Information Builders

iWay

Omni-Gen™ Integration Edition Operation and Management Guide

Version 3.14

February 11, 2020

Contents

Preface

This documentation provides operation and management information for Omni-Gen™ Integration Edition. It is intended for developers and administrators of Omni-Gen.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	Contents
1	Omni-Gen™ Integration Edition Operation and Management	Provides information for Omni-Gen™ Integration Edition 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
`THIS TYPEFACE` or `this typeface`	Denotes syntax that you must type exactly as shown.
this typeface	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 *http://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 *not* 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*.

Chapter 1

Omni-Gen™ Integration Edition Operation and Management

This document is designed to serve as a starting point and reference for an operator of Omni-Gen™ Integration Edition. 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

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) - Wipe out existing database tables, install new bundle, create new schema.

❑ **Update** (deploy-bundle) - Upgrade an existing bundle, update existing Data Model.

❑ **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

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-Gen 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*	
source_*	

Omni-Gen System Tables [Reset Only]	
Matched Patterns	**Specific Named Tables**
os_work_*	

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. Model
   b. Ramp
   c. Source
   d. Clean system tables (Clean only).
   e. System (Reset only).
   f. Delete existing Liquibase Database changelogs.
2. Custom(er) database pre-deployment migration scripts.
3. Omni-Gen pre-deployment migration scripts.
4. Create/Update database tables. This includes the following data sources:
   a. Model
   b. Ramp
   c. Source
   d. System
5. Custom(er) database post-deployment migration scripts.
6. Omni-Gen 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-Gen database, you are prompted to continue or quit the installation, based on the existence of the tables. This serves to prevent you from accidently 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. Generate Effective IDS documents.
5. Generate IDS documentation.
6. Generate IDS sample OIDs.
7. Generate XSD schemas for the IDS documents.
8. Generate JPA model for the IDS documents.
9. Compile the JPA model.
10. Weave the JPA model.
11. Package and move the model jar.

Deployment Logs

Each time a bundle or database deployment is undertaken from the command line or user interface, 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. For example, deploy_update_2019-11-06 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.

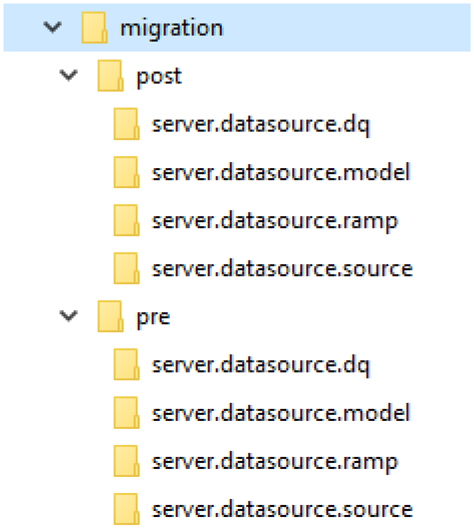
The following table provides a list of the possible generated log files and the corresponding deployment commands.

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

Database Migration

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



```
migration
  post
    server.datasource.dq
    server.datasource.model
    server.datasource.ramp
    server.datasource.source
  pre
    server.datasource.dq
    server.datasource.model
    server.datasource.ramp
    server.datasource.source
```

Customer: /omnigen/OmniGenData/migration

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

The following is an example of a 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-Gen environment.
  - ❑ Named: og_{subject_name}_r
- ❑ Source. **Note:** In Omni-Gen Integration Edition, source tables are not used.
  - ❑ Source data (does not get modified).
  - ❑ Named: og_{subject_name}_s

- Model
    - Instance data
    - Named: og_{subject_name}
- System tables
    - Used by Omni-Gen system, not dependent on the model.
    - Named: os_

Below is a list of Omni-Gen 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	
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-Gen system.
os_ramp_control	Data loading jobs and related information that were submitted to the ramp for processing.
os_reload_queue	

System Table Name	Description
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.

