iWay Software
An Information Builders Company

iWay

Omni-Patient™ Server User's Guide

Version 2.3.x

DN3502213.0216

Contents

Preface

This documentation describes how to configure and use all of the facilities that are available through Omni-Patient™ Server.

How This Manual Is Organized

This manual includes the following chapters:

	Chapter/Appendix	Contents
1	Introducing Omni-Patient Server	Provides an introduction to Omni-Patient Server and describes key features and functionality.
2	Configuring Omni-Patient Server	Describes configuration tasks related to Omni-Patient Server.
3	Anatomy of Omni Interface Documents	Describes the anatomy of Omni Interface Documents (OIDs).

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.
underscore	Indicates a default setting.

Convention	Description
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.
...	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 the deployment information that our consultants require.

Adapter Deployment	
Container	
Version	
Enterprise Information System (EIS) - if any	
EIS Release Level	
EIS Service Pack	
EIS Platform	

The following table lists iWay-related information needed by our consultants.

iWay Adapter	
iWay Release Level	

iWay Patch	

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.	
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

- Service Manager package to reproduce problem
- Custom functions and agents in use
- Diagnostic Zip
- Transaction log

For information on tracing, see the *iWay Service Manager User's Guide*.

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*.

1 Introducing Omni-Patient Server

The *Omni-Patient Server User's Guide* is designed to provide integrators, support staff, customers, and partners a better understanding of Omni-Patient Server processing and integrating with Omni-Patient Server. Specifically, this documentation defines the following objectives:

- Understanding the high-level components of the Omni-Patient Server.
- Understanding the specific logical process flows of the Omni-Patient Server.
- Understanding how to communicate with the Omni-Patient Server.

Topics:

Omni-Patient Server Overview

The Omni-Patient Server is a vital component of the Omni-Patient product, implemented on the iWay Service Manager platform.

The primary roles of the Omni-Patient Server are as follows:

- Process Omni Interface Documents (OIDs) and persist Source Instance records to the Enterprise Mastered Information Repository (EMIR).
- Communicate with iWay Data Quality Suite to Cleanse, Match, and Merge mastered Subjects.
- Process and persist the Golden Record for each mastered Subject in the EMIR.
- Provide a data access layer from the EMIR to outside consumers, such as Omni Patient Management Central (OPMC).

The Omni-Patient Server accomplishes these tasks through three primary components:

- **Omni Interface Document (OID) Handler**

  Source data is communicated to Omni-Patient through the use of Omni Interface Documents (OIDs). The OID Handler processes these XML documents, communicates with iWay Data Quality Suite, and persists updates to the EMIR.

- **Remediation Request Handler**

  The Remediation Request Handler manages the interaction between the Advanced Remediation tools in Omni-Patient Management Central, and components of the Omni-Patient Server. It supports both synchronous and asynchronous interaction with Advanced Remediation.

- **OmniServices Layer**

  The OmniServices Layer exposes the Omni-Patient data model through a RESTful web service interface. The Primary consumer of the OmniServices Layer at the present time is Omni-Patient Management Central, the primary user interface of Omni-Patient. The 360 Viewer, Data Dictionary, and Advanced Remediation functions are all expected to be supported to some degree by the OmniServices Layer.

Enterprise Mastered Information Repository (EMIR) Overview

In this section:

The EMIR is the data store that results from the processing and mastering of the customer's data. It contains all the Instance records provided by the various Source Systems, as well as Golden records for all Mastered Subjects implemented across the enterprise.

This data store can be used to support further exchange of data, as well as enterprise level Business Intelligence and Analytics.

The EMIR is comprised of the following components:

- **Transactional subject tables:**
  - Subject Instance tables
  - Subject Instance history tables
- **Mastered subject tables:**
  - Subject Instance tables (foreign key that links instance to UEMID)
  - Subject Instance history tables
  - Subject Master tables (UEMID is primary key)
  - Subject Master history tables
- **Core tables:**
  - SourceCodeSet
  - SourceCode
  - SourceCodeMap
  - SourceCodeRelation
  - OmniWorking

Universal Enterprise Master Identifier (UEMID)

During the mastering process, the Golden master record that represents one or many merged instances is given a unique identifier, known as the Universal Enterprise Master Identifier (UEMID). Each instance for a given master will be updated with a reference to the appropriate UEMID during processing.

Note: The UEMID is the Primary Key of the Golden record for a given subject.

Omni Interface Document (OID) Handler

In this section:

The OID Handler is the primary processing engine of the Omni-Patient Server. It is the conduit through which data is loaded in the Enterprise Master Information Repository (EMIR).

Omni Interface Documents (OIDs) are the primary means by which Source Instances are loaded into the EMIR through the Omni-Patient Server. These eXtensible Markup Language (XML) interface documents, implement an XML Schema Definition (.xsd) called an Interface Document Specification (IDS). For more information, see *Anatomy of Omni Interface Documents* on page 41.

