Rabobank – HCP Ingestion Engine

Administrative Guide

Author: Clifford Grimm

Date: 11/4/2015 5:56 PM

Revision: 2.0

Table of Contents

1 INTRODUCTION 1

1.1 Terminology 2

2 SOLUTION OVERVIEW 3

2.1 Data Set Processing 4

3 RUN-TIME ENVIRONMENT 5

3.1.1 Data Area Layout 6

4 CONFIGURATION 8

4.1 General Configuration 8

4.2 Data Set Specific Configuration 9

4.2.1 RegionMapper_*.properties File Format 10

4.2.2 AccountMapping.lst File Format [Bloomberg ONLY] 11

4.2.3 Configuration Notes 11

4.2.3.1 Reuters 12
4.2.3.2 Bloomberg 12
4.2.3.3 Verint 12
4.2.3.4 Algomi 13

5 EXECUTION SCRIPTS 14

5.1 Data Set Processing Scripts 14

5.2 Output Logging Wrapper 14

6 MONITORING/DEBUGGING 15

6.1 Completion Status Monitoring 15

6.1.1 Script Output Example 16

6.1.2 E-Mail Status Example 17

6.1.3 Search Console Example 18

6.2 Logging 20

6.2.1 Logging Configuration 21

6.2.1.1 BASH Script Debug Logging 21

6.2.1.2 COMET Java logging 22

6.2.1.3 HTTP Response Headers 23

6.2.2 Log File Search Failure Tips 24

7 FAILURE AND RECOVERY PROCEDURE 25

7.1 Collection Processing 27

7.1.1 Collection Warning 27

7.1.2 Collection Failure 27

7.2 Validation Failure 27

7.3 Duplication Failure 28

7.4 Ingestion Failure 29

7.5 Residual File Failure 30

7.5.1 Expanded Ingestion Failure 30

7.5.2 Unknown Region 31

7.5.2.1 New Region Introduced 32

7.5.3 Unexpected Files 33

Revision History

Date	Author	Revision	Description
11/4/2015	Clifford Grimm	2.0	Editorial Review. Added pictures of Search Console reporting. Finalized for V2.0 of the solution.
10/30/2015	KC Kancherla	1.1	Added new Algomi Data Set
9/2/2015	Clifford Grimm	1.0	Second pass review completed.
8/25/2015	Clifford Grimm	0.1	Initial Revision

1 Introduction

Rabobank is a consumer of the Hitachi Data Discovery Suite (HDDS) and the Hitachi Content Platform (HCP). These platforms will be used as part of a solution to provide a compliance application with an intuitive search interface focused on locating various communication data based on a comprehensive but specialized criteria.

The data that will be searched by this solution consists of:

1) E-mail
2) Reuters Instant Messaging
3) Bloomberg Messages
    a. IB Compliance
    b. Message Compliance
4) Verint call recordings
5) Algomi fixed income messaging

The overall solution consists of two parts:

1) Data Ingestion Process Service consisting of custom software that will ingest the Reuters Bloomberg, Verint and Algomi data into HCP in a form that is consumable by HDDS.
2) Search Interface Service custom built search interface with criteria specification as required by Rabobank.

This document will be focusing on the data ingestion portion of the solution and will be only ingesting the Reuters, Bloomberg, Verint and Algomi content.

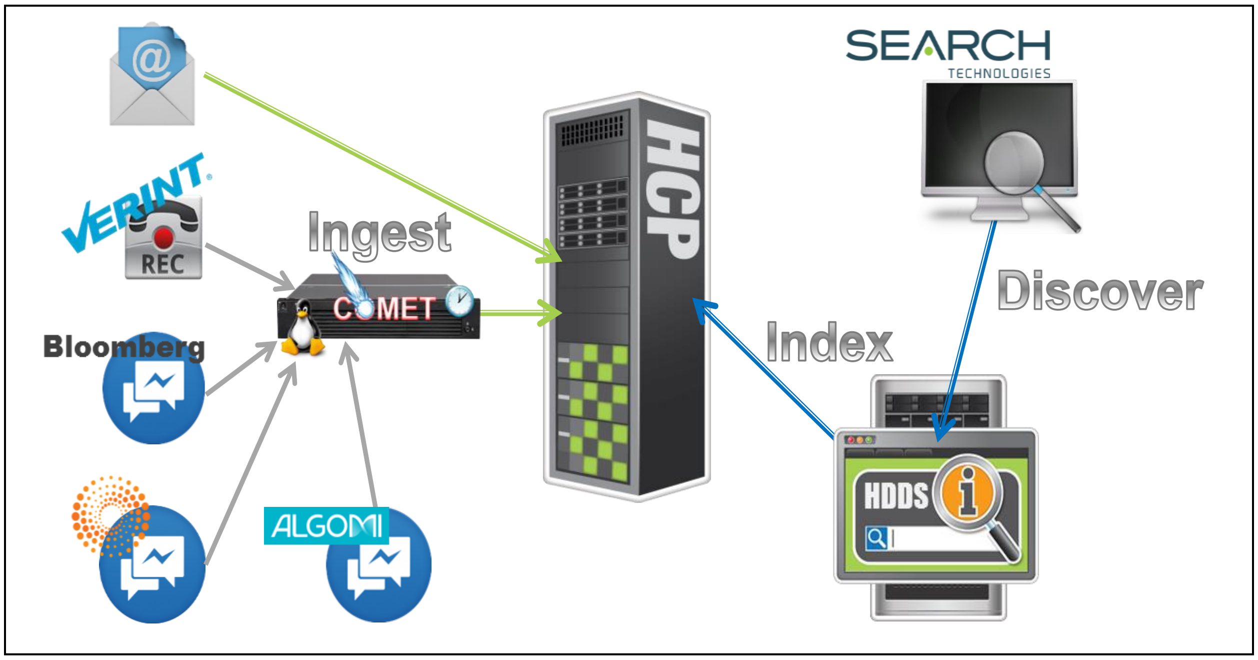


Figure 1: Solution Overview

1.1 Terminology

Term	Definition
Custom Metadata	An HCP concept where detailed information can be attached to HCP objects to be used for providing further value to the HCP objects. This custom metadata is usually in the form of XML to facilitate indexing.
Data Set	Refers to a type of data that is provided to this solution. The types discussed in this document are `Bloomberg`, `Reuters`, `Verint` and `Algomi`.
Data Set Collection	A Data Set specific group of files/objects for a period of period as designated by the Data Set gathering mechanism. Currently, every data set collection is performed every 24 hours, thus consists of 24 hours of data.
Data Set Item	Refers to the most basic item that is ingested for any given data set. A data set item could be a single message, chat/conversation session, or a call recording wav file.
Quarantine Area	The file system folder structure that is used to place content that cannot be processed due to various reasons. The content stored in this area needs to be manually evaluated for the cause of the failure and manual corrective actions taken.
Raw Data Set	The original form of the content for a given Data Set as provided by external processes for processing by this solution. (a.k.a. `RawData`)
Region Mapping	The action of taking a region code extracted from the Data Sets and mapping it to an HCP Namespace that is to receive the content.
Staging Area	The file system location that Raw Data Set is placed by the external systems and is used as the raw input for this solution.
Transformed Data Set	This solution receives the Raw Data Set and performs various transformation of the content to facilitate usage by HDDS and the Search Console part of the solution. (a.k.a `TransformData`)
Work Area	The file system folder structure that is used as a temporary storage area for assembling and transforming content in preparation for ingestion into HCP. (a.k.a. `WorkArea`)

2 Solution Overview

The ingestion portion of the solution is responsible for accepting the different data sets and ingesting them into the HCP system in a manner that will provide for robust search by the HDDS system. The current types of data sets to be processed are Reuters, Bloomberg, Verint and Algomi.

The data sets are written onto a file system of the ingestion system by a mechanism provided by Rabobank. The data sets are organized by the data set type and further are organized in a manner that identifies the geographic region for which they belong.

The ingestion engine detects the existence of the data sets and process them as appropriate for each data set. This processing consists of transforming the data prior to ingestion either by reorganizing or transforming in a manner that facilitates ingestion into HCP and indexing by HDDS.

During content ingestion into HCP, the content is organized into namespaces appropriate for that data and the region for which the data is for. In addition, when the content is ingested into HCP, it is further transformed into appropriate folder structure and/or generate custom metadata attached to the object. Folder structure is important to adhere to HCP best practices around folder and object count. The custom metadata will facilitate the indexing and search capabilities of the HDDS product.

