

iWay

Omni-Insurance Operation and
Management Guide

Version 3.12

Active Technologies, EDA, EDA/SQL, FIDEL, FOCUS, Information Builders, the Information Builders logo, iWay, iWay Software, Parlay, PC/FOCUS, RStat, Table Talk, Web390, WebFOCUS, WebFOCUS Active Technologies, and WebFOCUS Magnify are registered trademarks, and DataMigrator and Hyperstage are trademarks of Information Builders, Inc.

Adobe, the Adobe logo, Acrobat, Adobe Reader, Flash, Adobe Flash Builder, Flex, and PostScript are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Due to the nature of this material, this document refers to numerous hardware and software products by their trademarks. In most, if not all cases, these designations are claimed as trademarks or registered trademarks by their respective companies. It is not this publisher's intent to use any of these names generically. The reader is therefore cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

Copyright © 2020, by Information Builders, Inc. and iWay Software. All rights reserved. Patent Pending. This manual, or parts thereof, may not be reproduced in any form without the written permission of Information Builders, Inc.

Contents

Preface	7
Documentation Conventions	7
Related Publications	8
Customer Support	8
Help Us to Serve You Better	9
User Feedback	10
iWay Software Training and Professional Services	10
1. Omni-Insurance Operation and Management	13
Database Usage	13
Bundle Deployment Tasks.	13
Database Reset and Clean.	14
Order of Database Deployment Execution Tasks.	15
Changes to System Tables Work Order and Work Order Item.	16
Database Validation Step.	16
Deployment Process Detail.	17
Deployment Logs.	17
Data Quality Bundle Deployments.	18
Database Migration.	18
Description of Database Tables.	20
Change Log Generation and Execution.	24
ChangeLog Locations.	26
OmniGenData and OmniGovConsole Data Remediation.	27
Controller Database Tasks.	28
Console Database Connectivity Checks.	29
Operations Console.	29
Database Pools and Connection Information.	29
Liquibase.	34
Common Database Considerations.	36
Performance Tuning.	37
Database Maintenance Recommendations	38
Archiving Tables in the Omni-Insurance Database.	38
Archiving Error and Information Logs in Omni-Insurance.	39

General Maintenance Recommendations for Common Database Systems.....	40
Purging Old Data.....	42
Purging Inactive Data.....	43
Data Quality Components	43
Data Quality Processes.....	44
Data Quality Processing.....	48
DQ Process Activities.....	49
Warnings and Errors.....	49
Deployment Operations	51
Work Orders	51
Work Order Types.....	51
Work Order States.....	54
Work Order Items.....	55
START.....	55
RAMP_TO_SOURCE.....	55
TXN_ON_SOURCE and ON_SEL_SOURCES.....	56
SOURCE_TO_MODEL.....	56
RAMP_TO_MODEL.....	57
MASTER_REFERENCE.....	57
CLEANSE.....	57
TXN_ON_INSTANCE and ON_SEL_INSTANCES.....	58
MATCH.....	58
MATCH_SET_INACTIVE.....	60
MATCH_SET_DELETE.....	60
FILL_RELOAD_QUEUE.....	60
MERGE.....	61
PROMOTE_MASTER.....	61
REMEDiate.....	61
AUTO_CLOSE.....	62
HISTORY.....	63
PUBLISH.....	64
MASTER_REFERENCE_RELOAD.....	64
SET_RELOAD_TRANSACTION.....	64

CLEAR_RELOAD_QUEUE.....	64
CDC_RECORD.....	64
SUBJECT_GROUP_PROCESS.....	64
STOP.....	65
Work Order Automation.....	65
Executing the Automation.....	65
Creating an Automation.....	66
Starting the Automated Work Order (Submitting the Job).....	67
Remediation Ticket Submission	68
Omni-Insurance Operational Measures	71
Omni-Insurance Logging	75
Viewing System Logs.....	76
Configuration.....	78
File Organization.....	79
Log and Output File Details.....	80
Advanced Logging Options.....	83
Data Quality Tracing.....	83
Standard Logging Support.....	83
Log4J Logging Considerations.....	84
A. Reserved Words	87
Reserved Words List	87

Preface

This documentation provides operation and management information for Omni-Insurance. It is intended for developers and administrators of Omni-Insurance.

How This Manual Is Organized

This manual includes the following chapters:

Chapter/Appendix		Contents
1	Omni-Insurance Operation and Management	Provides information for Omni-Insurance operation and management.
A	Reserved Words	Provides information on system reserved words, which you should not use as part of the model definition or any user-defined fields.

Documentation Conventions

The following table lists and describes the documentation conventions that are used in this manual.

Convention	Description
<code>THIS TYPEFACE</code> or <code>this typeface</code>	Denotes syntax that you must type exactly as shown.
<i>this typeface</i>	Represents a placeholder (or variable), a cross-reference, or an important term. It may also indicate a button, menu item, or dialog box option that you can click or select.
<u>underscore</u>	Indicates a default setting.
Key + Key	Indicates keys that you must press simultaneously.
{ }	Indicates two or three choices. Type one of them, not the braces.
	Separates mutually exclusive choices in syntax. Type one of them, not the symbol.

Convention	Description
...	Indicates that you can enter a parameter multiple times. Type only the parameter, not the ellipsis (...).
.	Indicates that there are (or could be) intervening or additional commands.

Related Publications

Visit our Technical Documentation Library at <http://documentation.informationbuilders.com>. You can also contact the Publications Order Department at (800) 969-4636.

Customer Support

Do you have questions about this product?

Join the Focal Point community. Focal Point is our online developer center and more than a message board. It is an interactive network of more than 3,000 developers from almost every profession and industry, collaborating on solutions and sharing every tips and techniques. Access Focal Point at <http://forums.informationbuilders.com/eve/forums>.

You can also access support services electronically, 24 hours a day, with InfoResponse Online. InfoResponse Online is accessible through our website, <http://www.informationbuilders.com>. It connects you to the tracking system and known-problem database at the Information Builders support center. Registered users can open, update, and view the status of cases in the tracking system and read descriptions of reported software issues. New users can register immediately for this service. The technical support section of www.informationbuilders.com also provides usage techniques, diagnostic tips, and answers to frequently asked questions.

Call Information Builders Customer Support Services (CSS) at (800) 736-6130 or (212) 736-6130. Customer Support Consultants are available Monday through Friday between 8:00 A.M. and 8:00 P.M. EST to address all your questions. Information Builders consultants can also give you general guidance regarding product capabilities. Be prepared to provide your six-digit site code (xxxx.xx) when you call.

To learn about the full range of available support services, ask your Information Builders representative about InfoResponse Online, or call (800) 969-INFO.

Help Us to Serve You Better

To help our consultants answer your questions effectively, be prepared to provide specifications and sample files and to answer questions about errors and problems.

The following table lists the environment information that our consultants require.

Platform	
Operating System	
OS Version	
JVM Vendor	
JVM Version	

The following table lists additional questions to help us serve you better.

Request/Question	Error/Problem Details or Information
Did the problem arise through a service or event?	
Provide usage scenarios or summarize the application that produces the problem.	
When did the problem start?	
Can you reproduce this problem consistently?	
Describe the problem.	
Describe the steps to reproduce the problem.	
Specify the error messages.	

Request/Question	Error/Problem Details or Information
Any change in the application environment: software configuration, EIS/database configuration, application, and so forth?	
Under what circumstance does the problem <i>not</i> occur?	

The following is a list of error and problem files that might be applicable.

- ☐ Input documents (XML instance, XML schema, non-XML documents)
- ☐ Transformation files
- ☐ Error screen shots
- ☐ Error output files
- ☐ Trace files
- ☐ Custom functions and agents in use
- ☐ Diagnostic Zip
- ☐ Transaction log

User Feedback

In an effort to produce effective documentation, the Technical Content Management staff welcomes your opinions regarding this document. Please use the Reader Comments form at the end of this document to communicate your feedback to us or to suggest changes that will support improvements to our documentation. You can also contact us through our website, <http://documentation.informationbuilders.com/connections.asp>.

Thank you, in advance, for your comments.

iWay Software Training and Professional Services

Interested in training? Our Education Department offers a wide variety of training courses for iWay Software and other Information Builders products.

For information on course descriptions, locations, and dates, or to register for classes, visit our website, <http://education.informationbuilders.com>, or call (800) 969-INFO to speak to an Education Representative.

Interested in technical assistance for your implementation? Our Professional Services department provides expert design, systems architecture, implementation, and project management services for all your business integration projects. For information, visit our website, <http://www.informationbuilders.com/consulting>.

Omni-Insurance Operation and Management

This document is designed to serve as a starting point and reference for an operator of Omni-Insurance. It contains information on various underlying structures of the product, including some that should not be modified by the end-user, but are documented to assist during troubleshooting. For general operational use, see the *Omni Console User's Guide*.

In this chapter:

- ☐ [Database Usage](#)
 - ☐ [Database Maintenance Recommendations](#)
 - ☐ [Data Quality Components](#)
 - ☐ [Deployment Operations](#)
 - ☐ [Work Orders](#)
 - ☐ [Remediation Ticket Submission](#)
 - ☐ [Omni-Insurance Operational Measures](#)
 - ☐ [Omni-Insurance Logging](#)
-

Database Usage

This section describes important information on database interactions and usage.

Bundle Deployment Tasks

The following deployment actions will affect the database structure and/or data content.

- ☐ **Install/Replace** (deploy-bundle-clean) - Wipeout existing database tables, install new bundle, create new schema.
- ☐ **Update** (deploy-bundle) - Upgrade an existing bundle, update Data Model.
- ☐ **Update DQ Bundle** (deploy-dq-bundle) - Replace the existing Data Quality/Mastering configuration files in /omnigen/OmniServer/Mastering, no database tasks.
- ☐ **Reset** (deploy-db-clean) - Reset the existing database to its default state from the command line or console. Options include Model Tables or Model and System Tables (everything).

Database Deployment Phases

- 1. Drop
- 2. Create/Update

The following table lists the command line (omni.sh/omni.cmd) deployment commands and their Omni Console equivalents.

Command Line	Omni Console
deploy-bundle	Update bundle
deploy-bundle-clean	Install/Replace bundle
deploy-db-clean	Reset
deploy-dq-bundle	Update Data Quality bundle

Database Reset and Clean

During a database reset or a deployment that invokes the *clean* flag, the following database tables are dropped and recreated. These table identifiers are hardcoded in the server and will match exact names or character patterns if denoted by an asterisk (*).

Omni-Insurance Model Tables [Clean and Reset]	
Matched Patterns	Specific Named Tables
ids_*	job_request
md_*	omni_error_docs
og_*	omni_remediation_ticket
os_cdc_*	omnigen_interface
os_i*	
os_m*	
os_r*	
os_s*	

Omni-Insurance Model Tables [Clean and Reset]

source_*

repos_*

Omni-Insurance System Tables [Reset Only]**Matched Patterns**

os_work_*

Specific Named Tables

Development is underway to begin separating *system* tables from *bundle* tables that are associated with a specific deployment, as their lifecycles are different. Separate lists are now maintained internally for tables that belong to each category. However, some ambiguity remains. For example, tables such as *os_measure*, which are system tables by definition, since they are not inherently tied to a subject model, have not been fully incorporated into the new lifecycle.

Order of Database Deployment Execution Tasks

These are performed after a new bundle is deployed or updated.

1. Drop existing database tables (if clean is specified – Reset/Replace only). This includes the following data sources:
 - a. Workflow/Remediation (These are pre-written scripts, *not* generated dynamically.)
 - b. Model
 - c. Mastering
 - d. Ramp
 - e. Source
 - f. Clean system tables (Clean only). See *Changes to System Tables Work Order and Work Order Item*.
 - g. System (Reset only). See *Changes to System Tables Work Order and Work Order Item*.
 - h. Delete existing Liquibase Database changelogs.
2. Custom(er) database pre-deployment migration scripts.
3. Omni-Insurance pre-deployment migration scripts.

4. Create/Update database tables. This includes the following data sources:
 - a. Model
 - b. Ramp
 - c. Source
 - d. Cohort
 - e. System
 - f. Workflow/Remediation (This is *Create* only. These are pre-written scripts, *not* generated dynamically from changelogs.)
5. Custom(er) database post-deployment migration scripts.
6. Omni-Insurance post-deployment migration scripts.

Changes to System Tables Work Order and Work Order Item

The Work Order and Work Order Item (and other *os** table measures, ramp, and so on) are created as part of the bundle deployment process.

1. The Controller checks on startup to ensure the *os_work_order* and *os_work_order_item* tables exist. If not, it creates them.
2. There will be two options for Database reset from the Console, see *Deployment* below.
3. Deployment
 - a. Install/Replace - If the tables exist, all Work Orders with *omni_system_type* = "SERVER", and their related Work Order Items, are deleted.
 - b. Upgrade/Update - If the tables do not exist (for some reason) they are created or updated, as needed.
 - c. Database Reset
 - a. The *Model* option resets all managed tables as before, *except* Work Order and Work Order Item are handled as in *Install/Replace*.
 - b. The *System and Model* option resets all of the managed tables in the system (including Work Order and Work Order Item). This is the same behavior as the previous implementation of *Reset Database*.

Database Validation Step

There is a validation step during installation that checks for the existence of the system tables in the database. After you supply parameters for the Omni-Insurance database, you are prompted to continue or quit the installation, based on the existence of the tables. This serves to prevent you from accidentally overwriting database table information.

Deployment Process Detail

Generating JPA from IDS, changelogs, and so on.

1. Backup existing bundle.
2. Clean previous bundle files and generated artifacts.
3. Copy the bundle to your directory and unzip the contents.
4. Copy any bootstrap OID files (OHD only).
5. Generate Effective IDS documents.
6. Generate IDS documentation.
7. Generate IDS sample OIDs.
8. Generate XSD schemas for the IDS documents.
9. Generate JPA model for the IDS documents.
10. Compile the JPA model.
11. Weave the JPA model.
12. Package and move the model jar.
13. Copy the new DQ configurations for Cleansing, Merging, Matching, Remediation, and Relationships.

Deployment Logs

Each time a bundle or database deployment is undertaken from command line or UI, a log file is created with a name and timestamp corresponding to the event.

- ☐ The file name is created by concatenating the operation name and a timestamp of when the event started. Example: `deploy_update_2017-11-06 12-25-29.865.json`
- ☐ Location: `omnigen/OmniGenData/deployment`
- ☐ The information in these files is the content that is loaded on the deployment progress screen of the Omni Console.

The following table provides a list of the possible generated log files and the corresponding deployment commands.

Deployment Action	File name
Update bundle	<code>deploy_update_{timestamp}.json</code>

Deployment Action	File name
Install/Replace bundle	deploy_install_{timestamp}.json
Update DQ bundle	deploy_dq_update_{timestamp}.json
Reset database	deploy_reset_{timestamp}.json

Data Quality Bundle Deployments

Added an ability for *Data Quality Only* deployment bundles that skip all code generation/IDS operations and database processing. This will only replace the DQ configuration, and perform any DQ related operations.

The steps include copying the following configuration files from the exploded bundle:

1. Cleansing
2. Matching
3. Merging
4. Remediation
5. Relationships
6. Rebuilding the lookup sent with the bundle

This assumes that the IDS directory in the deployment bundle will be empty for a DQ only bundle.

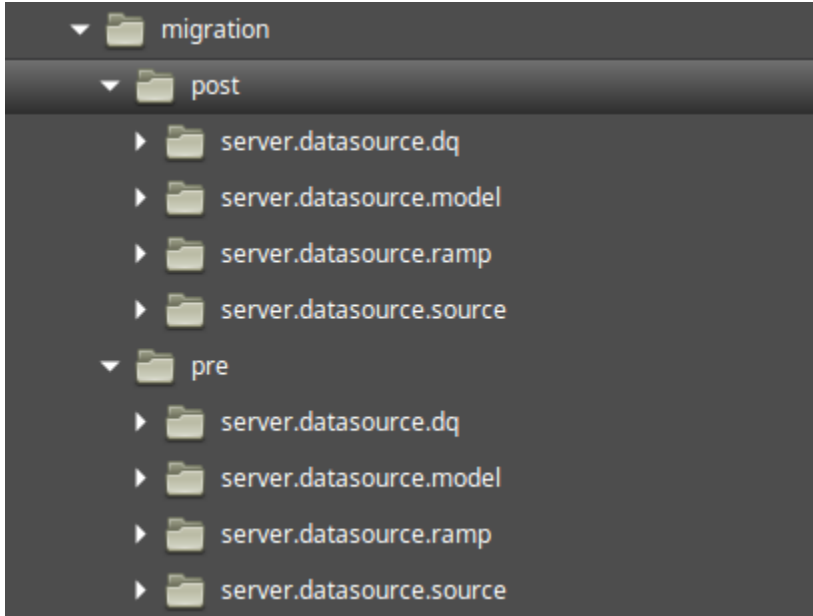
Changes:

- ☐ Add `deploy_dq_bundle` command to the command line. This will bypass all the database processes normally run during deployment.
- ☐ Added an additional button to the Deployment screen on the Omni Console. There are now two options attached to the *Update Bundle* button. One for the standard Deployment bundle, and one for the Data Quality Bundle.
- ☐ New command line task `deploy_dq_bundle` to execute this action.

