by the BSA common scripts.
BSA_LOG_HOME	Fully qualified path of the readable/writeable directory, where script logs will be rooted. Script logs are written into a hierarchy that parallels the script call tree, rooted in a date stamped directory located in this specified directory. A valid definition of this variable is required by the BSA common scripts.
BSA_LOG_TYPE	Integer parameter that specifies the type of script log files to be written. Must equal one of { 0 | 1 | 2 | 3 }. 0 = No logging; 1 = Text ".log" files; 2 = XML structured ".xml" file; 3 = Text and XML log files. A valid definition of this variable is required by the BSA common scripts.

Variable Name	Description
BSA_CONFIG_DIR	Fully qualified path to the directory that contains the BSA configuration files, bsa_cred.config and bsa_fetch_files.config. A valid definition of this variable is required by the BSA common scripts.
BSA_ARCHIVE_DIR	Fully qualified path to the directory to which BSA file transfer operations will archive files. A valid definition of this variable is required by the BSA common scripts.
RPAS_LOG_LEVEL	RPAS binary logging threshold severity. The log entries at only this or higher severity level will be written to the script logs from binaries that accept a –loglevel argument. Must be one of `{PROFILE | DEBUG | INFORMATION | WARNING | ERROR}`.
ONL_OUTBOUND_DIR	Fully qualified path to the directory that contains the AIP Online output data. This path should match the like definition in aip_env_online.sh.
RAW_RMS_DATA_DIR	Fully qualified path to the directory that contains the untransformed RMS output data.
RMS_SCHEMA_DIR	Fully qualified path to the directory that contains the RETL schema files corresponding to the untransformed RMS output data.
AIP_SCHEMA_DIR	Fully qualified path to the directory that contains the RETL schema files depicting the transformed RMS output data.
RPAS_PAGE_SPLIT_PERCENTAGE	This variable is used to optimize AIP performance. Do not alter this setting without consulting AIP Technical Management.
ORACLE_AIP_PERISHABLE_ON	This UNIX variable is set to **yes** (ORACLE_AIP_PERISHABLE_ON=yes) when AIP is replenishing perishable products. This setting is case sensitive.

In addition to setting the aip_env.sh values, you should review the settings of the system kernal. Specifically, the max processes need to be reviewed. Depending on the system, execute:

- AIX 5.3: /usr/sbin/lsattr –E –l sys0 –a maxuproc
- Solaris9: /usr/sbin/sysdef | grep v_maxup
- HP-UX 11i: /usr/sbin/sysdef | grep maxuprc

Because 11+ processes can be started per thread, the maximum processes should be greater than approximately BSA_MAX_PARALLEL * 20.

AIP Java/Oracle Environment Setup

The configuration files for AIP Java/Oracle, while similar, are distinct from those for AIP RPAS, and have similar but unique names to avoid possible collision if AIP Java/Oracle and AIP RPAS are housed on the same server or volume. The aip_common_online.sh script is included in all the AIP Oracle batch scripts. This script uses aip_env_online.sh to provide configurable environment information needed by the batch scripts. The variables defined in aip_env_online.sh and detailed in the table below, must not be removed or have their names altered. Any customization should only be made on the right hand sides of the variable assignment statements. They should not be altered with typeset, or exported.

The AIP Oracle instance of the environment settings file may be slightly different than the AIP RPAS instance.

Variable Name	Description
AIPDOMAIN	Fully qualified path of the AIP RPAS global domain.
INTEGRATION_HOME	Fully qualified path of the readable/writeable directory that serves as a base for other path definitions. This value is also referenced by numerous AIP Online batch scripts.
BSA_TEMP_DIR	Fully qualified path of the readable/writeable directory, where scripts can store temporary files. A valid definition of this variable is required by the BSA common scripts.
BSA_LOG_LEVEL	Script logging threshold severity. Log entries at only this or higher severity level will be written to the script logs. Must be one of `{ PROFILE | DEBUG | INFORMATION | WARNING | ERROR }. A` valid definition of this variable is required by the BSA common scripts.
BSA_MAX_PARALLEL	Script parallel process fan-out maximum. The number of processes that any given process (script instance) can spawn. A valid definition of this variable is required by the BSA common scripts.
BSA_LOG_HOME	Fully qualified path of the readable/writeable directory, where script logs will be rooted. Script logs are written into a hierarchy that parallels the script call tree, rooted in a date stamped directory located in this specified directory. A valid definition of this variable is required by the BSA common scripts.
BSA_LOG_TYPE	Integer parameter that specifies the type of script log files to be written. Must equal one of { 0 | 1 | 2 | 3 }. 0 = No logging; 1 = Text ".log" files; 2 = XML structured ".xml" file; 3 = Text and XML log files. A valid definition of this variable is required by the BSA common scripts.
BSA_CONFIG_DIR	Fully qualified path to the directory that contains the BSA configuration files, bsa_cred.config and bsa_fetch_files.config. A valid definition of this variable is required by the BSA common scripts.
BSA_ARCHIVE_DIR	Fully qualified path to the directory to which the BSA file transfer operations will archive files. A valid definition of this variable is required by the BSA common scripts.
HAVE_WIP	Must be set to "false."

Variable Name	Description
DEFAULT_BSA_SQL_CRED_APP	Default application name for use in server credential lookup. This definition must match the application column of a record in bsa_cred.config.
RETL_MAX_HEAP_SIZE	The maximum memory heap size for use by the RETL application. Raising this limit may improve the performance on production systems.
RETL_CONFIG_FILE	Fully qualified path to the RETL application configuration settings file. This file contains the database connect information for RETL import and export.
NLS_LANG	Default language used by the RETL application.
ONL_OUTBOUND_DIR	Fully qualified path to the directory to contain AIP Online output data. This path should match the like definition in aip_env_rpas.sh.
ONL_INBOUND_DIR	Fully qualified path to the directory to contain externally generated data files to be loaded by AIP Online.
ONL_SCHEMA_OWNER	The schema owner of the DDL and DML used by AIP Online. This value is used by store_source extract to retrieve the table partition data.

Server Access Credential Configuration

The bsa_cred.config file contains single-line records that specify computer and database server access credentials, as also the related connection information in a generic form, which is used by batch scripts that require machine and database access. This information is isolated in the bsa_cred.config file because of its sensitive nature. This file should be access controlled so that only the process(es) which run(s) the AIP RPAS and the AIP ONLINE batch can read its contents. The bsa_cred.config file must be located in the path defined by BSA_CONFIG_DIR in the aip_env_rpas.sh and aip_env_online.sh configuration files. If these settings agree and the AIP Online and AIP RPAS both reside on the same server, then, a single instance of bsa_cred.config is sufficient. If not, distinct copies must be placed in the defined paths.

Each record of bsa_cred.config has the following fields in a space-delimited format:

- Application: Unique key indicating the application for which the record exists. This name is used as a key by the dependent batch scripts.
- Server: Computer or database server name, or “NA” if not applicable for the given application.
- User: User name credential or “NA” if not required for the given application.
- Password: Password credential or “NA” if not required for the given application. Because passwords may contain spaces, passwords must be encoded with enclosing quotes (“ “).

Records for the following application keys must be present:

- BATCH
- AIP-ONLINE
- AIP-RPAS
- DATABASE

Below is an example of the complete bsa_cred.config content:

```
BATCH         mspdev03    NA           "NA"
AIP-ONLINE    mspdev03    onlineusr    "onlinepw"
AIP-RPAS      mspdev03    rpasusr      "rpaspw"
DATABASE      devrtk11    dbuser       "dbpw"
```

6

AIP Interfaces and Tranformation Scripts

The AIP system interfaces with the merchandizing 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 and RDF.

RMS to AIP Interfaces and Transformation Scripts

During AIP installation, several RMS-AIP transform scripts and schema file were installed to a directory specified during installation. This location is refered to as $RMS_AIP_TRANSFORM for this section. (Refer to the *AIP Installation Guide* for more information about the installation details.)

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. Therefore $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

The execution of these scripts is controlled by running the master script, aip_t_master.ksh. These scripts will be located on the RPAS server, under $RPAS_HOME/bin. These scripts are run before the AIP RPAS batch is run.

Input Schema Files

Location: $RMS_AIP_TRANSFORM/rms_schema_dir

Input Schema files are the schema files which are required by the Transform Interface scripts to read the RETL extracts from RMS (or an external system). These files are owned by RMS and are not included with the AIP Package.

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

The $RMS_AIP_TRANSFORM/rms_schema_dir directory must be populated by copying the following Input Schema Files:

- 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

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.

The $RMS_AIP_TRANSFORM/aip_schema_dir directory must contain following output schema files:

- 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_AIP_TRANSFORM/raw_rms_data

The RETL Extracts from RMS are used as AIP loads/feeds. There are two kinds of RETL Extracts: one, which require transformation before being fed into AIP, and second, the set of extracts which do not require any transformation.