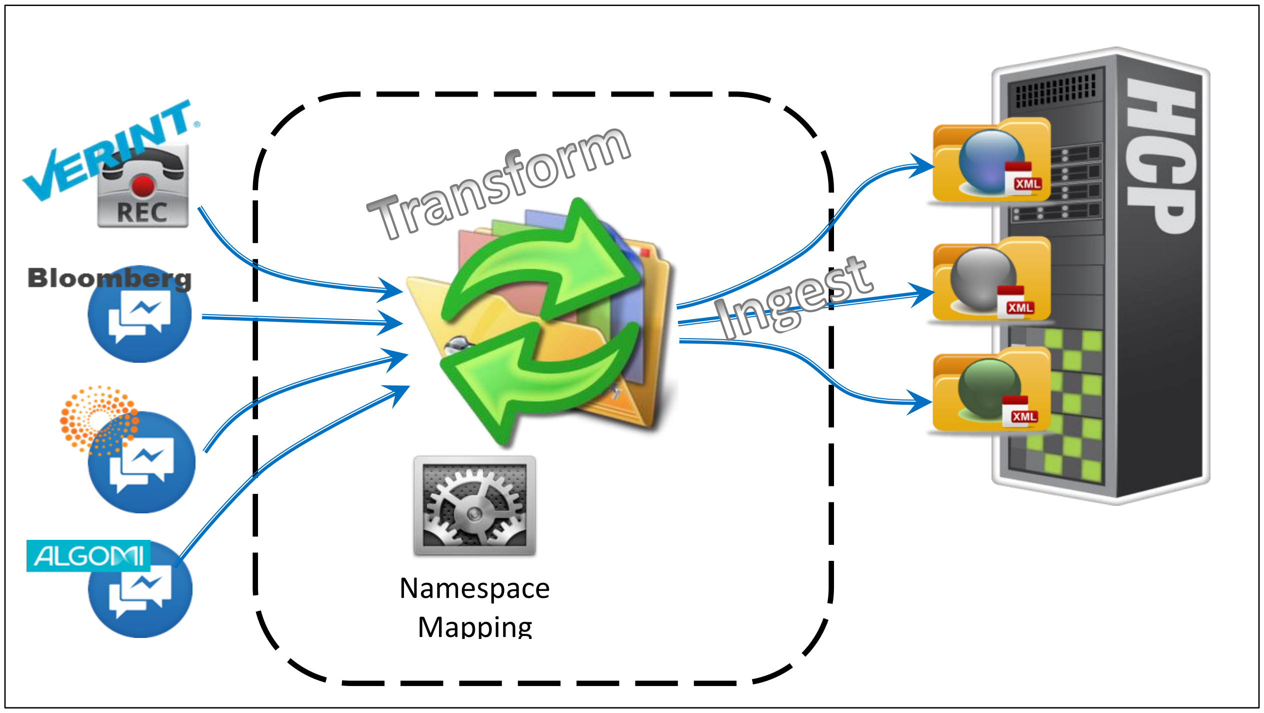


Figure 2: Ingest/Transform Overview

2.1 Data Set Processing

Each data set goes through multiple phases to reach the point of content being ingested into HCP in an index-able form. The general phases of processing are the following:

Phase Name	Description
Raw Data Set Capture	Evaluate content in drop zone for complete data set collections. When found, move data set collection into work area. There may be very basic data layout change for the raw data to adjust for Rabobank file transfer mechanism limitations.
Data Set Validation	Data Set is checked to ensure it is complete and intact. The level of checks is different for each data set.
Data Set Transformation	Data Set is transformed as needed to produce ingest-able and index-able content form.
Data Set Ingestion	Raw and/or Transformed content is ingested into HCP.
Cleanup and Failure Reporting	Work area is evaluated for failure conditions and cleaned up for next processing cycle.

Table 1: Data Set Processing Phases

3 Run-Time Environment

The solution run-time environment consists of a core folder structure to organize the run-time files consisting of libraries, configuration files, and run-time areas for log files, lock files, and minor scratch files. The following diagram shows the folder layout.

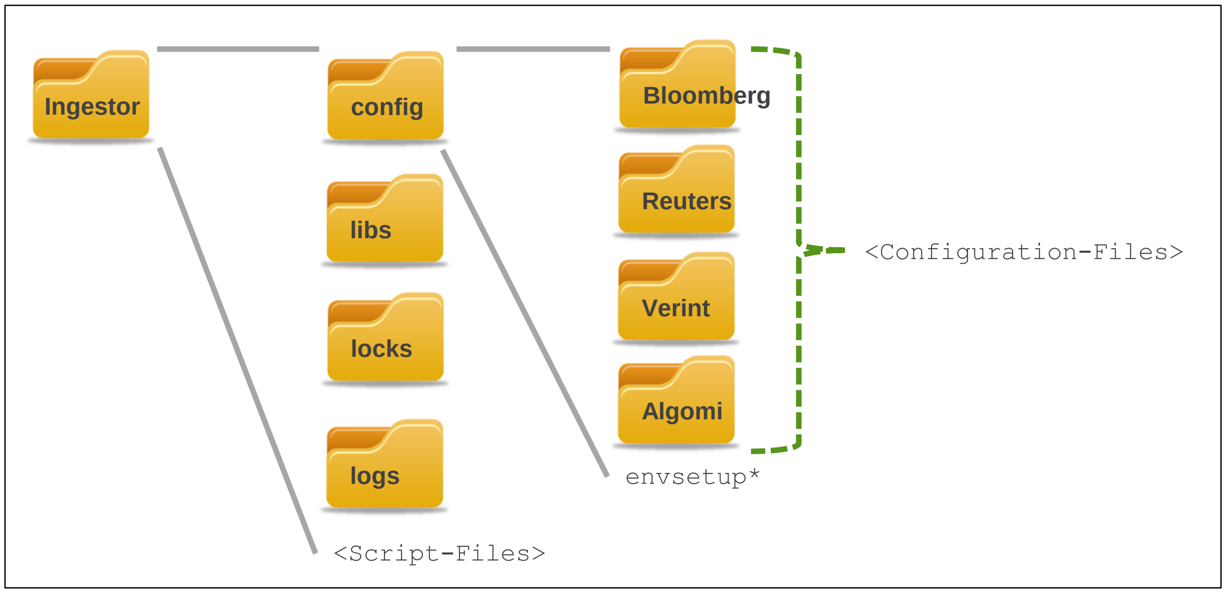


Figure 3: Deployment File/Folder Layout

In the top level folder resides the main scripts for running the data ingestion. While there are a good handful of scripts in the folder, the main scripts start with "`Process`". There is one process script for each data type implemented.

While these scripts can be run directly, the processing output goes to the console and will be difficult to look back at the execution to resolve any issues. The remedy this, there is a `cmdLogger` script. More on these scripts later in this document.

The `config` folder contains all run-time configuration required to run. Although there are quite a few files in this folder and its subfolders, there are relatively few files that may require changing for a given environment. These configuration files will be described in a later section.

The last folder that likely is of any interest to the solution administrator is the `logs` folder. This folder contains a subfolder for each data set supported. Within these subfolders are both the output when the `cmdLogger` script is used, as well as the logging from the COMET program used for ingesting content into HCP. More on this topic in a later section.

3.1.1 Data Area Layout

When content is received, it is captured in its raw form and transformed into a structure that is more suited for search purposes. As part of the collection and transformation, a three part folder structure is constructed.

The first, named `Ingest`, is used by the external systems for delivering the vendor specific content as defined by Rabobank processes. The second, named `WorkArea`, is a scratch area that is used to prepare content for ingestion. The third, named `Quarantine`, is an area where content that failed to be transformed and/or ingested in HCP is written for further evaluation.

In each of these area folders, the following diagram shows the folder layout for a production system.

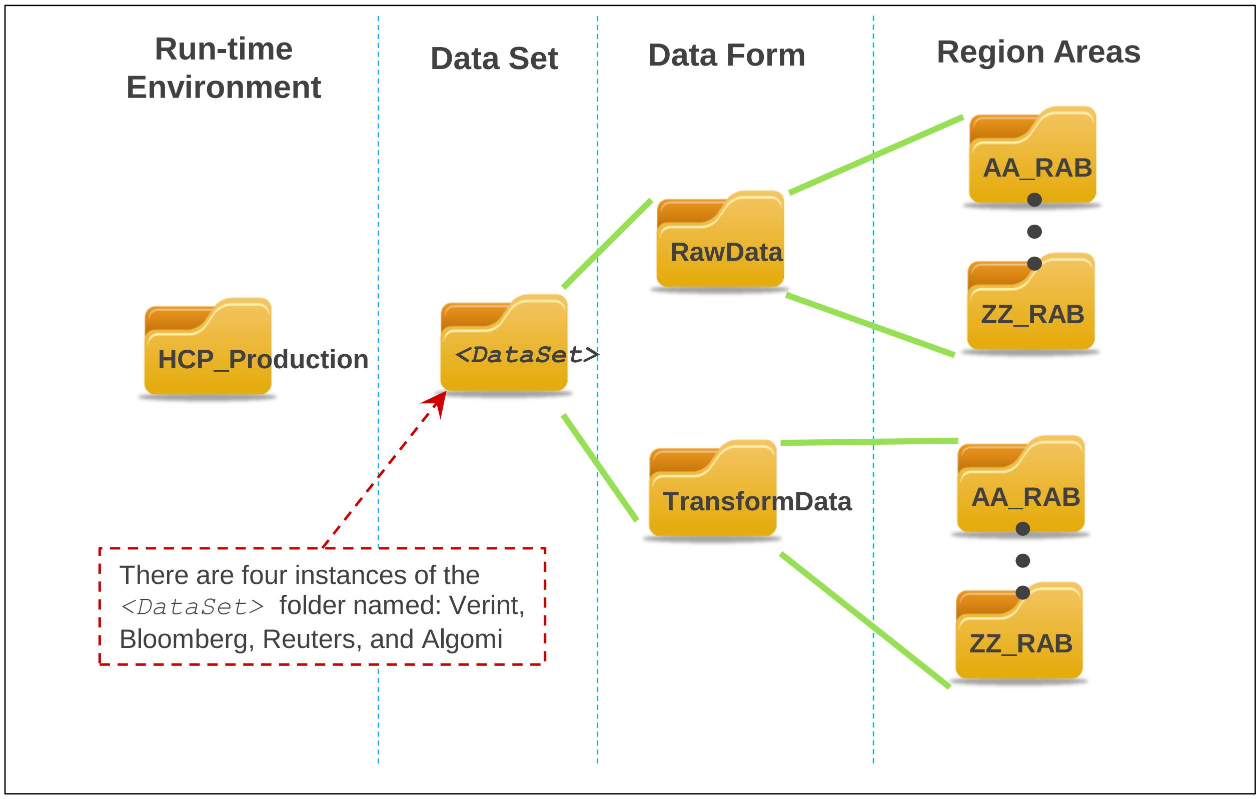


Figure 4: Data Area Folder Layout

The four levels in the folder layout consists of:

- The *Run-Time Environment* (shown as `HCP_Production` folder) provides separation of different solution deployments; for example production and test.
- The *Data Set* (shown as *`<DataSet>`* folder) consists of the 4 possible data sets to be processed; that is, `Verint`, `Bloomberg`, `Reuters` and `Algomi`.
- The *Data Form* level consists of both a `RawData` and a `TransformData` folder. The `RawData` content is a close copy of the collection as received by the solution. The

`TransformData` folder contains the fully transformed content that will be used for search purposes.