Database Migration

This process allows pre- and post-deployment database scripts to be run from separate locations for customers and the internal Omni-Insurance team.

The following directory structure exists to hold the source files used in the migration process. There are *pre* and *post* folders that contain a directory for each data source. The migration scripts should be added to the appropriate directory.



Customer: /omnigen/OmniGenData/migration

Omni-Insurance: /omnigen/OmniServer/dbms/migration

Example of SQL wrapper for a Liquibase changelog:

- ☐ It is assumed that the end-user will be using this format, so the SQL syntax should be database vendor specific.
- ☐ This changeSet executes native SQL against the specified database (H2 in this case).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://www.liquibase.org/xml/ns/
dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-
3.3.xsd">
    <changeSet author="liquibase-docs" id="sql-example">
        <sql dbms="h2"
            endDelimiter="\nGO"
            splitStatements="true"
            stripComments="true">
            <comment>You must specify the Schema if it is not the
default for the datasource</comment>
            <comment>This varies between databases</comment>
            create table person(name varchar(255));
            insert into person (name) values ('Bob');
        </sql>
    </changeSet>
</databaseChangeLog>
```

If you are not using the default schema for a data source, it must be explicitly specified in the migration script as shown in the changelog comment above.

It is recommended to continue using the standard changeset format to accommodate as many database vendors as possible. However, there may be vendor specific issues where the SQL version is more appropriate.

Description of Database Tables

This section includes information on the database tables, as well as noted naming convention for the deployment specific tables.

☐ Liquibase:

- ☐ Holds transactions for all the operations performed by Liquibase on the given database.
- ☐ Databasechangelog, databasechangeloglock

☐ Ramp:

- ☐ Used to load data from the ramp into the Omni-Insurance environment.
- ☐ Named: og_{subject_name}_r

☐ Source:

- ☐ Source data (does not get modified).
- ☐ Named: og_{subject_name}_s

- ☐ Instance:
 - ☐ Instance data (Source data after Cleansing process, if any.)
 - ☐ Named: og_{subject_name}
- ☐ Master:
 - ☐ Holds the mastered records for a given subject.
 - ☐ Named: og_{subject_name}_m
- ☐ History:
 - ☐ Named: og_{subject_name}_h
- ☐ Work Flow:
 - ☐ Holds information related to the remediation process.
 - ☐ Named: wf{}
- ☐ System tables:
 - ☐ Used by Omni-Insurance system, not dependent on the model.
 - ☐ Named: os_

Below is a list of Omni-Insurance system tables. These are not subject specific and exist regardless of the subject deployment bundle.

System tables and descriptions – DO NOT MODIFY TABLES

System Table Name	Description
ids_override_match	
ids_override_property	
job_request	
md_ids_document	
md_ids_document_change	
md_ids_document_element	

System Table Name	Description
md_ids_document_group	
md_ids_document_list	
md_ids_document_promote	
md_ids_document_type	
os_measure	Measures (metrics/timings) for various processes across the Omni-Insurance system.
os_ramp_control	Data loading jobs and related information that were submitted to the ramp for processing.
os_reload_queue	
os_source_code_xref	Source code cross-reference tables.
os_subject_group	Grouping
os_subject_group_relation	Grouping relation
os_word_order	Subject specific jobs to be processed by the system.
os_work_order_item	A detailed list of steps that are undertaken in each work order.
Grouping Table Name	Description
os_subject_group	Also listed in System table.
os_subject_group_relation	Also listed in System table.
subject_group	
group_node	
Workflow Table Name	Description
scxml_instance	

Workflow Table Name	Description
scxml_lock	
wfCases	
wfDocument	
wfEvents	
wfMetaData	
wfPayload	
wfRemedyRef	
wfTicketLock	
wfTickets	A list of remediation tickets to be resolved.
workflow_version	Metadata on the creation of Workflow Tables.
case_doc_xref	View
wfUserCase	View

Data Quality tables – DO NOT MODIFY TABLES

System Table Name	Description
repos_{subject_name}_data	
repos_{subject_name}_form	
repos_{subject_name}_keys	
repos_{subject_name}_meta	
repos_{subject_name}_wdkey	
repos_{subject_name}_wgid	
repos_{subject_name}_wkey	

System Table Name	Description
repos_{subject_name}_wpk	

Change Log Generation and Execution

The fundamental building block of Liquibase is the Changelog.

These are stored in omnigen/OmniServer/dbms/changelogs.

A set of changelogs is generated for each of the main data sources in the Omni-Insurance system.

1. MigrateChangeLog.xml
- ☐ A Liquibase changelog denoting the differential between the existing database and the generated JPA classes (Model Jar).
2. DropChangeLog.xml
- ☐ A Liquibase changelog denoting the operations necessary to drop all of the specified tables in the given schema.
3. JpaModel.xml
- ☐ A Liquibase changelog created directly from the JPA class definitions according to the persistence.xml.

Source	omnigen-sourceMigrateChangeLog.xml
	omnigen-sourceDropChangeLog.xml
	omnigen-sourceJpaModelChangeLog.xml
Ramp	omnigen-rampMigrateChangeLog.xml
	omnigen-rampDropChangeLog.xml
	omnigen-rampJpaModelChangeLog.xml
Model	omnigen-modelMigrateChangeLog.xml
	omnigen-modelDropChangeLog.xml
	omnigen-modelJpaModelChangeLog.xml

Mastering	omnigen-masteringDropChangeLog.xml
System	omni-systemMigrateChangeLog.xml
	omni-systemJpaModelChangeLog.xml
	omni-systemDropChangeLog.xml
Entire database schema	{hashkey}-DatabaseChangeLog.xml

Below are example subsets of the different changelogs.

DropChangeLog.xml

```

omnigen-modelDropChangeLog.xml
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
   http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.3.xsd">
3      <changeSet id="OmniGen-2017-10-31-04:51:46" author="OmniGen" runAlways="false">
4          <dropTable cascadeConstraints="true" tableName="ids_override_match"/>
5          <dropTable cascadeConstraints="true" tableName="os_subject_group_relation"/>
6          <dropTable cascadeConstraints="true" tableName="og_person"/>
7          <dropTable cascadeConstraints="true" tableName="og_facility_relation"/>
8          <dropTable cascadeConstraints="true" tableName="og_organization_address"/>
9          <dropTable cascadeConstraints="true" tableName="og_person_m"/>
10         <dropTable cascadeConstraints="true" tableName="os_ramp_control"/>
11         <dropTable cascadeConstraints="true" tableName="os_measure"/>
12         <dropTable cascadeConstraints="true" tableName="os_subject_group"/>
13     </changeSet>
14 </databaseChangeLog>
15

```

MigrateChangeLog.xml/JpaChangeLog.xml

```

omnigen-sourceMigrateChangeLog.xml •
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <databaseChangeLog xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
   http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.3.xsd">
3    <changeSet id="OmniGen-2017-10-31-04:52:20" author="controller" runAlways="false">
4      <createTable tableName="og_organization_name_s">
5        <column name="id" type="VARCHAR(255)">
6          <constraints nullable="false"/>
7        </column>
8        <column name="status" type="VARCHAR(255)"/>
9        <column name="source_name" type="VARCHAR(255)"/>
10       <column name="source_instance_id" type="VARCHAR(255)"/>
11       <column name="source_instance_id_name" type="VARCHAR(255)"/>
12       <column name="type_code" type="VARCHAR(255)"/>
13       <column name="start_date" type="DATE"/>
14       <column name="end_date" type="DATE"/>
15       <column name="full_name" type="VARCHAR(255)"/>
16       <column name="status_reason" type="VARCHAR(255)"/>
17       <column name="omni_created_date" type="DATETIME"/>
18       <column name="omni_modified_date" type="DATETIME"/>
19     </createTable>
20
21     <addPrimaryKey columnNames="id" constraintName="pk_og_organization_name_s"
   tableName="og_organization_name_s"/>
22
23     <createIndex unique="false" indexName="ix_og_organization_name_s_organization_id"
   tableName="og_organization_name_s">
24       <column name="organization_id"/>
25     </createIndex>
26   </changeSet>
27 </databaseChangeLog>

```

ChangeLog Locations

Omni-Insurance Liquibase Changelogs are stored in the following locations, under omnigen/OmniServer/dbms/:

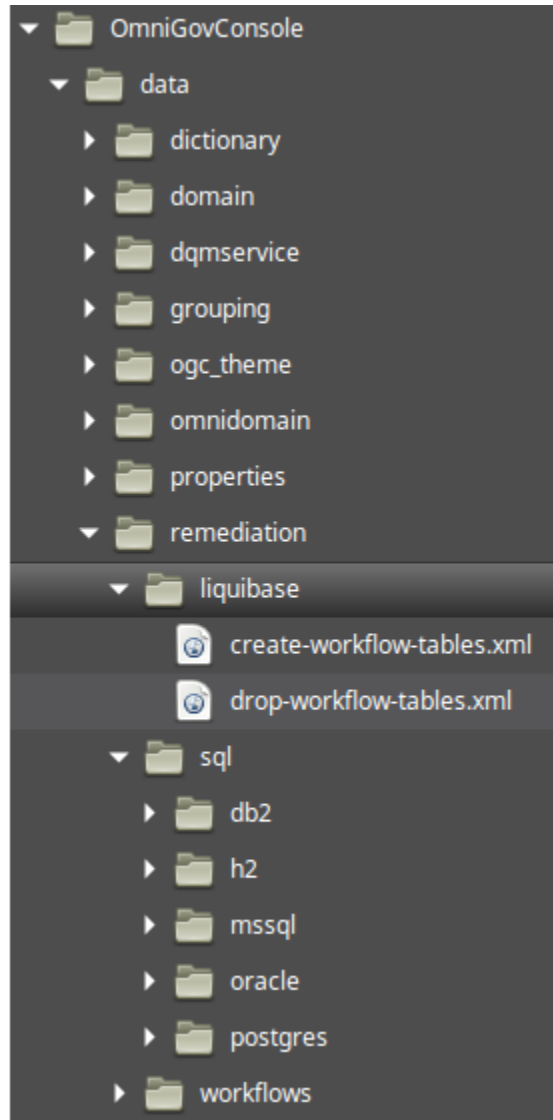
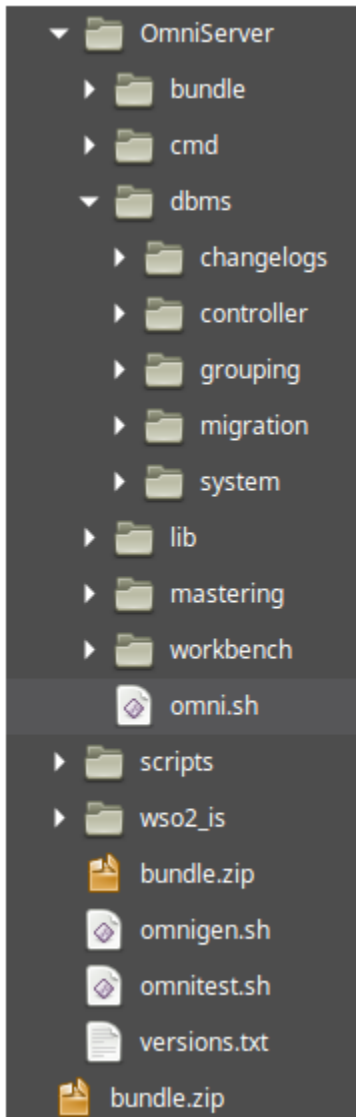
- ☐ Changelogs: All dynamically generated changelogs created by Omni-Insurance during the deployment process. This folder gets cleared during a *Reset* or *Bundle Clean* operation.
- ☐ Controller: Pre-written Liquibase scripts that run during the startup phase of the Controller.
- ☐ Grouping:
 - ☐ A Liquibase script that resets the `subject_group` and `group_node` tables related to grouping during a bundle reset or installation/replace.

- ❑ A Liquibase script to create the tables, sequence, and stored procedures during the update database phase of deployment.
- ❑ Migration:
 - ❑ Contains pre-written database scripts to aid in migration between versions or feature releases. These run every time deployment is executed.
 - ❑ Separate folders for each data source.
 - ❑ Different location for Information Builders system and customer-specific scripts.
- ❑ System: Contains pre-written scripts related to *System* database table maintenance. Currently, this only includes cleaning Work Order and Work Order Item tables, and their creation.

OmniGenData and OmniGovConsole Data Remediation

- ❑ Liquibase: Contains the Drop and Create Liquibase scripts that run raw SQL statements, based on the type of database being used.

- ❑ **Sql:** Contains the raw SQL scripts to drop and create all necessary Workflow/Remediation tables for each database.



Controller Database Tasks

The Controller may execute a number of database operations, depending on the state of the system.

The root Liquibase script for the controller is stored in `/OmniServer/dbms/controller`. This script contains a list of scripts and files to execute that includes creating the Work Order, Work Order Item, and Workflow/Remediation tables, if they do not already exist. If any issues occur during this process, the Controller will continue to start, either by executing an empty Changeset, or ignoring the exception. A message will be included in the Controller log file indicating the issue, and the Console will show the relative configuration errors.

Console Database Connectivity Checks

The controller will validate all database connections at a set interval (every 20 seconds) and display an error regarding any issue, if one exists. You will be prohibited from taking any actions you would normally take with an invalid configuration. If the database reconnects or the settings are fixed, the errors will resolve and the console will return to normal.

Operations Console

The Operations pages on the Console provide a better idea of the inner workings of your Omni-Insurance system. The database activity tab shows the slowest ten, and the most recent ten queries from the Server. This feature must be enabled from the Database/Configuration screens (Model, Ramp, Source) before data is run through the system in order to gather the metrics.

Database Pools and Connection Information

The Omni application makes use of multiple database connection pools to manage database connections. Each is separately configurable, but a mechanism is provided to allow common-configuration (across server and controller) using the 'default' settings. Configuration for controller and server can be changed in the `OmniGenConfiguration.property` file. Configuration for Tomcat is controlled in files found in `OmniGovConsole/conf/Catalina`.

- ☐ Settings: `defaultDataSourceSettings`
- ☐ Property Prefix: `server.datasource.default`
- ☐ Max-active: 50
- ☐ Initial Size: 2

Omni Controller Application

- ☐ Pool: `modelConnectionPool`
- ☐ Construction: Eager
- ☐ Settings: `modelDataSourceSettings`

- ☐ Property Prefix: server.datasource.model
- ☐ Max-active: 50
- ☐ Initial Size: 2
- ☐ Derives From: defaultDataSourceSettings

Omni Controller Application

- ☐ Pool: systemConnectionPool
- ☐ Construction: Eager
- ☐ Settings: systemDataSourceSettings
- ☐ Property Prefix: server.datasource.system
- ☐ Max-active: 50
- ☐ Initial Size: 2
- ☐ Derives From: modelDataSourceSettings

Omni Server Application

- ☐ Pool: consumptionConnectionPool
- ☐ Construction: Lazy
- ☐ Settings: consumptionDataSourceSettings
- ☐ Property Prefix: server.datasource.consumption
- ☐ Max-active: 50
- ☐ Initial Size: 2
- ☐ Derives From: defaultDataSourceSettings

Omni Server Application

- ☐ Pool: modelConnectionPool
- ☐ Construction: Lazy
- ☐ Settings: modelDataSourceSettings
- ☐ Property Prefix: server.datasource.model

- ☐ Max-active: 50
- ☐ Initial Size: 2
- ☐ Derives From: defaultDataSourceSettings

Omni Server Application

- ☐ Pool: masteringConnectionPool
- ☐ Construction: Lazy
- ☐ Settings: masteringDataSourceSettings
- ☐ Property Prefix: server.datasource.dq
- ☐ Max-active: 50
- ☐ Initial Size: 2
- ☐ Derives From: defaultDataSourceSettings

Omni Server Application

- ☐ Pool: rampConnectionPool
- ☐ Construction: Lazy
- ☐ Settings: rampDataSourceSettings
- ☐ Property Prefix: server.datasource.ramp
- ☐ Max-active: 50
- ☐ Initial Size: 2
- ☐ Derives From: defaultDataSourceSettings

Omni Server Application

- ☐ Pool: sourceConnectionPool
- ☐ Construction: Lazy
- ☐ Settings: sourceDataSourceSettings
- ☐ Property Prefix: server.datasource.source
- ☐ Max-active: 50

- ☐ Initial Size: 2
- ☐ Derives From: defaultDataSourceSettings

Omni Server Application

- ☐ Pool: systemConnectionPool
- ☐ Construction: Eager
- ☐ Settings: modelDataSourceSettings
- ☐ Property Prefix: server.datasource.model
- ☐ Max-active: 50
- ☐ Initial Size: 2
- ☐ Derives From: defaultDataSourceSettings

Tomcat Application

- ☐ Pool: ogc.xml
- ☐ Construction: Eager
- ☐ Settings: grouping_config.db
- ☐ Max-active: 10
- ☐ Initial Size: 10

Tomcat Application

- ☐ Pool: OmniDomain.xml
- ☐ Construction: Eager
- ☐ Settings: jdbc/OmniWorkflow
- ☐ Max-active: 10
- ☐ Initial Size: 10

Tomcat Application

- ☐ Pool: OmniDomain.xml
- ☐ Construction: Eager

☐ Settings: jdbc/OmniGen

☐ Max-active: 10

☐ Initial Size: 10

Tomcat Application

☐ Pool: RemediationService.xml

☐ Construction: Eager

☐ Settings: jdbc/OmniWorkflow

☐ Max-active: 10

☐ Initial Size: 10

The following are descriptions of the database pools and connection information.

Application. The name of the Omni component that holds the pool.

Pool. The name of the JDBC pool. The pool name should be unique for each application and is not configurable. For Tomcat, this is the name of the file in conf/Catalina that configures the pool, and must coincide with the corresponding webapp.

Construction. Whether the pool is created at application startup (Eager), or is done when features needing the pool are activated (Lazy). Not configurable.

Settings. The name of the settings that govern the pool. For Tomcat, this is the Resource name.

Derives From. If a pool property is not provided, the corresponding value from this setting will be used.

Property Prefix. The property prefix to alter the settings in the OmniGenConfiguration.property file.

Max-active. The maximum number of active connections that can be allocated from this pool at the same time. For JDBC, if additional connections are requested, they will block waiting for a free connection, and may timeout if one does not become available within the timeout period. Max-active is configurable by setting a property of PROPERTY_PREFIX.max-connections (for example, server.datasource.default.max-connections). In Tomcat, it is configured by setting the *maxActive* property of the Resource.

Max-idle. The maximum number of connections that should be kept in the idle pool. In Tomcat, it is configured by setting the *maxIdle* property of the Resource.

Min-idle. The minimum number of connections that should be kept in the pool at all times. For JDBC, the default value for this property is derived from 'Initial Size'. In Tomcat, it is configured by setting the *minIdle* property of the Resource.

Initial Size. The number of connections that will be established when the connection pool is started. Initial Size is configurable by setting a property of PROPERTY_PREFIX.initial-connections (for example, server.datasource.default.initial-connections). In Tomcat, it is configured by setting the *initialSize* property of the Resource.

Ramifications: With all services running, a minimum of 56 connections would exist, and at most, 440 connections would be open. Using the default settings, it is recommended that a database support 500 maximum connections. This is configured differently for each database (PostgreSQL: the *max_connections* property in *pg_hba.conf*, Oracle: TBD, SQL-Server: TBD).

Liquibase

Source control for your database Liquibase is a software framework that provides a database vendor agnostic abstraction over common database operations. Utilizing the Changeset structure (differing formats are XML, JSON, YAML), you define database operations in a Liquibase specific format. When the changeset is executed, it will manage any idiosyncrasies under the hood for the specific database vendor (SQL Server, Oracle, Postgres, and so on).

<http://www.liquibase.org/>

Liquibase Notes: A version of Liquibase is being used that has been patched to address an issue. Deployment was failing against a SQL Server database with a case sensitive collation (for example, SQL_Latin1_General_CP1_CS_AS). It is version 3.5.3 available from Maven with an additional few commits.

Change tracking: There is a special table that is utilized by Liquibase for the management and record keeping of changelog actions called *databasechangelog*. It contains a hash of the designated changeset, as well as other identifying information. Once created, the Omni-Insurance system will not reset this table, it must be manually dropped.

Fields contained in this table:

Column	Standard Data Type	Description
ID	VARCHAR(255)	Value from the changeSet "id" attribute.
AUTHOR	VARCHAR(255)	Value from the changeSet "author" attribute.

Column	Standard Data Type	Description
FILENAME	VARCHAR(255)	Path to the changelog. This may be an absolute path or a relative path depending on how the changelog was passed to Liquibase. For best results, it should be a relative path.
DATEEXECUTED	DATETIME	Date/time of when the changeSet was executed. Used with ORDEREXECUTED to determine rollback order.
ORDEREXECUTED	INT	Order that the changeSets were executed. Used in addition to DATEEXECUTED to ensure order is correct even when the databases datetime supports poor resolution. Note: The values are only guaranteed to be increasing within an individual update run. There are times where they will restart at zero.
EXECTYPE	VARCHAR(10)	Description of how the changeSet was executed. Possible values include "EXECUTED", "FAILED", "SKIPPED", "RERAN", and "MARK_RAN",
MD5SUM	VARCHAR(35)	Checksum of the changeSet when it was executed. Used on each run to ensure there have been no unexpected changes to changeSet in the changelog file.
DESCRIPTION	VARCHAR(255)	Short auto-generated human readable description of changeSet.
COMMENTS	VARCHAR(255)	Value from the changeSet "comment" attribute.
TAG	VARCHAR(255)	Tracks which changeSets correspond to tag operations.

Column	Standard Data Type	Description
LIQUIBASE	VARCHAR(20)	Version of Liquibase used to execute the changeSet.

Database Locking: Liquibase uses a distributed locking system to only allow one process to update the database at a time. The other processes will simply wait until the lock has been released. There is a special table created for this purpose called *databasechangelock*.

Fields contained in this table

Column	Standard Data Type	Description
ID	INT	Id of the lock. Currently there is only one lock, but is available for future use.
LOCKED	INT	Set to "1" if the Liquibase is running against this database. Otherwise set to "0".
LOCKGRANTED	DATETIME	Date and time that the lock was granted.
LOCKEDBY	VARCHAR(255)	Human-readable description of who the lock was granted to.

Common Database Considerations

SQL Server

- ☐ All indexes are explicitly created as non-clustered. This was requested as a performance enhancement. By default, SQL Server will create a clustered index on the primary key, unless instructed otherwise.
- ☐ File locks are common.
- ☐ Limit to the length of indexes across columns (900 characters for releases prior to SQL Server 2016, 1700 for SQL Server 2016 and higher).
- ☐ No Drop with Cascade.
- ☐ Maximum object name is 100 characters.

Oracle

- ☐ Limit to the length of indexes across columns.
- ☐ Maximum object name is 30 characters.
- ☐ No Drop with Cascade.

PostgreSQL

- ☐ Maximum object name is 63 characters.

H2

- ☐ This is used as the default by Omni-Insurance if the system cannot configure a connection to an external database.

Db2

- ☐ No Drop with Cascade.
- ☐ To use Db2 as a repository database, the following tuning steps are required. This is due to the nature of Db2 and its requirement for higher memory consumption during the deployment phase. If the memory is not increased, you might encounter an *OutOfMemoryError* exception when resetting the environment or the deployment phase.

For new installations:

1. The Db2 JDBC URL should include a *traceLevel=0* option during the configuration.
2. Prior to running *config* on the binary, set *cfg.server.commandline.max-memory=2048M* in the configuration file.
3. After *config* completes, verify *server.commandline.max-memory=2048M* in the *OmniGenconfiguration.properties* file.

For existing installations:

1. The Db2 JDBC URL should include a *traceLevel=0* option during the configuration.
2. In the Omni Console, navigate to *Configuration, Runtime*, and click the *Command Line* tab. Set the *JVM Process Max Memory* parameter to 2048M.
3. Stop all processes and then restart.

Performance Tuning

When handling a large number of records, you may want to optimize the database being used for Omni-Insurance. One way to tune the performance of the database is to increase the amount of available memory. For example, for one million records, increasing the memory by 2GB could improve performance.

In addition, removing the following indexes can also improve performance:

❑ `REPOS_SUBJECT_WGID`

❑ `REPOS_SUBJECT_WPK`

where:

SUBJECT

Is the customer subject.

Database Maintenance Recommendations

This section provides maintenance recommendations for the Omni-Insurance database.

Archiving Tables in the Omni-Insurance Database

The following table lists and describes the tables that can be archived in the Omni-Insurance database.

Table	Table Name	Description	Archive Strategy
Measures	os_measure	Operational status information (usually timed) about processes that take place in the system (deploying a bundle, starting a service, merging, and so on).	As needed.
Work Orders	os_work_orders	A list of Work Order tasks generated by the system for execution.	As needed.
Work Order Items	os_work_order_item	A list of Work Order Items that represent the individual operations that take place during the execution of a Work Order.	As needed.

Table	Table Name	Description	Archive Strategy
Change Data Capture (If enabled)	os_cdc_change	Captures changes that take place to subjects within the system.	As needed.
Change Data Capture Subscription	os_cdc_subscription	A list of subscribers to change data capture events.	
Change Data Capture Trace	os_cdc_trace	Detailed information on a change data capture event.	
Ramp Control	os_ramp_control	A list of subject operations that load data into the system from the relational on ramp.	

Archiving Error and Information Logs in Omni-Insurance

All logs and system-generated information is stored in the [omnigen/OmniGenData](#) directory. The following list describes the location, purpose, and archiving recommendations of the system logs and deployment logs.

❑ System Logs

❑ Location:

```
omnigen/OmniGenData/logs/{command, controller, server,
OmniDesignerRepository, ...etc}
```

System logs are separated into subdirectories by application name.

❑ Purpose: Provides detailed records of a particular application component of the Omni-Insurance system.

❑ Archiving: In a high volume system, the controller and server log directories can grow to be quite large. These should be monitored and archived according to the requirements of the specific system (frequency and size threshold) the software is running on.

❑ **Deployment Logs**

❑ **Location:**

`omnigen/OmniGenData/deployment`

❑ **Purpose:** Provides a detailed record of the steps that occurred during a bundle deployment.

❑ **Archiving:** The deployment logs are smaller in size (less than 20kB) and do not need to be archived.

General Maintenance Recommendations for Common Database Systems

The following list describes the general maintenance recommendations for common database systems.

❑ **Oracle**

Tools: Oracle Database Resource Manager

Maintenance: Many of the generic maintenance tasks can be automated to run automatically during specific maintenance intervals.

1. **Automatic Optimizer statistics collection:** Collects optimizer statistics for all schema objects in the database for which there are no statistics or only stale statistics. The statistics gathered by this task are used by the SQL query optimizer to improve the performance of SQL execution.
2. **Automatic segment advisor:** Identifies segments that have space available for reclamation, and makes recommendations on how to defragment those segments.
3. **Automatic SQL tuning advisor:** Examines the performance of high-load SQL statements, and makes recommendations on how to tune those statements. You can configure this advisor to automatically implement SQL profile recommendations.

❑ **PostgreSQL**

Tools:

Cron scripts, Windows Task Scheduler, and `check_postgres` are available for monitoring database health and reporting unusual conditions.

Maintenance:

1. **Periodic vacuuming:** Either manual or through a daemon.

The PostgreSQL VACUUM command has to process each table on a regular basis for the following reasons:

- a. To recover or reuse disk space occupied by updated or deleted rows.
 - b. To update data statistics used by the PostgreSQL query planner.
 - c. To update the visibility map, which speeds up index-only scans.
 - d. To protect against the loss of very old data due to *transaction ID wraparound* or *multixact ID wraparound*.
2. **Update planner statistics:** Analyze
 3. **Prevention transaction ID wraparound failures:** Must be vacuumed at least once every two billion transactions.
 4. **Routine reindexing:** Reclaim space and improve performance.
 5. **Maintaining log files:** Log rotation, archiving, and deletion.

❑ SQL Server

Tools: SQL Server Management Studio (SSMS)

Maintenance:

The Maintenance Plan Wizard can be started from SSMS and can be found in the Management section of the SSMS tree. It creates scheduled jobs, which are run by the SQL Server Agent and can perform the following tasks:

1. **Reorganize index pages:** The Reorganize Index task runs the ALTER INDEX statement with the REORGANIZE option on the indexes in the selected databases. This task helps to remove index fragmentation, but does not update index and column statistics. If you use this option to remove index fragmentation, then you will also need to run the Update Statistics task as part of the same Maintenance Plan.
2. **Rebuild indexes:** The Rebuild Index task runs the ALTER INDEX statement with the REBUILD option on indexes in the selected databases, by physically rebuilding indexes from scratch. This removes index fragmentation and updates statistics simultaneously.
3. **Update statistics on the indexes:** The Update Statistics task runs the sp_updatestats system stored procedure against the tables of the selected databases, updating index, and column statistics. It is normally run after the Reorganize Index task is run. Do not run it after running the Rebuild Index task, as the Rebuild Index task performs this same task automatically.
4. **Backup database and transaction logs:** The Back Up Database (Transaction Log) task allows you to specify the databases, destination files, and overwrite options for a transaction log backup.
5. **Perform internal consistency checks:** The Internal Consistency Checks task checks data and data pages within the database to make sure that a system or software problem has not damaged data.




6. **Delete Backup and Restore History:** The History Cleanup task deletes historical data from the SQL Server database, including historical data regarding backup and restore, SQL Server Agent, and Maintenance Plans. If you do not perform this task periodically, then over time, the SQL Server database can grow very large.
7. **Cleanup tasks:** The Cleanup task allows you to define the databases for which you want to delete database task history.

DB2

1. **REORGCHK / REORG:** After many changes to table data that are caused by the insertion, deletion, and updating of variable length columns activity, logically sequential data can be located in on non-sequential physical data pages. Because of that, the database manager performs extra read operations to access data. Reorganize DB2 tables to eliminate fragmentation and reclaim space by using the REORG utility.
2. **AUTO_RUNSTATS (automatic statistics collection):** RUNSTATS (manual statistics collection).
3. **DB2 Optimizer:** Uses information and statistics in the DB2 catalog to determine the best access to the database, which is based on the query that is provided.
4. **DB2 Health Monitor:** Calculates health indicators based on data retrieval from database system monitor elements, the operating system, and the DB2 database.

Purging Old Data

Omni-Insurance systems maintain some runtime data, which diminishes over time. This data resides in a relational database as follows:

-  **Work Orders** (os_work_order)
-  **Work Order Items** (os_work_order_item)
-  **Measures** (os_measure)

The following server-based runtime settings are available to control purge events:

1. **Purge Time.** A cron-based expression that defines when to run the purge job. The default value is `0 0 3 * * ?`, which indicates every day at 3:00am. For more information on cron expressions, see the following website:
https://docs.oracle.com/cd/E12058_01/doc/doc.1014/e12030/cron_expressions.htm
2. **Purge Age.** Defines the age of data (in days) that should be purged. The default value is 30 days.

Purging Inactive Data

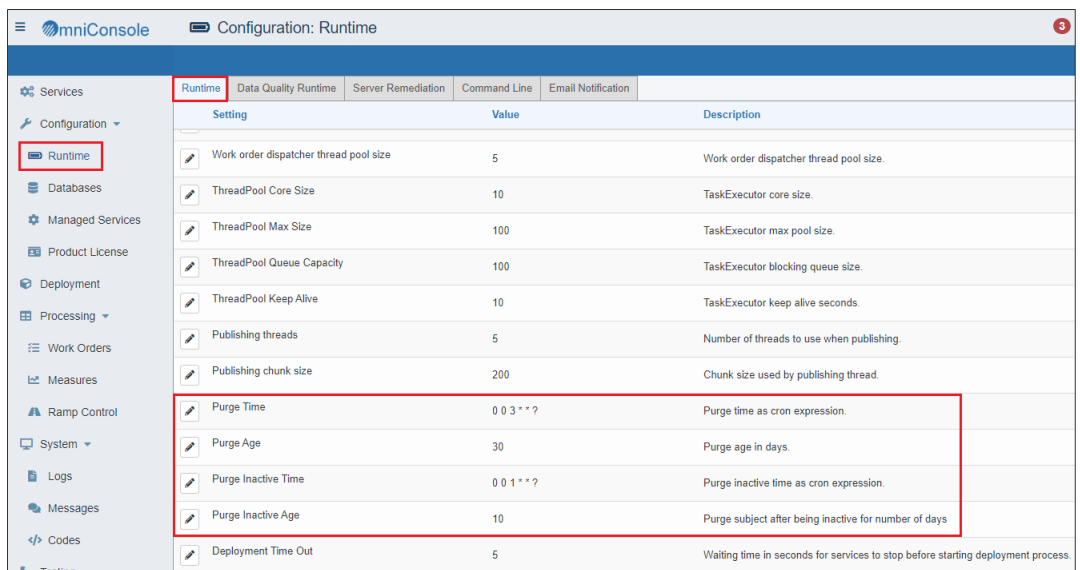
The following Omni-Insurance server-based runtime settings are available to control the purging of inactive user data:

- ❑ **Purge Inactive Time.** A cron-based expression that defines when to run a purge job for inactive data. The default value is `0 0 1 * * ?`, which indicates every day at 1:00am. When this cron-based expression evaluates as true, a `SETUP_DELETE_STALE` work order is created, and only becomes active if there are no other work orders currently in progress.

When a `SETUP_DELETE_STALE` work order becomes active, a corresponding `DELETE_STALE` work order is issued for each subject in the model that is populated with data. In a `DELETE_STALE` work order, all inactive records of a given subject that were last updated before the value specified by the *Purge Inactive Age* setting are deleted. This includes source, instance, master, history, and master-master reference records.