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-Gen 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
System	omni-systemMigrateChangeLog.xml
	omni-systemJpaModelChangeLog.xml
	omni-systemDropChangeLog.xml
Entire database schema	{hashkey}-DatabaseChangeLog.xml

ChangeLog Locations

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

- ❑ Changelogs: All dynamically generated changelogs created by Omni-Gen 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.
- ❑ 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.

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 and Work Order Item 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-Gen system. The database activity tab shows the slowest ten, and the most recent ten queries from the Server. You can enable this feature using the SQL Profiling Enabled option under *Configuration, Databases* for Model, Ramp, and Source before data is run through the system in order to gather the metrics.

Database Pools and Connection Information

The Omni-Gen 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.

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

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

Application. Name of the Omni component that holds the pool.

Pool. Name of the JDBC pool. The pool name should be unique for each application and is not configurable.

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

Settings. Name of the settings that govern the pool.

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

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

Max-active. 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).

Max-idle. Maximum number of connections that should be kept in the idle pool.

Min-idle. 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*.

Initial Size. 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).

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.

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 for the specific database vendor (SQL Server, Oracle, PostgreSQL, 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).

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-Gen 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.
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 (0).
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.

Column	Standard Data Type	Description
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.
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 *databasechangeloglock*.

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.

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.propertie*s 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.

H2

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

Performance Tuning

When handling a large number of records, you may want to optimize the database being used for Omni-Gen. 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.

Database Maintenance Recommendations

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

Archiving Tables in the Omni-Gen Database

The following table lists and describes the tables that can be archived in the Omni-Gen 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	List of Work Order tasks generated by the system for execution.	As needed.

Table	Table Name	Description	Archive Strategy
Work Order Items	os_work_order_item	List of Work Order Items that represent the individual operations that take place during the execution of a Work Order.	As needed.
Ramp Control	os_ramp_control	List of subject operations that load data into the system from the relational on ramp.	

Archiving Error and Information Logs in Omni-Gen

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, …}
    ```

    System logs are separated into subdirectories by application name.
  - **Purpose:** Provides detailed records of a particular application component of the Omni-Gen 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 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-Gen 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

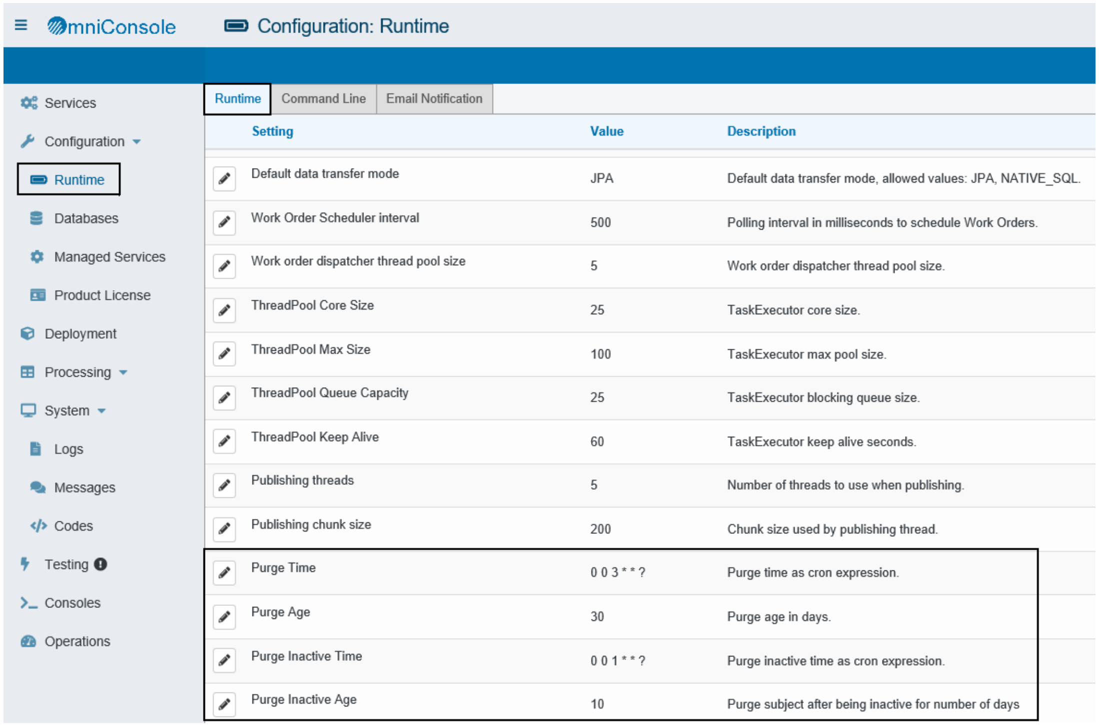
The following Omni-Gen server-based runtime settings are available to control the purging of inactive user data:

- **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, and history 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
Default data transfer mode	JPA	Default data transfer mode, allowed values: JPA, NATIVE_SQL.
Work Order Scheduler interval	500	Polling interval in milliseconds to schedule Work Orders.
Work order dispatcher thread pool size	5	Work order dispatcher thread pool size.
ThreadPool Core Size	25	TaskExecutor core size.
ThreadPool Max Size	100	TaskExecutor max pool size.
ThreadPool Queue Capacity	25	TaskExecutor blocking queue size.
ThreadPool Keep Alive	60	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 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 box 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-Gen 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-Gen from an Omni Interface Document using the REST API service.

  The following syntax shows the REST API service:

  ```
  /server/processInstance
  ```

- **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, and history 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, and history tables. Other records such as overrides and code-field references are deleted.

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_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 instance record and/or its transaction ID and Omni-Gen 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-Gen Integration Edition Relational OnRamp 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.

HISTORY

The following steps will be included if the captureHistory attribute in the instance 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.

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 records in the current transaction for which a CDC subscription exists.

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 following is a high-level summary of how to automate a sequential set of batches to 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.

To support automation, the following functions were added:

- **START_BATCH.** The start batch function 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.

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
   ```

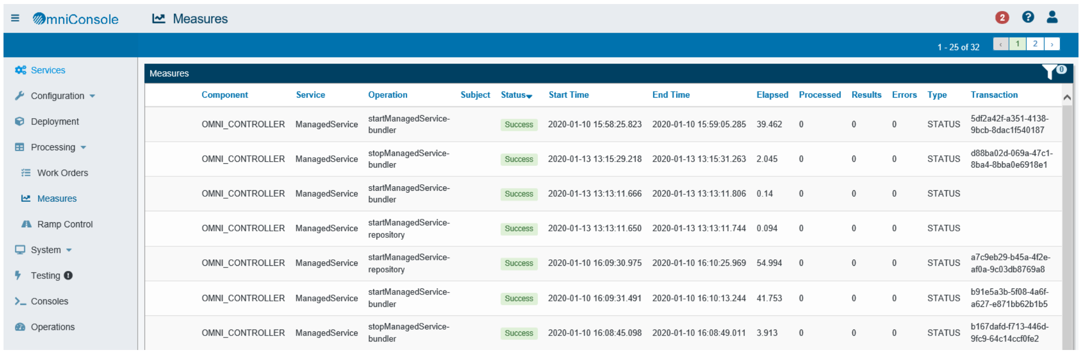
2. Copy and paste the JSON document into the work order Swagger API.

3. Click *Try it out*.

   View the automation executing on the console work order screen.

Omni-Gen 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.

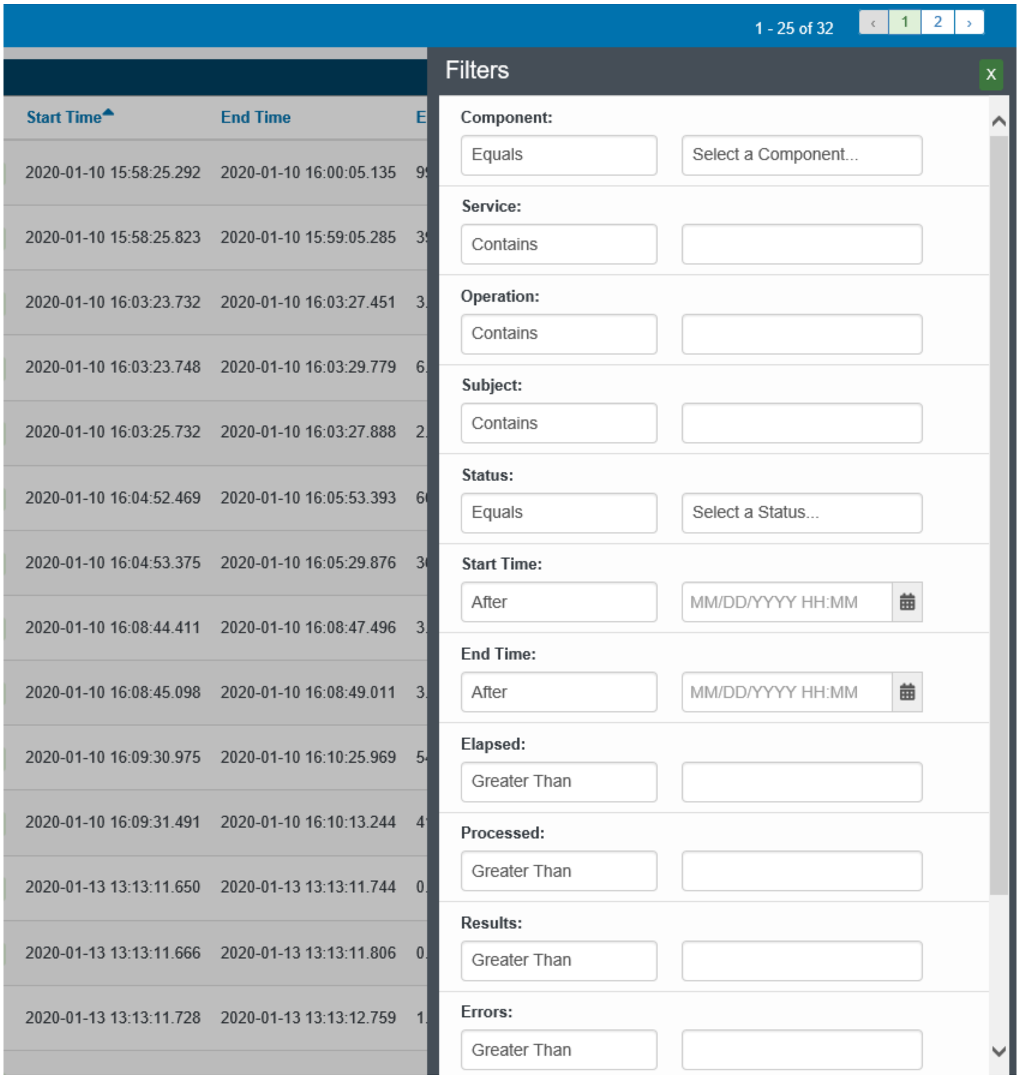


Measures

Component	Service	Operation	Subject	Status	Start Time	End Time	Elapsed	Processed	Results	Errors	Type	Transaction
OMNI_CONTROLLER	ManagedService	startManagedService-bundler		Success	2020-01-10 15:58:25.823	2020-01-10 15:59:05.285	39.462	0	0	0	STATUS	5df2a42f-a351-4138-9bcb-8dac1f540187
OMNI_CONTROLLER	ManagedService	stopManagedService-bundler		Success	2020-01-13 13:15:29.218	2020-01-13 13:15:31.263	2.045	0	0	0	STATUS	d88ba02d-069a-47c1-8ba4-8bba0e6918e1