- The *Region Areas* folders are used to separate the region content from each other. Depending on the configuration, each region will be mapped to an HCP namespace.

4 Configuration

The configuration consists of a hierarchical structure of files and reside in the `config` folder. At the top level are the solution configuration files, in each subfolder are data set specific configuration.

There are a fair number of files that direct the software how to operate, however, only a small number of these files need maintaining for installation into a production environment. This section will describe the files and the typical settings that may require attention.

4.1 General Configuration

The top level configuration folders contains the solution level configuration files. The two files that are of interest for general site specific configuration are the `envsetup` and `envsetup_Core` files. Any other files in this top level folder generally do not require changes for site specific needs.

The `envsetup` file contains the site specific core configuration parameters in BASH script format. The following table lists the configuration settings and their meaning:

Setting Name	Description
BASE_DATA_AREA	Full folder path to the root of the data area to be used for processing the data sets. This folder must be on a file system with a lot of free space to receive, expand, and preprocess the content for ingestion into HCP. This specifies the data layout described in section *3.1.1 Data Area Layout*.
HCP_DNS_NAME	The fully qualified DNS name of the HCP system that the content is to be ingested.
HCP_ENC_PWD	The md5sum encoded password of the user configured in HCP_USER parameter. This is the default solution wide setting and can be changed per data set by editing the data set specific `envsetup_Core` file.
HCP_USER	The HCP data access user account that will be used to ingest content into the tenants and namespaces configured for this solution. This is the default solution wide setting and can be changed by editing the data set specific `envsetup_Core` file.
PROCESSING_AREA	Derived from `BASE_DATA_AREA`. Defines the folder path of the processing area used to expand and perform pre-processing content prior to HCP ingestion. This is also known as `WorkArea`.
QUARATINE_AREA	Derived from `BASE_DATA_AREA`. Defines the folder path of the root folder for placing content that failed processing. Under error free operation, this area will remain empty. This is also known as `Quarantine`.
SEARCH_CONSOLE_HOST	The fully qualified DNS name (or IP Address) of the solution Search Console that will be contacted to record ingestion processing statistics. If this value is not defined or blank, no statistics will be sent to the Search Console.

Setting Name	Description
SEARCH_CONSOLE_PWD	The password of the HDDS user that should be passed to the Search Console API for authentication for recording ingestion processing statistics. Note: This password is base64 encoded.
SEARCH_CONSOLE_USER	The user name of the HDDS user that should be passed to the Search Console API for authentication for recording ingestion processing statistics. This user must have permissions in HDDS to execute the "`GetUser`" command.
STAGING_AREA	Derived from `BASE_DATA_AREA`. Defines the root folder path of the data area that the raw content is being placed. This is content is expected to be placed by an external processing mechanism and just picked up by this solution. This is also known as `Ingest`.
STATUS_EMAIL_RECIPIENTS	Comma separated list of e-mail addresses of person(s) that should receive an e-mail upon processing completion. If this value is blank, no e-mail will be sent.

Table 2: ***envsetup*** Site Specific Settings

Also in the top level `config` folder, there is the `envsetup_Core` file. This file contains more internal processing configuration settings as well as general purpose routines used by the solution. It is not expected that any content in this file requires changing unless certain execution behavior changes is desired. The following table describes the configuration parameters in this file.

Setting Name	Description
MAX_FILE_AGE	Maximum number of days to let stray files stay around in the `STAGING_AREA` before considering them stale and thus be moved to the `QUARANTINE_AREA`.
MAX_HISTORY_IN_DAYS	Maximum number of days to keep track of the history of data set collections that were processed. This history is used to determine if a data set collection found in STAGING_AREA has already been processed and if so, should be sent to the `QUARANTINE_AREA`.

Table 3: ***envsetup_Core*** Site Specific Settings

4.2 Data Set Specific Configuration

The `config` folder contains subfolders where each folder contains data set specific configuration. In each of these folders there are a number of files that provide both site specific configuration as well as operational configuration. Many of these files are for internal processing configuration and are not an end-user changeable configuration.

Of the files in these folders, the following table lists the files which may be of interested to the administrator for site specific configuration. For many of these files, there is a `Raw` and a `Transform` version. The processing of the data sets consist of ingestion of the `Raw` and `Transform` content. The `Raw` content is mostly in the same form the data was received by the solution. The `Transform` content has gone through substantial transformation and is what is used by the overall solution for finding content. The processing of these two forms of the data is performed by separate COMET software executions, therefore, they require separate configuration to operate appropriately.

File	Description
`envsetup_Core`	Contains data set specific values and possible override values from the solution level `envsetup*` file(s). Two configuration values of interest in these file(s) are: • `HCP_TENANT_NAME` – HCP Tenant name for which the data set is to be written. • `HCP_TEST_NAMESPACE_NAME` – HCP Namespace name that is used to test for connectivity and to determine appropriate HCP version features that can be used during ingest. This file is a BASH script file format.
`RegionMapper_*.properties`	Maps data set region codes to what HCP namespace the data for that region should be written. There is one for Raw and Transform subset of data. This is a Java properties file format.
`AccountMapping.lst`	*[Bloomberg Only]* Provides a mapping from Bloomberg account to the region that it belongs. The layout of this file is described in following subsection.

4.2.1 RegionMapper_*.properties File Format

The `RegionMapper_*.properties` files are in the form of Java properties files. In general, Java properties files consist of a name/value pair.

The values in these files are intended to define for any given region code, the HCP information required to ingest the content into HCP. The information required for each region is: HCP DNS name, Tenant Name, Namespace Name, Data Access User, Data Access User Password, and time zone information for the region.

To avoid having to define this information for every region, this mapping file allows for specifying “default” values for regions, and then allowing values to be overridden for regions as necessary. The default values are taken from values configured by the `envsetup*` scripts both from the solution and data set levels, so generally, there is no reason to configure these for any specific data set. The following is an example of the “default” values in the file:

```
namespace.default=${HCP_TEST_NAMESPACE_NAME}
tenant.default=${HCP_TENANT_NAME}
hcpname.default=${HCP_DNS_NAME}
username.default=${HCP_USER}
encodedPassword.default=${HCP_ENC_PWD}
timeZone.default=UTC
```

The bulk of the configuration in these files consist of constructing the list of mappings where each mapping consists of a region id and the HCP namespace for that regions data. An example mapping is the following:

```
id.1=AR_RAB
namespace.1=BLMARRAB400
```

This example defines `AR_RAB` region code content is to be written into `BLMARRAB400` namespace in the tenant defined in the default configuration values at the top of the configuration file.

In any given mapping file, there could be multiple mappings. Each mapping set of values has different numbers as part of the value name. For example, the second mapping set could be:

```
id.2=AU_RAB
namespace.2=BLMAURAB400
```

This defines a second mapping where the `AU_RAB` region code content is to be written to the `BLMAURAB400` namespace.

NOTE: There must NOT be any gaps or mis-ordering in the numbering of the region identifiers and namespaces, otherwise region mappings will not be recognized and content may be placed into `Quarantine`.

4.2.2 AccountMapping.lst File Format [Bloomberg ONLY]

The `AccountMapping.lst` file is only used for the Bloomberg data set. This file defines what Bloomberg account numbers belong to which region code. The format of this file is simple in that it consists of the first element on the line being the region code, and all subsequent items on the line consist of the Bloomberg account numbers. For example:

```
SG_RAB 117481 117589 30242675 30243089
```

The usage of this file is that when a region code is needed for a Bloomberg account, the file is quickly scanned for that account number and the region is the first space separated item in the line for which the account number was found.

With this method, it is acceptable to have multiple lines for any region code to help format the file in a reasonable manner. For example:

```
# Netherlands
NL_RAB 30537 30009756 30197429 196970 30184577 30140309 554332
NL_RAB 196976 217426 30208420 179739 128915 3616 201308
NL_RAB 748220 30135751 102834 707527 321502
```

This file format also allows for comments. Any comment starts with the "#" character and thus any characters after the comment character will be ignored.

4.2.3 Configuration Notes

The section contains special configuration notes relative to the different data sets.

4.2.3.1 Reuters

The only files of interest for Reuters configuration are the `RegionMapper_*.properties` and the `envsetup_Core` properties. No other configurations files should need to be modified unless it is the intent to change the overall operation of the Reuters content.

The `RegionMapper_*.properties` file has two forms. One is for `Raw` data content mapping and the other is for `Transform` data content. For Reuters, these two files are identical since the `Raw` and the `Transform` content are going into the same namespace.