The $RMS_AIP_TRANSFORM/raw_rms_data directory holds both the RETL extracts that are to be transformed by the Transform Interface scripts, and the RETL extracts which do not require any transformation. These extracts will be moved from the RMS server (or another external system) to the $RMS_AIP_TRANSFORM/raw_rms_data location, by the user using a batch scheduler (which is not part of the AIP application suite).

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

- 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

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

- splr.txt
- dmx_dirspl.txt
- dmx_prdspllks.txt
- sr0_curinv.txt
- sr0_curinv_*.txt
- wr1_curinv.txt
- wr1_curinv_*.txt

Note: As shown above, sr0_curinv and wr1_curinv data feeds may be partitioned. For example: sr0_curinv_1.txt, sr0_curinv_2.txt, etc. AIP RPAS batch steps will combine the partitions for each current inventory feed, if they exist, before loading them into the sr0_curinv and wr1_curinv measures.

Finally, there are several RMS RETL extracts which are not used by AIP RPAS batch. However, the first two (closed_order.txt and received_qty.txt) are used by AIP Oracle batch. See below for guidance on these two feeds.

- closed_order.txt
- received_qty.txt
- dm0_onseffdt_.txt
- dm0_ofseffdt_.txt

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. AIP currently expects to load these files from an external system which produces AIP loadable file formats.

Running the Transform Scripts

Execute aip_t_master.ksh from location $RPAS_HOME/bin. This script will execute the 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 new files are created by the transforms, the files are moved to $AIPDOMAIN/interface/config/rms, which is a location defined by $AIPRMS in aip_constants_rpas.sh.

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

File Transfer for Order Management

The following two files must be moved, by using some kind of scheduled batch job, from the RMS server to the AIP Oracle server location defined by $ONL_INBOUND_DIR in the aip_env_online.sh file:

- closed_order.txt (must be renamed to closed_order.dat during the transition)
- received_qty.txt (must be renamed to received_qty.dat during the transition)

cron_import.sh 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.

RDF to AIP Interfaces

AIP does not provide transformation scipts 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. Customers 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

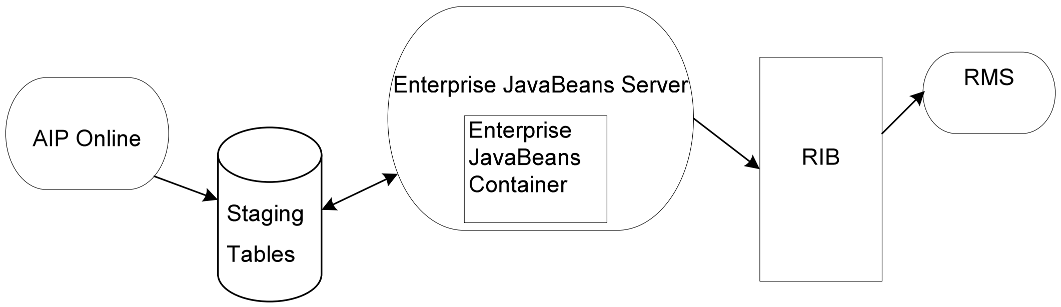
The table below displays the forecast data files that are loaded from the forecasting system into AIP.

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.
- EJBs “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)	
wi1_frcpo_.dat	This file contains warehouse Purchase Orders and Transfers to be released today. It also contains all Purchase Orders and Transfers forecasted fort he entire planning horizon. This file is loaded into the 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. 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 it 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/store_source/store_orders_tsf.

   Note: There is also a store_source directory in ONLINELOC/dm. This is **not** the correct directory.
5. Execute the in.sh script found in the store_orders_tsf 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.sh "srp".

Release Remaining Orders That Have Met Their Lead Time

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

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, or wi1_frcpo_.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.sh while loading strsplrord.dat (store_source/store_orders_po/in.sh).

Corrupt Data File

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

Load Remaining Order Files

1. Attempt to complete the load of the rest of the orders. Navigate to $INTEGRATION_HOME/config. Execute the remaining in.sh scripts listed in import_store_source.config.

Release Orders That Have Met Their Lead Time

2. Navigate to $INTEGRATION_HOME and execute cron_release_store_order.sh
3. Execute cron_release_non_contents_order.sh

Post Load Procedure Error

If the error is returned by import_store_orders the contingency purchase orders have likely been deleted in order to prepare the database to import the new purchase order plan.

Reload Contingency Purchase Orders and Remaining Order Files

1. 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.
2. Make a backup of strsplrord.dat by renaming it strsplrord.datbak.
3. 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.
4. Navigate to $ONL_INBOUND_DIR and rename strsplrord.dat1 to strsplrord.dat.
5. Navigate to $INTEGRATION_HOME/scripts. Execute: process_aiponline_data.sh -l "${INTEGRATION_HOME}/config/import_store_source.config". This will reload the contingency purchase orders produced yesterday as well as complete the load of the remaining order files.

Release Orders That Have Met Their Lead Time

6. Navigate to $INTEGRATION_HOME and execute cron_release_store_order.sh
7. Execute cron_release_non_contents_order.sh.

   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 where 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.sh while loading strwhord.dat (store_source/store_orders_tsf/in.sh).

Reload Contingency Purchase Orders and Remaining Order Files

1. Attempt to complete the load of the rest of the orders. Navigate to $INTEGRATION_HOME/config. Execute the remaining in.sh scripts listed in import_store_source.config.
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/store_source/store_orders_tsf. Execute the 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_store_order.sh
8. Execute cron_release_non_contents_order.sh.

Restore Backup Files

9. 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 wi1_frcpo_.dat

During the run of cron_import.sh while loading wi1_frcpo_.dat (store_source/non_contents_order_forecast/in.sh).

Corrupt Data File

If the error is not an error returned by the procedure import_non_contents_order or 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_store_order.sh.
2. Execute cron_release_non_contents_order.sh.

Post Load Procedure Error

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

No contingency file is available in the AIP release. The wi1_frcpo_.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 could be reloaded (including any orders that were released early).

Failure Point 5: Any other failure point in cron_import.sh

Failure to load any other data file will not impact the system's ability to load the store and warehouse orders listed in the table above. These files should be loaded either manually or via the scheduler. Subsequently, the scripts cron_release_store_order.sh and cron_release_non_contents_order.sh should be run.

7 AIP Batch Processing

The complete AIP batch processing comprises 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.

AIP Generic Batch Process Execution

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 Online platform. Foundation data is first populated on the RPAS platform. Then, the Java/Oracle data is moved to the Oracle database. Using this data, the DM user logs in to 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.

- 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.

AIP RPAS Batch Processing

The aip_batch.sh script is provided with the AIP installation, and it is used to run the entire RPAS 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. These top level scripts 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.

The AIP Batch Script (aip_batch.sh)

The AIP 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 aib_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
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.	✓	✓
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.		✓

Step Name	Description of Action	Initial Batch Run	Daily Batch Run
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.	✓	
get_onl_data	Gets the daily export from AIP Online and prepares files for loading onto 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.	✓	✓
load_rms_replenishment_data	Preprocesses and loads all the SRP and WRP measures that come from RMS.		✓
create_empty_archive_files	This script creates new item alerts for all the items.	✓	
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.	✓	✓

Step Name	Description of Action	Initial Batch Run	Daily Batch Run
auto_commit_wkbooks_batch	This script commits all the workbooks (SRP and WRP) on the AIP RPAS platform.		✓
set_implementation_parameters	Sets all the DM implementation parameters.	✓	
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.		✓
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.		✓
copy_sister_data	Copies the sister store and warehouse data by running the command xml files, copySisterStore.xml and copySisterWarehouse.xml.		✓
run_replenishment	Runs the replenishment and reconciliation actions, across the supply chain.		✓
run_replenishment_post_ processing	Executes the replenishment post-process by sequentially running the defined rule group.		✓
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.		✓
run_dm_alerts	Executes the 'dmb_master20.sh "${AIPDOMAIN}"'script to create alerts, if any.	✓	✓
export_dm_data	Extracts all the DM hierarchy and measure data required to keep DM RPAS and DM Online in sync.	✓	✓
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.		✓