- ❑ **Purge Inactive Age.** Specifies the number of days after which inactive data (records) are purged. The default value is 10 days.

You can access these runtime settings from the Runtime tab, located under Configuration on the left pane of the Omni Console, as shown in the following image.



Setting	Value	Description
Work order dispatcher thread pool size	5	Work order dispatcher thread pool size.
ThreadPool Core Size	10	TaskExecutor core size.
ThreadPool Max Size	100	TaskExecutor max pool size.
ThreadPool Queue Capacity	100	TaskExecutor blocking queue size.
ThreadPool Keep Alive	10	TaskExecutor keep alive seconds.
Publishing threads	5	Number of threads to use when publishing.
Publishing chunk size	200	Chunk size used by publishing thread.
Purge Time	0 0 3 * * ?	Purge time as cron expression.
Purge Age	30	Purge age in days.
Purge Inactive Time	0 0 1 * * ?	Purge inactive time as cron expression.
Purge Inactive Age	10	Purge subject after being inactive for number of days
Deployment Time Out	5	Waiting time in seconds for services to stop before starting deployment process.

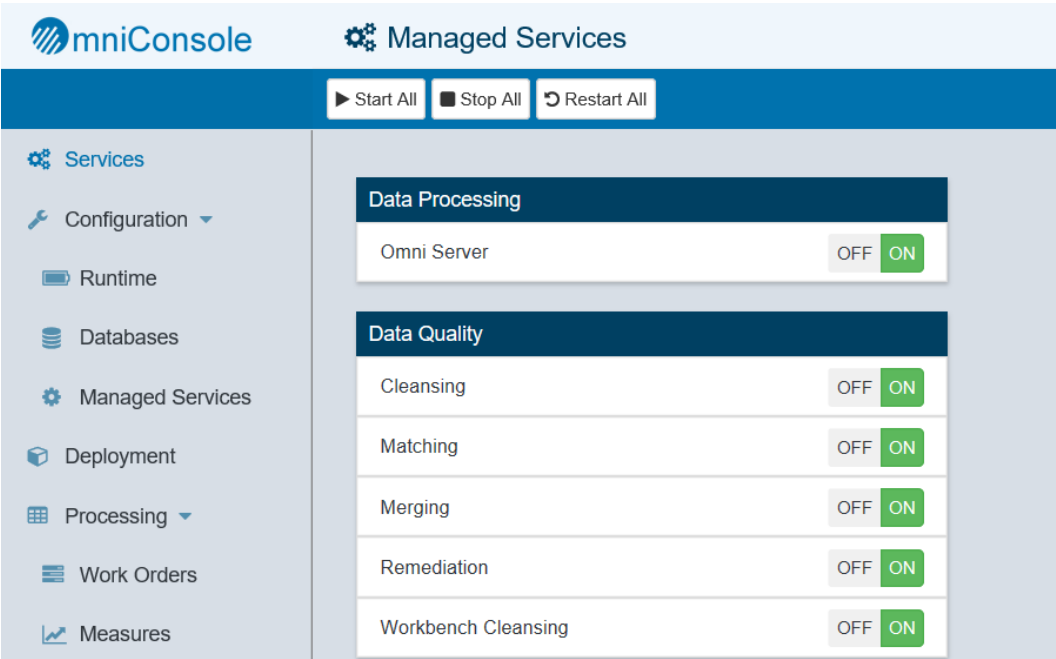
Data Quality Components

This section describes the related Data Quality (DQ) components.

Data Quality Processes

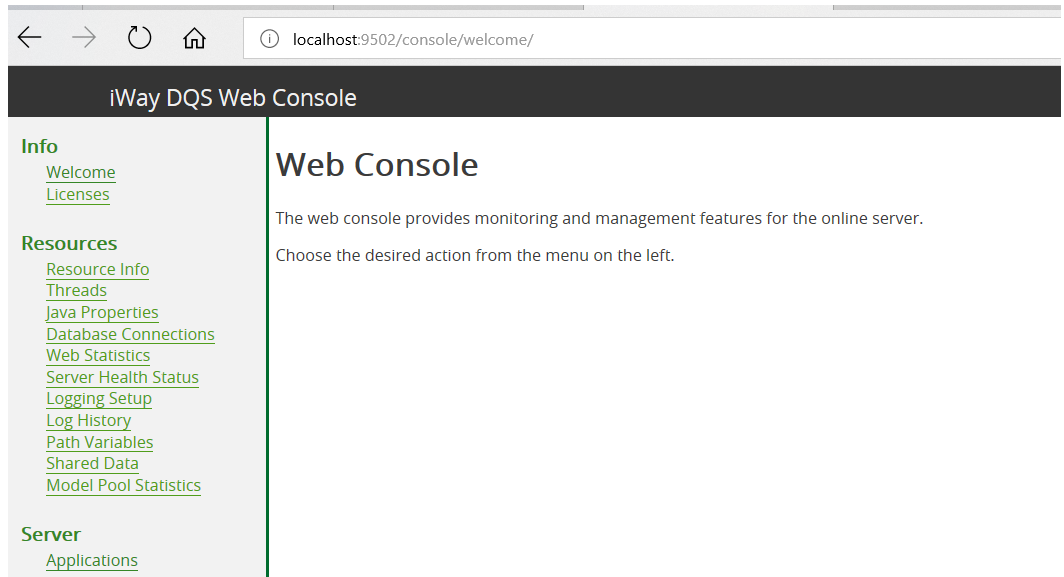
The DQ processes for Cleansing, Matching, Merging, and Remediation can be started and stopped using the Omni Console. Please note that Merging is available only for the Omni-Insurance MDM Edition and not the DQ Edition.

The services shown in the following image can be managed using the Omni Console.

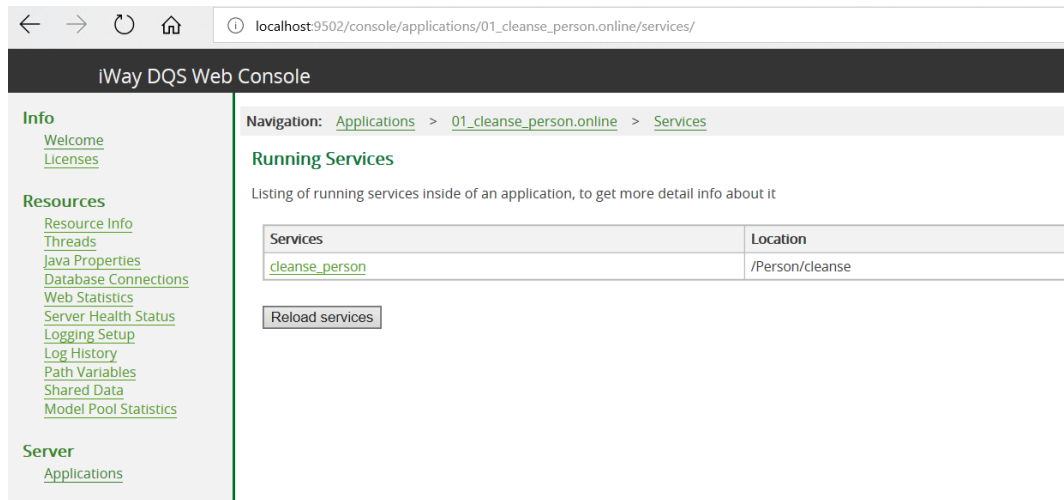


There is a link to access Data Quality Console form the Omni Console for further details. The console only shows if a DQ process has successfully been launched. It does not check if the services defined with the process have been loaded. If a deployment bundle contains an erroneous or incorrectly configured plan, then the process may start, but the service may be unable to load.

The status of the services within a DQ process can be seen in the console of the process. The console is available under the HTTP port defined for the process, for example:



The list of loaded services can be found in the Applications section.

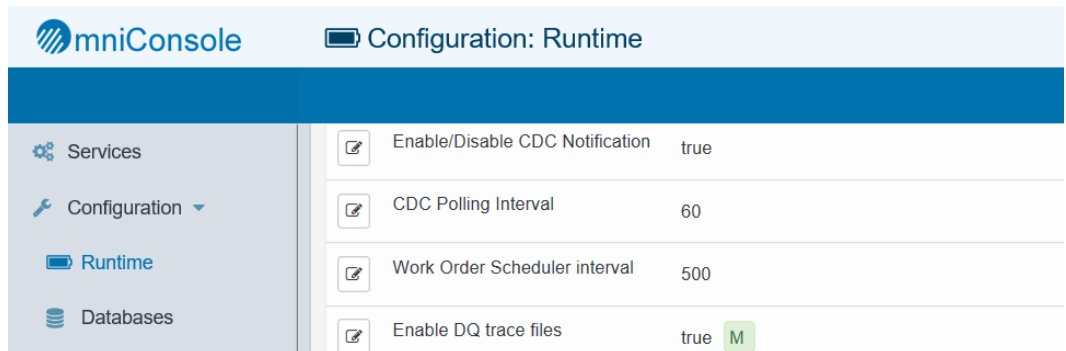


If an expected service is not listed, it generally indicates an error in the plan implementing the service. In this case, the DQ logs should indicate an error.

Logs for each of the processes are in OmniGenData/logs/dq. Each DQ process writes four logs, as described in the following table.

Log file suffix	Contents
_access	HTTP requests that are received by the server.
_perf	Execution times for service invocations.
_err	Messages that occur during execution of a plan.
_online	Messages that occur during execution of a service. Generally this duplicates the _err file.

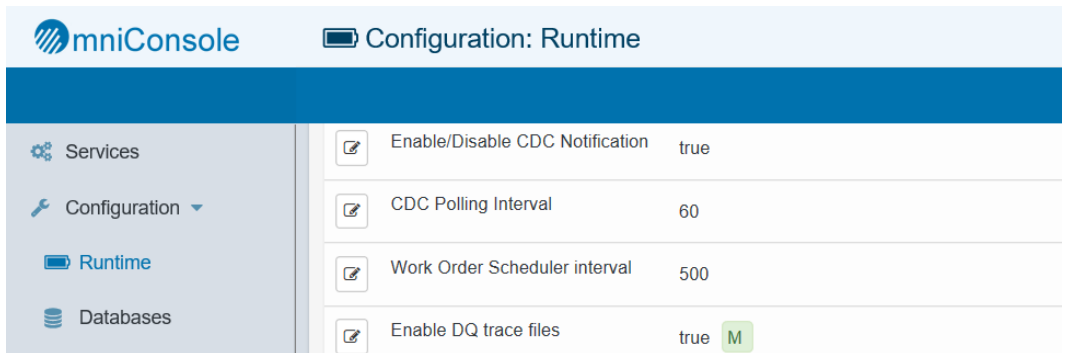
In addition, it is possible to log the data exchanged between Omni-Insurance and each DQ process by enabling the *DQ Trace* option in the Omni Console:



After enabling the option, Omni-Insurance Server must be restarted. The option should be enabled only for temporary debugging purposes on small loads. When enabled, Omni-Insurance Server will write a set of CSV files into OmniGenData/logs. Each file is named according to the DQ process, the transaction ID of the work order being executed, and "send" or "receive" to indicate whether the file contains data sent from Omni-Insurance to DQ or received by Omni-Insurance from DQ.

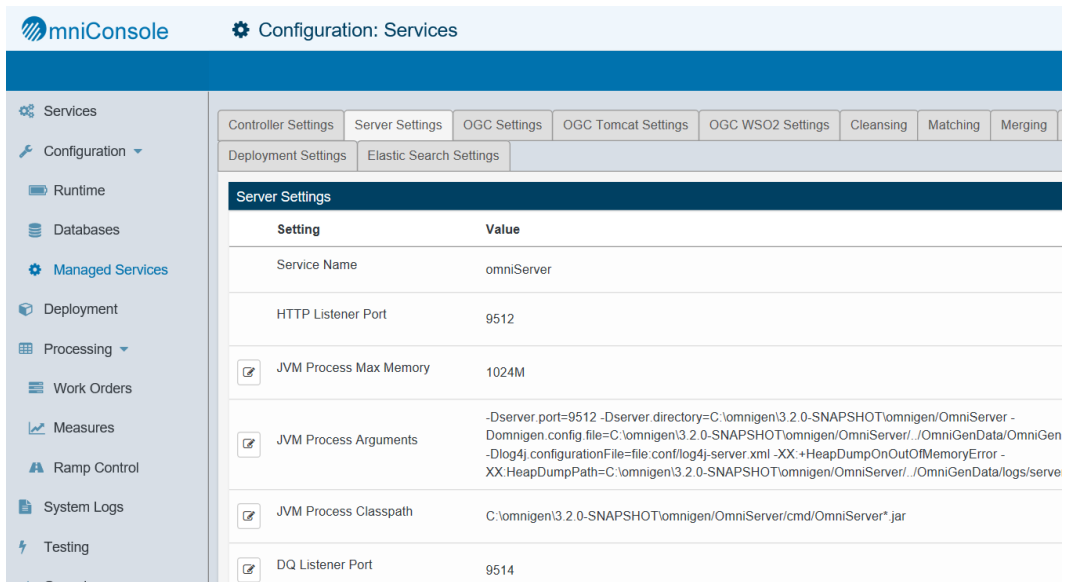
Configuration Options

The HTTP listener port and JVM properties for each process can be modified in the console in the appropriate tab under *Managed Services*.



Setting	Value
Enable/Disable CDC Notification	true
CDC Polling Interval	60
Work Order Scheduler interval	500
Enable DQ trace files	true M

The TCP port used by Omni-Insurance to send and receive data to executing DQ plans – the DQ Listener Port – is defined under *Server Settings* in the console. This is not an HTTP port and should never be opened in a browser or by any program except the plugin components embedded within a DQ plan.

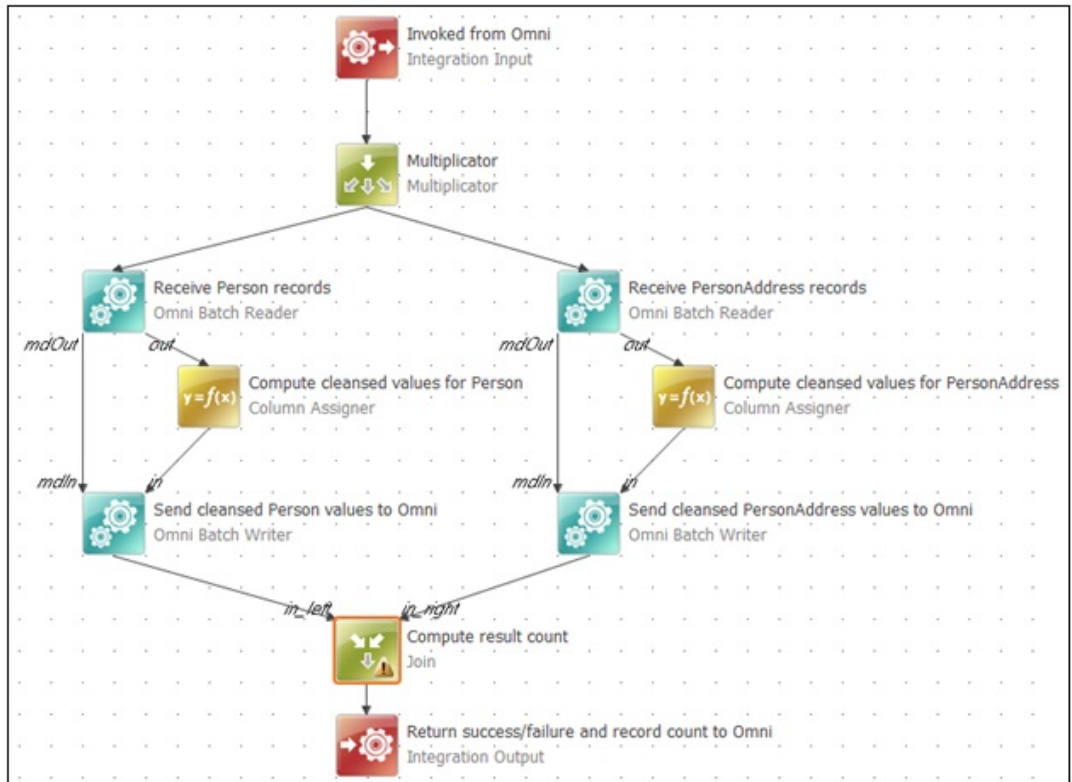


Setting	Value
Service Name	omniServer
HTTP Listener Port	9512
JVM Process Max Memory	1024M
JVM Process Arguments	-Dserver.port=9512 -Dserver.directory=C:\omnigen\3.2.0-SNAPSHOT\omnigen\OmniServer -Domnigen.config.file=C:\omnigen\3.2.0-SNAPSHOT\omnigen\OmniServer\..\OmniGenData\OmniGen -Dlog4j.configurationFile=file:conf/log4j-server.xml -XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=C:\omnigen\3.2.0-SNAPSHOT\omnigen\OmniServer\..\OmniGenData\logs\serve
JVM Process Classpath	C:\omnigen\3.2.0-SNAPSHOT\omnigen\OmniServer\cmd\OmniServer*.jar
DQ Listener Port	9514

It is not recommended that any of these settings be modified, but if they are modified then the DQ process and the Omni-Insurance Server process should be restarted.

Data Quality Processing

The following diagram illustrates the process flow within an Omni-Insurance DQ plan. This example describes a cleansing process. Matching, merging, and remediation process flows are similar.



The general flow is:

1. Omni-Insurance invokes a DQ REST service that is linked to a Cleansing, Matching, Merging, or Remediation plan. Omni-Insurance provides the subject and the transaction id associated with the work order being processed.
2. The plan executes, in parallel, branches for the root subject and each of its subcollections.
3. Each branch begins with an "OmniBatchReader" component. Each reader opens a TCP connection to the Omni-Insurance DQ server and requests a set of columns. Omni-Insurance then streams the requested data through the connection and the reader sends the records into the plan for processing.

4. After a record is processed, it is sent to an "OmniBatchWriter" component. Upon first access, the component sends to Omni-Insurance the set of entity attributes which it will send. It then streams the input records it receives from the plan back to Omni-Insurance through another TCP connection.
5. When all branches have read, processed, and written, all the records provided by Omni-Insurance, a count of all the records processed is computed and returned as the result of the REST invocation.

DQ Process Activities

The following table describes the data sent and received in each of the DQ processes.

Process	Sent	Received
cleansing	Cleansing overrides and source records associated with the work order	Cleaned values. Instance records are updated.
matching	Instance records associated with the work order	Ids, master ids, and match quality values for all root subject instances affected by the plan execution. This may be a super set of the instance records sent into the plan.
merging	Master ids and match quality values for all root subject records affected by the matching results	A set of master root subject records and all the subcollection records associated with them. Master root subject records are inserted or updated. Any existing subcollection records associated with the root subject masters are deleted and the new subcollection records are inserted.
remediation	Instance records associated with the work order	Cleansing and matching tickets. Inserted into omni_remediation_ticket.

Warnings and Errors

This section describes warnings and errors with related workarounds to resolve the issue.

Server fails to connect

If Omni-Insurance Server is unable to connect to a DQ process, OmniGenData/logs/server/server.log will contain a "Not Found for URL" message such as:

```
com.ibi.omni.server.services.ServiceException: Not Found for
URL http://localhost:9502/Person/cleanse
```

The most likely cause is that the plan failed to load due to an error in the plan definition. This can be verified by loading the DQ console for the process and checking that the referenced service is available. A new deployment bundle will have to be generated with a corrected plan.

Invalid name warnings

When a reader component in a plan requests a column that is undefined, a warning message is generated in OmniGenData/logs/server/server.log as for example:

```
WARN com.ibi.omni.cleanse.CleansingSender:64 [] [] Requested
column ssn not available in entity Person
```

Omni-Insurance will allow the plan to continue execution, however requested value will not be transmitted to the DQ process and the DQ process log will also include a warning, as for example:

```
<message>[306] ssn not sent by omni</message>
```

Similarly, if a writer component in a plan indicates to Omni-Insurance that it will write a column whose name is not recognized by Omni-Insurance, a "No field found" warning message is generated:

```
WARN com.ibi.omni.cleanse.CleansingReceiver:59 [] [] No field
found for Person.firstName
```

Process Failure

Omni-Insurance Server errors that occur while processing DQ streams are logged in the server log. If the DQ plan is still receiving data, an error message is also sent to the executing DQ plan, causing it to abort. The DQ plan will log the error message it receives from Omni-Insurance and also log its own failure message, which is typically just:

```
com.ataccama.dqc.online.core.RuntimeErrorReporterException: Configuration
execution failed.
```

If an error occurs within the DQ plan itself, it will attempt to send an error message to Omni-Insurance Server. Omni-Insurance Server will log this as a com.ibi.omni.dq.ReceivedErrorException and to stop all active senders and receivers.

Deployment Operations

This section provides an overview of the deployment operations, which are available in the Deployment section of the Omni Console.

Install / Replace Bundle

- ☐ Opens a file explorer dialog to choose a bundle to deploy.
- ☐ The option is labeled *Install Bundle* or *Replace Bundle* if an existing bundle is available.

Note: If a database exists, then it will be cleaned and all data will be lost.

Update Bundle

- ☐ Allows you to select a bundle from the file system.
- ☐ It is expected to be a derivative of the current bundle. For example, a new column/attribute or a plan change.

Note: The database is not cleaned.

Reset Environment

- ☐ Resets the environment back to the state it was in when last deployed.

Note: This is a destructive operation and should be used with caution.

Work Orders

A work order processes data for a given subject through a series of work order items. A work order is associated with a transaction ID. Records marked with the work order transaction ID will be included in subsequent processing.

Work Order Types

The following list describes the types of work orders.

- ☐ **BULK.** In a BULK work order, data is loaded into Omni-Insurance from the ramp tables for a given subject. This work order is created when an `os_ramp_control` record is processed or when an Omni Interface Document (OID) is loaded from the File Input Location.
- ☐ **IMMEDIATE.** In an IMMEDIATE work order, data is loaded into Omni-Insurance from an Omni Interface Document using the REST API service.

The following syntax shows the REST API service:

`/server/processInstance`

- ❑ **RELOAD.** In a RELOAD work order, master records are recreated for select instance records of a given subject. This work order is created through a MASTER_REFERENCE_RELOAD work order item.

Relevant work order items include:

- ❑ SET_RELOAD_TRANSACTION
 - ❑ MERGE
 - ❑ CLEAR_RELOAD_QUEUE
 - ❑ HISTORY_MASTER (optional)
 - ❑ CDC_RECORD (optional)
- ❑ **MASTER_PLAN_CHANGE.** In a MASTER_PLAN_CHANGE work order, all data for a given subject is reprocessed from the source tables. This work order is created using the Reprocess subject option on the Deployment screen in the console.

The following syntax shows the REST API service:

```
/server/quality/reprocess/{subject}
```

- ❑ **OVERRIDE_RELOAD.** In an OVERRIDE_RELOAD work order, an instance record is updated with specific value(s) and reprocessed beginning with CLEANSE. This work order is initiated through a request from OGC.

The following syntax shows the REST API service:

```
/remediation/PropertyOverride
```

- ❑ **MANUAL_MASTER_OVERRIDE.** In a MANUAL_MASTER_OVERRIDE work order, records in a match override are reprocessed beginning with MATCH. This work order is initiated through a request from OGC.

The following syntax shows the REST API service:

```
/remediation/MatchOverride
```

- ❑ **SUBJECT_GROUP_PROCESS.** In a SUBJECT_GROUP_PROCESS work order, data is generated for subject groups. For more information, see [SUBJECT_GROUP_PROCESS](#) on page 64.

The following syntax shows the REST API service:

```
/server/subjectGroups
```

- ❑ **LOAD_ODS.** In a LOAD_ODS work order, consumption tables are populated with data from instance or master records. This work order is initiated using the Consumption View console.

The following syntax shows the REST API service:

```
/consumption/work/{ods}
```

- ❑ **CLEANSE_PLAN_CHANGE.** In a CLEANSE_PLAN_CHANGE work order, records of a given subject are processed using CLEANSE.

The following syntax shows the REST API service:

```
/server/quality/cleanse/{subject}
```

- ❑ **MATCH_PLAN_CHANGE.** In a MATCH_PLAN_CHANGE work order, records of a given subject are processed using MATCH and MERGE.

The following syntax shows the REST API service:

```
/server/quality/match/{subject}
```

- ❑ **READ_ONLY_MATCH.** This is used to find a master record based on specific data.

The following syntax shows the REST API service:

```
/server/master?responseType=oid|masterId
```

- ❑ **ADTEvent.** In an ADTEvent work order, non-IDS xml that represents ADT (HL7) is translated into an OID. This is relevant only to OHD.

- ❑ **SETUP_DELETE_STALE/DELETE_STALE.** A SETUP_DELETE_STALE work order is created at the Purge Inactive Time (Runtime setting), an interval specified as a cron expression. It will only become ACTIVE if there are no other work orders in progress. When a SETUP_DELETE_STALE work order becomes ACTIVE, a DELETE_STALE work order will be created for each subject in the model that is populated with data.

In a DELETE_STALE work order, all INACTIVE records of a given subject that were last updated before the Purge Inactive Age are deleted. This includes source, instance, master, history, and master-master reference records. The Purge Inactive Age is a Runtime setting to specify the number of days after which INACTIVE records will be deleted.

❑ **RESET_SUBJECT.** In a RESET_SUBJECT work order, all data for a given subject is purged from the system. This work order is created using the Reset button on the Deployment screen in the console. It will only become ACTIVE if there are no other work orders in progress. All tables associated with the subject are truncated, including ramp, source, instance, master, and history tables. Other records such as remediation cases and tickets, overrides and master-master, and code-field references are deleted. If applicable, Elastic indexes are dropped and recreated. For a mastered subject, the DQ match index is reinitialized as follows:

- ❑ The Matching service is stopped.
- ❑ The DQ index tables (repos_subject_*) are dropped.
- ❑ The Matching service is restarted, which in turn, recreates the tables.
- ❑ REST API service:

`/api/v1/server/reset/{subject}`

Work Order States

The following list describes the work order states.

- ❑ **NEW.** The work order is created, but not ready to process until all work order items are added. This state is managed in general code.
- ❑ **READY.** The work order is ready to be scheduled. Generally, a subject-based work order will remain in READY state until prior work orders of this subject are complete. This state is managed in general code.
- ❑ **SCHEDULED.** This is a temporary state, set by the scheduler, indicating the work order will be executed immediately.
- ❑ **ACTIVE.** The work order is actively running. The first step of every work order is the START work order item, which moves the work order to ACTIVE state.
- ❑ **PAUSED.** The Pause console menu item on the Work Orders screen can be used to pause an ACTIVE work order. Note that the work order will not go into PAUSED state until the currently executing work order item can safely come to a stop. A PAUSED work order must be resumed using the Resume console menu item. The work order will then resume processing from the last active work order item.

- ❑ **COMPLETE.** The work order has finished executing. The result can be PASS or FAIL. A failed work order must be restarted or ignored to proceed. The Restart console menu item on the Work Orders screen can be used to restart work order processing beginning with the failed work order item. The Ignore console menu item will set the result of the work order to IGNORE and allow the next READY work order of this subject to be scheduled.

Work Order Items

This section lists and describes the work order items.

START

In this step, the work order state moves from SCHEDULED to ACTIVE. This is the first item in every work order.

RAMP_TO_SOURCE

The following processes describe the steps involved in the RAMP_TO_SOURCE work order item, which is invoked for mastered or cleansed subjects.

- ❑ **Ramp Processing.** In this step, data of a subject is copied from ramp tables to source tables based on a ramp batch ID and optionally, a source name. Each subject collection is copied independently on its own thread. The source record and/or its transaction ID and Omni-Insurance modified date are updated if the ramp data differs from the source data (business fields only), if there is a change in the status of the record, or if the ChangeDetection IGNORE ramp load option is specified. The ramp load policy can cause a record status change. The default ramp load policy is UPSERT. The default record status is ACTIVE and it can become INACTIVE under REPLACE or DELETE ramp load policies.

When all data in the ramp batch has been processed, a root source record not yet in the transaction will be added to the transaction if any of its collection items have the current transaction ID.

For more information on the Ramp Load policy, see the *Omni-Insurance Integration Services User's Guide*.

- ❑ **Code Processing.** All unique source codes are collected in memory during ramp processing. `checkForMissingCodes` determines which of those codes are new. A ramp batch is created for the new codes and a `SourceCodeSet BULK` work order is submitted for processing.

A list of codes and the fields which reference those codes are also collected in memory during ramp processing. `checkForMissingCodeXRefs` identifies the new code-field references and adds them to the `os_source_code_xref` table.

Note: When a SourceInstanceID is trimmed of leading and trailing whitespace, a warning message is logged. If the trimmed SourceInstanceID contains a space or a colon, the record will not be moved to the source table and an error is logged.

Parent records are not auto-generated for orphan records.

TXN_ON_SOURCE and ON_SEL_SOURCES

The following steps describe TXN_ON_SOURCE and TXN_ON_SEL_SOURCES.

- ❑ **TXN_ON_SOURCE.** This step in a MASTER_PLAN_CHANGE work order sets the transaction ID on all the source records for the subject.
- ❑ **TXN_ON_SEL_SOURCES.** This step in an OVERRIDE_RELOAD work order sets the transaction ID on select source records.

SOURCE_TO_MODEL

This section describes the step involved in the SOURCE_TO_MODEL work order item, which is invoked for mastered or cleansed subjects.

Source Processing

In this step, data of a subject is copied from the source tables to the instance tables based on the transaction ID. Each subject collection is copied independently on its own thread. The instance record and/or its transaction ID and Omni-Insurance modified date are updated if the source data (business fields only) or status differs from the instance data or status, or if the ChangeDetection IGNORE ramp load option was specified.

When all source data in the transaction has been processed, a root instance record not yet in the transaction will be added to the transaction if any of its collection items have the current transaction ID.

The instance data in the transaction is now pre-cleansed.

RAMP_TO_MODEL

The following processes describe the steps involved in the RAMP_TO_MODEL work order item, which is invoked for non-mastered, non-cleansed subjects.

- ❑ **Ramp Processing.** In this step, data of a subject is copied from ramp tables to instance tables based on a ramp batch ID and optionally, a source name. Each subject collection is copied independently on its own thread. The source record and/or its transaction ID and Omni-Insurance modified date are updated if the ramp data differs from the source data (business fields only), if there is a change in the status of the record, or if the ChangeDetection IGNORE ramp load option is specified. The ramp load policy can cause a record status change. The default ramp load policy is UPSERT. The default record status is ACTIVE and it can become INACTIVE under REPLACE or DELETE ramp load policies.

For more information on the Ramp Load policy, see the *Omni-Insurance Integration Services User's Guide*.

- ❑ **Code Processing.** All unique source codes are collected in memory during ramp processing. checkForMissingCodes determines which of those codes are new. A ramp batch is created for the new codes and a SourceCodeSet BULK work order is submitted for processing.

A list of codes and the fields which reference those codes are also collected in memory during ramp processing. checkForMissingCodeXRefs identifies the new code-field references and adds them to the os_source_code_xref table.

Note: When a SourceInstanceID is trimmed of leading and trailing whitespace, a warning message is logged. If the trimmed SourceInstanceID contains a space or a colon, the record will not be moved to the source table and an error is logged.

Parent records are not auto-generated for orphan records.

MASTER_REFERENCE

This step only applies to mastered subjects. The instance records in the transaction are traversed for other mastered instances that are referred to by the participating instances. A record is inserted into the os_master_reference table for each new mastered instance reference. These records are used by the FILL_RELOAD_QUEUE work order item in preparation for a RELOAD work order.

CLEANSE

In this step, data for a given subject is cleansed. For a non-mastered subject, this step will only be included if the cleanse attribute in the IDS is true.

The DQ Cleansing service is called to start processing. DQ calls back to Omni-Insurance Server to retrieve the data to cleanse, identifying the fields in the Cleansing service definition.

- ❑ **Cleansing Sender.** ACTIVE source records with the current transaction ID are sent to the Cleansing service. For code fields requested by the Cleanse plan, if the source code is mapped to a standard code, the standard code is transmitted. Otherwise, the source code is transmitted.

The Cleansing service runs the rules defined in the Cleanse plan and returns the cleansed data. The set of values returned is independent of the values that were sent, though generally there is a correlation.

- ❑ **Cleansing Receiver.** The instance record is updated with the cleansed value if it differs from its current value, and the meta_tags column is updated with the tags returned by the service or with an empty representation. The transaction ID and Omni-Insurance modified date are also updated. The Cleansing service generally does not return code values so the source code is the value persisted on the instance record.
- ❑ **Overrides.** Property overrides, performed by an OVERRIDE_RELOAD work order, are applied during the CLEANSE process. This includes overrides of fields that are not defined in the cleanse plan. If the field is defined in the cleanse plan, the cleansed value supersedes the override value.

TXN_ON_INSTANCE and ON_SEL_INSTANCES

The following steps describe TXN_ON_INSTANCE and TXN_ON_SEL_INSTANCES.

- ❑ **TXN_ON_INSTANCE.** This step in a MASTER_PLAN_CHANGE work order sets the transaction ID on all the instance records for the subject.
- ❑ **TXN_ON_SEL_INSTANCES.** This step in an OVERRIDE_RELOAD or MANUAL_MASTER_OVERRIDE work order sets the transaction ID on select instance records.

MATCH

In this step, records of a mastered subject are processed by match rules to generate the master ID. The DQ Matching service is called to start processing. DQ calls back to Omni-Insurance Server to retrieve the data to use in the match plan, identifying the fields in the Matching service definition.

- ❑ **Matching Sender.** ACTIVE and INACTIVE instance records with the current transaction ID are sent to the Matching service.

The Matching service runs the rules defined in the Match plan and returns the results. The number of records received may be larger than the number of records sent because all instances of an affected master ID are returned.

- ❑ **Matching Receiver.** The instance record is updated with results of the match plan, as follows:
 - ❑ **master_id.** Current master_id last assigned to this instance by the matching engine.
 - ❑ **prev_master_id.** Previous master_id assigned to this instance by the matching engine. In the event that the instance changed its master during the current work order, this data element is used to determine whether to re-merge the remaining instances assigned to prev_master_id, or *soft delete* the master from the system in the event that there are no instances tied to it.
 - ❑ **match_rule.** Human readable string describing the matching rule used in identifying the master_id. Values will be determined by the mastering resource implementing the Matching Rules.
 - ❑ **match_candidate_id.** Identifier of the *pre-matching* group, defined during the unification step in the matching plan.
 - ❑ **match_quality.** Contains digital representation of the match quality (how good the match was). In this case, the lower the overall score, the better the match becomes.
 - ❑ **match_quality_detail.** Additional supporting metadata for match_quality.
 - ❑ **match_role.** The role that the instance record played in the match. The following list describes the valid values:
 - ❑ **N.** Record has no regular key for candidate grouping.
 - ❑ **M.** Best record of one candidate group. It represents the center of the initially established matching group within the candidate group.
 - ❑ **I.** Next selected centers of other matching groups in the candidate group.
 - ❑ **S.** Slaves or records similar to some center and attached to its matching group.
 - ❑ **R.** Renegades or records not similar to any center in a candidate group.
 - ❑ **transaction_id.** Affected instance records not previously in the transaction.

If there is a change to the master ID of an instance, any instances of the previous master ID are added to the transaction. This ensures the complete set is provided to the MERGE process.

- ❑ **Overrides.** Match overrides, performed by a `MANUAL_MASTER_OVERRIDE` work order, are applied during the `MATCH` process.
- ❑ **Whitelist.** A space delimited string of instance IDs that should be placed in the same match group, though they may not be considered a match by the match rules.
- ❑ **Blacklist.** A space delimited string of instance IDs that should never be placed in the same match group, though they may be considered a match by the match rules.

A white or black list is maintained for each participating instance on a white or black list, referencing the other list participants.
- ❑ **Pre-mastered survivor contract.** An agreement that effectively whitelists items from the source system. This does not involve a `MANUAL_MASTER_OVERRIDE` work order, but is specified when data is loaded from the ramp.

MATCH_SET_INACTIVE

This step will run for a mastered subject if the Enable/Disable Match Post Processing for Inactive runtime setting is true. Turning off this feature may result in unreliable master data.

The status of an `ACTIVE` master record is set to `INACTIVE` if it has no active instances. The `prev_master_id` of instance records in the transaction is used to optimize this process.

MATCH_SET_DELETE

This step will run for a mastered subject if the Enable/Disable Match Post Processing for Delete runtime setting is true.

The status of a master record is set to `DELETE` if it is in a `PREDELETE` state and has no active instances.

FILL_RELOAD_QUEUE

In this step, the `os_reload_queue` table is populated with all mastered instances that are referenced by a mastered instance in the current transaction, as found in the `os_master_reference` table, in preparation for a `RELOAD` work order.

The following list shows the related work order items:

- ❑ `MASTER_REFERENCE`
- ❑ `MASTER_REFERENCE_RELOAD`

MERGE

In this step, master records are created for a given subject. The DQ Merging service is called to start processing. DQ calls back to Omni-Insurance Server to retrieve the data to use in the Merge plan, identifying the fields in the Merging service definition (all fields, plus the match response fields).

- ❑ **Merging Sender.** ACTIVE instance records with the current transaction ID are sent to the Merging service. For code fields, if the source code is mapped to a standard code, the standard code is transmitted. Otherwise, the source code is transmitted. For link fields, the master ID of the referenced instance record is transmitted. If it does not exist, an empty string is sent.

The Merging service runs the rules defined in the Merge plan and returns the master record(s).

- ❑ **Merging Receiver.** All previously existing master records returned by the Merge plan are updated and new master records are created. New collection items are created and previous records are deleted afterwards. For code fields, the standard code is persisted to the master record. For link fields, the master ID of the referenced record is persisted.

PROMOTE_MASTER

In this step, instructions are applied to promote master collection data to the master record root.

REMEDiate

In this step, remediation tickets are created for a given subject. For a non-mastered subject, this step will only be included if the remediate attribute in the IDS is true.

The DQ Remediation service is called to start processing. DQ calls back to Omni-Insurance Server to retrieve the data to use in the remediation plan, identifying the fields in the Remediation service definition (all fields, plus the match response fields).

- ❑ **Remediation Sender.** ACTIVE instance records with the current transaction ID are sent to the Remediation service.

The Remediation service runs the rules defined in the Remediation plan and returns cleansing and/or matching remediation tickets.

- ❑ **Cleansing Ticket Receiver.** The Remediation plan returns the following attributes:
 - ❑ ID of the instance record to which the remediation ticket applies.

- ☐ Name of the field whose value requires remediation.
- ☐ Reason code.
- ☐ Reason code description.
- ☐ Severity.
- ☐ Subject or entity to which the remediation ticket applies.

The values of the above attributes are used to build the cleansing ticket records, which are continued to the omni_remediation_ticket table.

☐ **Matching Ticket Receiver.** The Remediation plan returns the following attributes:

- ☐ Subject to which the remediation ticket applies.
- ☐ Reason code.
- ☐ Reason code description.
- ☐ References to instance records to which the remediation ticket applies.
- ☐ Severity.

The values of the above attributes are used to build the matching ticket records, which are persisted to the omni_remediation_ticket table along with the master ID to which the remediation ticket applies.

Related work order item: AUTO_CLOSE

AUTO_CLOSE

This step determines if any open remediation tickets can be closed. It also updates the status of tickets to indicate they are ready to be sent to the OGC Remediation Service.

All remediation tickets of instances or master records in the current transaction that are not in a closed state and were not updated in the current transaction will be closed. If the ticket has not yet been sent to the OGC Remediation Service, it is closed directly. Otherwise, the ticket status and destination are updated and the ticket will be closed by the TicketOutboundService after it is sent to OGC.

Remediation tickets from the omni_remediation_ticket table are sent to OGC in a background process.

Remediation Ticket States

- ❑ **PENDING_NEW.** Initial status of remediation tickets created in the REMEDIATE step.
- ❑ **PENDING_CURRENT.** Intermediate state for PENDING tickets updated in the REMEDIATE step of the current transaction.
- ❑ **PENDING.** Set by the AUTO_CLOSE process for tickets in PENDING_NEW or PENDING_CURRENT state. It is also set for ACK tickets to be closed. Tickets in this state will be sent to the OGC Remediation Service.
- ❑ **ACK_CURRENT.** Intermediate state for ACK tickets updated in the REMEDIATE step of current transaction.
- ❑ **ACK.** Set by the AUTO_CLOSE process for tickets in ACK_CURRENT state. Tickets in this state were already sent to the OGC Remediation Service.

ACK is set by the TicketOutboundService after successful transmission of an open ticket to the OGC Remediation Service.

- ❑ **CLOSED.** Set by the AUTO_CLOSE process if the PENDING ticket is repaired in the current transaction.
- ❑ **ERROR.** Set by the TicketOutboundService when a failure occurs in the transmission of a ticket to the OGC Remediation Service.

CLOSED is set by the TicketOutboundService based upon response from the OGC Remediation Service of a close ticket request.

HISTORY

The following steps will be included if the captureHistory attribute in the instance and/or master IDS is true.

- ❑ **HISTORY_INSTANCE.** All instance records in the current transaction are copied to the corresponding history table. A snapshot of the entire record is recorded in history. Each entity is copied independently on its own thread.
- ❑ **HISTORY_MASTER.** All master records in the current transaction are copied to the corresponding history table. A snapshot of the entire record is recorded in history. Each entity is copied independently on its own thread. History records of collections are not related to each other by ID because master records are recreated in entirety every time.
- ❑ **HISTORY_INSTANCE_MASTER.** This step runs when history is enabled for both instance and master records of a mastered subject.

PUBLISH

In this step, a JSON representation of records along with statistics is compiled and sent to the Elastic index. For the `_data` index, the entire record is replaced. For the `_hist` index, the new record is added, and the `_endDate` is set on the previous version of the record.

- ☐ **PUBLISH_SOURCES.** All source records in the current transaction are processed.
- ☐ **PUBLISH_INSTANCES.** All instance records in the current transaction are processed.
- ☐ **PUBLISH_MASTERS.** All master records in the current transaction are processed.

MASTER_REFERENCE_RELOAD

In this step, a RELOAD work order is created for `os_reload_queue` records added by the `FILL_RELOAD_QUEUE` step in the current transaction. A separate RELOAD work order is created for each relevant subject.

Related work order items: `FILL_RELOAD_QUEUE`, `MASTER_REFERENCE_RELOAD`

SET_RELOAD_TRANSACTION

In this step, the transaction ID of instance records is updated based on the contents of the `os_reload_queue` table for a particular RELOAD work order.

CLEAR_RELOAD_QUEUE

In this step, records are deleted from the `os_reload_queue` table after the corresponding instance records have been processed by the RELOAD work order.

CDC_RECORD

This step runs only if the Enable/Disable CDC Notification runtime setting is true.

The `os_cdc_change` table is populated with the XML representation (Omni Interface Document) of instance and/or master records in the current transaction for which a CDC subscription exists.

SUBJECT_GROUP_PROCESS

This step runs in a `SUBJECT_GROUP_PROCESS` work order. A `SUBJECT_GROUP_PROCESS` work order is launched only under the following conditions:

- ☐ The `mastering/services/relationships` directory exists.
- ☐ The server has been idle for the Work Order Scheduler idle interval (default is 5 minutes) after a successful MERGE in a completed BULK or `MASTER_PLAN_CHANGE` work order.

- ❑ A SUBJECT_GROUP_PROCESS work order has not been executed since the last successful MERGE in a completed BULK or MASTER_PLAN_CHANGE work order.

Subject Group Process

The /relations endpoint of the DQ Merging service is invoked, which runs a DQ plan that generates data for the os_subject_group_relations table in its entirety (data from previous iterations is deleted).

STOP

In this step, the status, result type, and end date of the work order are updated. This is the last item in every work order.

Work Order Automation

The ability to automate a sequential set of batches to process was added to Omni-Insurance.

The following is a high-level summary of the process:

1. The integrator creates a series of batches to process with batch control records.
2. Using the batch IDs, create a work order automation JSON document that includes entries for each batch to process.
3. Copy and paste the JSON document into the work order Swagger API.
4. Click *Try it out* to test the work order.
5. View the automation executing on the console work order screen.

Executing the Automation

The automation job is a work order of its own. It is composed of a series of work order items that may also create other work orders. With this in mind, you are able to track the progress of the automation from the work order processing screen.

Each automation work order item will be attempted until one fails or the work is complete. This is no different than regular work order processing.

New Functions

To support automation, the following new functions were added:

- ❑ **START_BATCH.** The start batch takes batch the information and starts the batch as a BULK work order. In the work order processing screen, the batch will be visible as normal and the batch parameters are provided in the *jsonContext* aggregate.

WAIT_FOR_ALL_COMPLETE. Using the automation work order as a starting point, this function polls the work order table looking for any new work orders to complete. It will wait a configurable number of iterations with configurable *n* second pause between iterations. If any subsequent work order fails, then the automation work order will also fail. This function will also wait for an additional duration, making sure no other work comes into the system. This is also a configurable wait. This combination is intended to support not only the immediately created batch, but also any work orders it may create, for example a *RELOAD*.

The following image shows how work orders appear in the console processing screen.

Work Orders							
	Type	Subject	Batch Id	Source	Status	Result	Reason
+	BULK	Pet	e24f889a-bc4d-473b-85e3-3cd97d05d542	TestSource	COMPLETE	PASS	2018-01-31 16:08:05.966
+	BULK	Car	7c3cd1c9-95bc-431a-a6d0-c09b08baa3c5	TestSource	COMPLETE	PASS	2018-01-31 16:06:40.961
-	USER_DEFINED	dynamic			COMPLETE	PASS	2018-01-31 16:06:40.424
	START	dynamic			COMPLETE	PASS	2018-01-31 16:06:40.419
	START_BATCH	dynamic			COMPLETE	PASS	2018-01-31 16:06:40.435
	WAIT_FOR_ALL_COMPLETE	dynamic			COMPLETE	PASS	2018-01-31 16:06:40.561
	START_BATCH	dynamic			COMPLETE	PASS	2018-01-31 16:08:05.658
	WAIT_FOR_ALL_COMPLETE	dynamic			COMPLETE	PASS	2018-01-31 16:08:05.757
	STOP	dynamic			COMPLETE	PASS	2018-01-31 16:09:30.860

Creating an Automation

You are required to use a text editor and a Swagger interface for automation purposes.

Automation creates a work order and corresponding work order items from the provided JSON document. The tags match the role and purpose of the tags in their respective tables.

The following syntax shows a sample automation JSON document.

```

{
  "sourceType": "USER_DEFINED", <-- Required.
  "subject": "dynamic", <-- Required, value up to user, should not be the
name of a subject.
  "sourceName": "devops", <-- Required, value up to user.
  "omniSystemType": "SERVER", <-- Only SERVER supported at the moment.
  "workOrderItem": [
    {
      "workOrderItemName": "START_BATCH", <-- Required as is.
      "workOrderItemNameExtension": "car-batch", <-- Used to
differentiate different batches.
      "processorOrder": 1, <-- Required and must be sequential across
work order items.
      "jsonContext": { <-- These are the parameters passed in to
starting the batch.
        "subject": "Car",
        "batchId": "abb5bca6-0565-4344-b85a-fa975456ec11",
        "sourceSubType": "MERGE_PRESERVE_ON_NULL",
        "source": "TestSource"
      }
    },
    {
      "workOrderItemName": "WAIT_FOR_ALL_COMPLETE", <-- This task has
an 1 min idle time, be patient.
      "processorOrder": 2 <-- Make sure to increment this.
    },
    {
      "workOrderItemName": "START_BATCH",
      "workOrderItemNameExtension": "pet-batch",
      "processorOrder": 3,
      "jsonContext": {
        "subject": "Pet",
        "batchId": "e24f889a-bc4d-473b-85e3-3cd97d06dd42",
        "sourceSubType": "MERGE_PRESERVE_ON_NULL",
        "source": "TestSource"
      }
    },
    {
      "workOrderItemName": "WAIT_FOR_ALL_COMPLETE",
      "processorOrder": 4
    }
  ]
}

```

Starting the Automated Work Order (Submitting the Job)

To start the automated work order:

1. Use Swagger on the controller. For example:

```

https://<hostname>:9500/swagger-ui.html#!/WorkOrder/
postWorkOrderUsingPOST

```

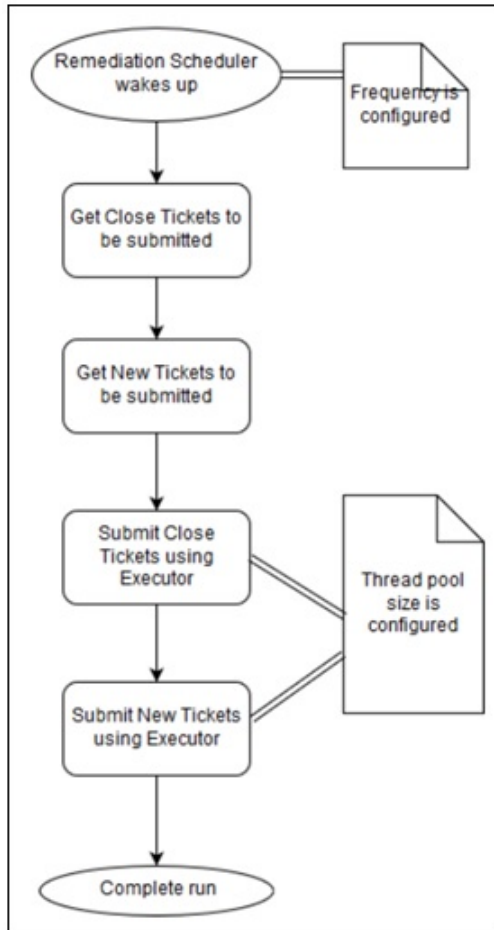
2. Copy and paste the JSON document into the work order Swagger API.

3. Click *Try it out*.









View the automation executing on the console work order screen.

Remediation Ticket Submission

This section describes the remediation ticket submission process as it is viewed by the Omni Server. Data Quality (DQ) remediation is responsible for generating remediation tickets that are stored in the *omni_remediation_ticket* table. Omni Server remediation is responsible for submitting these tickets to the Remediation Service. The Remediation Service exposes the tickets through the Omni Governance Console (OGC) user interface and allows them to be managed by a data steward. This section focuses on the ticket submission process from Omni Server to the Remediation Service.

High Level Workflow

Remediation tickets that are in the *PENDING* state are candidates for submission. The submission of these tickets is executed using a scheduler that fires off repeatedly. The frequency of the scheduler is configurable and controlled by the *Outbound Service Frequency* setting, whose value is specified in milliseconds, as shown in the following image.

Runtime			
Data Quality Runtime			
Server Remediation			
Command Line			
Setting	Value	Description	
 Outbound Service Frequency	8000 	Polling interval for sending information to the Remediation Server	
 Max Records to Poll	100	Maximum number of records processed during each poll	
 Remediation Services Base Uri	http://omnigen-VirtualBoxSRLib.com:5090/RemediationService/ /ul/workflow.svc/	Base URL for end points on the Remediation Server	
 Remediation Server About Uri	http://omnigen-VirtualBoxSRLib.com:5090/RemediationService/ /ul/workflow.svc/About	Endpoint URL for the About service on the remediation server	
 Remediation Server New Ticket Uri	http://omnigen-VirtualBoxSRLib.com:5090/RemediationService/ /ul/workflow.svc/NewTicket	Endpoint URL for new ticket on the remediation server	
 Remediation Server Close Ticket Uri	http://omnigen-VirtualBoxSRLib.com:5090/RemediationService/ /ul/workflow.svc/CloseTicket	Endpoint URL for close ticket on the remediation server	
 Max Remediation ticket threads	10	Maximum number of concurrent remediation ticket threads.	

For example, setting this value to 8000 will cause the scheduler to fire off every 8 seconds. During a given run, due to the way the remediation tickets are managed by the Remediation Service, any close ticket requests must be submitted before new ticket requests. The ticket submission is multi-threaded. This is done using an *Executor*, which services incoming submission requests using a thread pool. The size of this pool is controlled by the *Max Remediation ticket threads* setting, which can be configured. Processing is blocked until all of the close remediation ticket and new remediation ticket requests are submitted. The run cycle is considered finished after all of the submissions are completed. This entire process flow is repeated when the scheduler wakes up again.

The recommended default values for these two settings are:

- ☐ Outbound Service Frequency=10000
- ☐ Max Remediation ticket threads=10

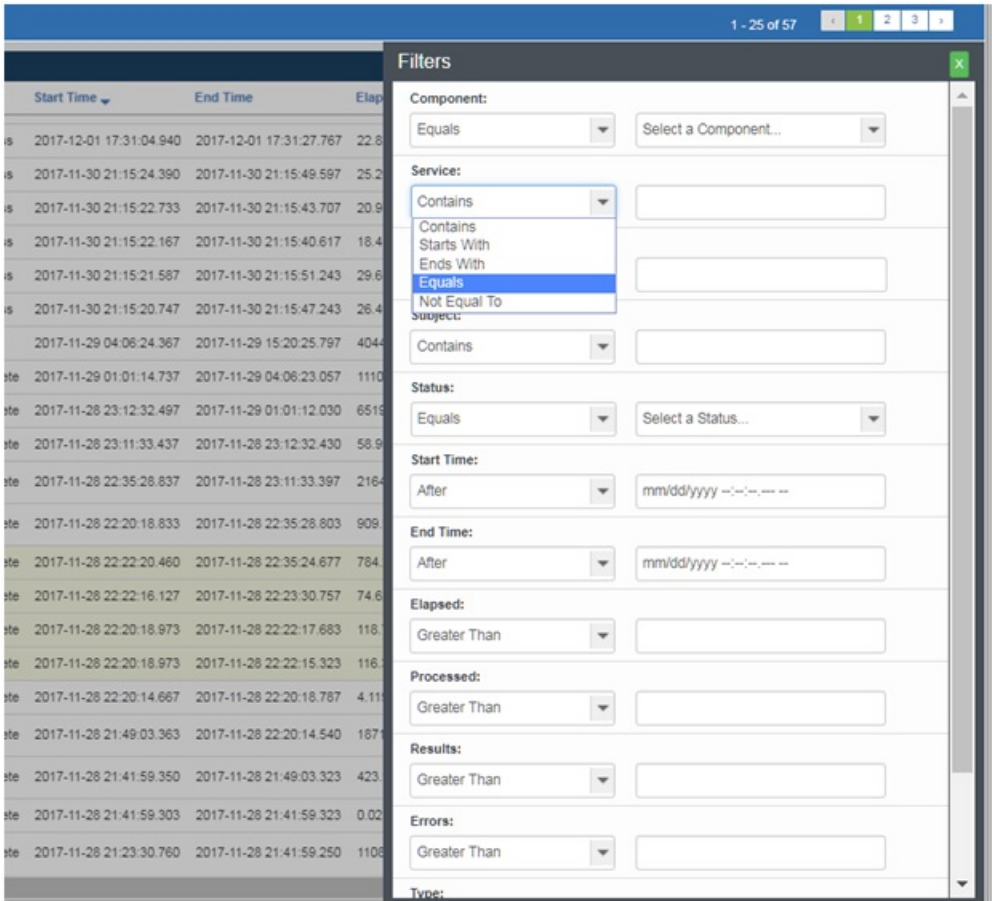
Omni-Insurance Operational Measures

The Omni Console allows you to display measures for specific operations. For example, you can check the execution status and processing time for each operation. To display these measures, click *Processing* in the left pane and then click *Measures*. A list of measures is displayed in the right pane, as shown in the following image.

Component	Service	Operation	Subject	Status	Start Time	End Time	Elapsed	Processed	Results	Errors	Type	Transaction
OMNI_CONTROLLER	ManagedService	startManagedService-matching		Success	2017-12-05 19:17:05.303	2017-12-05 19:17:42.473	36.28	0	0	0	STATUS	243e1e5-2665-4325-af52-04520225641e
OMNI_CONTROLLER	ManagedService	startManagedService-cleaning		Success	2017-12-05 19:17:04.193	2017-12-05 19:17:38.467	34.274	0	0	0	STATUS	611a0995-7411-4b49-8943-0900103c55ba
OMNI_CONTROLLER	ManagedService	startManagedService-elasticSearch		Success	2017-12-05 19:16:30.463	2017-12-05 19:17:00.620	30.356	0	0	0	STATUS	7b9e4d5b-7472-4d8c-8b42-03a37162d550
OMNI_CONTROLLER	ManagedService	startManagedService-elasticSearch		Failed	2017-12-05 19:13:17.230	2017-12-05 19:13:20.630	3.401	0	0	0	STATUS	1119c569-32b4-4ee1-8973-3631e573a492
OMNI_CONTROLLER	ManagedService	stopManagedService-wordbenchCleaning		Success	2017-11-30 21:15:24.360	2017-11-30 21:15:49.587	25.226	0	0	0	STATUS	6269501c-3a89-4d2a-8af6-85c457559a65
OMNI_CONTROLLER	ManagedService	stopManagedService-remediation		Success	2017-11-30 21:15:22.733	2017-11-30 21:15:43.707	20.975	0	0	0	STATUS	18795007-d442-4f82-af12-6935a78c369
OMNI_CONTROLLER	ManagedService	stopManagedService-merging		Success	2017-11-30 21:15:22.167	2017-11-30 21:15:42.617	15.453	0	0	0	STATUS	8752380c-d6ea-431b-836a-8d8ed136c229
OMNI_CONTROLLER	ManagedService	stopManagedService-matching		Success	2017-11-30 21:15:21.567	2017-11-30 21:15:51.243	29.656	0	0	0	STATUS	005f865b-8736-4d5a-abc3-a2865140bc15
OMNI_CONTROLLER	ManagedService	stopManagedService-cleaning		Success	2017-11-30 21:15:20.747	2017-11-30 21:15:47.243	26.497	0	0	0	STATUS	369c916e-4007-4290-4a15-25a197f9e1ed
OMNI_SERVER	Publish	masters	Customer	Failed	2017-11-29 04:06:24.367	2017-11-29 15:20:29.797	40441.431	13455	0	0	TMED	369c916e-4007-4290-4a15-25a197f9e1ed
OMNI_SERVER	Publish	instances	Customer	Complete	2017-11-29 01:01:14.737	2017-11-29 04:06:23.057	11108.321	1000000	0	0	TMED	369c916e-4007-4290-4a15-25a197f9e1ed
OMNI_SERVER	Publish	source	Customer	Complete	2017-11-29 23:12:32.467	2017-11-29 01:01:12.030	6519.536	1000000	0	0	TMED	369c916e-4007-4290-4a15-25a197f9e1ed
OMNI_SERVER	AutoClose	autoClose	Customer	Complete	2017-11-29 23:11:33.437	2017-11-29 23:12:32.430	58.983	0	0	0	TMED	369c916e-4007-4290-4a15-25a197f9e1ed
OMNI_SERVER	Remediation	remediate	Customer	Complete	2017-11-29 22:35:28.837	2017-11-29 23:11:33.397	2194.961	90519	90519	0	TMED	369c916e-4007-4290-4a15-25a197f9e1ed
OMNI_SERVER	Merging	merge	Customer	Complete	2017-11-29 22:20:18.833	2017-11-29 22:35:28.803	909.971	90541	90541	0	TMED	369c916e-4007-4290-4a15-25a197f9e1ed
Public	GDPR-002	GDPRBatchProcess	GDPRBatchProcess	Customer	Complete	2017-11-16 10:10:14.867	2017-11-16 10:10:14.867	4.116	0	0	TMED	369c916e-4007-4290-4a15-25a197f9e1ed

You can click on any column name to order them based on that column. In addition, you can toggle descending and ascending ordering by clicking a column name. Some operations contain sub-operations. The measures of such operations have a preceding plus sign (+) button. Clicking this button expands and displays the measures of the sub-operations.

Each screen can display 25 measures. The upper-right corner contains page control buttons to view additional measures. You can click the page number, and left or right arrow buttons, to navigate to other pages. Below the page control buttons is a filter button. Clicking the filter button opens the filter window, as shown in the following image.



The filter window allows you to configure filters, which allow you to search for specific measures. You can create a filter for one column or a combination of multiple filters for multiple columns. It is very convenient and useful to display the measures of interest.

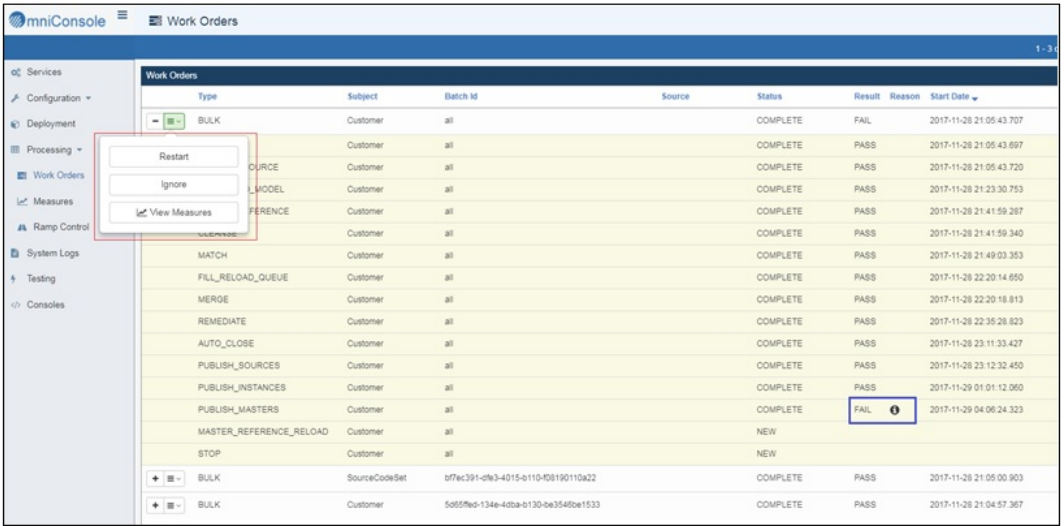
There is another way to quickly show the measures for a work order. Click *Processing* in the left pane and then click *Work Orders*. All of the work orders are listed in the right pane, as shown in the following image.

The screenshot shows the OmniConsole interface for Work Orders. The left sidebar has a tree view with 'Processing' expanded and 'Work Orders' selected. A red box highlights the 'View Measures' button. The main table lists work orders with columns: Type, Subject, Search ID, Source, Status, Result, Message, and Start Date. A red arrow points to the 'Filter button' in the top right corner. A red arrow also points to the 'Work order items' label in the left sidebar.

Type	Subject	Search ID	Source	Status	Result	Message	Start Date
BULK	Customer	all		COMPLETE	FAIL		2017-11-28 21:05:43.707
BULK	SourceCodeSet	8f7ec391-d9e3-4d15-b110-65810210a22		COMPLETE	PASS		2017-11-28 21:05:00.903
BULK	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:04:57.367
SOURCE_CODE	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:04:57.363
SOURCE_CODE_MODEL	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:04:57.400
MASTER_REFERENCE	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:00.473
CLEARAGE	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:00.873
WATCH	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:02.143
FULL_RELOAD_QUEUE	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:03.113
MERGE	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:03.179
REMEDATE	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:06.343
AUTO_CLOSE	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:07.737
PUBLISH_SOURCES	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:07.870
PUBLISH_INSTANCES	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:11.847
PUBLISH_MASTERS	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:12.790
MASTER_REFERENCE_RELOAD	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:14.367
STOP	Customer	5d89fed-134e-4db8-b130-be3548be1533		COMPLETE	PASS		2017-11-28 21:05:14.107

The work order screen shows the status of work orders. You can also order them by clicking a column name. You can also navigate pages using the upper right corner buttons and add filters in the filter windows. Clicking the plus sign (+) button to the left of a work order will show the status of individual work order items. There is also a small *manual* button next to the plus sign (+) button. Clicking this button will display a context menu. Clicking *View Measures* will display all of the measures for the work order. This is a quick way for you to check the measures for a specific work order.

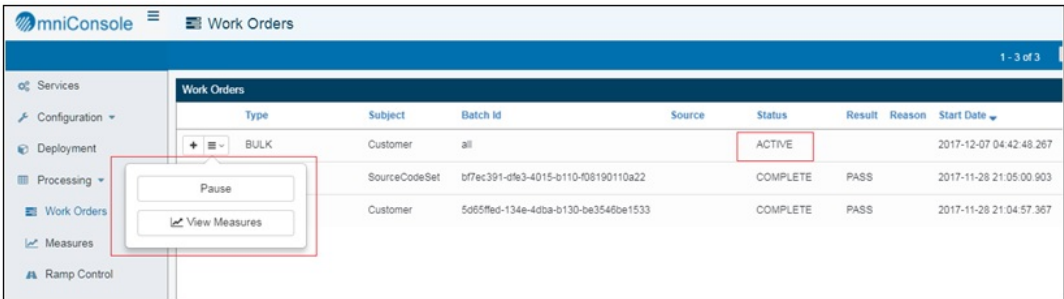
If a work order fails, the result will display *FAIL* in the work order screen. To determine at which particular step a work order has failed, you can expand the work order. An expanded failed work order is shown in the following image.



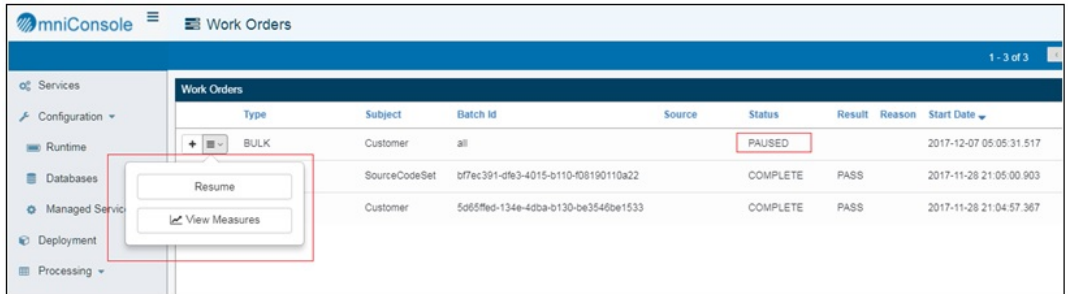
This particular work order shows a failure at the *PUBLISH_MASTERS* step. Clicking the small "i" icon in the blue rectangle displays a short message about the failure, which helps you understand the cause of the failure.

If you click the small *manual* button to the left of the work order, the context menu (in the red rectangle) will show different options. *Restart* will restart the failed step. *Ignore* will skip this work order. Otherwise, the failed work order will block the execution of the other work orders for the same subject. *View Measures* will show detailed measures for this work order.

When the status of a work order is active, the context menu shows *Pause* and *View Measures*, as shown in the following image.



You can pause a work order if required. The status of the work order will change to *PAUSED*. The context menu will change to *Resume* and *View Measures*, as shown in the following image.



The screenshot shows the mniConsole interface with a sidebar on the left containing 'Services', 'Configuration', 'Runtime', 'Databases', 'Managed Service', 'Deployment', and 'Processing'. The main area is titled 'Work Orders' and displays a table with columns: Type, Subject, Batch Id, Source, Status, Result, Reason, and Start Date. A context menu is open over the first row, which has a status of 'PAUSED'. The menu contains two options: 'Resume' and 'View Measures'.

Type	Subject	Batch Id	Source	Status	Result	Reason	Start Date
BULK	Customer	all		PAUSED			2017-12-07 05:05:31.517
	SourceCodeSet	b77ec391-dfe3-4015-b110-408190110a22		COMPLETE	PASS		2017-11-28 21:05:00.903
	Customer	5065f9ed-134e-4d8a-b130-be3546be1533		COMPLETE	PASS		2017-11-28 21:04:57.367

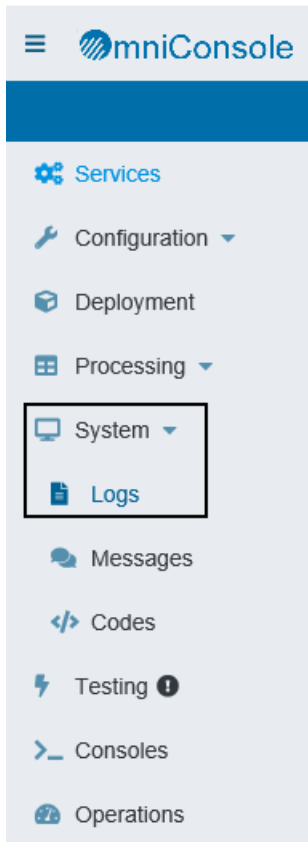
Resume allows you to resume the operation of the work order.

Omni-Insurance Logging

This section describes how to view logging information for Omni-Insurance, configure logging options, and key considerations.

Viewing System Logs

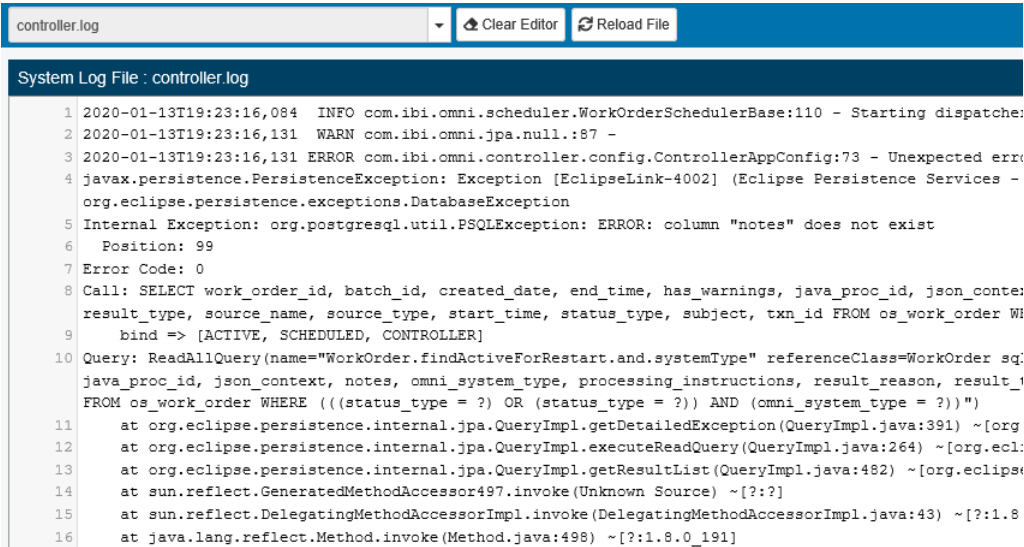
Most of the Omni-Insurance log files that are used most often can be viewed through the Omni Console. Click System Logs, as shown in the following image.



