Information Builders

iWay

Omni-HealthData™ Integration Services User's Guide

Version 3.11

November 13, 2019

Contents

Preface

This documentation provides the details needed to understand and use the Omni-HealthData™ Relational OnRamp. It assumes an understanding of Omni concepts and familiarity with relational databases. The abbreviation, ROR, will be used in this documentation to stand for Relational OnRamp, where convenient.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	Contents
1	Omni-HealthData Integration Services	Describes how to load source data into an Omni system.
A	Using the Omni Command Line Tool	Describes how to use the Omni command line tool to generate test data in OnRamp tables.

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 *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-HealthData Integration Services

This documentation provides the details needed to understand how to load source data into an Omni system.

In this chapter:

Omni Integration Overview

Omni-HealthData™ is a platform that enables the creation and maintenance of cleansed and mastered data for a business domain. A data model, reflecting the business requirements of a customer, is deployed into Omni-HealthData. This data model is divided into *subjects*, each of which is described by a document hierarchy. Specifically, data for a particular subject is represented by a set of root instances, each logically linked to any number of descendants (for example, child, grandchildren, and so on).

The data model is packaged as a .zip file and formatted as an Omni *bundle*. For Omni products targeted for specific industry verticals, such as Omni-HealthData™, the data model is pre-packaged as a bundle with the product. Otherwise, the data model is designed, and the bundle is generated, on site. Deploying an Omni bundle to be used for a particular data model, creates various artifacts and utilities for testing and integration with the Omni-HealthData system. Once initialized, data may be fed into an Omni-HealthData system. In particular, each subject is described by an IDS specification.

There are two facilities available for data to be added:

- ❑ **Relational OnRamp.** Sets of records to be loaded to an Omni-HealthData system are first staged in relational tables.
- ❑ **XML.** Sets of records to be loaded to an Omni-HealthData system are first assembled as XML-formatted Omni Interface Documents (OIDs).

A spreadsheet, generated by the model deployment for each subject, provides the names of the ramp tables, attributes, descriptions, and formats of the ramp representation. Similarly, an XML schema defines the XML interface for each subject.

Typically, the data being loaded into an Omni-HealthData system originates from source systems, which are formatted differently from the target model. As a result, a conversion step is required to bring this data to an Omni-HealthData system.

Integration With the Relational OnRamp

The Omni-HealthData Relational OnRamp is a set of relational tables and protocols used to load data into an Omni system. This section describes how data can be integrated with the Relational OnRamp.

Tables

Any subject of an Omni system is logically represented as a hierarchy of documents described by the IDS: the top-level subject document and child instance documents, or collections. For each such document, parent or child, there is a corresponding ROR table with name, nameOfSubjectOrCollection_r. For example, the Facility subject contains a Facility parent and an Address child, which are represented and loaded using the facility_r and facility_address_r tables, respectively.

When a new bundle is deployed, this will initialize empty OnRamp tables.

Table Mappings

Omni products provide an Excel workbook for each IDS subject, which describes the layout of the ramp and model tables. In the Excel workbook, the top-level subject and each child collection are described by separate tabs. Table mappings can be obtained from the Deployment area of the Omni Console, where a *Download Documentation* option is available.

Table Keys

Each OnRamp table has a group of identifying columns, which taken together, map to the source record. These columns will be part of the primary key.

For top-level subjects, the identifying columns are:

- **source_name.** Name of the source system (for example, source_name = TestEpic).
- **subject_sid (Source Instance ID).** Unique Business identifier of the Subject within the named source system (for example, facility_sid = 123).

For child collections, the identifying columns are:

- **source_name.** Same as the source name of the parent (for example, source_name = TestEpic).
- **top_level_subject_sid.** Unique identifier of top-level parent subject (for example, facility_sid = 123).
- **intermediate_level_collection_sid [0-n].** Identifiers of each intermediate collection separating the child from the root subject. There is one sid per level. (for example, facility_child_sid = CHILDTYPE1, facility_grandchild_sid = GRANDCHILDTYPE1, and so on).
- **child_sid.** Unique Business identifier of the child collection object within the hierarchy defined by the key (for example, a discriminator with respect to the parent or facility_address_sid = PrimaryLocation).

Equivalently, the identifying columns of a child collection may be defined recursively as the combination of the following:

- Identifying columns of the direct parent.
- sid of the child.

Finally, the full primary key for any of these tables, whether top-level subject, or child collection, is comprised of the identifying columns described above and the batch_id column, which is present in each ROR table.

Non-Key Data Columns

The following are conventions for forming OnRamp fields from their IDS names, based on their type.