The `envsetup_Core` properties file is not likely to need to be changed unless the Reuters tenant that will be receiving the content has changed, or the namespace used to test connectivity either has changed name or has been removed.

4.2.3.2 Bloomberg

The only files of interest for Bloomberg configuration are the `RegionMapper_*.properties` and the `envsetup_Core` properties. No other configurations files should need to be modified unless it is the intent to change the overall operation of the Bloomberg content.

The `RegionMapper_*.properties` file has two forms. One is for `Raw` data content mapping and the other is for `Transform` data content. For Bloomberg, these two files are quite different. For Bloomberg all `Raw` content is being written to a separate HCP namespace than the `Transform` content. In the event that a region is being added or removed for Bloomberg, only the `Transform` flavor of the configuration file will need to be changed.

The `envsetup_Core` properties file is not likely to need to be changed unless the Bloomberg tenant that will be receiving the content has changed, or the namespace used to test connectivity either has changed name or has been removed.

Also note that Bloomberg has an additional configuration file named `AccountMapping.lst`. This file is described in section *4.2.2.*

4.2.3.3 Verint

The only files of interest for Verint configuration are the `RegionMapper_*.properties` and the `envsetup_Core` properties. No other configurations files should need to be modified unless it is the intent to change the overall operation of the Verint content.

The `RegionMapper_*.properties` file has two forms. One is for `Raw` data content mapping and the other is for `Transform` data content. For Verint, these two files are identical since the `Raw` and the `Transform` content are going into the same namespace.

The `envsetup_Core` properties file is not likely to need to be changed unless the Verint tenant that will be receiving the content has changed, or the namespace used to test connectivity either has changed name or has been removed.

4.2.3.4 Algomi

The only files of interest for Algomi configuration are the `RegionMapper_*.properties` and the `envsetup_Core` properties. No other configurations files should need to be modified unless it is the intent to change the overall operation of the Algomi content.

The `RegionMapper_*.properties` file has two forms. One is for `Raw` data content mapping and the other is for `Transform` data content. For Algomi, these two files are quite different. For Algomi all `Raw` content is being written to a separate HCP namespace than the `Transform` content. In the event that a region is being added or removed for Algomi, only the `Transform` flavor of the configuration file will need to be changed.

The `envsetup_Core` properties file is not likely to need to be changed unless the Algomi tenant that will be receiving the content has changed, or the namespace used to test connectivity either has changed name or has been removed.

5 Execution Scripts

In the top level installation folder, there are a number of Linux BASH scripts. Many of them are supporting scripts and typically will not be run directly except for perhaps advanced debugging purposes.

There are two sets of scripts that are used as the main execution point for the data ingestion and are described in the following sections.

5.1 Data Set Processing Scripts

Of the scripts in the top level installation folder, there are three processing scripts that are the main execution point for the data ingestion, one for each data set:

`ProcessBloomberg`, `ProcessReuters`, `ProcessVerint` and `ProcessAlgomi`

These scripts do not take any input parameters and output all logging to the console that executed the script. If desired to capture the output, it is necessary to utilize standard Linux BASH operations like redirection and pipe. For example, to redirect the script input and output both to a file and the console output, the following command would be reasonable:

```
./ProcessBloomberg 2>&1 | tee TestingBloomberg.log
```

5.2 Output Logging Wrapper

To help with capturing logs of each run into a standard mechanism, a wrapper script named `cmdLogger` is available in the top level installation folder. This BASH script fundamentally runs the command provided in the parameters and captures the output into a log file with date/time stamp information under the appropriate data set log folder. The usage for this command is the following:

```
cmdLogger -d=<data-set> <Command>
```

The `<data-set>` qualifier value is used to decide what subfolder in the `logs` folder the log file should be placed in. The `<Command>` parameter is the command to run and redirect output to a log file.

The following is an example usage:

```
./cmdLogger -d=Bloomberg ./ProcessBloomberg
```

With this command, the following log file would be generated:

```
logs/Bloomberg/ProcessBloomberg_2015-08-18T18:07:12.log
```

Running of this script will block until the `<Command>` completes. In addition, the output of the `<Command>` is not displayed on the console.

This command can be integrated with scheduling software like cron or manually executed and use Linux BASH functionality to run the command in the background.

6 Monitoring/Debugging

6.1 Completion Status Monitoring

Daily Monitoring of ingestion completion status consists of an overall completion status indicating the most severe status during the run (SUCCESS, WARNING, FAILURE). In addition, it may have a message associated with the completion status. Also included are statistics about how many items were encountered/processed and a count of any warning/failures that may have occurred.

The following table describes the warning/failure counts that may be reported in the completion status:

Failure/Warning Name	Description
`Collect File Failure`	This counter indicates how many collections presented to the solution could not be collected for further processing. This is usually caused by inability to move the collection for further processing and likely a file system issue.
`Collect File Warning`	This counter indicates how many warnings were encountered with collections presented to the solution. Typically, the cause may be either unexpected files or collections in a transient state and thus not ready to be collected for further processing.
`Validate Failure`	After a complete collection is identified, a number of checks are performed to ensure correctness of the content of the collection. This counter identifies how many collections failed validation.
`Duplicate Failure`	After collecting and validating a collection, a lookup is performed to ensure that this collection has not been seen prior. If it has, this statistic will be updated and the collection will be moved to `Quarantine` area.
`Residual File Failure`	This counter indicates that after an attempt to ingest all the content into HCP, there were a number of files that were left behind. This could be due to various reasons like HCP unavailability, file(s) are not part of the configured files to be ingested, etc.

Table 4: Failure/Warning Statistic Types

This solution has the ability to be configured to report the completion status to multiple destinations. The three destinations are the processing script output, E-mail recipients, and Search Console Alerts page. The following subsections show example completion status output for these destinations.

6.1.1 Script Output Example

The output of the processing script contains a lot of content, but at the very end of the output contains an overview of the processing success/failure. Depending on how the processing script was executed, it may show on the command line output or redirected to a log file for future consideration.

The following is an example completion status that completed with warning(s):

```
INFO: TIMESTAMP - 2015-08-24T19:20:02-0400 Message: (none)
Exit Status: WARNING
Message:     Detected Warning(s) during file collection from:
/data/Ingest/HCP_Production/Reuters

Statistics
  Object Types:          [ "Reuters Messenger" ]
  Processed Items:       [ 11 ]

  Collect File Warning:  1
INFO: [Reuters] Finished Processing (2 - Detected Warning(s) during
file collection from: /data/Ingest/HCP_Production/Reuters)
INFO: TIMESTAMP - 2015-08-24T19:20:02-0400 Message: (none)
```

6.1.2 E-Mail Status Example

An optionally configured status may be sent via e-mail to a list of recipients. To configure the recipients, set the `STATUS_EMAIL_RECIPIENTS` configuration to a comma separated list of e-mail addresses to receive the status. If the configuration parameter is empty or undefined, no e-mails will be sent.

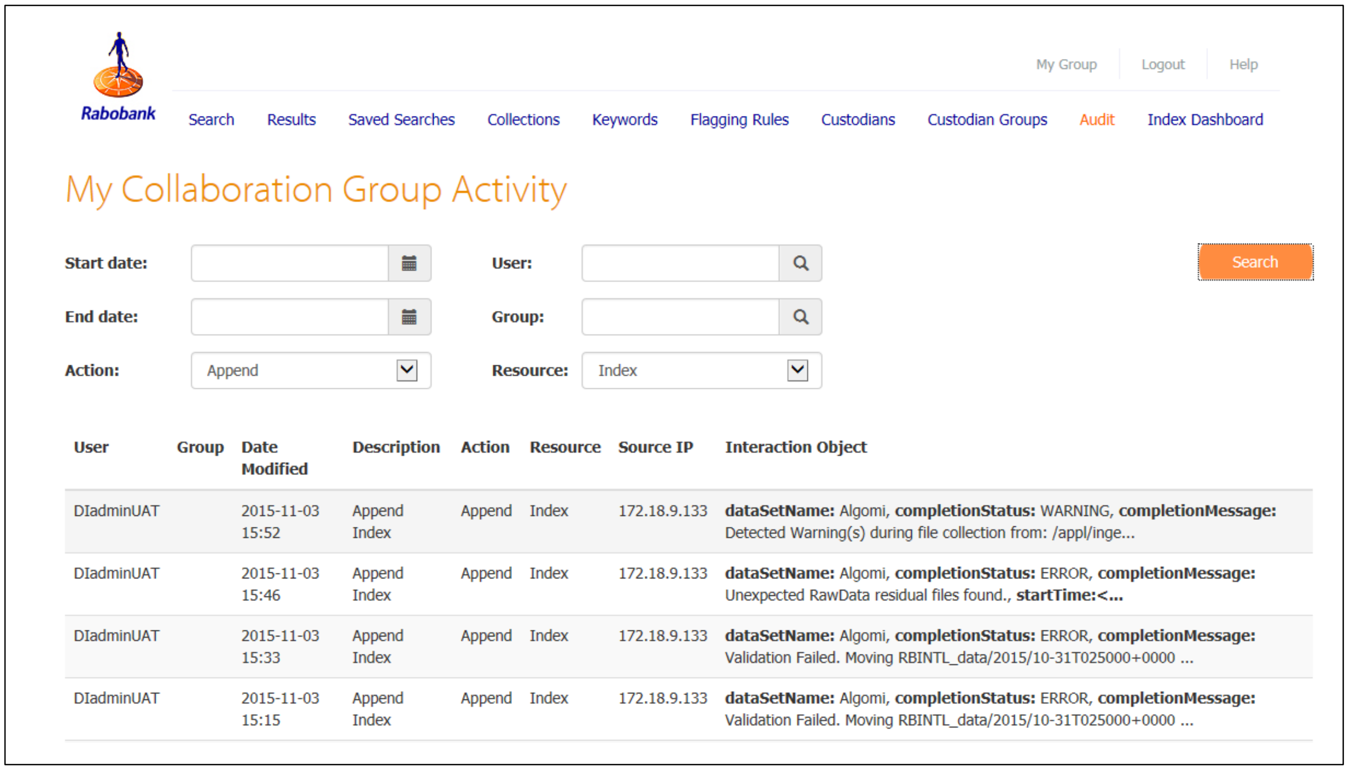
The following is an example e-mail status sent at processing script completion:

```
From comet@localhost.localdomain  Mon Aug 24 19:20:03 2015
Return-Path: <comet@localhost.localdomain>
From: Comet Service <comet@localhost.localdomain>
Date: Mon, 24 Aug 2015 19:20:02 -0400
To: jdoe@hds.com
Subject: [WARNING] ProcessReuters Data Ingestion Completion Status
User-Agent: Heirloom mailx 12.5 7/5/10
Content-Type: text/plain; charset=us-ascii
Status: R

Message: Detected Warning(s) during file collection from:
/data/Ingest/HCP_Production/Reuters

Statistics
  Object Types:           [ "Reuters Messenger" ]
  Processed Items:        [ 11 ]

  Ingest Failure:         0
  Validation Failure:     0
  Duplicate Failure:      0
  Residual File Failure:  0
  Collect File Warning:   1
  Collect File Failure:   0
```