OMNI_CONTROLLER	ManagedService	startManagedService-bundler		Success	2020-01-13 13:13:11.666	2020-01-13 13:13:11.806	0.14	0	0	0	STATUS	
OMNI_CONTROLLER	ManagedService	startManagedService-repository		Success	2020-01-13 13:13:11.650	2020-01-13 13:13:11.744	0.094	0	0	0	STATUS	
OMNI_CONTROLLER	ManagedService	startManagedService-repository		Success	2020-01-10 16:09:30.975	2020-01-10 16:10:25.969	54.994	0	0	0	STATUS	a7c9eb29-b45a-4f2e-af0a-9c03db8769a8
OMNI_CONTROLLER	ManagedService	startManagedService-bundler		Success	2020-01-10 16:09:31.491	2020-01-10 16:10:13.244	41.753	0	0	0	STATUS	b91e5a3b-5f08-4a6f-a627-e871bb62b1b5
OMNI_CONTROLLER	ManagedService	stopManagedService-bundler		Success	2020-01-10 16:08:45.098	2020-01-10 16:08:49.011	3.913	0	0	0	STATUS	b167dafd-f713-446d-9fc9-64c14ccf0fe2

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.

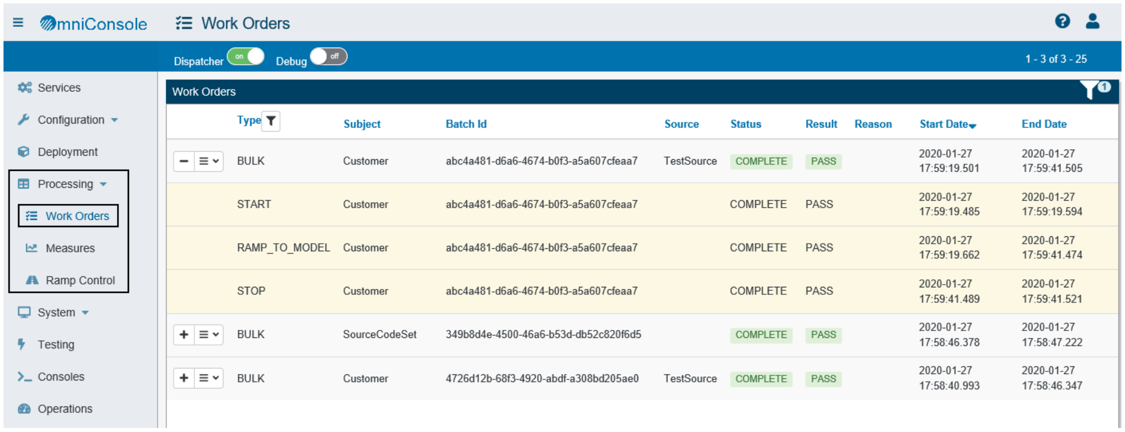


Start Time	End Time	E
2020-01-10 15:58:25.292	2020-01-10 16:00:05.135	9
2020-01-10 15:58:25.823	2020-01-10 15:59:05.285	3
2020-01-10 16:03:23.732	2020-01-10 16:03:27.451	3.
2020-01-10 16:03:23.748	2020-01-10 16:03:29.779	6.
2020-01-10 16:03:25.732	2020-01-10 16:03:27.888	2.
2020-01-10 16:04:52.469	2020-01-10 16:05:53.393	6
2020-01-10 16:04:53.375	2020-01-10 16:05:29.876	3
2020-01-10 16:08:44.411	2020-01-10 16:08:47.496	3.
2020-01-10 16:08:45.098	2020-01-10 16:08:49.011	3.