You can select the desired log from the drop-down list to view details, as shown in the following image.

Select a System Log File...	
Command Line 10	
command_configure.log	2020-01-10 14:35:23
command_controller_deploy_bundle.log	2020-01-13 13:58:55
command_controller_deploy_bundle_1.log	2020-01-13 13:37:03
command_controller_deploy_bundle_2.log	2020-01-13 13:38:33
command_install_controller_winsvc.log	2020-01-10 14:37:37
command_install_prop_encrypt.log	2020-01-10 14:37:31
command_start_controller.log	2020-01-10 15:50:06
command_start_controller_1.log	2020-01-10 14:55:59
command_start_controller_2.log	2020-01-10 15:35:55
command_start_controller_3.log	2020-01-10 15:43:45
Controller 4	
controller-1.log	2020-01-13 17:10:24
controller-2.log	2020-01-13 18:16:50
controller-3.log	2020-01-13 19:23:16
controller.log	2020-01-13 19:39:44
Deployment Bundler 1	
bundler.log	2020-01-13 13:17:25
Repository 17	
bridge.log	2020-01-10 15:58:33
catalina.2020-01-10.log	2020-01-10 16:08:45
catalina.2020-01-13.log	2020-01-13 13:15:41
host-manager.2020-01-10.log	2020-01-10 15:58:26
host-manager.2020-01-13.log	2020-01-13 13:15:33

A log viewer for the selected log is displayed, as shown in the following image.



Configuration

Modifying the log file locations is currently not supported, but many of them can be viewed through the configuration tabs in the controller console, as described in the following table.

Application	Configuration Page	Configuration Tab	Parameter Name	Parameter Key
command line	Runtime	Command Line	Log Directory	server.commandline.log-directory
controller service	Managed Services	Controller	Log Directory	server.controller.log-directory
DQ services (except workbench)	Runtime	Data Quality Runtime	Data Quality Logs	server.dq.mastering.log-directory
deployment bundle service	Managed Services	Deployment	Log Directory	server.bundler.log-directory
omni server	Managed Services	Server	Log Directory	server.omni-server.log-directory

Application	Configuration Page	Configuration Tab	Parameter Name	Parameter Key
OGC web applications	Managed Services	OGC	Log Directory	ogc.log-directory
OGC tomcat server	Managed Services	OGC Tomcat	Log Directory	server.ogctc.log-directory
repository server	Managed Services	Repository	Log Directory	server.repository.log-directory
workbench cleansing service	Managed Services	Workbench Cleansing	Workbench Cleansing Logs	server.dq.workbench.cleansing.log-directory

File Organization

For convenience, commonly-used log files generated by most Omni-Insurance processes can be found in the OmniGenData/logs directory, inside the Omni-Insurance install directory. The log files are further organized into subdirectories based on the process that generated them:

- ❑ bundler: Deployment bundle service logs.
- ❑ command: Output from any "omni" shell command.
- ❑ controller: Omni controller service logs.
- ❑ dq: Logs from the data quality services.
- ❑ OGC: The OGC tomcat standard output.
- ❑ OmniDesignerRepository: All repository service tomcat logs (including web applications).
- ❑ server: Omni server logs.

In some cases, more detailed logs or output data can be found in other locations:

- ❑ deploymentbundle: Saved copies of deployed bundles.
- ❑ deploymentbundle/logs: Zipped archives of deployment bundle service logs.
- ❑ install/Omnigen_install_logs: Installer logging and debug output.
- ❑ OmniDesignerRepository/webapps/Bridge/WEB-INF/lib/configuration: EMF bridge web application detail messages.

- ❑ OmniGenData/deployment: Detailed deployment event timings.
- ❑ OmniServer/dbms/changelogs: most-recent LiquiBase migration changesets.
- ❑ OmniGovConsole/logs: OGC tomcat and web application logs.
- ❑ wso2_is/repository/logs: WSO2 server logs.

Log and Output File Details

Application	File	Location	Archival	Description
command line	command_ [command].log	OmniGenData/ logs/command	Numbered	Output of running "omni [command]" from the command line.
	command_ controller_ [command].log			Output of running "omni [command]", but via the controller service.
controller service	controller.log	OmniGenData/ logs/controller	Numbered	Controller service output.
deployment bundle service	bundler.log	OmniGenData/ logs/bundler	Dated and numbered, zipped, saved to deploymentbundle/ logs	Deployment bundle service (bundler) output.
	[deploy_bundle].zip	deploymentbundle	Epoch timestamped, all previous files	Input files uploaded to the controller console when Deploy Bundle button is clicked.

Application	File	Location	Archival	Description
data quality services	[dq service type]_online_0.log	OmniGenData/ logs/dq	None	General messages about the running service.
	[dq service type]_err_0.log		None	Service error messages.
	[dq service type]_perf_0.log		None	Service performance timings and events.
	[dq service type]_request.log		None	Detailed service request data.
	cleansing_*.log	OmniGenData/ logs/dq/workbench	None	Data quality service logs for workbench cleansing. Same semantics as for other DQ services.
installer	Omnigen_install_[timestamp].log	install/ Omnigen_install_logs	Timestamped	Detailed OmniGen installer messages and output.
	installer_debug[n].txt		Numbered	OmniGen installer configuration dump.
OGC tomcat server	ogc.log	OmniGenData/ logs/OGC	None	Output of OGC tomcat server.
omni server	server.log	OmniGenData/ logs/server	Numbered	Output of Omni server.

Application	File	Location	Archival	Description
repository tomcat server	repository.log	OmniGenData/ logs/ OmniDesignerRepository	None	Output of repository tomcat server.
	catalina.[date].log		Dated	Repository tomcat general messages.
	host-manager. [date].log		Dated	Repository tomcat manager web application output.
	[host].[date].log		Dated	Repository tomcat virtual host manager app messages.
	[host]_access. [date].log		Dated	Repository tomcat web request log.
	manager.[date].log		Dated	Repository tomcat manager app messages.
	bridge.log		Numbered	EMF bridge web application log.
	repository_ service.log		Numbered	Repository service web application log.
	workbench_ service.log		Numbered	Workbench service web application log.
	[epoch timestamp].log	OmniDesignerRepository/webapps/ Bridge/WEB-INF/lib/ configuration	Epoch timestamped	Detailed EMF repository messages.

Application	File	Location	Archival	Description
WSO2 server	audit.log	wso2_is/ repository/logs	Dated	WSO2 security messages, such as login.
	http_access_[date].log		Dated	WSO2 web request log.
	wso2carbon.log		Dated	WSO2 main server log.
	wso2carbon-trace-messages.log		Dated	WSO2 trace-level messages (not common).

Advanced Logging Options

Sometimes it may be necessary to exercise control over how specific Omni-Insurance services perform logging. This section describes several options that are available for advanced users.

Data Quality Tracing

The Data Quality services (cleanse, match, merge, remediate, and workbench cleanse) can be set to log transaction data to log files and real records to CSV-formatted files. In the Controller Console (under Configuration, Runtime, Runtime tab), set Enable DQ trace files to true. The parameter key is `server.runtime.server.dqTraceEnable`. This will cause Omni-Insurance to write two CSV files every time a DQ service processes any records. One will be a "send" file, the other a "receive" file. Each will contain a header row and a row for each record sent to, or received by, a DQ service. The files will be named to indicate the DQ service, the record subject, the transaction ID, and whether the data was sent or received.

Standard Logging Support

Omni-Insurance services rely on Apache Log4J to provide logs. This allows configuration of logging through standard Log4J/Slf4J configuration files, or through a wrapper framework, as described in the following table.

Framework	Application	Logging Configuration Files
Log4J	command line	OmniServer/cmd/conf/log4j-command.xml
	controller service	OmniServer/cmd/conf/log4j-controller.xml
	data quality services	OmniServer/mastering/services/[dq service type]/services/_logging.xml, OmniServer/workbench/services/cleansing/services/_logging.xml
	omni server	OmniGenData/OmniServer/cmd/conf/log4j-server.xml
	WSO2 server	wso2_is/repository/conf/log4j.properties
Spring Boot	deployment bundle service	deploymentbundle/application.properties
Apache Tomcat	OGC	OmniGenData/OmniGovConsole/conf/logging.properties
	repository server	OmniGenData/OmniDesignerRepository/conf/logging.properties

Log4J Logging Considerations

Omni-Insurance uses standard Apache Log4J logging levels. This means that:

1. Log messages will be emitted with one of the following severity levels: FATAL, ERROR, WARN, INFO, DEBUG, or TRACE
2. Logging severity level filters can be set per service or per code structure to filter out messages with a lesser severity.
3. Omni-Insurance log messages have been tagged to best represent their relevance to the user, where possible, for example:
 - a. **FATAL.** A fatal error, meaning the process must stop or, sometimes, the operation has failed.
 - b. **ERROR.** An error that is typically recoverable.
 - c. **WARN.** An unusual or non-ideal condition has been detected, but the operation will continue.
 - d. **INFO.** General messages about the normal behavior of the process or operation.
 - e. **DEBUG.** Messages generally useful in troubleshooting or debugging, typically short messages containing a small amount of data or internal state.

- f. **TRACE.** Larger messages useful in troubleshooting or debugging, typically filtered out by default and containing full contents of commands or internal structures.

For more information about configuration and log levels, see the following online Apache Log4J documentation:

<https://logging.apache.org/log4j/2.x/manual/configuration.html#ConfigurationSyntax>

<https://logging.apache.org/log4j/2.x/manual/configuration.html#Properties>

<https://logging.apache.org/log4j/2.x/manual/configuration.html#SystemProperties>

<https://logging.apache.org/log4j/2.x/manual/architecture.html>

<https://logging.apache.org/log4j/2.x/log4j-api/apidocs/org/apache/logging/log4j/Level.html>

Reserved Words

This section provides information on system reserved words, which you should not use as part of the model definition or any user-defined fields.

Note: Contents of the *source name* and *source instance id* fields, which are populated by the user-defined Omni Integration processes, should not contain white space characters, for example, ASCII blank and below, as well as colons. These characters are prohibited.

In this appendix:

- ❑ [Reserved Words List](#)
-

Reserved Words List

The following list of words is reserved for system use.

- ❑ id
- ❑ MasterChildId
- ❑ MasterId
- ❑ MasterStatus
- ❑ MasterStatusCode
- ❑ MasterStatusReason
- ❑ master_id
- ❑ MatchCandidateGroupId
- ❑ MatchOverrideBlacklist
- ❑ MatchOverrideWhitelist
- ❑ MatchProcessingTimestamp
- ❑ MatchQualityDetail
- ❑ MatchQualityScore
- ❑ MatchRole

- ☐ MatchRule
- ☐ match_can_group_id
- ☐ match_override_blacklist
- ☐ match_override_whitelist
- ☐ match_proc_tstamp
- ☐ match_quality_detail
- ☐ match_quality_score
- ☐ match_role
- ☐ match_rule
- ☐ OmniCreatedDate
- ☐ OmniModifiedDate
- ☐ OmniStatus
- ☐ OmniStatusReason
- ☐ omni_created_date
- ☐ omni_modified_date
- ☐ omni_status
- ☐ omni_status_reason
- ☐ PreviousMasterId
- ☐ prev_master_id
- ☐ SourceCreatedBy
- ☐ SourceCreatedDate
- ☐ SourceInstanceId
- ☐ SourceInstanceIdName
- ☐ SourceModifiedBy
- ☐ SourceModifiedDate

- ☐ SourceName
- ☐ SourceStatusCode
- ☐ source_created_by
- ☐ source_created_date
- ☐ source_instance_id
- ☐ source_instance_id_name
- ☐ source_modified_by
- ☐ source_modified_date
- ☐ source_name
- ☐ source_status_code



Feedback

Customer success is our top priority. Connect with us today!

Information Builders Technical Content Management team is comprised of many talented individuals who work together to design and deliver quality technical documentation products. Your feedback supports our ongoing efforts!

You can also preview new innovations to get an early look at new content products and services. Your participation helps us create great experiences for every customer.

To send us feedback or make a connection, contact Sarah Buccellato, Technical Editor, Technical Content Management at Sarah_Buccellato@ibi.com.

To request permission to repurpose copyrighted material, please contact Frances Gambino, Vice President, Technical Content Management at Frances_Gambino@ibi.com.

iWay

/ Omni-Insurance Operation and Management Guide

Version 3.12

DN3502367.0520

Information Builders, Inc.
Two Penn Plaza
New York, NY 10121-2898