- **String, Double, Integer fields.** <CamelCase> becomes <lower_case>, as before.
- **Date/Time fields (idsIdsDateTimeType).** It is recommended that date and time elements terminate with tokens *Date*, *Time*, or *DateTime* to indicate the desired precision. If these tokens are found, they are converted to lowercase in the OnRamp field name using the usual transformation.
- **Codes.** Three columns are used to comprise the code reference:

```
<lower_case> + _src
<lower_case> + _set
<lower_case> + _val
```

- **Links.** The element name changes from <CamelCase> to <lower_case> and takes the following suffixes and links:
    - **Sample 1.** Link is defined to refer to a single subject.

      ```
      <lower_case> + _subjectName + _snm
      <lower_case> + _subjectName + _sid
      ```

    - **Sample 2.** Link is defined to refer to one of several possible subjects.

      ```
      <lower_case> + _subjectName +_sbj
      <lower_case> + _subjectName + _snm
      <lower_case> + _subjectName + _sid
      ```

- **Group Names.** Elements within groups work unchanged.

  ```
  <CamelCase>
  ```

  becomes

  ```
  <grp_id> + <lower_case>
  ```

Additional Columns

- **batch_id.** This is part of the key for each OnRamp table. Every OnRamp row is associated with an input batch.
- **onr_created_datetime.** Not currently implemented. If the integrator sequences entries in a batch by this field, the OnRamp may in the future use it for recovery.

Source Codes

Source Codes and Source Code Sets are loaded through the OnRamp tables like other subjects. The top-level subject to use when submitting a batch with either or both is SourceCodeSet. However, the IDS structure for SourceCode and SourceCodeSet is slightly different from standard subjects, so the following considerations apply.

- Duplicate naming of fields
    - **source_code_set_r.** Source_code_set_sid and code_set_name must be populated with duplicate values of the code set name.
    - **source_code_r.** Source_code_sid and source_code must be populated with duplicate values of the source code.
- Source Codes reference their sets as part of the key. The records for these sets (from source_code_set_r) must either be included in the OnRamp batch with the source codes or have already been loaded.

- Omni-HealthData uses standard codes (SourceCodeStandard). SourceCodeSet and data that references those codes, including SourceCodeStandard, can be loaded in any order.

OnRamp Operational Specification

This section describes how various integration scenarios will be supported through the relational OnRamp in Omni-HealthData™. Specifically, the purpose of this section is to make the rules governing how the OnRamp changes Omni Source tables to explicit, and to describe how these changes relate to and reflect different integration tasks. It assumes an understanding of the structure of a ramp table, for example, its keys, the mapping of Omni types to columns, and the hierarchical organization of the different ramp tables belonging to a subject.

The concept is that of a ramp batch, which is a set of records in relational ramp tables aggregated by means of a batch_id and then submitted as a data update by an integrator.

Each record in the OnRamp table must correspond to a unique record in the source system. This is enforced by the OnRamp table keys.

Omni-HealthData maintains *source* tables (identified by a _s suffix), which contain the latest version of a given source instance. Source and ramp instances are compared with each ramp batch in order to prevent instances, which have not materially changed from being reprocessed. For any ramp operation, the resulting state of the Omni Source tables is dependent on:

- The prior state of the Omni Source tables.
- The input ramp batch data.
- The input batch operational instructions (batch type, parameters, and options).

Integration Modes

Two integration modes can be usefully distinguished:

- **Full Data.** The integrator provides a *complete set of replacement data* for a subject from one or more source systems. Full data loads can be used for initial loads or subsequent loads. This mode is suitable when source data is provided cumulatively. New records are added to the repository, existing records are replaced, and records no longer present are deleted.

- **Incremental Data.** The integrator provides a set of data consisting of *only the records that are to be changed*, for example, data to be added, modified, or deleted for a subject from one or more source systems. Previously loaded Omni data not referenced by an Incremental Data batch (directly or indirectly) are left unchanged.

Batch Types

A set of operations are available to support integration needs. These operations are specified by batch type and are given as an execution parameter on a particular batch. The following table provides a high-level summary of the supported batch types and the scenarios in which they are used.

Batch_type	Integration Mode	Scenario
UPSERT	Incremental	The integrator presents a set of records for a given subject to be added or replaced.
INSERT_ONLY	Incremental	The integrator presents a set of NEW instances or instance hierarchies (no implied hierarchies) to add to a given subject.
REPLACE_SELECTED	Incremental	The integrator presents a set of records for a given subject to be added or replaced.
REPLACE_ALL	Full	The integrator presents a complete replacement set for a given subject.
DELETE	Incremental	The integrator presents a subset of records for a given subject to be (logically) deleted from the repository.

Detailed Batch Types

This section describes the various batch types in detail.

UPSERT

Use cases:

- ❑ The integrator can access or compute incremental data from a source system.
- ❑ The integrator wants to replace a set of instances or instance children (no implied hierarchies).
- ❑ Migration path for OID policy="merge".

When an UPSERT batch is processed:

- ❑ New instances from the batch are added.
- ❑ Pre-existing instances (top level or subcollection) changed in any way by the batch are replaced. NULLS are handled according to the upsertNullHandling option with values PRESERVE and IGNORE. PRESERVE is the default.
- ❑ Pre-existing instances not in the batch are unchanged.