2020-01-10 16:09:30.975	2020-01-10 16:10:25.969	5
2020-01-10 16:09:31.491	2020-01-10 16:10:13.244	4
2020-01-13 13:13:11.650	2020-01-13 13:13:11.744	0.
2020-01-13 13:13:11.666	2020-01-13 13:13:11.806	0.
2020-01-13 13:13:11.728	2020-01-13 13:13:12.759	1.

Component:	Equals	Select a Component...
Service:	Contains	
Operation:	Contains	
Subject:	Contains	
Status:	Equals	Select a Status...
Start Time:	After	MM/DD/YYYY HH:MM
End Time:	After	MM/DD/YYYY HH:MM
Elapsed:	Greater Than	
Processed:	Greater Than	
Results:	Greater Than	
Errors:	Greater Than	

The filter window allows you to configure filters, 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.



Type	Subject	Batch Id	Source	Status	Result	Reason	Start Date	End Date
BULK	Customer	abc4a481-d6a6-4674-b0f3-a5a607cfeaa7	TestSource	COMPLETE	PASS		2020-01-27 17:59:19.501	2020-01-27 17:59:41.505
START	Customer	abc4a481-d6a6-4674-b0f3-a5a607cfeaa7		COMPLETE	PASS		2020-01-27 17:59:19.485	2020-01-27 17:59:19.594
RAMP_TO_MODEL	Customer	abc4a481-d6a6-4674-b0f3-a5a607cfeaa7		COMPLETE	PASS		2020-01-27 17:59:19.662	2020-01-27 17:59:41.474
STOP	Customer	abc4a481-d6a6-4674-b0f3-a5a607cfeaa7		COMPLETE	PASS		2020-01-27 17:59:41.489	2020-01-27 17:59:41.521
BULK	SourceCodeSet	349b8d4e-4500-46a6-b53d-db52c820f6d5		COMPLETE	PASS		2020-01-27 17:58:46.378	2020-01-27 17:58:47.222
BULK	Customer	4726d12b-68f3-4920-abdf-a308bd205ae0	TestSource	COMPLETE	PASS		2020-01-27 17:58:40.993	2020-01-27 17:58:46.347