Step Name	Description of Action	Initial Batch Run	Daily Batch Run
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 newtork alerts by running the defined WRP network alert rule group.		✓
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 aib_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
```

AIP Java/Oracle Batch Processing

The cron_export.sh, cron_import.sh, cron_release_store_order.sh, and cron_release_non_contents_order.sh scripts 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 three scripts that may have have multi-threaded process calls; however, all the scripts called directly by cron_export.sh, cron_import.sh, cron_release_store_order.sh, and cron_release_non_contents_order.sh, are run inline.

These top level scripts 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 dependancies and limitiations on what can be run in parallel.

Prerequisites for the AIP Online Batch Processing

The following prerequisites must be met before executing the AIP Online batch scripts:

- The machine running the interface scripts must have the ability to schedule CRON jobs to execute scripts on a timed schedule.
- RETL version 11.2 or later must be installed and available in the PATH.
- The Oracle 9i database schema must be installed and built with all the interface and "automation" procedures and functions compiled and available.

AIP Java/Oracle Interface Directory Structure

The table below provides information about the AIP Java/Oracle fiels and directories.

Script / Directory	Description
config.xml	The RETL configuration file containing the database connection information.
cron_import.sh	Executable automated import script
cron_export.sh	Executable automated export script
cron_release_non_contents_order.sh	Release into-warehouse purchase orders and transfers from the non_contents_order table to the po_/tsf_mfqueue tables
cron_release_store_order.sh	Release into-store purchase orders and transfers from the store_order table to the po_/tsf_mfqueue tables.
hierarchy/	The directory containing the hierarchical RETL scripts.
dm/	The directory containing the data management RETL scripts
store_source/	The directory containing the store order and warehouse order RETL scripts.
data/	The directory for the inbound data files.
scripts/	The location of automation scripts for data processing.
temp/	The temporary folder required for Batch Script Architecture (BSA).
config/	The location of configuration files, such as bsa_cred.config and import_dm.config.

Script / Directory	Description
logs/	The location of log files.
archive/	The location of the archived input data files.

Execution Sequence of AIP Java/Oracle Batch Scripts

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

Step Name	Description of Action	Initial Batch Run	Daily 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	Imports all the DM and hierarchy information into the Online database	✓	✓
cron_release_store_order.sh	Releases store orders to the external system, through RIB		✓
cron_release_non_contents_order.sh	Releases orders to the external system, through RIB		✓

8 AIP RPAS 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.

External Integration Data Processing

- Verify and Process Foundation Data from External System
  - check_process_external_data.sh
- Verify and Process Inventory Data from External System
  - check_process_inventory_data.sh
- Create Empty Hierarchy files
  - create_empty_default_files.sh

Internal Integartion Data Processing

- Retrieve Data from AIP Java/ Oracle platform
  - get_from_aiponline.sh

Process Inventory Management 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 Data into AIP RPAS

- Load All Hierarchies
  - load_all_hierarchies.sh
- Load Measures with AIP Online Data
  - load_onl_data.sh
- Load Data Management Inventory Data
  - load_rms_dm_measures.sh
- Load Replenishment Inventory Data
  - load_rms_replenishment_measures.sh

Generate New Alerts

- Create Empty Archive Files
  - create_empty_archive_files.sh
- Generate Batch and Online Alerts
  - load_all_newitem_alert_measures.sh

Load from External Systems

- Load Non-RMS External Files
  - load_non_rms_external.sh

Commit Workbooks before Batch Run

- Auto Commit Workbooks
  - workbook_batch.sh COMMIT

Set Implementation Parameters

- Set Implementation Parameters for DM and Relenishment
  - set_implementation_parms.sh

Perform Data Management Calculations

- Run initial load DM Batch
  - run_partial_dm_batch.sh
- Run DM Batch
  - dmb_master.sh

Calculate Replenishment across Stores And Warehouses

- Check Load Forecast Data
  - check_load_forecast_data.sh
- Purge and Advance Low-Variability Data
  - for_each_local_domain.sh -p purge_low_variability_advance_local.sh [DOMAIN]
- Copy Sister Stores and Warehouses
  - for_each_local_domain.sh -p copy_sister_data_local.sh [DOMAIN]
- Run Replenishment
  - scrp.sh
- Run Replenishment Post-processing
  - for_each_local_domain.sh -p scrp_post_local.sh [DOMAIN]

Export to AIP Online

- Export Replenishment Data
  - for_each_local_domain.sh -p export_scrp_inter_meas_local.sh [DOMAIN]
- Send_Replenishment to Online
  - send_scrp_measures_to_online.sh
- Run Non-critical Data Management Alerts
  - dmb_master_alerts.sh
- Export DM Data
  - export_dm_data.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

Build Workbooks after Batch Run

- Auto Build Workbooks
  - workbook_batch.sh BUILD

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

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 transformations 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.

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 Transformations. These data are also loaded into the RPAS domain as either hierarchy or measure 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

File Name	Explanation
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_prmspasc__i.txt	Store Promotional Space Shelf Capacity
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
srx_rltnte.txt	SKU Ad/Rollout Notes
ipfctwkprfd.txt	Week to Day Forecast Percentage Default (Un-Normalized)
ipfctwkprfe.txt	Store Week to Day Forecast Percentage Override (Un-Normalized)
sr0_prmprsstk_g.txt	Store Global Default Promotional Presentation Stock
sr0_prmprsstk_s.txt	Store Default Promotional Presentation Stock
sr0_prmprsstk_i.txt	Store Promotional Presentation Stock
ipwhhldcpci.txt	Stocking Point Holding Capacity
srx_poidst.txt	Poisson Distribution Lookup Table

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.

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
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/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
dmx_pcktyp.txt	Pack Type

   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 12.1 Implementation Guide for details on this configuring.

   Note that dmx_rmsskumap measure data is no longer loaded into RPAS effective with AIP 12.1. 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. Rename dmx_pcktyp.txt (output from interutil call) to dmx_pcktyp.ovr.
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, the 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.

Input Files from interface/config/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
splr.txt	hspl.config	hspl.txt
loc.txt	loc.config	loc.txt

8. Move non-RMS, external, measure data listed in $AIPDOMAIN/interface/config/external/measdata_from_external.config from the $AIPDOMAIN/interface/config/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. 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 the above tables should be copied by the client's batch scheduler into the $AIPDOMAIN/interface/config/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/config/rms directory. If the files were not FTPed correctly, rerun the script, but ensure that the required RMS files are present 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."

Verify and Process Inventory Data from External System

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 Appendix B – AIP RPAS File Formats. The flat files should have a *.txt extension.

File Name	Description
sr0_curinv.txt	Store Current Inventory
sr0_it_.txt	Store Intransits
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.txt is partitioned like sr0_curinv_X.txt (where X is 1, 2, 3…), merge all of the partitioned files. Rename single or consolidated data file to sr0_curinvinit.txt. Process similarly for wr0_curinv.txt.

3. 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
4. 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/config/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/config/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.

Create Empty Hierarchy 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.

Retrieve Data from AIP Java/Oracle Platform

Script Call

get_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 retrieves data from AIP Online. AIP Online sends hierarchy information and DM measures to AIP RPAS batch.

The get_from_aiponline.sh script is called to get data from the AIP Online export data directory. The data files have been placed in the export directory by the cron_export.sh script which is run on the Oracle server.