The following table shows the status of a source instance after an UPSERT operation.

Instance in Ramp?	**Instance in Source?**	**Source Status Before Operation**	**Source Status After Operation**
Yes	Yes	ACTIVE	ACTIVE
Yes	Yes	INACTIVE	ACTIVE
Yes	No	N/A	ACTIVE
No	Yes	ACTIVE	ACTIVE
No	Yes	INACTIVE	INACTIVE

Guidelines for a batch composition

The set of instances in an UPSERT batch can update any existing instances. New instances must be introduced as complete hierarchies, either from the subject root of the child or an already existing hierarchy. This condition will be satisfied when one of the following holds for each instance in the batch:

- The instance is a top-level instance.
- The instance is a child instance and all its ancestors are in the batch OR all its ancestors are in Omni Source.

INSERT_ONLY

Use cases:

- The integrator wants to perform an initial load of subjects into the Omni database.
- The integrator wants to add a set of instance records not yet known to be in the Omni database.

Note: In some cases, INSERT_ONLY with truncateBeforeInsert = TRUE may outperform REPLACE/UPSERT operations (this is only appropriate for transactional subjects for which capturing changes between loads in unnecessary).

The following table shows the status of a source instance after an INSERT_ONLY with truncateBeforeInsert = FALSE operation.

Instance in Ramp?	Instance in Source?	Source Status Before Operation	Source Status After Operation
Yes	Yes	ACTIVE	*INSERT_ONLY operation fails with error
Yes	Yes	INACTIVE	*INSERT_ONLY operation fails with error
Yes	No	N/A	ACTIVE
No	Yes	ACTIVE	ACTIVE
No	Yes	INACTIVE	INACTIVE

The following table shows the status of a source instance after an INSERT_ONLY with truncateBeforeInsert = TRUE operation.

Instance in Ramp?	Instance in Source?	Source Status Before Operation	Source Status After Operation
Yes	Yes	ACTIVE	ACTIVE
Yes	Yes	INACTIVE	ACTIVE
Yes	No	N/A	ACTIVE
No	Yes	ACTIVE	N/A
No	Yes	INACTIVE	N/A

Guidelines for a batch composition

The set of instances in an INSERT_ONLY batch must be introduced as complete hierarchies, either as the subject root or the child of an existing hierarchy.

REPLACE_SELECTED

Use cases:

- ❑ The integrator can access or compute incremental data from a source system.
- ❑ The integrator wants to replace a full hierarchy of an instance or instance child.
- ❑ Migration path for OID policy="replace".

When a REPLACE_SELECTED batch is processed:

- ❑ New instances from the batch are added.
- ❑ Pre-existing instances (top level or subcollection) changed in any way by the batch are fully replaced. NULLS override existing values.
- ❑ Pre-existing instances not in the batch, such as:
  - ❑ Top level: No action
  - ❑ Child level: If an ancestor is in the batch, it is deleted. Otherwise, no action is taken.

If an instance or instance child is included in a REPLACE_SELECTED batch, all descendants which are still required must also be included as part of the batch. Any omitted descendants are deleted.

The following table shows the status of a source instance after a REPLACE_SELECTED operation.

Instance in Ramp?	Instance in Source?	Source Status Before Operation	Source Status After Operation
Yes	Yes	ACTIVE	ACTIVE
Yes	Yes	INACTIVE	ACTIVE
Yes	No	N/A	ACTIVE
No	Yes	ACTIVE	ACTIVE, if no direct ancestor instance is in ramp. Otherwise, INACTIVE.
No	Yes	INACTIVE	INACTIVE

Guidelines for a batch composition

The set of instances in a REPLACE_SELECTED batch must form fully connected hierarchies, but not necessarily from the subject root. No two hierarchies within the batch, however, can share the same subject root. This condition will be satisfied when one of the following holds for each instance in the batch:

- ❑ The instance is a top-level instance.
- ❑ If the parent of the instance is not in the batch, then both:
    - ❑ All the ancestors of the instance *are not* in the batch.
    - ❑ All the ancestors of the instance *are* in Omni Source.

REPLACE_ALL

Use Cases:

The data from a source system is only available in its entirety. The integrator supplies an initial load or a complete replacement load for the source system.

When a REPLACE_ALL batch is processed:

- ❑ New instances from the batch are added.

- Pre-existing instances (top level or subcollection) changed in any way by the batch are replaced.
- Pre-existing instances (top level or subcollection) not in the batch are deleted.

The following table shows the status of a source instance after a REPLACE_ALL operation.