The work order screen shows the status of work orders. You can order the work orders 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 drop-down arrow next to the plus sign (+) button. Clicking this button will display a context menu. Clicking *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.

Clicking the *i* icon in next to FAIL displays a short message about the failure, which helps you understand the cause of the failure.

If you click the drop-down arrow to the left of the work order, the context menu 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. *Measures* will show detailed measures for this work order.

When the status of a work order is active, the context menu shows *Pause* and *Measures* options.

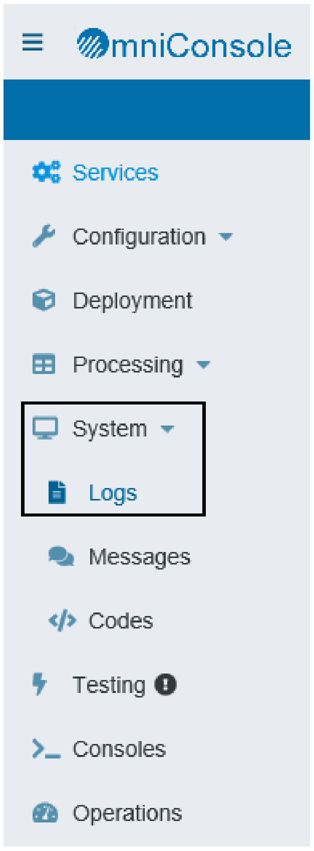
You can pause a work order if required. The status of the work order will change to *PAUSED*. The context menu will change to *Resume* and *Measures*. *Resume* allows you to resume the operation of the work order.

Omni-Gen Logging

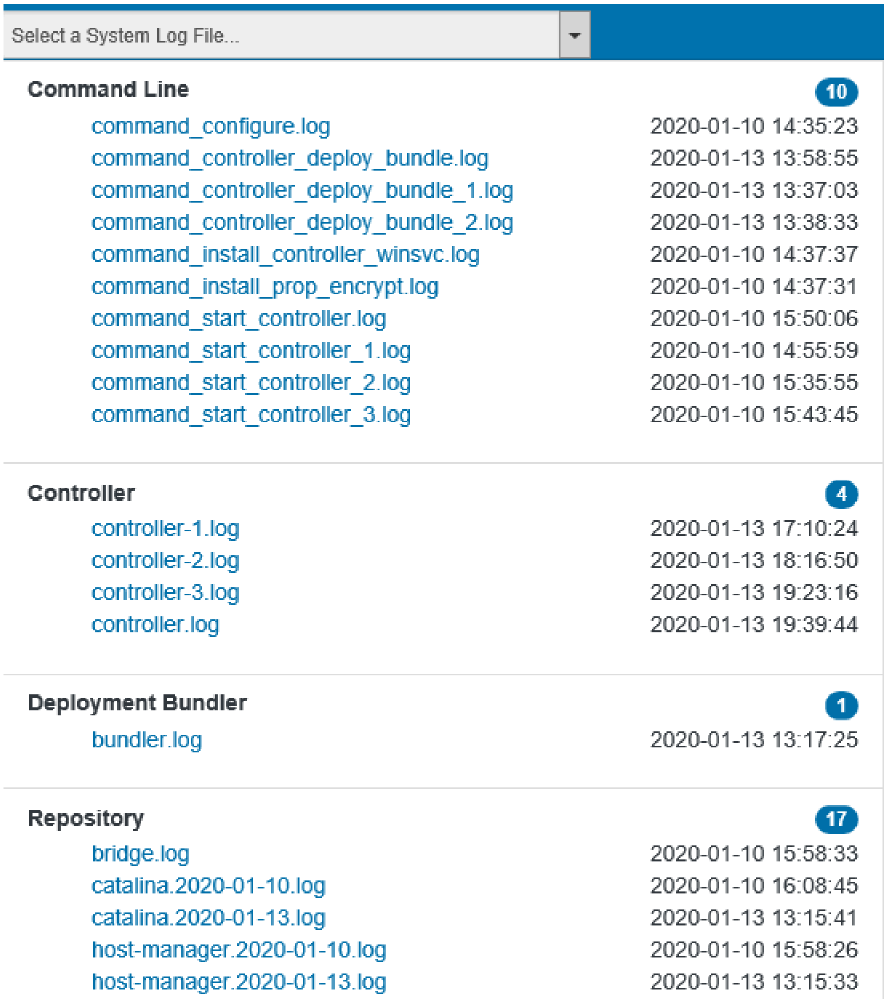
This section describes how to view logging information for Omni-Gen and configure logging options.

Viewing System Logs

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

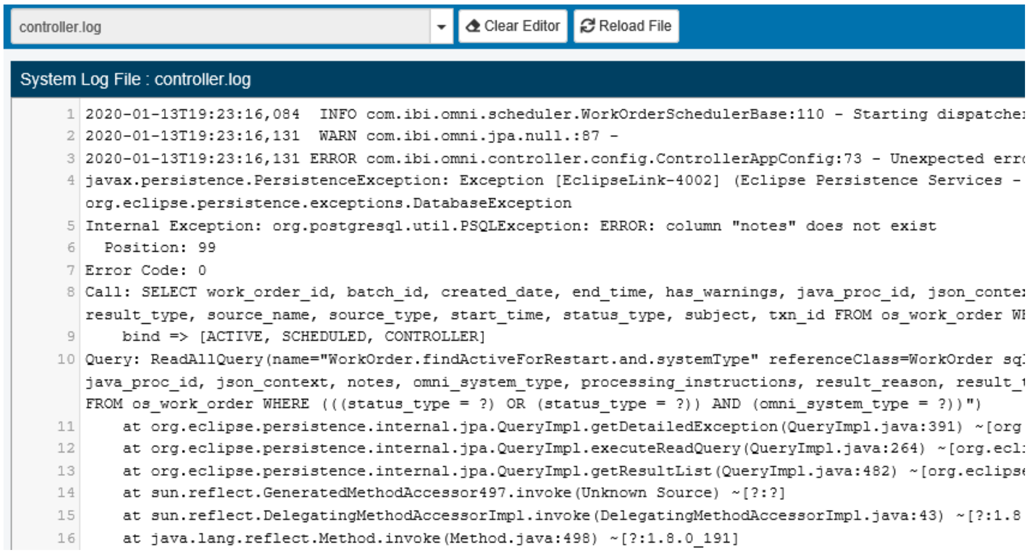


You can select a log file from the drop-down list to view details, as shown in the following image.



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.