Day on Day Processing

1. fetch_files.sh is called two times. The first will fetch the AIP Online hierarchy export. The second will fetch the AIP Online measure export. The fetch_files.sh script is invoked with two keys that are listed inside $AIPDOMAIN/interface/config/bsa_fetch_files.config:
   - **DMo_hier_export** is keyed to retrieve the file hierarchy.tar.Z from the $ONL_OUTBOUND_DIR defined in aip_env_rpas.sh, and unpack the data contained within to the $AIPDOMAIN/interface/import/hier directory.
   - **DMo_meas_export** is keyed to retrieve the file aip.tar.Z from the $ONL_OUTBOUND_DIR, 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.
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 overwriten 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
4. Create the Direct Supply Point Flag data dm0_dirsppflg by transforming dm0_src_i into Boolean TRUE for supplier (V*, S*) sources and false for non-supplier sources.

This script calls the following scripts:

fetch_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 theAIP Oracle schema. If it did not, perform the restart/recovery steps for cron_export.sh.
4. Ensure that *.zip files were correctly FTPed to RPAS. If not, archive or remove the RPAS zip files under $AIPDOMAIN/interface/import.
5. Restart the batch from this step.

Notes

This step can be run in parallel with check_process_external_data.

Process Inventory Management 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 inventory management system (such as RMS) as well as from AIP Online.

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.

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

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 Online hierarchy file (matching the pattern *.dat) found in $AIPDOMAIN/interface/import/hier/, spawn a parallel instance of run_interutil.sh to execute the interutil binary with the LOAD_ALL_HIERARCHIES option. For each file $AIPDOMAIN/interface/import/hier/${hier}.dat, this creates an output file $AIPDOMAIN/input/${hier}.dat, formatted according to $AIPDOMAIN/interface/config/hier/${hier}.config.

This script calls the following scripts:

run_interutil.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:

merge_hierarchies.sh

Restart/Recovery

If this step fails, perform the following:

1. Examine the log filesm 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.
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

Script Call

reconfigAIPDomainPartitions.ksh

Functional Overview

This script is a wrapper script to the RPAS_HOME/bin utility reconfigGlobalDomainPartitions, to be used within AIP. It will check for the existence of specific configuration files that will trigger the addition or removal of partitions to the prod hierarchy

Technical Details

Adding new partitions to the domain is triggered by the existence of AIPDOMAIN/config/reconfigpartdim.xml. This file specifies new partition positions, and the local domain to which they are assigned. Removing positions along the partitioning dimension is triggered by the existence of a preformatted file, AIPDOMAIN/config/reconfigpartdim_remove.cfg. This file must contain a comma-separated list of positions to be removed.

Day on Day Processing

1. If both of the following files exist in directory $AIPDOMAIN/config:
   - reconfigpartdim_remove.cfg
   - prod.dat

   Then trigger dimension removal, and prepare hierarchy files for loading:
   - Temporarily move $AIPDOMAIN/input/prod.dat to a backup location
   - Call reconfigGlobalDomainPartitions to remove the positions indicated in reconfigpartdim_remove.cfg.
   - Move reconfigpartdim_remove.cfg to the $AIPDOMAIN/input/processed directory.
   - Restore prod.dat to $AIPDOMAIN/input/prod.dat
2. If both of the following files exist in directory $AIPDOMAIN/config:
   - reconfigpartdim.xml
   - prod.dat

   Then trigger dimension addition, and prepare hierarchy files for loading:
   - Temporarily move $AIPDOMAIN/input/prod.dat to a backup location
   - Call 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.
   - Restore prod.dat to $AIPDOMAIN/input/prod.dat

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 Data into AIP RPAS

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 fext, 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 Measures with AIP Online 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 calls 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 Measure Data

Script Call

load_rms_dm_measures.sh

Functional Overview

This step preprocesses and loads all measures that come from RMS or another external inventory system into the RPAS domain.

Technical Details

This script loads all measure data in $AIPDOMAIN/interface/config/rms which were deposited from the inventory system are considered Data Management measure data.

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

Prerequisits

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 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.

Load Replenishment Inventory Data

Script Call

load_rms_replenishment_measures.sh

Functional Overview

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

Technical Details

This script loads all measure data in $AIPDOMAIN/interface/config/rms which were deposited from the inventory system are considered as Replenishment measure 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.

This script calls the following scripts:

- loadmeasure.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 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.

Generate New Alerts

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

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 uses <hierarchy>.dat.last files to compare the new hierarchies with the old hierarchies. If used with the true flag, which implies that it is the first time running this step, empty hierarchy_name.dat.last files are generated for comparison purposes.

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.

This script calls the following scripts:

- load_one_newitem_alert_measure.sh

Appropriate Batch Run (First Time and/or Daily)

- First Time
- Daily

Prerequisits

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 from External Systems

Load Non-RMS External Files

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 non-RMS external system data into the RPAS domain.

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.

Committing Workbooks before Batch Run

Auto Commit Workbooks

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.

Set Implementation Parameters

Set Implementation Parameters for DM and Replenishment

Script Call

set_implementation_parms.sh

Functional Overview

This script is called to set configurable parameters for DM and replenishment. All parameters are contained in measure data and are set by rules whose text is editable in the shell scripts: set_implementation_parms_dm.sh and set_implementation_parms_aip.sh.

Technical Details

The script set_implementation_parms.sh calls the customizable scripts set_implementation_parms_dm.sh and set_implementation_parms_aip.sh. Each of the customizable scripts is a simple list of rules to be executed as mace commands on the global $AIPDOMAIN.

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 the desired value changes

Prerequisites

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 batch from this step.

Notes

The scripts set_implementation_parms_dm.sh and set_implementation_parms_aip.sh designed with the intention of being modified on a per-implementation basis. This is a special case; in general, it is expected that only top-level batch step scripts will be subject to modification in the field.

Perform Data Management Calculations

Run Initial Load 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, and resulting output data dm1_prfhme is loaded into the dm1_prfhme measure array.

Day On Day Processing

1. Runs dmcreateprf.sh shell script wrapper to the dmcreateprf AIP binary on the $AIPDOMAIN.
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.
4. The dm1_prfhme overlay (.ovr) file, created by dmcreateprf, is loaded into the $AIPDOMAIN with a call to loadmeasure.

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:

- Verify and Process Data from External Systems
- Load All Hierarchies
- Load Measure Data
- Load Non-RMS External Data
- Generate Batcn and Online Alerts
- set_implementation_parms.sh

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

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
- the store release schedule
- the store placement schedule
- the warehouse fixed period

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.

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 and C++ binaries. Each module uses some integration (e.g. RMS, RDF) and intragration (AIP Oracle/Online) inputs and creats 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 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 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	Description
1	dmcreateprf.sh	Create Profiles
2	dmwhsrc.sh	Calculate Warehouse Source
3	dmxwhchtowh.sh	Transform Warehouse Chamber Data to Warehouse