Instance in Ramp?	Instance in Source?	Source Status Before Operation	Source Status After Operation
Yes	Yes	ACTIVE	ACTIVE
Yes	Yes	INACTIVE	ACTIVE
Yes	No	N/A	ACTIVE
No	Yes	ACTIVE	INACTIVE
No	Yes	INACTIVE	INACTIVE

Source system isolation: A source system name or comma-delimited list of source systems names, is given in a column on ramp_control. These system names become a filter to the RampToSource function. This supports loading from different source systems on separate schedules. No activity will occur for source systems not on the list.

Hazard: Mass deletion will result if a batch is launched in error with partial data. The missing section will be made inactive. A subsequent full REPLACE_ALL will restore the integrity of the repository. As a precaution against a misspelled batch_id, no records will be changed if an empty REPLACE_ALL batch is received.

Guidelines for batch composition

The set of instances in a REPLACE_ALL batch must form fully connected hierarchies from the subject root. This will be the case when all instances in the batch satisfy one of the following:

- The instance is a top-level instance.
- All the ancestors of the instance are in the batch.

DELETE

Use cases:

- The integrator selects instances of hierarchies to delete, either from the root level of a subject or a subcollection level.

When a DELETE batch is processed:

- ❑ New instances from the batch are ignored.
- ❑ Pre-existing instances (top level or subcollection) are marked as inactive. All descendants of the instances are marked inactive.
- ❑ Pre-existing instances are not in the batch:
  - ❑ Top level: no action
  - ❑ Child level: If the ancestor is in the batch, it will be marked as inactive. Otherwise, no action is taken.

The following table shows the status of a source instance after a DELETE operation.

Instance in Ramp?	Instance in Source?	Source Status Before Operation	Source Status After Operation
Yes	Yes	ACTIVE	ACTIVE
Yes	Yes	INACTIVE	ACTIVE
Yes	No	N/A	ACTIVE
No	Yes	ACTIVE	ACTIVE, if no direct ancestor instance is in ramp. Otherwise, INACTIVE.
No	Yes	INACTIVE	INACTIVE

Guidelines for a batch composition

There are no restrictions on a DELETE batch.

Deleting Omni Data

A delete instruction for a particular instance (either top level or subcollection) is an indication that it should no longer be considered a logical member of the subject set. This instance is then given inactive status. The consequences of an instance becoming inactive this way are outside the scope of this document but include suppression of cleansing, possible mastering reconciliation and removal from the visibility of consumption views, and the Management Console. The history of the inactive instance should be maintained and the contents of the instance, aside from being marked inactive, are otherwise left intact.

An instance (top level or subcollection) will be deleted by the OnRamp as the result of the following operations:

- The key for the instance is referenced in a DELETE batch.
- The key for the parent of the instance is referenced in a DELETE batch.
- The key for the instance is omitted in a REPLACE_ALL batch.
- The key for the instance is omitted in a REPLACE_SELECTED batch, and the key for an ancestor of the instance is referenced by the batch.

An inactive instance can be returned to active status if it appears in a subsequent REPLACE_ALL, REPLACE_SELECTED, or UPSERT batch.

Inactive instances may be scheduled for physical removal by using the Purge Inactive Time and Purge Inactive Age settings.

For more information on purging inactive data, see the *Omni-Gen™ Operation and Management Guide*.

Orphan Omni Data

Orphan Omni data are active child instances that have at least one ancestor that is either inactive or does not exist. Using the different ramp operations, you can introduce orphan records into Omni-HealthData. These orphans are either inaccessible or have undefined behavior with respect to Omni-HealthData processing. If the usage guidelines given below for each of the ramp operations are followed, orphan records will not be introduced.

Nulls

The upsertNullHandling option controls the handling of a NULL column input for the batch_type UPSERT.

- **PRESERVE.** Preserves the existing value.
- **OVERRIDE.** Replaces the existing value with NULL.

If neither option is specified, PRESERVE is used as the default.

Note: Non-null columns always replace existing values.

OnRamp Operational Details

This section describes the OnRamp operational details.

Parallelism

The Relational OnRamp will process batches for different top-level subjects concurrently. Multiple batches for the same top-level subject will be processed serially.

Launching Relational OnRamp Batches

Omni-HealthData supports the following methods of launching an OnRamp batch:

- Directly updating *os_ramp_control*.
- Calling the *ProcessRamp* web service.

Directly Updating os_ramp_control

You can add a row to *os_ramp_control*, setting the following columns.