6.1.3 Search Console Example

The data ingestion solution consists of the ability to send the status information for a run via an HTTP request to the Search Console server. This information is displayed in an Alerts page in the GUI. To send a status, the `envsetup` configuration file has three settings starting with the name `SEARCH_CONSOLE_`. See *Table 2:* `envsetup` *Site Specific Settings* for details of these configuration values. If the `SEARCH_CONSOLE_HOST` value is not set or blank, no status will be sent to the Search Console.

The following are some screen snapshots of what the status looks like in the GUI:

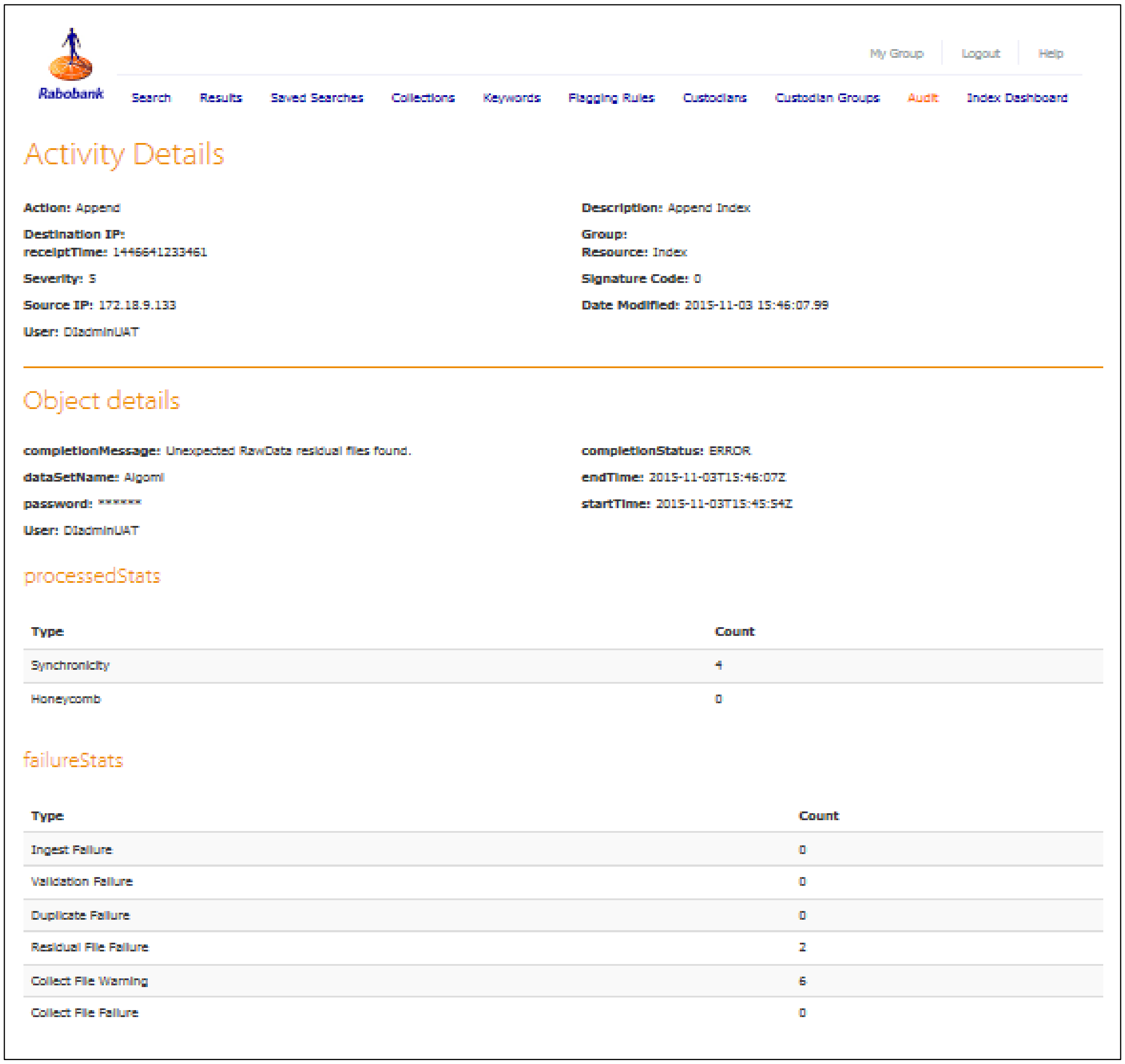


My Collaboration Group Activity

Start date:		User:	
End date:		Group:	
Action:	Append	Resource:	Index

User	Group	Date Modified	Description	Action	Resource	Source IP	Interaction Object
DIadminUAT		2015-11-03 15:52	Append Index	Append	Index	172.18.9.133	**dataSetName:** Algomi, **completionStatus:** WARNING, **completionMessage:** Detected Warning(s) during file collection from: /appl/inge...
DIadminUAT		2015-11-03 15:46	Append Index	Append	Index	172.18.9.133	**dataSetName:** Algomi, **completionStatus:** ERROR, **completionMessage:** Unexpected RawData residual files found., **startTime:<...**
DIadminUAT		2015-11-03 15:33	Append Index	Append	Index	172.18.9.133	**dataSetName:** Algomi, **completionStatus:** ERROR, **completionMessage:** Validation Failed. Moving RBINTL_data/2015/10-31T025000+0000 ...
DIadminUAT		2015-11-03 15:15	Append Index	Append	Index	172.18.9.133	**dataSetName:** Algomi, **completionStatus:** ERROR, **completionMessage:** Validation Failed. Moving RBINTL_data/2015/10-31T025000+0000 ...

Figure 5: Search Console Status Summary



Activity Details

Action: Append

Description: Append Index

Destination IP:
receiptTime: 1446641233461

Group:
Resource: Index

Severity: 5

Signature Code: 0

Source IP: 172.18.9.133

Date Modified: 2015-11-03 15:46:07.99

User: DIadminUAT

Object details

completionMessage: Unexpected RawData residual files found.

completionStatus: ERROR

dataSetName: Algomi

endTime: 2015-11-03T15:46:07Z

password: ******

startTime: 2015-11-03T15:45:54Z

User: DIadminUAT

processedStats

Type	Count
Synchronicity	4
Honeycomb	0

failureStats

Type	Count
Ingest Failure	0
Validation Failure	0
Duplicate Failure	0
Residual File Failure	2
Collect File Warning	6
Collect File Failure	0

Figure 6: Search Console Detail Completion Status

6.2 Logging

This data ingestion solution has a robust logging capability that can be configured at multiple levels. The default configuration is to provide basic information messages to ensure it is evident that the processing is making progress. All log files are written to the `logs` folder of the solution installation folder.

The logging that is performed by the solution has multiple aspects since both BASH scripts and Java software is being used to construct this solution. It is not feasible to have one log containing all the logging, so it may seem a bit complex.

At the highest level, the processing scripts are written in BASH and output logging to standard output. This output can also be redirected by the `cmdLogger` script described in section *5.2 Output Logging Wrapper*. The logging by this wrapper will consist of all script output as well as the console logging performed by the COMET Java software executed. The usage of this wrapper script for Bloomberg will create a file with the path name like:

```
logs/Bloomberg/ProcessBloomberg_2015-08-18T18:07:12.log
```

During execution of the processing scripts, there are two execution instances of COMET Java software; one for ingesting the `RawData` content representation and another for ingesting the `TransformData` content representation. See section ***Error! Reference source not found. Error! ference source not found.*** for details on the two types of content . Each COMET Java run against a representation will create its own log file as configured for that data set. For the Bloomberg data set, the two log files created for the most recent run will be named:

```
logs/Bloomberg/COMET_Raw.log
logs/Bloomberg/COMET_Transform.log
```

Depending on the configuration, the level of logged information in these files may be more detailed than what is provided as console output and captured by the `cmdLogger` logs.