```
2020-01-13T19:23:16,084  INFO com.ibi.omni.scheduler.WorkOrderSchedulerBase:110 - Starting dispatche
2020-01-13T19:23:16,131  WARN com.ibi.omni.jpa.null.:87 -
2020-01-13T19:23:16,131 ERROR com.ibi.omni.controller.config.ControllerAppConfig:73 - Unexpected err
javax.persistence.PersistenceException: Exception [EclipseLink-4002] (Eclipse Persistence Services -
org.eclipse.persistence.exceptions.DatabaseException
Internal Exception: org.postgresql.util.PSQLException: ERROR: column "notes" does not exist
  Position: 99
Error Code: 0
Call: SELECT work_order_id, batch_id, created_date, end_time, has_warnings, java_proc_id, json_conte
result_type, source_name, source_type, start_time, status_type, subject, txn_id FROM os_work_order W
    bind => [ACTIVE, SCHEDULED, CONTROLLER]
Query: ReadAllQuery(name="WorkOrder.findActiveForRestart.and.systemType" referenceClass=WorkOrder sq
java_proc_id, json_context, notes, omni_system_type, processing_instructions, result_reason, result_
FROM os_work_order WHERE (((status_type = ?) OR (status_type = ?)) AND (omni_system_type = ?))")
    at org.eclipse.persistence.internal.jpa.QueryImpl.getDetailedException(QueryImpl.java:391) ~[org
    at org.eclipse.persistence.internal.jpa.QueryImpl.executeReadQuery(QueryImpl.java:264) ~[org.ecl
    at org.eclipse.persistence.internal.jpa.QueryImpl.getResultList(QueryImpl.java:482) ~[org.eclips
    at sun.reflect.GeneratedMethodAccessor497.invoke(Unknown Source) ~[?:?]
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43) ~[?:1.8
    at java.lang.reflect.Method.invoke(Method.java:498) ~[?:1.8.0_191]
```

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
deployment bundle service	Managed Services	Deployment	Log Directory	server.bundler.log-directory
omni server	Managed Services	Server	Log Directory	server.omni-server.log-directory
repository server	Managed Services	Repository	Log Directory	server.repository.log-directory

File Organization

For convenience, commonly-used log files generated by most Omni-Gen processes can be found in the OmniGenData/logs directory, inside the Omni-Gen 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.
- 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.

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 from the controller service.

Application	File	Location	Archival	Description
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.
installer	Omnigen_install_ [timestamp].log	install/ Omnigen_install_lo gs	Timestamped	Detailed OmniGen installer messages and output.
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.

Advanced Logging Options

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

Standard Logging Support

Omni-Gen 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
	omni server	OmniGenData/OmniServer/cmd/conf/log4j-server.xml
Spring Boot	deployment bundle service	deploymentbundle/application.properties
Apache Tomcat	repository server	OmniGenData/OmniDesignerRepository/conf/logging.properties

Log4J Logging Considerations

Omni-Gen 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-Gen 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

Appendix A

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

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.*

Information Builders

iWay

Omni-Gen™ Integration Edition Operation and Management Guide

Version 3.14

DN3502358.0220

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