- **batch_id (string).** Batch to execute.
- **subject (string).** Subject to process as part of the batch.
- **source_name (string).** Restrict the source system to participate in the batch.
- **batch_type (string).** Processing mode. [UPSERT, INSERT_ONLY, REPLACE_SELECTED, REPLACE_ALL, DELETE]
- **batch_options (string).** Deprecated. Use data_transfer_mode, change_detection, upsert_null_handling, and truncate_before_insert columns, instead.
- **load_type (string).** Deprecated.
- **data_transfer_mode (string).** [JPA, Native_SQL]. JPA - database neutral, Native_SQL - high-performance database specific. Applies to SQL Server and Postgres only.
- **change_detection (string).** [ENFORCE, IGNORE]. ENFORCE - identifies business duplicates and suppresses them from further processing. IGNORE - all records in batch are fully processed (including duplicates). Useful for error recovery.
- **upsert_null_handling (string).** [PRESERVE, OVERRIDE]. Governs how to handle missing values on an update (replaces deprecated batch options PRESERVE_ON_NULL and OVERRIDE_ON_NULL). Applies to UPSERT only.
- **truncate_before_insert (string).** [FALSE, TRUE]. Applies to INSERT_ONLY mode only. Truncate data before an INSERT_ONLY operation.
- **state (string).** Set to READY when batch is fully assembled.

- **active (string).** Set to *Y* to enable the subject/batch to be loaded.

The *os_ramp_control* can be monitored to follow the status of a ramp batch.

The following are possible values of the *state* field:

- **PENDING.** Indicates the ramp will begin loading.
- **LOADING.** Indicates the ramp is being loaded.
- **READY.** Load complete and ready for processing.
- **SCHEDULED.** Schedules for processing.
- **PROCESSING.** Omni is currently processing the subject batch.
- **COMPLETE.** Omni completed processing of the subject batch.
- **FAILED.** Processing of the batch has failed.

Note: This information is also available in the RampControl section of the Omni Console.

Invoking the ProcessRamp Web Service

This section describes the *ProcessRamp* web services that can be used to launch a ramp batch.

V4 Web Service

`https://server_host:server_port/server/api/v1/server/processRamp.v4`

The HTTP PUT request accepts the parameters listed and described in the following table.

Parameter	Type	Description and Values
batchId	string	Batch to execute.
subject	string	Subject to process as part of the batch.
sourceName	string	Restrict the source system to participate in the batch.

Parameter	Type	Description and Values
batchType	string	Mode corresponding to the *batch_type*: ❑ UPSERT (default) ❑ INSERT_ONLY ❑ REPLACE_SELECTED ❑ REPLACE_ALL ❑ DELETE

The following table provides the batch options (sub directives) for the process ramp service and the corresponding *batch_options* column in *os_ramp_control*.

Name	Type	Description and Values
dataTransferMode	string	Overrides the default data transfer mode runtime configuration setting. ❑ JPA (default) Database neutral. ❑ Native_SQL High performance, bulk database-specific. Currently supported for SQL Server and PostgreSQL. **Note:** Native_SQL generates a runtime error for unsupported databases.

Name	Type	Description and Values
changeDetection	string	Determines whether or not to eliminate duplicates from further processing (data quality operations). ❑ ENFORCE (default) Eliminates the duplicates. ❑ IGNORE Guarantees processing of all records in a batch. Useful at certain times for error recovery. **Note:** ChangeDetection = IGNORE replaces batch option *FORCE_REPROCESS*.
upsertNullHandling (previously *addUpdateNullHandling*)	string	UPSERT only. Governs how to handle missing values on an update. ❑ PRESERVE (default) ❑ OVERRIDE

Name	Type	Description and Values
truncateBeforeInsert	boolean	Truncate data before an INSERT_ONLY operation. Used for a full replacement of the data set. Applies only to transactional subjects. ❏ FALSE (default) ❏ TRUE **Note:** Is an error unless loadType is INSERT_ONLY.

Integration With XML (Omni Interface Document)

All subjects of a deployed model in an Omni-HealthData system are described by an Omni Interface Document (OID). This is a document in XML format by which records may be added, modified, or removed from the system. From the Deployments area in the Omni Console, you can download the schema of a particular OID for each subject along with a sample instance document.

The OID represents an instance of the subject as a full hierarchy (the root object and all descendants).

OID Element Types

Each business attribute in an OID may be represented by various character, numeric, and data types as specified by the schema. There are also several Omni-defined constructs that control parent/child relationships, links to other subjects, and codes and groups.

Instance Child

One or multiple instances of a child collection can be included in an OID.

Example:

```
<Client>    // subject name

      <OtherPhoneNumbers>    // enclosing element of child collection –
list in IDS and mapping document

      <PatientPhoneNumber version="3.0.0">    // subcollection name
        <SourceName>System_A</SourceName>    // subcollection source name
        <SourceInstanceId>1</SourceInstanceId>    // subcollection
identifier
        <Type sourceName="OMNI" codeSet="0185">H</Type>
        <StartDate format="yyyy-MM-dd">2018-01-15</StartDate>
        <EndDate format="yyyy-MM-dd"></EndDate>
        …
      </PatientPhoneNumber>

     <PatientPhoneNumber version="3.0.0">
        <SourceName>System_A</SourceName>
        <SourceInstanceId>2</SourceInstanceId>
        <Type sourceName="OMNI" codeSet="0185">C</Type>
        <StartDate format="yyyy-MM-dd"></StartDate>
        <EndDate format="yyyy-MM-dd"></EndDate>
        …
      </PatientPhoneNumber>

    </OtherPhoneNumbers>

</Client>
```