4	dmassignprf.sh	Profile Assignment

Order of Execution	Script	Description
5	dmcatchup.sh	Store Source Assignment
6	dmstrrelsch.sh	Calculate Store Release Schedule
7	dmstrplcsch.sh	Calculate Store Placement Schedule
8	dmnewcmd.sh	New SKU Ranging
9	dmcmwhal.sh	Ranging Warehouses to SKU-pack sizes
10	dmmntsupspl.sh	Maintain Source Splits
11	dmwhfsrcsch.sh	Calculate Warehouse Release Schedule
12	dmallwh.sh	Calculate Ranging status for Warehouse/SKU-pack sizses
13	dmstrordpks.sh	Calculate Store Ordering Pack Size
14	dmstrordunt.sh	Calculate Store Ordering Unit
15	dmconwhpalmul.sh	Convert Warehouse Pallet Multiple
16	dmsmlordpksz.sh	Calculate Smallest Ordering Pack Size
17	dmdgrspec.sh	Calculate Demand Group Specifier
18	dmpreprcstchg.sh	Identify Pre-Priced Status Change
19	dmwhrollups.sh	Warehouse Rollups
20	dminheritattr.sh	Inherit Attributes

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_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. 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.

Calculate Replenishment across Stores And Warehouses

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.

Purge and Advance Low-Variability Data

Script Call

for_each_local_domain.sh -p purge_low_variability_advance_local.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_low_variability_advance_local.sh script for each local domain.

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

This step's command line script call is a composite of two scripts, for_each_local_domain.sh and purge_low_variability_advance_local.sh, which cause the purge and advance 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 purgeLowVariabilityAdvance.xml. This XML resource file contains the list of measures on which the purge and advance operation applies. Please see file purgeLowVariabilityAdvance.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_low_variability_advance_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 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

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.

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_place	Generates store placement masks and orders
scrp_pre	Runs all the pre processing steps for AIP. Generates masks, store demand, and allocation boundaries for stores as well as other frequently used computation results.
scrp_replFixed	Runs the replenishment batch run within the fixed period and generates an unconstrained receipt plan
scrp_reconcShort	Runs the shortfall reconciliation when the demand is greater than the available inventory at the source and generates a constrained receipt plan
scrp_reconcSub	Runs the substitution reconciliation and substitutes alternate allowed pack sizes when there is still some unmet demand from destinations
scrp_stklessRec	Runs the stockless reconciliation for stockless items and pushes the excess quantity to the destinations
scrp_replPstFxd	Runs the replenishment batch run post fixed period and generates an unconstrained receipt plan for post fixed period
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. Restart the batch from this step.

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.

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 Replenishment & Reconciliation batch process

Rule Groups	Description
Local Rule Groups	The following rule groups are called by scrp_local.sh script
scrp_post	Does the post processing on local domains and generates

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

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.

Exporting to AIP Online

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.
- wi1_frcpo_.dat – Supplier to warehouse orders and 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)
- exportPlanWhOrdXfer.xml (wi1_frcpo_.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.

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. Tar and compress the global domain export files 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.

Run Data Management Alerts

Script Call

dmb_master_alerts.sh

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 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	Description
1	dmcmdstralerts.sh	Commodity/Store Alerts
2	dmmsgdatalert.sh	Missing Data Alert
3	dmnasprfalert.sh	Not Assigned to Profile Alert
4	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 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 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.
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 imported by AIP Online's initial batch step.
   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.

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

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
sr0_art__9	Store Alert 9: RDF Detail Alert
sr0_art__10	Store Alert 10: No Like SKU Found
sr0_arthstsum	History Availability Alert
sr0_artprjsum	Projected Availability Alerts

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 wrp_network_alerts.sh will run the RPAS utility alertmgr to find all alerts registered in the $AIPDOMAIN domain.

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
IpStlSurAltV	Stockless Surplus 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 wrp_network_alerts.sh will run the RPAS utility alertmgr to find all alerts registered in the $AIPDOMAIN domain.

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.

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

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.
5. Call alertmgr to find all generated alerts on $AIPDOMAIN.
6. 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:

- 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.

Building Workbooks after Batch Run

Auto Build Workbooks

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 aotumatically 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

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.

9

AIP Java/Oracle Batch Process

Directory Structure

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

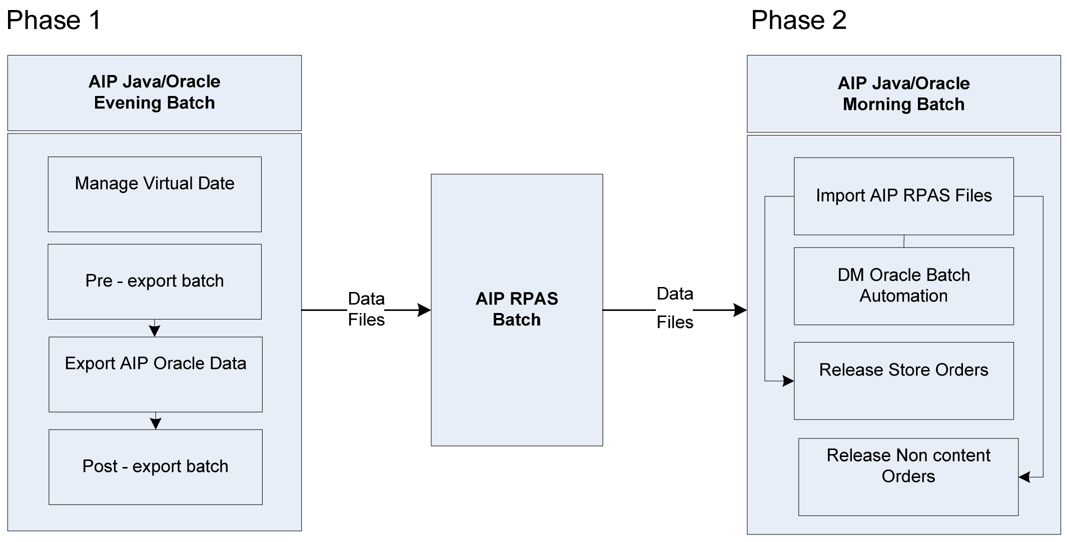
Files and Directories	Explanation
config.xml	The RETL configuration file containing database connection information.
cron_import.sh	The executable automated import script.
cron_export.sh	The executable automated export script.
hierarchy/	The directory containing hierarchy RETL scripts
dm/	The directory containing data management RETL scripts.
store_source/	The directory containing into-store and into-warehouse order RETL scripts.
data/	The directory for inbound data files.
scripts/	The location of SQL scripts for data processing.
temp/	The temporary folder required for Batch Script Architecture (BSA).
config/	Location of config files such as bsa_cred.config, import_dm.config
logs/	The location of log files.
archive/	The location of archived input data files.

Prerequisites for AIP Java/ Oracle Platform

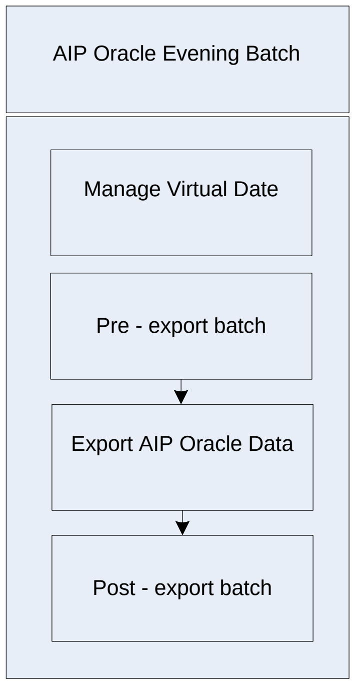
- The machine running interface scripts must have the ability to schedule CRON jobs to execute scripts on a timed schedule.
- RETL version 11.2 or later must be installed and available in the PATH.
- Oracle 9i database schema must be installed and built with all interface procedures compiled and available.

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 - This is a Morning batch executed in AIP Java/Oracle platform after the RPAS batch run. During this batch process, the data files are loaded from the RPAS platform, processed and finally the orders are released to the external system.



Configuration Setup

1. Edit the config.xml file with the correct parameters for your Oracle database.

   Example:

   ```
   <CONFIG>
     <DEFAULTS operator="oraread">
       <PROPERTY name="arraysize"      value="@oraread.arraysize@" />
       <PROPERTY name="hostname"      value="@oraread.hostname@"/>
       <PROPERTY name="port"          value="@oraread.port@"/>
       <PROPERTY name="dbname"        value="@oraread.dbname@"/>
       <PROPERTY name="connectstring" value="@oraread.connectstring@"/>
     </DEFAULTS>
     <DEFAULTS operator="orawrite">
       <PROPERTY name="hostname"      value="@orawrite.hostname@"/>
       <PROPERTY name="port"          value="@orawrite.port@"/>
       <PROPERTY name="dbname"        value="@orawrite.dbname@"/>
       <PROPERTY name="dbuserid"      value="@orawrite.dbuserid@"/>
       <PROPERTY name="method"        value="@orawrite.method@"/>
      </DEFAULTS>
   </CONFIG>
   ```

2. Edit the aip_env_online.sh file.

   a. Set the value of following variables.

      – The INTEGRATION_HOME variable, as the name suggests, should be the full path to the the interface home directory.

      – ONL_OUTBOUND_DIR is defaulted to ${INTEGRATION_HOME}/outbound is the directory location where cron_export.sh puts the exported files from AIP Online.

      – ONL_INBOUND_DIR (defaulted to ${INTEGRATION_HOME}/inbound) is the directory location from which cron_export.sh expects the inbound files from RPAS to be sourced.

      – ARCHIVE_DIRECTORY (defaulted to ${INTEGRATION_HOME}/archive) is the directory location where cron_import.sh sends the input data files for archiving.

   b. Edit the BSA variables, if necessary.