As part of the COMET Java logging configuration, the COMET Java log files will be rolled over, compressed, and stored in a date subfolder. An example set of files created during log file rollover is:

```
logs/Bloomberg/2015-08/COMET_Raw-08-24-2015-1.log.gz
logs/Bloomberg/2015-08/COMET_Transform-08-24-2015-1.log.gz
```

The following diagram shows the representative layout of the log files:

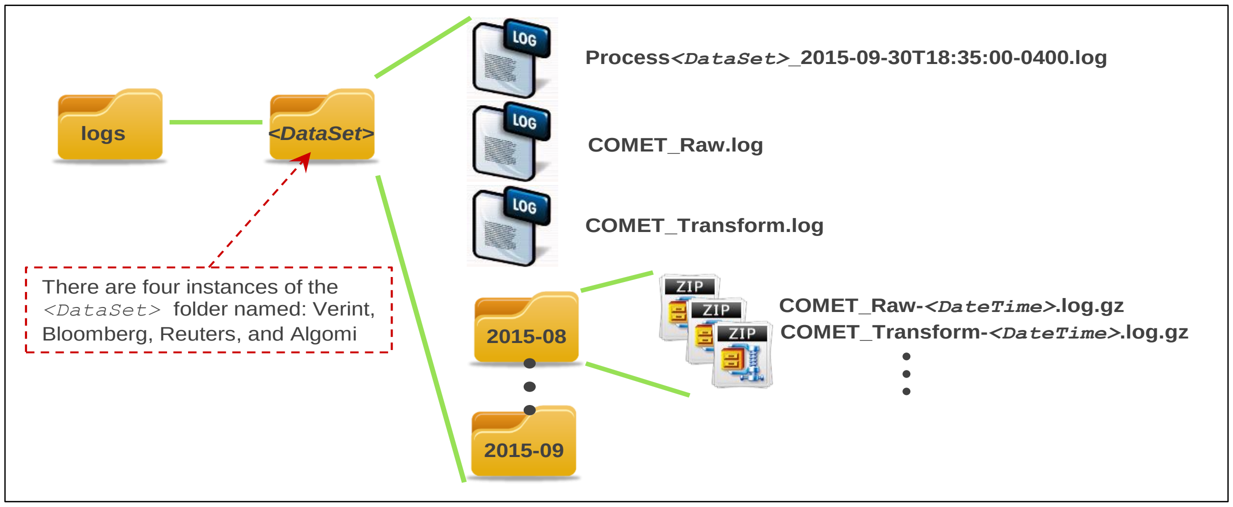

logs
<DataSet>
There are four instances of the <DataSet> folder named: Verint, Bloomberg, Reuters, and Algomi
Process<DataSet>_2015-09-30T18:35:00-0400.log
COMET_Raw.log
COMET_Transform.log
2015-08
2015-09
COMET_Raw-<DateTime>.log.gz
COMET_Transform-<DateTime>.log.gz

Figure 7: Log file layout

6.2.1 Logging Configuration

There are two fundamental logging configurations that can be utilized: BASH debugging and COMET Java logging. There is also an additional output of HTTP response information that can be obtained that is useful for debugging failed HTTP communication with HCP.

The following subsections will describe each.

6.2.1.1 BASH Script Debug Logging

Throughout the BASH scripts for this solution, there is additional information that can be displayed to help with understanding the processing flow for debugging possible problems. Enabling debugging simply comes down to defining the BASH environment variable `DEBUG` at the start of the execution of the script. This can be accomplished in many different ways from all processing actions to just turn on debugging for a specific data set. The following table outlines how to enable debugging at various levels.

Logging Level	Procedure
Interactive Shell	When logged into a Linux shell, one can turn on debugging for all executions of the processing scripts by first issuing the following command in the shell: `export DEBUG=1` This will keep it enabled in the shell session until either the shell is exited, or the value is removed with the following command: `unset DEBUG`

Logging Level	Procedure
All process script executions	To turn on debugging for all solution scripts regardless of the Linux shell instance, turn on debugging by adding the following configuration line to the `config/envsetup` setup script: `export DEBUG=1` To remove the setting, simply remove the line from the configuration file. All future executions will no longer display the debug output.

Individual data set execution	If it is desired to only turn on debugging output for a specific data set, add the following line to the data set specific configuration file `config/<DataSet>/envsetup_Core`:
	`export DEBUG=1`
	For example, to add debugging to the Reuters data set, modify the configuration file named:
	`config/Reuters/envsetup_Core`
	To remove the setting, simply remove the line from the configuration file. All future executions will no longer display the debug output.

Table 5: BASH Script Log Settings

6.2.1.2 COMET Java logging

The core of ingestion of content into HCP uses the COMET Java software. With Java software, there are multiple comprehensive logging libraries that can be used. The COMET implementation uses log4j v2 libraries for logging. Detailed information about this logging library can be found at:

```
http://logging.apache.org/log4j/2.x/manual
```

This logging library allows for logging at various levels to provide more information for debugging. The logging levels supported by this library are:

```
TRACE, DEBUG, INFO, WARN, ERROR, FATAL
```

Each data set has its own separate configuration for the logging that is performed via COMET. The configuration file can be found at the general path:

```
config/<DataSet>/log4j2.xml
```

The default configuration for COMET logging sends logging to two different destinations: COMET console output and rotated log files residing in `logs/<DataSet>` folder, for example `logs/Reuters`. In the `log4j2.xml` file, each destination can be configured to output different logging levels. The console logging is defaulted to `INFO` level, and the rotated log files is defaulted to `DEBUG` level logging.

The following `log4j2.xml` file snippet shows the XML tags that configure these levels:

```
<Loggers>
    <Root level="TRACE">
        <AppenderRef ref="Console" level="INFO"></AppenderRef>
        <AppenderRef ref="LogFiles" level="DEBUG"></AppenderRef>
    </Root>
</Loggers>
```

To change the logging level for a destination, change the text for the `level` value to one of the desired level listed previously. For more detailed information about how to configure log4j v2 logging, see the following web page:

```
http://logging.apache.org/log4j/2.x/manual/configuration.html
```

6.2.1.3 HTTP Response Headers

When COMET communicates with HCP via the REST API, each HTTP response contains status and header information from HCP. This information can be valuable when trying to identify failures with respect to interactions with HCP. Some failures could be permissions issues, misconfigured HCP settings, etc.

COMET has the ability to dump the HTTP Response headers to the COMET console output. This output will accompany the solutions processing script output which depending on how it is executing may end up in a BASH Script log file.

To turn on this HTTP Response headers, there is a Java properties in the core COMET properties file that can be change. The properties file is named `comet.properties` and resides in each data set folder `config/`*`<DataSet>`*. In this file, there is a name/value property definition called `execution.debugging.httpheaders`. The default configuration is that this value is commented out in the property file, and thus defaults not outputting the HTTP headers. If you desire to turn on this logging, edit the file and ensure the configuration line looks like the following:

```
# Indicate whether HTTP headers should be written to the console (stdout)
# [Default: false]
execution.debugging.httpheaders=true
```

6.2.2 Log File Search Failure Tips

As mentioned in the prior sections there are 2 core types of logs: BASH Script logs and COMET Java logs. The BASH Scripts logging will contain the logging output by the BASH scripts but will also have the console level logging from the COMET Java processes that are executed by the scripts.

To help isolate the reason for a failure reported via the monitoring described in section *6 Monitoring/Debugging*, there are certain string patterns that can be found in the various log files to help isolate the cause for the failure. As previously mentioned, the monitoring status only contains the most recent and highest level of failure along with some helpful text as to the cause. There could be other same and lower level error messages in the log files. Utilizing the search for the text strings below will help find all the failures/warnings and help isolate the overall problem. The following table describes the strings to search for:

Source	Text String	Description
BASH Scripts	ERROR	Any failures that occur is output with the string “ERROR” followed by some helpful text. Near that ERROR report there may be more Linux command failures to help further understand the failure.
BASH Scripts	WARN	An operational warning for which the scripts has recovered from. Any warnings should be investigated since while it does indicate that content may have been processed successfully, the scripts decided that an anomaly encountered might require corrective action.
COMET Java	WARN	Indicates unexpected but recoverable conditions that may require investigation to see if there is any corrective action needed. These will also show up in the BASH Script logging as they are output to the COMET console output. However, the COMET Java logs will contain quite a bit more other debugging information that would be helpful to isolate the cause of the issue.
COMET Java	ERROR	Indicates an unexpected failure occurred and corrective action could not be taken. All errors should be reviewed, and take any possible corrective actions. These will also show up in the BASH Script logging as they are output to the COMET console output. However, the COMET Java logs will contain quite a bit more other debugging information that would be helpful to isolate the cause of the issue.
COMET Java	FATAL	Fatal conditions and errors mentioned above are intermixed and essentially mean the same.
COMET Java	Exception	When Java programs encounter a run-time exception it will show the name of the exception and the call stack. All exception names end with the string Exception. While most exceptions are caught and handled, it should still be considered error and understood as to the cause.

Table 6: Log File Search Strings

7 Failure and Recovery Procedure

The data ingestion process involves a handful of processing phases to transition from receipt of the raw content to the point of having content ingested into HCP. The main phases are described in *Table 1: Data Set Processing Phases*.