Link to Another Subject

Example:

```
<PrimarySupplier>    // link element in schema
   <Supplier>    // linked subject type (subject name)
      <SourceName>System_A</SourceName>    // subject instance source system
      <SourceInstanceId>123</SourceInstanceId>    // subject instance
identifier
   </Supplier>
</PrimarySupplier>
```

Variable Link to Another Subject

Example:

```
<ChargeableEvent>    // variable link element in schema
      <Subject>EventSubject1</Subject>    // linked subject type (subject
name)
      <SourceName>System_B</SourceName>    // subject instance source
system
      <SourceInstanceId>456</SourceInstanceId>    // subject instance
identifier
</ChargeableEvent>
```

Code

Example:

```
<AccountType sourceName="System B" codeSet="AccountTypes">C1</AccountType>
```

where:

AccountType

Is the code element in the schema.

sourceName="System B"

Is the code source system.

codeSet="AccountTypes”

Is the code set name.

C1

Is the code value.

Boilerplate Fields/Elements

The following table lists and describes the supported elements that can be configured within OIDs.

Element	Required?	Description
SourceName	Required	Defines the source system from which the OmniObject originated. It is used in conjunction with the SourceInstanceId to uniquely identify the OmniObject.
SourceInstanceId	Required	A repeatable unique identifier which can be constructed from the available data in the originating system. It is used in conjunction with the SourceName to uniquely identify the OmniObject.
SourceInstanceIdName	Optional	This element can be used to give an indication of where the data came from in the source system to provide traceability.
SourceStatusCode	Optional	A user-defined status value, which may be used as a screening condition for reporting off of the model.
SourceCreatedDate	Optional	DateTime that the record was created in the originating system.

Element	Required?	Description
SourceCreatedBy	Optional	Description of the entity that created the record in the source system.
SourceModifiedDate	Recommended	DateTime that the record was modified in the originating system.
SourceModifiedBy	Optional	Description of the entity that modified the record in the source system.

SourceStatusCode is a user-defined status value. It may be used as a screening condition for reporting off of the model.

```
<SourceName>
<SourceInstanceId>
<SourceInstanceIdName>
<SourceStatusCode>
<SourceCreatedDate format="yyyy-MM-dd hh:mm:ss">
<SourceCreatedBy>
<SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">
<SourceModifiedBy>
```

OID Policies

An OID may direct the system to perform an upsert, replace, or delete operation on the instance data. The specific operation type is indicated by the *policy=" "* attribute on the root element of the OID.

Example:

```
<Member xmlns="http://www.ibi.com/2013/OP/Interface" policy="replace">
```

The following operations correspond to the OnRamp batch types:

- Policy="merge" corresponds to an UPSERT with the upsertNullHandling option with value PRESERVE.
- Policy="replace" corresponds to REPLACE_SELECTED.
- Policy="delete" corresponds to DELETE.

By default, this operation will be an upsert (*policy="merge"*) if no policy is specified.

Batch Interface

It is possible to submit several OIDs as a logical batch. To do this, multiple OIDs are wrapped into an *OmniInterface* document. For example:

```
<OmniInterface>

  <Member>
     ….
 <Member>
  <Member>
     ….
 <Member>
    …
</OmniInterface>
```

OmniInterface documents are processed as text files that are read from the following folder:

```
omnigen\OmniGenData\input\oid\input
```

Successfully submitted documents are copied to the following folder:

```
omnigen\OmniGenData\input\oid\processed
```

Unsuccessful ones are copied to the following folder:

```
omnigen\OmniGenData\input\oid\error
```

An *OmniInterface* document may combine OIDs of different subjects and with different policies. However, all records in an *OmniInterface* document sharing the same subject and policy will be submitted together as sub-batches in no specified order.

Immediate Interface

Two REST-based services are available to submit OIDs to an Omni-HealthData system.

Load OID to Ramp

```
PUT "api/v1/server/bulk/load/{subject}"
```

Parameter	Description	Parameter Type	Data Type
xml	OID	Body (application/ xml)	String
subject	Subject name	Path	String

Parameter	Description	Parameter Type	Data Type
batchId	Batch ID of the Ramp record(s)	query	String

This service populates the appropriate Ramp tables (parent and child collections) with the data from the OID. Multiple OIDs can be loaded into the Ramp using the same *batchId* and submitted together. For more information, see the Ramp submission options.

The following Swagger interface is available for testing:

```
https://omni-host:9512/server/swagger-ui.html#!/Bulk32Processing/
bulkLoadUsingPUT
```

Process OID

```
POST "api/v1/server/processImmediate”
```

Parameter	Description	Parameter Type	Data Type
xml	OID	Body (application/xml)	String

This service takes a single OID and submits it to an Omni-HealthData system.