      The following BSA variables are defaulted, but they can be changed if required.

      – BSA_LOG_HOME is set to ${INTEGRATION_HOME}/logs.

      – BSA_CONFIG_DIR is set to ${INTEGRATION_HOME}/config. The BSA config files bsa_cred.config and bsa_fetch_files.config are located here.

      – BSA_TEMP_DIR points to the /temp folder required for Batch Script Architecture (BSA).

      – BSA_LOG_LEVEL is defaulted to INFORMATION. Valid values are PROFILE, DEBUG, INFORMATION, WARNING and ERROR. This determines the minimum level of logging performed.

      – BSA_LOG_TYPE is defaulted to 3. Valid values are 0=No logging; 1=Text Only; 2=XML Only; 3=Text & XML.

      – BSA_MAX_PARALLEL is defaulted to 4. This is the maximum number of processes that can be spawned from any other process.

- DEFAULT_BSA_SQL_CRED_APP is defaulted to DATABASE. It is used by bsa_sql.sh to do a lookup from bsa_cred.config to connect to AIP online database.

  Note: The value of DEFAULT_BSA_SQL_CRED_APP should not be changed.

  ---

c. RETL configuration variables:

- RETL_MAX_HEAP_SIZE is defaulted to 500M. Raise this limit to improve performance on production systems.

  Note: RETL runs within a Java Virtual Machine (JVM). Errors concerning the JVM stack size may be encountered when executing AIP Oracle batch processes. This value represents the amount of memory allocated to a single JVM thread and is defaulted by the JVM. The user may override it by setting the RETL_THREAD_STACK_SIZE variable in aip_env_online.sh or in their user profile.

  Example:
  ```
  export RETL_THREAD_STACK_SIZE=200000
  ```

  It can also be set in rfx.conf, the configuration file for RETL itself. However, modifying rfx.conf will affect all users accessing the RETL installation, not just those using AIP. When manipulating the JVM stack size, extreme care should be taken to prevent RETL from using an inordinate amount of the available physical memory.

- RETL_CONFIG_FILE is defaulted to ${INTEGRATION_HOME}/config.xml. Config.xml contains database connection information. This variable is used by RETL in.sh and out.sh extracts.

d. Batch server variable: The AIPDOMAIN variable needs to be set to the path on the BATCH server.

e. HAVE_WIP indicates if WIP is enabled to be exported. Its value is defaulted to false. WIP will not be implemented for AIP 12.1.

f. ONL_SCHEMA_OWNER should be set to the database schema owner. It is used by store_source extract. For example, if you are running AIP online extracts as aipdev121user or aipdev121, but the schema owner is aipdev121, then regardless of the running database user, ONL_SCHEMA_OWNER should be set up as aipdev121.

g. NLS_LANG is defaulted to AMERICAN_AMERICA.UTF8.

3. Edit the config files bsa_cred.config and bsa_fetch_files.config located in the ${INTEGRATION_HOME}/config directory to enter implementation specific parameters.

   a. The bsa_cred.config file contains the credentials for the various servers used by AIP.

   b. The bsa_fetch_files.config file contains file names and source/destination servers/directories for files used by AIP intragration. The unpack directory and required/optional config file location are applicable to .tar.Z files.

4. Configure a CRON job that will execute vdate.sh once daily, after the close of business, and before cron_export.sh has begun.

5. Configure a CRON job that will execute cron_export.sh once daily, after the close of business, and before RPAS batch has begun.
6. Configure a CRON job that will execute cron_import.sh once daily, after RPAS batch has completed for the modules that need to be imported. Possible options: [dm | alerts | srp | om]. For example:

   ```
   cron_import.sh dm
   ```
7. Configure a CRON job that will execute cron_release_store_order.sh once daily. This must be executed after SRP data has been imported into the AIP Oracle database.
8. Configure a CRON job that will execute cron_release_non_contents_order.sh once daily. This must be executed after warehouse replenishment data has been imported into the AIP Oracle 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 transfered to RPAS before RPAS batch begins.
2. The cron_export.sh process begins by running a pre-export, pre_aiponline_extract.sh, script. This scripts 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 to AIP Online

1. After RPAS batch has successfully completed, 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 [DOMAIN]/interface/export directory of each DM, and SRP domains.
2. When the CRON job begins the cron_import.sh script, first the 'data_directory' is cleaned out.
3. The script uses FTP to retrieve the inbound files from the RPAS server. If all files do not exist, the process repeats according to the max_retries parameter, waiting sleep_time seconds between tries.
4. Once files are successfully transferred, the remote files are renamed with a date and time stamp, so the same files are not imported again the next day.
5. Files are uncompressed into the data_directory and the loading process begins.
6. A log file is generated at ${INTEGRATION_HOME}/logs.

Purchase Order/Transfer RELEASE from AIP Online to Merchandising System

The into-store and into-warehouse PO/TSF release processes are both preceeded 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.

10 AIP Java/Oracle Batch Scripts

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 Oracle Data
- Export DM and OM Data for AIP RPAS Processing
- Import RPAS Data into AIP Oracle Database
- Release Orders to RMS
  - into-Store Purchase Orders and Transfers
  - into-Warehouse Purachase Orders and Transfers

Batch Execution for AIP Java/Oracle Platform

The batch scripts are executed on the AIP Online platform 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 most, if not all, 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 low level scripts will be modularized such that they can be setup in the scheduler, with the appropriate dependencies, and run individually (followed by the scheduled dependent process(es)).

The DM scripts that are executed prior to RPAS AIP Batch scripts are called DM Pre-batch.

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 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 should not be run twice in the same day's batch run.

Input Parameters

This script will accept the following commands:

- **set** [date in YYYYMMDD]

  This function performs the following:
  - calls the SET_VDATE function (using call_bsa_sql.sh)
  - Write vdate value to text file.
  - If transfer true, transfer file to AIP-RPAS server (using BSA file transfer function)
- **inc**

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

  This function performs the following:
  - calls the INC_VDATE function using call_bsa_sql.sh)
  - calls the GET_VDATE function (using call_bsa_sql.sh)
  - echo the output of GET_VDATE and write vdate value to text file
  - if transfer=true, transfer file to AIP-RPAS server (using BSA file transfer function)

For example,

Script Call	**Explanation**
vdate.sh	retrieves the vdate
vdate.sh get	retrieves the vdate
vdate inc	increments the vdate, without transfering it
vdate inc notransfer	increments the vdate, without transfering it
vdate inc transfer	increments the vdate and transfers it
vdate set notransfer 20001130	sets the vdate, without transfering it
vdate set transfer 20001130	set the vdate and transfers 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 configurable 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 configurable 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.

It is important to note that this export planning horizon is not directly linked to the planning horizons maintained in the DM Online application. However the data that is exported directly effects the replenishment plan batch calculations. It is pertinent that all data is exported for at least the duration of its corresponding planning horizon. The system only maintains a single global export planning horizon. Therefore, it is recommended that the export planning horizon be equal to or greater than the largest planning horizon defined in the AIP system.

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.

Export DM Oracle Data

The data created or modified in the Data Management application must be exported for communication to AIP RPAS. All export data files will be created in an outbound data directory. The files will be grouped into Hierarchy and DM. Each group will be compressed for retrieval by AIP RPAS.

Post-export Batch

Export Timestamp Maintenance

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.

Export DM and OM Data for AIP RPAS Processing (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 in SRP and WRP in order 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 to export all the data from DM and hierarchy data for 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 scripts loop through the following directories and execute each of the sub-directory's corresponding out.sh script:

- Hierarchy
- DM

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

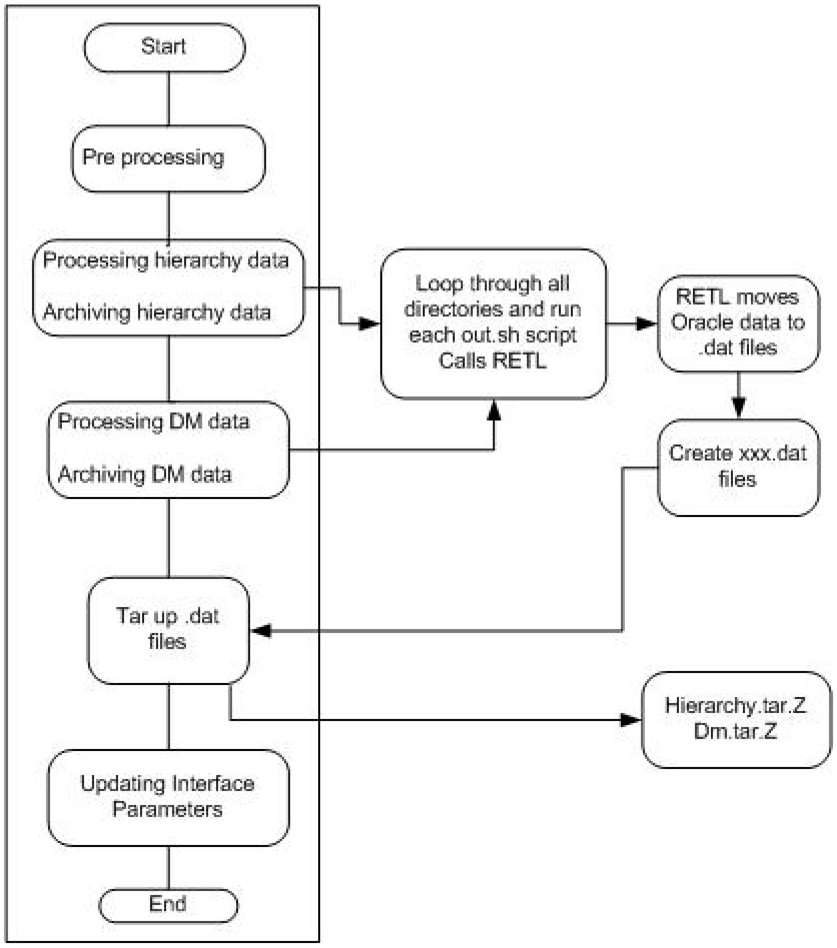
- out.sh - contains the SQL query or a call to outflow.xml