During any of these phases there can be different types of failures that occur. These failures will be identified in the status report as one of the following:

Failure Class	Description
Collection Warning	Encountered a raw data collection that is missing files. In general, the files will be left alone, but if the files are older than a configured threshold `MAX_FILE_AGE`, the content will be moved to `Quarantine`.
Collection Failure	A file manipulation failure occurred. Likely cause is disk space limitations. Files will be left in place and likely processed once the issue is resolved.
Validation Failure	After extensive evaluation of a data set collection, the collection seems to be corrupted. The raw data set will be moved to `Quarantine`.
Duplicate Failure	The checksum for the data set collection matches one seen in the last `MAX_HISTORY_IN_DAYS` days. The raw data set will be moved to `Quarantine`.
Ingestion Failure	After performing transformation of content and attempted ingestion, there were some items that were not ingested. An item consists of a call recording, message, or chat session and does not include any supporting temporary files.
Residual Failure	After performing ingestion cycle of raw and transform data, there were a number of files remaining in the `WorkArea`. These residual files could consist of both temporary files required for ingestion and items that failed ingestion.

Upon failure, the end result will be that any non-processed content will be set aside into the `Quarantine` area. The `Quarantine` area is rooted at the path defined by `QUARATINE_AREA` configuration parameter. An example path is:

```
/appl/ingest/Quarantine/HCP_Production
```

Under this folder will be two subfolders named `RawData` and `TransformData`. The organization under this folder will be identical to the `WorkArea` structures described in *section 3.1.1 Data Area Layout*.

To evaluate the reason for content being placed in the `Quarantine` area, the data set specific log file for the failed run can be analyzed to root out the details of the failure. The following is the log file name template for a specific *<DataSet>*:

```
logs/<DataSet>/Process<DataSet>_*.log
```

where `<DataSet>` is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

Appropriate manual actions can be taken to resolve the issue (be it environmental or unexpected data set form), re-inserting the current state of the data set collection into the processing work flow for retry, then re-initiate the data ingestion processing.

The following subsections will discuss the possible failures, where to look for failure details, typical actions to resolve the issues, and how to appropriately re-introduce the failed content into the multi-phased processing.

7.1 Collection Processing

Collection Processing consists of identifying content dropped into a staging area and moving the collections into the solution `WorkArea` folder. The staging area for production environment is rooted at path defined by the `STAGING_AREA` configuration parameter. This typically will point to a path like:

```
        /appl/ingest/HCP_Production/<DataSet>
```

where `<DataSet>` is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

Once a full data set collection is identified, it will be moved to the data ingestion `RawData` work area described in section *3.1.1 Data Area Layout*. The move operation will consist of copy of all relevant files to the data ingestion `WorkArea` and upon success will remove files from raw input file folder.

During this processing, there could be warnings and failures that are identified.

7.1.1 Collection Warning

A collection warning will be generated when there are files in the staging area, but there seem to be some missing top level files. It is considered a warning since content may be still be placed into the staging area and thus likely processed upon the next processing cycle.

However, if the content remains incomplete for `MAX_FILE_AGE` minutes, it is considered that the file transfer has failed and the data will never become complete. At this point, the content will be moved to the `Quarantine` area path defined by QUARANTINE_AREA configuration parameter. This typically resolves to a path like:

```
        /appl/ingest/Quarantine/HCP_Production/<DataSet>
```

where `<DataSet>` is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

Any subfolder structure, if any, will remain the same as found in the staging area for the specific data set.

Any attempts at resolving issues with the data set collection will require regenerating a correct collection from the data set source and remove the content in `Quarantine` area.

7.1.2 Collection Failure

Any collection failure is likely due to a file system condition like out of space or read only. Once this issue is resolved, the collections typically will be processed on the next data set processing execution.

There should be no actions required once this condition is resolved as the data set collections should still be in tact in the staging area.

7.2 Validation Failure

A validation failure can occur if the data set collection is considered invalid after doing extensive evaluation on the collection and its contents. Each data set type has different validation that it will perform. To find out what validation failed look in the appropriate `logs/<DataSet>/Process<DataSet>*.log` file for other errors, where `<DataSet>` is replaced by Reuters, Bloomberg, Verint or Algomi.

There is no procedure for fixing invalidated collections except to contact the people responsible for generating the data set collections and provide them with the invalid collection and the reason. They must regenerate the content and have it delivered to the staging area in the form required for the specific data set.

Once resolved, the data set collection can just be removed from the `Quarantine` area.

7.3 Duplication Failure

A duplication failure occurs when a checksum is calculated for a collection and another collection in the last MAX_HISTORY_IN_DAYS days has the same checksum value. Any data set collections that encounters the duplication failure will be placed in the `Quarantine` area under the `RawData` folder structure found at:

```
/appl/ingest/Quarantine/HCP_Production/<DataSet>/RawData
```

where *`<DataSet>`* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

The history of the collections is stored in the file with the path template of :

```
/appl/ingest/WorkArea/HCP_Production/<DataSet>/history.log
```

where *`<DataSet>`* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

The history file consists of lines that have the ingestion date/time, checksum, and the `RawData` area folder path of the collection. An example line for Bloomberg `history.log` file content is:

```
2015 08 24 19 18 38,f82d432a70e9d4e5199cc2be7ef859693a78a35c,RBINTL_data/3616/2015-02-25T190000+0000
```

Examine the data set specific `Process*.log` file to determine what data set collection failed. Search for an error entries like:

```
ERROR: Collection checksum found: 7c33e832cb0eb2610c8d3bf603bde1e986c6ea7d
ERROR: Found collection already processed: NL_data/43282/2015-03-26T1934+0000
```

Look in the `history.log` file for the same checksum and perhaps even the same collection to determine if the failure is valid. If the checksum and the collection names are identical, investigate why the collection would be delivered twice to the staging area. If the checksum is the same, but the collection name is different, the method used to compute the checksum may be insufficient and may require software changes.

If the collection is being processed because of an attempted re-ingestion, remove the line from the `history.log` file and try ingestion again.

7.4 Ingestion Failure

An ingestion failure occurs when a piece of content (voice recording, message, or conversation/chat) was not ingested into HCP. The ingestion failure count is the number of these items detected. The cause for this fundamentally comes down to inability to write content to HCP for which causes could be due to HCP connectivity, HCP space limits, improper HCP configuration, or improper solution configuration. The first location to find out details of the failure is to examine the data set specific processing log files:

```
logs/<DataSet>/Process<DataSet>_*.log
```

where *`<DataSet>`* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

Once the issue is resolved. The content placed in `Quarantine` can be re-introduced back into the `WorkArea`. First, the data set collection checksum record, if existing, needs to be removed. The checksum record resides in the file path specified by:

```
/appl/ingest/WorkArea/HCP_Production/<DataSet>/history.log
```

where *`<DataSet>`* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

Using your favorite editor, edit the `history.log` for the specific *`<DataSet>`*. If the failure was recent, it is most likely the last entry at the bottom of the file. To identify the correct entry, look at the last comma separated field in the file. This field is the path of the data set collection and should match the folder structure/names of the data set collection to be re-ingested. Remove the relevant line in the file.

Next, the following commands will move the content from the `Quarantine` area to the `WorkArea`:

```
cd /appl/ingest/Quarantine/HCP_Production/<DataSet>
mv RawData /appl/ingest/WorkArea/HCP_Production/<DataSet>/
mv TransformData /appl/ingest/WorkArea/HCP_Production/<DataSet>/
```

where *`<DataSet>`* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

Note: Depending on the failure, there may not be a `RawData` or `TransformData` folder to move as all this type of content may have been ingested successfully. So as long as one of the above `mv` commands does something useful, the next data set processing will attempt the ingestion again.

NOTE: If the reason for the failure is not clear, there are separate log files are created from the COMET ingestion processes. There is one file for the processing the `RawData` content and one for the `TransformData` content. The paths to those log files are:

```
logs/<DataSet>/COMET_Raw.log
logs/<DataSet>/COMET_Transform.log
```

where *`<DataSet>`* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

7.5 Residual File Failure

Residual File Failure is a failure counter for which is the number of files left behind in the `WorkArea` after ingestion processing has completed. Residual File Failure count is different from the Ingestion Failure in that its count includes ALL files found in the `WorkArea` after ingestion processing. These files could be a result of one or more of the following:

- Expanded Ingestion Failure counts
- Unknown Region
- Unexpected files were found in the data set collection

The cause of the failure could be for many reasons and the logs/<DataSet>/Process<DataSet>*.log files should be examined for any failures or warnings. The most common reason is due to HCP ingestion failure, however, the other common reason is Unknown Region ingestion exception.

7.5.1 Expanded Ingestion Failure

When an Ingestion Failure occurs as described in section *7.4 Ingestion Failure*, the count for this failure only includes data set items like call recordings, messages, and chat/conversation sessions. The Residual File Failure counts includes *all* files including those files which may be temporary files used to generate HCP metadata attached to the HCP objects. Under normal situations, if any given data set collection ingested all the data set items, the temporary files are removed.

For example, when processing Reuters Messages, there are a number of files named `UserInfo*` and `*_Metadata` created for the purpose of generating HCP custom metadata for the messages objects. These files are temporary files and not ingested.

In the event that a data set item is remaining, the temporary files will not be removed and thus reported in this count of failures.

Investigate and correct the cause of the failure via the log files. If the cause due to HCP availability issue or mis-configured environment like HCP permissions issue, the content can be reintroduced into the processing stream.

Once the issue is resolved. The content placed in Quarantine can be re-introduced back into the `WorkArea`. First, the data set collection checksum record, if existing, needs to be removed. The checksum record resides in the file path specified by:

```
/appl/ingest/WorkArea/HCP_Production/<DataSet>/history.log
```

Using your favorite editor, edit the `history.log` for the specific *`<DataSet>`*. If the failure was recent, it is most likely the last entry at the bottom of the file. To identify the correct entry, look at the last comma separated field in the file. This field is the path of the data set collection and should match the folder structure/names of the data set collection to be re-ingested. Remove the relevant line in the file.