The following Swagger interface is available for testing:

```
https://omni-host:9512/server/swagger-ui.html#!/Processing/
processImmediateUsingPOST
```

Comparing OnRamp and XML Interfaces

- OnRamp is native to Omni-HealthData. OIDs are converted into Ramp batches prior to being submitted.
- Full hierarchies are collected in a single document with OIDs. They are separated into different tables for the Ramp.
- OnRamp allows for incremental upserts only at the child level.
- OnRamp supports REPLACE_ALL and the upsertNullHandling option with values PRESERVE or OVERRIDE.

Migration Consideration for OID Users in Omni-HealthData Version 2.x

In Omni-HealthData version 2.x, the SourceInstanceId of an instance child was required to be unique within the source system. In Omni-HealthData version 3.x, the SourceInstanceId of an instance child must be unique with respect to its parent instance.

Appendix A

Using the Omni Command Line Tool

This chapter describes how to use the Omni command line tool to generate test data in OnRamp tables.

In this appendix:

Generating Test Data in RAMP Tables

This command line tool generates random data in RAMP tables for a given subject. From the OmniServer installation location, run omni.cmd on Windows or omni.sh on Linux, as shown in the following syntax:

```
>omni.cmd createTestRampData -Dramp.subject=Person -Dramp.batch=1
-Dramp.count=100 -Dramp.withOrphans=yes
–Dgroup.number=20 –Dvalues.file=C:/person-values.properties
```

where:

ramp.subject

Is required and case-sensitive. It specifies the subject test data to be created.

ramp.batch

Is optional and the default value is 1.

ramp.count

Is the number of records to be generated. The default value is 10.

ramp.withOrphans

Is optional and the default is set to *False*. If this value is set to *yes*, *y*, or *true*, then orphans will be created.

values.file

Is optional, but it is used to specify values for some fields.

group.number

Is optional, but specifies the number of master records you want the test data to be grouped for the subject. If it is set, the matching field names need to be specified in the *values.file*.

A generated subject entity has random values for most of the fields. However, there are some exceptions:

- test_system is used for sourceName.
- IdsCodeType fields are set with test_system as the source, where field names plus "s" are code sets, and field names plus "_code" are code values.

More specific instructions for the values to use specific fields may be accomplished by referencing a "values.file".

The following syntax shows a sample values.file.

```
personaddress.city=Staten Island|New York|Brooklyn
personaddress.zip=10304|10121|11209
person.dob=2018-11-25T15:24:30.294+0400|2016-12-31T8:24:30.005|
2017-2-2T3:8:5
person.maritalStatus=source1:set1:value1|source2:set2:value2
person.ssn=
```

Lists of Individual Values

The fields specified in the *values.file* option will have one of the values from a ''|'' separated list. The format of *date* type is listed as *person.dob*. For the IdsCodeType type, the following should be separated by colons (:), as well as person.maritalStatus:

- source
- set
- value

The "|" separated values will be used sequentially. If the number of the record is larger than the size of the list, then the list values will be reused starting from the beginning.

You can provide the list of values for group matching fields. The number of values should be equal to the number of groups you expect. If the values are not provided for a field, for example, *person.ssn=*, the tool will use random values for the field and all the records in the same group will have the same value. Therefore, you do not have to specify the values for matching fields if the number of matching groups is larger.

Date Options

The following list describes the date options that are used in testing data in RAMP.

- **Individual Dates.** Date value in one of the accepted formats. For example:

  ```
  Encounter.ArrivalDateTime=2015-01-01
  ```

- **Date Range.** Low and high dates that are given. The utility selects randomly from within the range. For example:

  ```
  Encounter.ArrivalDateTime=(2015-01-01,2017-10-20)
  ```

- **Derived Date.** A specified base date and interval. The derived date is computed from the base date and an interval or an interval range. The use case is to compute departure dates which occur after arrival dates.

The format for interval values can be found at:

https://docs.oracle.com/javase/8/docs/api/java/time/Duration.html#parse-java.lang.CharSequence-

Related to ISO-8601, the following examples are offered:

- **PT20.345S** – Parses as *20.345 seconds*.
- **PT15M** – Parses as *15 minutes* (where a minute is 60 seconds).
- **PT10H** – Parses as *10 hours* (where an hour is 3600 seconds).
- **P2D** – Parses as *2 days* (where a day is 24 hours or 86400 seconds).
- **P2DT3H4M** – Parses as *2 days, 3 hours, and 4 minutes*.
- **P-6H3M** – Parses as *-6 hours and +3 minutes*.
- **-P6H3M** – Parses as *-6 hours and -3 minutes*.
- **-P-6H+3M** – Parses as *+6 hours and -3 minutes*.