- schema.xml - This defines the .dat file format using xml tags.
- outflow.xml - (optional) contains the SQL query when used

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 tomorrow through the entire planning horizon (as set in the Config file).
  - This table is used as a join in the SQL statement within out.sh or the outFlow.xml.
  - 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.


Start
Pre processing
Processing hierarchy data
Archiving hierarchy data
Loop through all directories and run each out.sh script Calls RETL
RETL moves Oracle data to .dat files
Processing DM data
Archiving DM data
Create xxx.dat files
Tar up .dat files
Hierarchy.tar.Z
Dm.tar.Z
Updating Interface Parameters
End

Export Flow Diagram

cron_export.sh script

Input Parameters

noTimestamp • 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 directories named hierarchy, dm, store_source in the location specified, with out.sh scripts that execute the out flows.

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 out.sh 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 of the export

pre_aiponline_extract.sh script

This script executes by incrementing the planning horizon and 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

The out.sh scripts make a single call to RETL with a database connection parameter. The out.sh scripts will either contain the SQL query or will tell what flow file contains the SQL query. One out.sh script exists for each directory. When more than one file is being generated, the out.sh script will contain calls to flow scripts, which each contain the SQL query and unique file name being generated.

schema.xml: This shows the format of the .dat file using xml tags.

outFlow.xml: This tells RETL the exact name of the .dat file, and it contains the SQL statement that is used to extract the data from the oracle table.

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) 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

The out.sh scripts make a single call to RETL with a database connection parameter. The out.sh scripts will either contain the SQL query or will tell what flow file contains the SQL query. One out.sh script exists for each directory. When more than one file is being generated, the out.sh script will contain calls to flow scripts, which each contain the SQL query and unique file name being generated.

schema.xml: This shows the format of the .dat file using xml tags.

outFlow.xml: This tells RETL the exact name of the .dat file, and it contains the SQL statement that is used to extract the data from the oracle table.

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

- assigned_commodity
- case_weight
- 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
- home_warehouse
- 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
- push_singles_flag
- ranged_commodity_pack_size
- rdc_reconciliation
- rdc_reconciliation_exceptions
- receipt_to_availability
- release_schedule_exceptions
- singles_enabled_commodity
- singles_enabled_warehouse
- sister_store
- sister_wh
- source_split_tb
- stackability
- 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
- supplier_placement_exceptions
- supplier_release_exceptions

- valid_warehouse
- vendor_lock
- warehouse_coupled_flag
- warehouse_pallet
- warehouse_planning_horizon
- warehouse_promotional_dates
- warehouse_rollout_status
- warehouse_schedule

e) 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.

e) 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 can be run in parallel with check_process_external_data. This step must run before load_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.

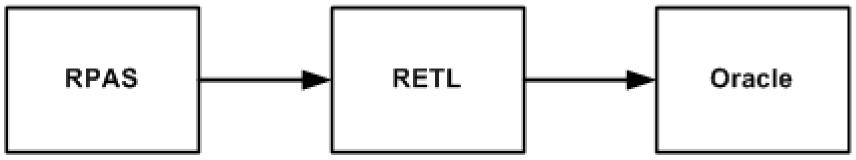
Import RPAS Data into AIP Oracle Database

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 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. The Oracle import process begins the data load by retrieving the flat files from the RPAS domains. Once retrieved, 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.

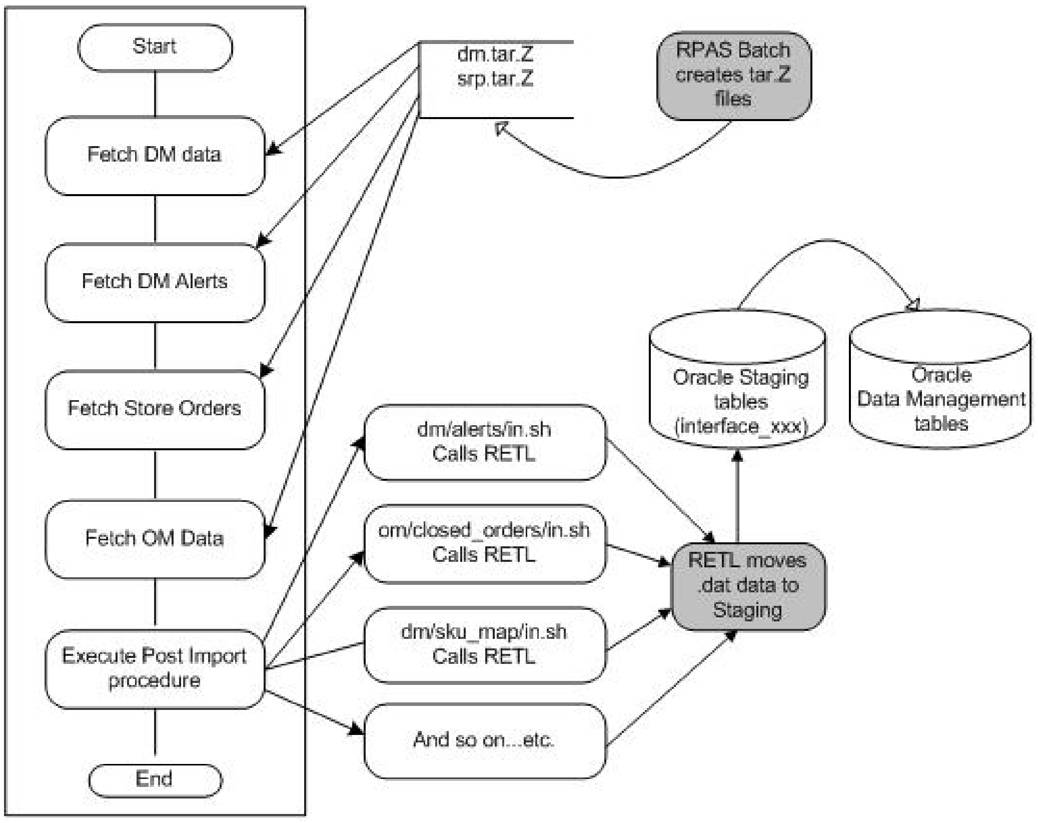


Importing Flow

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:

- in.sh
- schema.xml


Start
Fetch DM data
Fetch DM Alerts
Fetch Store Orders
Fetch OM Data
Execute Post Import procedure
End
dm.tar.Z
srp.tar.Z
RPAS Batch creates tar.Z files
Oracle Staging tables (interface_xxx)
Oracle Data Management tables
dm/alerts/in.sh Calls RETL
om/closed_orders/in.sh Calls RETL
dm/sku_map/in.sh Calls RETL
And so on...etc.
RETL moves .dat data to Staging

The cron_import.sh script currently performs three main functions:

1. Fetch data from the AIP RPAS server or location (fetch_ files.sh)
2. Import data into the AIP Online database (process_aiponline_data.sh)
3. Call any post-load processing (if necessary) (post_retl_import_wrapper.sh)

This script is a wrapper for a series of scripts.

This script will be used to retrieve the following data sets:

Hierarchy data: This will include Hierarchy data files. The 'process_aiponline_data.sh' script is executed with the parameter 'import_hierarchy.config'.

DM data: This will include DM data files. The 'process_aiponline_data.sh' script is executed with the parameter 'import_dm.config'.

DM Alerts data: This will include non-critical path DM Alerts.

Store and Warehouse Orders: The 'process_aiponline_data.sh' script is executed with the parameter 'import _store_source.config'.

OM data: This will include the files to be loaded into OM.

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 new post-load SQL processing script to execute the Automated Data Maintenance batch scripts.
- Import Store and Warehouse Orders
  - Retrieve the file containing Store Order data and Warehouse Order data
  - Call the import processing script with "store_source" 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

Scripts

The scripts fetch_ files.sh and process_aiponline_data.sh are executed to fetch and process the files by passing the appropriate parameters.

Input Parameters

Module name (options: [dm|alerts|srp|om])

Restart and Recovery

The file retrieval step simply retrieves a file from a remote location, and then archives that file. If the retrieval fails, then the process should be restarted once the cause of the error has been corrected. If the archive step fails, then the file should be manually archived and the next step in the process started.

The import process will execute many in.sh scripts in parallel. If this fails, it will be because one or more of the in.sh 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 in.sh 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 in.sh scripts from the command line.
3. Create a config file containing the list of only the in.sh 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 after the AIP Oracle laod scripts are called DM Post-Load batch. It is also refered to as Automated Data Maintenance Batch.

The following descriptions of each process provides 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 warehouse profile. This profile is created during DM RPAS batch.

The new supplier will also 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.

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.

The automatic assignment of SKUs to warehouse profiles is performed in DM 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 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 Flag – can override the demand group assignment configuration option. The Inventory Tracking Flag is a global Boolean set at implementation time. When it is set to “yes” 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" values 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 vadate. 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 recommened 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 arbitrairily 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 succesfull copy to occur. Otherwise the user must manually complete the supply chain setup.

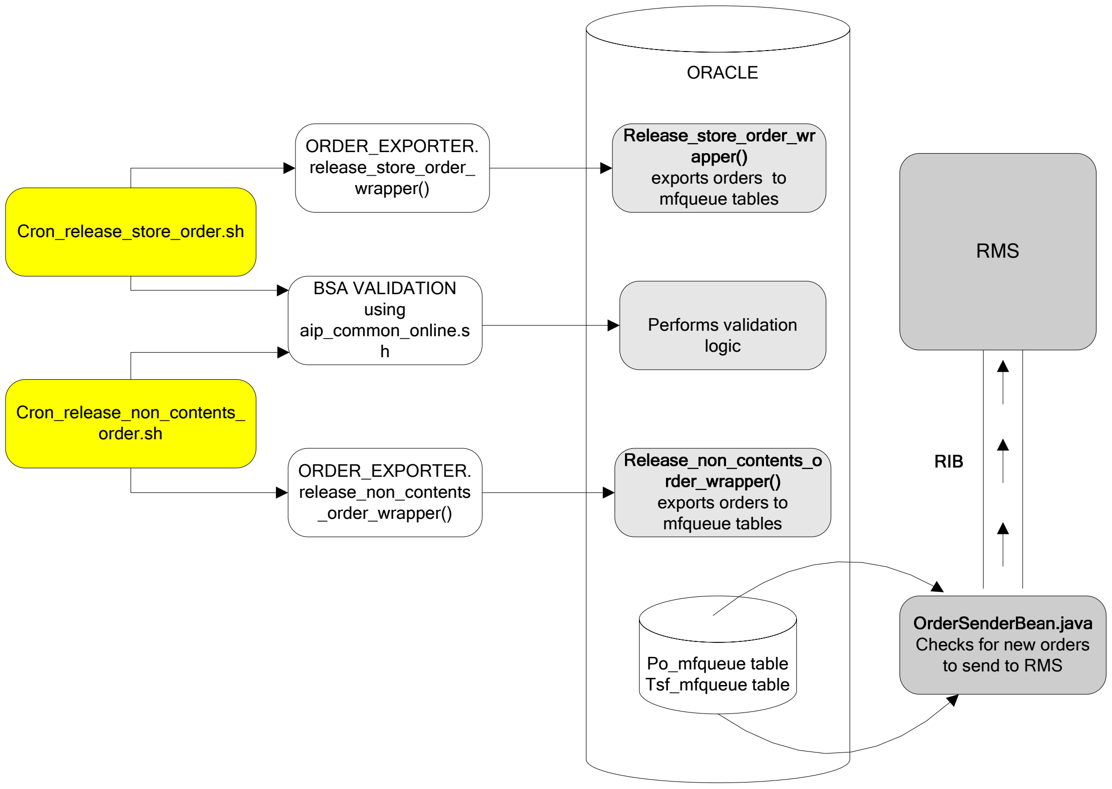
Because the only recourse for a failed copy is manual correction or setup it is highly recommened that the introduction of new warehouses with a sister warehouse copy be a tightly controlled process. To control the process the introduction of the new warehouse should not occur until the warehouse open date falls within the copy timeframe. To additionally safeguard this process the 'sister warehouse offset weeks' system parameter can be set to an arbitrairily large number that is less than the maximum export horizon system parameter (converted to weeks).

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 OM. OM 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.


ORACLE
ORDER_EXPORTER. release_store_order_ wrapper()
Release_store_order_wr apper() exports orders to mfqueue tables
Cron_release_store_order.sh
RMS
BSA VALIDATION using aip_common_online.s h
Performs validation logic
Cron_release_non_contents_ order.sh
RIB
ORDER_EXPORTER. release_non_contents _order_wrapper()
Release_non_contents_o rder_wrapper() exports orders to mfqueue tables
Po_mfqueue table
Tsf_mfqueue table
OrderSenderBean.java Checks for new orders to send to RMS