Next, the following commands will place the content into the processing folder framework:

```
cd /appl/ingest/Quarantine/HCP_Production/<DataSet>
mv RawData /appl/ingest/WorkArea/HCP_Production/<DataSet>/
mv TransformData /appl/ingest/WorkArea/HCP_Production/<DataSet>/
```

where *<DataSet>* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

Note: Depending on the failure, there may not be a `RawData` or `TransformData` folder to move as all this type of content may have been ingested successfully. So as long as one of the above `mv` commands does something useful, the next data set processing will attempt the ingestion again.

NOTE: If the reason for the failure is not clear, there are separate log files are created from the COMET ingestion processes. There is one file for the processing the `RawData` content and one for the `TransformData` content. The paths to those log files are:

```
logs/<DataSet>/COMET_Raw.log
logs/<DataSet>/COMET_Transform.log
```

where *<DataSet>* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

7.5.2 Unknown Region

During the transformation processing, the content is assigned region(s) that dictates which namespace(s) the content will reside for regional privacy reasons. The mechanism to determine the region is different for each data set type. The region is assigned a code, and the code is used to lookup the namespace for which the content should be written.

The following table summarizes how each data set type determines the region code:

Data Set Type	Region Determination Method
Bloomberg	The Bloomberg content is delivered in a very compact form and does not have the concept of a region. To determine the region for content, the compact content is substantially transformed to consumable chunks. Within these chunks, there are Bloomberg Account Number(s). The first step of the mapping is to lookup what the region code is for the Bloomberg Account Number. This mapping is configured via a Bloomberg only configuration file.
Reuters	When data sets are presented to the solution, the top level folder in the staging area is of the form *<RegionCode>*`_data`. These folders and their content are moved/transformed into the `RawData` and `TransformData` folder in the `WorkArea`.
Verint	The `RawData` folder layout has a collection folder where the first word of the folder name delimited by "." character is the region code. This folder name is used to construct the ingested folder structure that contains the `RawData` and the `TransformData` folders, and those contain a folder of the form *<RegionCode>*`_data`.

Data Set Type	Region Determination Method
Algomi	The Algomi data collection consists of a manifest file and a group of XML files containing the chat records. The XML files naming consists of "." character separated fields. The second field is the region code. This region code field is used to construct the ingested folder structure that contains the `RawData` and the `TransformData` folders, and those contain a folder of the form *`<RegionCode>`*`_data`.

For more information on the region mapping configuration see sections:

- *4.2.1 RegionMapper_*.properties File Format*
- *4.2.2 AccountMapping.lst File Format [Bloomberg ONLY].*

In the event that either a region code could not be determined, or the region code is not a known region code for the data set, the content will be considered "Unknown" and thus not ingested into HCP. At the end of processing, these unknown files are moved to the `Quarantine` area.

7.5.2.1 New Region Introduced

Upon investigation, if the cause of the unknown region mapping is the introduction of a new region, the first step is to adjust the region mapping files for the applicable data set. See the sections mentioned in the bullets above for more information on how to configure a new region. A ***special note*** is that Bloomberg configuration requires configuration for account to region, and also region to namespace.

For Reuters, Verint, and Algomi re-introducing content into the ingestion process is straight forward. Once the issue is resolved. The content placed in `Quarantine` can be re-introduced back into the `WorkArea`. First, the data set collection checksum record, if existing, needs to be removed. The checksum record resides in the file path specified by:

```
/appl/ingest/WorkArea/HCP_Production/<DataSet>/history.log
```

where *`<DataSet>`* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

Using your favorite editor, edit the `history.log` for the specific *`<DataSet>`*. If the failure was recent, it is most likely the last entry at the bottom of the file. To identify the correct entry, look at the last comma separated field in the file. This field is the path of the data set collection and should match the folder structure/names of the data set collection to be re-ingested. Remove the relevant line in the file.

Next, the following commands will place the content into the processing folder framework:

```
cd /appl/ingest/Quarantine/HCP_Production/<DataSet>
mv RawData /appl/ingest/WorkArea/HCP_Production/<DataSet>/
mv TransformData /appl/ingest/WorkArea/HCP_Production/<DataSet>/
```

where *`<DataSet>`* is one of `Reuters`, `Bloomberg`, `Verint` or `Algomi`.

Note: Depending on the failure, there may not be a `RawData` or `TransformData` folder to move as all this type of content may have been ingested successfully. So as long as one of the above `mv` commands does something useful, the next data set processing will attempt the ingestion again.

For Bloomberg, this is a bit more complex. Not only are there two configuration files that need to be changed, the Bloomberg content goes through a far more complex transformation process to ensure content is ingested properly. The easiest approach is the following procedure:

- Using a tool like HCP-DM, locate the `RawData` form of the data set collection(s). These collections should reside in the `blmrdrab*` namespace in the `hcpblm*` tenant. Save all the files of the collection onto the data ingestion server. The easiest is to grab the date/time formatted folder and all its contents.
  For this discussion, assume the following information:
  - Files are saved to `/appl/ingest/tmp` folder on the data ingestion system.
  - The collection date/time folder is: `02-25T190000+0000`
  - And the parent folder in HCP is: `2015`.
- Once copied, remove the `RawData` form of all the collections collected in the first bullet.
- Remove all `TransformData` data set collection(s) from all regional namespaces for the data sets collected in the first bullet.
- Remove the checksum record for the data set collection(s) to be re-processed. The checksum record resides in the file path specified by:

  ```
  /appl/ingest/WorkArea/HCP_Production/Bloomberg/history.log
  ```

  Using your favorite editor, edit the `history.log` file. If the failure was recent, it is most likely the last entry at the bottom of the file. To identify the correct entry, look at the last comma separated field in the file. This field is the path of the data set collection and should match the folder structure/names of the data set collection to be re-ingested. Remove the relevant line in the file and save.
- Edit the data set specific `history.log` file by removing the checksum lines
- Issue the following commands to put the content back into the processing flow:

  ```
  cd /appl/ingest/WorkArea/HCP_Production/Bloomberg
  mkdir -p RawData/RBINTL_data/3616/2015
  mv /appl/ingest/tmp/02-25T190000+0000 RawData/RBINTL_data/3616/2015/
  ```

- Run the Bloomberg processing. This will retransform the content using the new mapping configuration.

7.5.3 Unexpected Files

Encountering unexpected files is when there are files for which this solution is not configured to process. This condition will only occur if the files included in the data sets have changed or the processing to deliver the data sets to the ingestion engine has changed.

If the new file is not one configured for the solution and is a new file that should be processed, the solution configuration and/or software will have to be changed. Contact HDS for guidance as this kind of recovery is out of the scope of this document.

Appendix A

HCP Tenant/Namespace Mapping

The tables below represent the tenants and namespaces for each data source type. For the full mapping and additional information see the document *HCP Data Ingestor: Directory, Tenant, and Namespace Naming Standards* created by Rabobank.

Bloomberg

The Bloomberg tenant for production will be named HCPBLM400. Within this tenant there will be the following namespaces created and the owning country:

Namespace	Customer IDs	Country
BLMARRAB400	548478	Argentina
BLMAURAB400	30146097 624225	Australia
BLMBERAB400	30204878	Belgium
BLMBRRAB400	43673	Brazil
BLMCLRAB400	30086012	Chile
BLMCNRAB400	30167443 30167694	China
BLMHKRAB400	30114181	Hong Kong
BLMINRAB400	30214976	India
BLMIDRAB400	213002	Indonesia
BLMIERAB400	30082955	Ireland
	30122731	Japan
BLMMERAB400	30009312	Mexico
BLMNLRAB400	30537 30009756 30197429 196970 30184577 30140309 554332 196976 217426 30208420 179739 128915 3616 201308 748220 30135751	Netherlands

Namespace	Customer IDs	Country
	102834 707527	
BLMSGRAB400	117481 117589	Singapore
BLMTRRAB400	30218535	Turkey
BLMGBRAB400	30217083 73828 206856 30211910 30218690	United Kingdom
BLMUSNAW400	30219040 220733 8229 319647 30052667 585552 30050391 30201800 30216818 30087853	United States (North America Wholesale)

Reuters

The Reuters tenant for production will be named HCPREU400. Within this tenant there will be the following namespaces created and the owning country:

Namespace	TRMC ID	Country
REUAURAB400	52100	Australia
REUBRRAB400	58010	Brazil
REUCLRAB400	319686	Chile
REUCNRAB400	54760	China
REUHKRAB400	56686	Hong Kong
REUINRAB400	472643	India
REUIDRAB400	141416	Indonesia
REUIERAB400	9456	Ireland
REUNLRAB400	499212	Netherlands
REUSGRAB400	82665	Singapore
REUESRAB400	347910	Spain
REUTRRAB400	470343	Turkey
REUGBRAB400	237	United Kingdom
REUUSNAW400	56769	United States (North America Wholesale)

Verint

The Verint tenant for production will be named HCPVER400. Within this tenant there will be the following namespaces created and the owning country:

Namespace	Country
VERNLRAB400	Netherlands
VERGBRAB400	United Kingdom

Algomi

The Algomi tenant for production will be named `HCPALG400`. Within this tenant there will be the following namespaces created and the owning country:

Namespace	Country
ALGNLRAB400	Netherlands
ALGGBRAB400	United Kingdom
ALGUSRAB400	United States
ALGRDRAB400	RawData