Example 1:

```
Encounter.ArrivalDateTime=(2015-01-01,2017-10-20)
Encounter.DepartureDateTime=[ArrivalDateTime,PT10M]
```

The departure date is 10 minutes after the arrival date.

Example 2:

```
Encounter.ArrivalDateTime=(2015-01-01,2017-10-20)
Encounter.DepartureDateTime=[ArrivalDateTime,(PT10M,P4D)]
```

The departure date is in range between 10 minutes and 4 days of the arrival date.

Code Options

This section describes the code options that are found when generating test data in RAMP.

- **Individual Codes.** Codes that are indicated by Source:Set:Value. For example:

  ```
  Patient.BiologicalSex=OMNI:0001:M|OMNI:0001:F
  ```

  When constructing test data, it is recommended to have codes come from a specific set.

- **Codes from a SourceCodeSet.**

  CodeSet is indicated by 'SourceCodeSet':SourceName:SetName and optionally followed by a count. The count is the number of codes from the CodeSet to be used in populating the test data. If no count is specified, then all of the codes in the CodeSet are used. The CodeSet must be populated in the instance model to be used by the command line tool.

  Format:

  ```
  SourceCodeSet:SourceName:SetName[,count]
  ```

  Example:

  ```
  Encounter.HospitalService=SourceCodeSet:OMNI:0069
  ```

- **Codes from a SourceCodeMap.** You can select codes that are aggregated in a value set. The SourceCodeMap representing the ValueSet must be populated in the instance model to be used by the command line tool.

  Format:

  ```
  SourceCodeMap:  SourceCodeMap:SourceName:MapName[,count] (ValueSets
  Only)
  ```

  Example:

  ```
  DiagnosisEvent.Diagnosis= SourceCodeMap:NIH-VSAC:
  2.16.840.1.113883.3.464.1003.103.12.1001^DiagnosisCode,50
  ```

  This uses 50 codes from the Diabetes Diagnosis codes map.

Link Options

When constructing test data, it is recommended to populate links with actual records in the Omni Model. A count, if provided, is the number of subject instances to be used in populating the test data. If there is no count, then all instances for the subject are used. The records need to already be in the model.

Format:

```
OMNILINK:Subject[,count]
```

Example:

```
Encounter.Patient=OMNILINK:Patient,7
```

Note: A pool of different sets and codes can be combined for a given field. For example:

```
ProcedureEvent.Procedure=SourceCodeSet:HLI-V3:2.16.840.1.113883.6.4|
SourceCodeSet:HLI-V3:2.16.840.1.113883.6.12|MySource:MySet:ExtraCode
Generate test data in RAMP to test DELETE, REPLACE_ALL, REPLACE_SELECTED,
PRESERVE_ON_NULL
```

A few additional parameters for the createTestData command are introduced to allow even more control over test data generation. The following table lists and describes those parameters.

Parameter	Description
`-Dramp.sid`	Specifies the prefix of all sids in the batch. If none is provided, then the sids are randomly generated. This parameter allows you to produce batches of matching resources, but with different randomly generated field values and different batch sizes.
`-Dramp.withMissingChildren`	The data generator randomly skips a generation of some collection resources. This would allow you to test REPLACE_SELECTED.

Note that not all field values are generated. Some field values are randomly set to *null* to allow PRESERVE_ON_NULL testing.

Generating OID Test Data

This command line tool generates Omni Interface Document (OID) sample XML files with random values for a specific subject. From the Omni Server installation location, run omni.cmd on Windows or omni.sh on Linux, as shown in the following syntax:

```
>omni createTestOIDFiles -Dsubject=Person -Dout.dir=c:/temp/OIDGeneration
-Dcount=4 -Dids.dir=C:/Temp/omni30_1/3.0.0-SNAPSHOT/OmniServer/bundle/
artifacts/server/IDS
-Dvalues.file=c:/temp/OIDGeneration/valueset.txt
```

where:

`subject`

Is required and is case-sensitive.

`out.dir`

Is optional, but defines where to save the test files. The default value is system property *user.dir*.

`count`

Is the number of OID files. The default is 10.

`ids.dir`

Is an optional directory where the IDS document of the subject is stored. The default is:

```
$install_location/OmniServer/bundle/generated/ids
```

`values.file`

Is optional, but can be used to specify a value for an element or attribute in the generated XML files, instead of random values. The file contains key=value pairs. A key is an Xpath of a particular element or an attribute in the XML file. Values are a list of values separated by a pipe (|). For example, the following are valid key=value pairs for the subject, *Person*.

```
/OmniInterface/Person/ssn=123456789|223456789|323456789|423456789
/OmniInterface/Person/maritalStatus/@codeSet=test_code_set
/OmniInterface/Person/names/PersonName/SourceStatusCode/
@codeSet=test_code_set
/OmniInterface/Person/names/PersonName/SourceStatusCode=test_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-HealthData™ Integration Services User's Guide

Version 3.11

DN3502336.1119

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