Order Release Flow

1) Release Store Orders

Functional Overview

A batch release of store orders is performed once in the morning before the online application is made available. All store orders scheduled for release today will be communicated to RMS. The orders are assigned a purchase order number or transfer number, and the order status is then set to 'Open.'

Technical Details

This process releases store transfers and 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 in order to generate the messages sent to RMS via the RIB.

cron_release_store_order.sh script

Input Parameter

None

Day on Day Processing

The script calls PL/SQL procedure in order to perform the release of store purchase orders and transfers:

- ORDER_EXPORTER package
  - release_store_order_wrapper

ORDER_EXPORTER.release_store_order_wrapper

Input/Output Parameters

error_message

string (VARCHAR2 256) output parameter to pass back to the calling function for error message logging.

Day on Day Processing

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.
- Checks to see if order purge periods exist for both purchase orders and transfers.

  The order purge period is used to purge past, closed orders.

- 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.
- 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.'
- Sets VALIDATION_FAILURE_IND to 'Y' if order quantity is 0.

  This prevents the order from being released to RMS.
- Calls ORDER_EXPORTER.release_store_order to execute the release logic on orders that do not have a VALIDATION_FAILURE_IND equal to 'Y'.

ORDER_EXPORTER.release_store_order_wrapper

Input/Output Parameters

string (VARCHAR2 256) output parameter to pass back to the calling function for error message logging.

Day on Day Processing

This function performs the following:

- 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.
- 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 'XORDERCRE' message is inserted into the PO_MFQUEUE table for each purchase order being released.
- An 'XTSFCRE' message is inserted into the TSF_MFQUEUE table for each transfer being released.
- The status of the released POs and TSFs are updated to 'O' – open.
- POs and TSFs that have been closed for X days will be purged from the STORE_ORDER table. X is defined in the ORDER_PURGE_PERIOD table for each order type.

Restart/Recovery

If this step fails, 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.

> **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.

> **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	N	N	Y	Y
W	P	Y	N	N	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	Y	Y	Y	Y
W	P	Y	Y	Y	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
AssignPONumToStoreOrde	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
PurgeStoreOrder	O_errorMessage	0 for success; 1 for failure

The PurgeStoreOrder function is not a critical step. It purges historical closed orders. A failure in any other function will require a restart of cron_release_store_order.sh once the error is resolved.

Dependencies

The ability to release orders to RMS depends upon having the RPAS Replenishment Planning batch generated orders loaded in the Oracle database. Therefore, the script should not run until import is complete. This script must run prior to the Online day.

2) Release Non-Contents Orders

Functional Overview

A batch release of non-contents orders (warehouse orders) are performed once in the morning before the online application is made available. All non-contents orders 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 non-contents warehouse transfers and 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.

cron_release_non_contents_order.sh script

Input Parameter

None

Day on Day Processing

The script calls the PL/SQL procedure in order to perform the release of store purchase orders and transfers:

- ORDER_EXPORTER package
  - Release_non_contents_order_wrapper

ORDER_EXPORTER.release_non_contents_order_wrapper

error_message

string (VARCHAR2 256) output parameter to pass back to the calling function for error message logging.

Day on Day Processing

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.
- Checks to see if order purge periods exist for both purchase orders and transfers.

  The order purge period is used to purge past, closed, store orders.
- 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 you to view in DM Online.

  This does not cause the release process to halt.

  The remaining SKU/pack sizes with valid mappings are released.
- Sets the status of any overdue orders to a status of 'V.' Overdue orders are orders where the delivery date has passed and the order is not fully received or closed.
- 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

Input/Output Parameters

O_error_message

string (VARCHAR2 256) output parameter to pass back to the calling function for error message logging.

Day on Day Processing

This function performs the following:

- The release candidate table (CANDIDATE_PO_DU) is built.
- The release date is calculated for each non-contents order that is 'Un-released' and had not failed validation (VALIDATION_FAILURE_IND = 'N').
- 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.
- 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 'XORDERCRE' message is inserted into the PO_MFQUEUE table for each purchase order being released.
- An 'XTSFCRE' message is inserted into the TSF_MFQUEUE table for each transfer being released.
- The status of the released POs and TSFs are updated to 'O' – open.
- POs and TSFs that have been closed for X days will be purged from the NON_CONTENTS_ORDER table. X is defined in the ORDER_PURGE_PERIOD table for each order type.

Restart/Recovery

If this step fails, perform the following steps:

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.

> **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.

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 release_store_order in the order listed:

Function	Parameters	Return Value
buildCandidatePODU	O_errorMessage, I_firstDeliveryDate, I_planningHorizon	0 for success; 1 for failure
preprocessNonContentsOrder	O_errorMessage	0 for success; 1 for failure
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
purgeNonContentsOrder	O_errorMessage	0 for success; 1 for failure

The purgeNonContentsOrder function is not a critical step. It purges historical closed orders. A failure in any other function will require a restart of cron_release_non_contents_order.sh once the error is resolved.

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 import is completed.

This script must run prior to the Online day.

11

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 Installation 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 Installation 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.gem – The master domain contains the Acumate database configmeasdata.gem, located in $AIPDOMAIN/data directory. This database 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.gem – Each local domain contains the Acumate database admin.gem, located in each local domain's data directory. This database 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.

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 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 several supported operations that can be performed 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."

Repairing / Updating Domain References

If the array references between local and master domains are ever left in an incorrect state due to movement or copying of domain pieces that do not keep these references up to date (such as using mv or cp commands to move or copy domain pieces), these directory path references must be corrected. The following utility, updateGlobalPointers, is used to update these directory paths:

```
updateGlobalPointers –d  [ Path to the master domain]
```

This will prompt with old local domain paths and ask for new local domain paths to be entered. All paths in the globaldomainconfig.xml, as well as the Acumate databases admin (for locals) and configmeasdata (for master), are overwritten with absolute directory paths to the domain components. If the domain references were relative before (that is, the local domain's reference to the master are simply "..", and the master domain's references to the local domains are simply, e.g., "ldom0" and "ldom1")

Moving a Local Lomain

If required, a local 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.

Note: `moveDomain` only works on local domains.

Moving a Master Domain

If required, the master domain may be moved from one directory to another. This requires a combination of UNIX command usage as well as AIP utility usage.

1. Execute the appropriate `mv` UNIX command to move the master domain to the desired new path.
2. Run the AIP utility, `updateGlobalPointers`, to update the references between the master and local domains.

Note: `moveDomain` only works on local domains.

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 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 12 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.