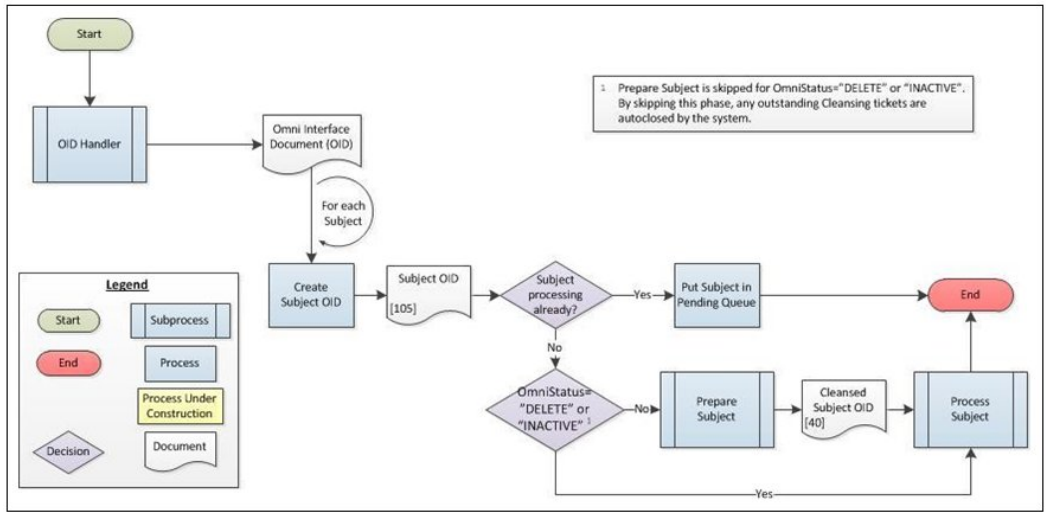
As a precursor to a deeper dive into the OID Handler, it is necessary to understand the various states of a Source Instance, since these will drive different processing paths.

The state of a Source Instance record is stored in the *OmniStatus* data element.

- **ACTIVE.** The default state.
  - Records are visible in Management Central
  - Records are eligible for consideration by the Matching Engine
  - Records are available for consumption by downstream applications and analytics.
- **INACTIVE.** Considered a *soft-delete* from the system.
  - Records are not visible in Management Central.
  - Records are not eligible for consideration by the Matching Engine.
  - Records are not available for consumption by downstream applications and analytics.
  - Records remain in the EMIR.

- **DELETE.** Marks the record for *hard-delete* from the system.
  - Records are not visible in Management Central.
  - Records are not eligible for consideration by the Matching Engine.
  - Records are not available for consumption by downstream applications and analytics on ACTIVE records.
  - Records remain in the EMIR until processed by the Delete Handler.

With a basic understanding of the OmniStatus for Source Instance records, it is possible to explore a more detailed view of the functionality of the OID Handler, as illustrated in the following diagram:



Since OmniInterface supports the ability to submit multiple subjects in a single payload, the OID must be split to define a Subject-level OID for each instance. To eliminate possible race conditions or database locks, the system also ensures that no more than one version of the same instance is processing at any one time.

If the Subject OID happens to currently be processing through the engine, it is temporarily placed into a Pending Queue for subsequent processing. For "ACTIVE" records that do not go to the Pending Queue, the values submitted by the source will be validated, standardized, and cleansed during the *Prepare Subject* subprocess, as configured for each specific implementation.

If any issues are identified through the preparation of the document, a case will be created for the Subject Instance, and any necessary remediation tickets will be communicated to the Advanced Remediation engine for follow-up.

Since the point of an OmniStatus of "DELETE" or "INACTIVE" is to physically or logically remove the record from the system, it is not necessary to execute the *Prepare Subject* subprocess. The side effect of skipping this phase is that no cleansing remediation tickets will be created, and any outstanding tickets from the previous version of the document will be automatically closed by the system.

The pending queue will be evaluated upon completion of the *Process Subject* subprocess, to determine if any records are now eligible for further processing.

Prepare Subject

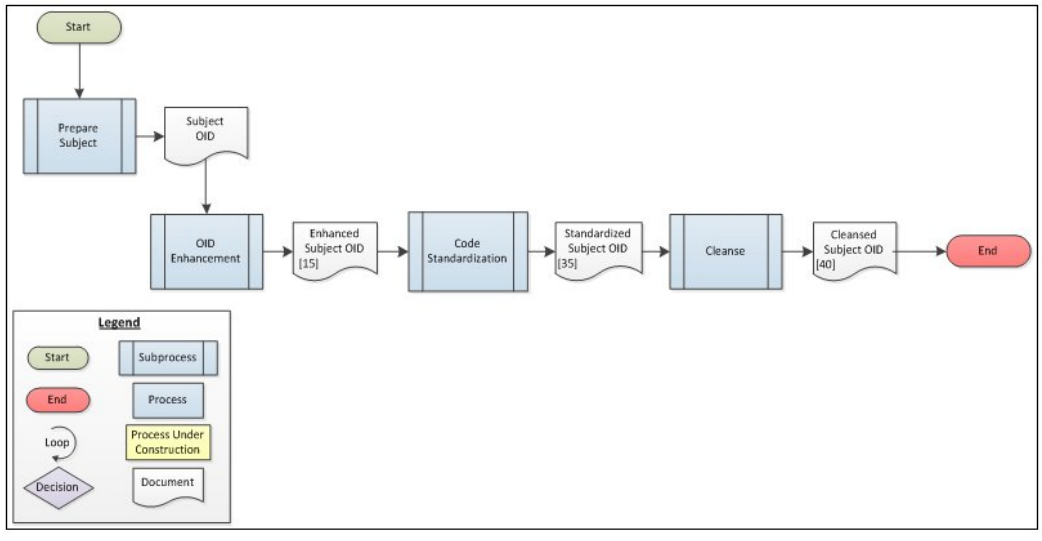
In this section:

The goal of preparing the subject is to enrich the document with the information that is required for mastering and persistence to the EMIR, as well as consumption by downstream applications or analytics.

The key features of the *Prepare Subject* subprocess will subsequently be discussed in further detail, but are listed as follows:

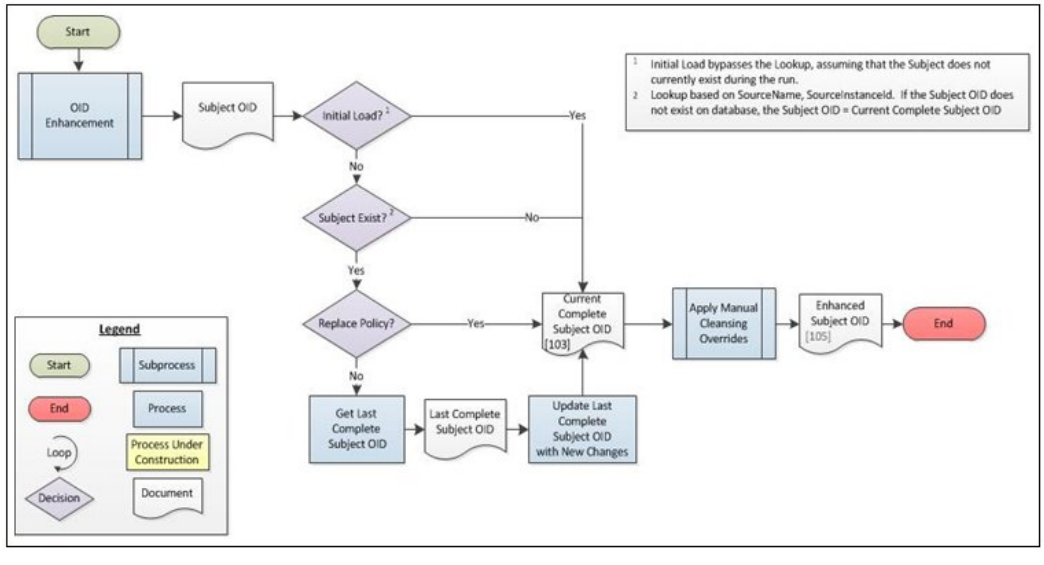
- **OID Enhancement.** Depending on the processing policy of the Subject, the document being processed will either replace the existing Subject, or will apply any updated data to the most recently processed complete Subject OID.
- **Code Standardization.** Converts code values from the various source systems to a standardized value set for use in Cleanse, Match, and Merge processes.
- **Cleanse.** Formats, validates, and standardizes values supplied from the source system.

The following diagram illustrates the process flow that is followed by the *Prepare Subject* subprocess.



OID Enhancement

The Omni-Patient Server is constructed to support standard CRUD (Create, Read, Update, Delete) operations. The initial enforcement of Create and Update policies are managed during the *OID Enhancement* subprocess, as shown in the following image.



During Initial Load of the system (as configured through a system property), it is assumed that all records will be Creates. That being the case, the system bypasses the document lookup and update, and effectively recognizes the submitted document as the complete document.

Note: During Initial Load, it is considered a system error if more than one instance of each unique Subject OID is submitted.

Beyond Initial Load, update operations are managed through a Policy configuration, which allows the record to be updated in whole or in part. The following Policy values may be specified globally, at the Source level, at the Subject level, or by Subject within Source:

- **Replace.** Takes the document *as is*, deleting any unspecified nodes from the prior version of the OID.
- **Merge.** Updates the most recent version of the OID with only those updates specified in the new document.

When the integrator configures a *Replace* policy, the document is taken *as is* for further processing, and data sent previously for the subject will be deleted.

The system may also be configured to accept updates in whole or in part, allowing a trading partner to send an update to even a single field under a *Merge* policy. When this type of update is provided, it is necessary to apply the latest changes to the last known Complete Subject OID in order to create the most current representation of the Subject.

As a final step to the *OID Enhancement* subprocess, Manual Cleansing Overrides (MCOs) are applied to the document. These overrides can be entered by a data steward in Omni Patient Management Central (OPMC) in response to a remediation ticket, and are simply a replacement for values submitted (or omitted) by the trading partner.

Code Standardization

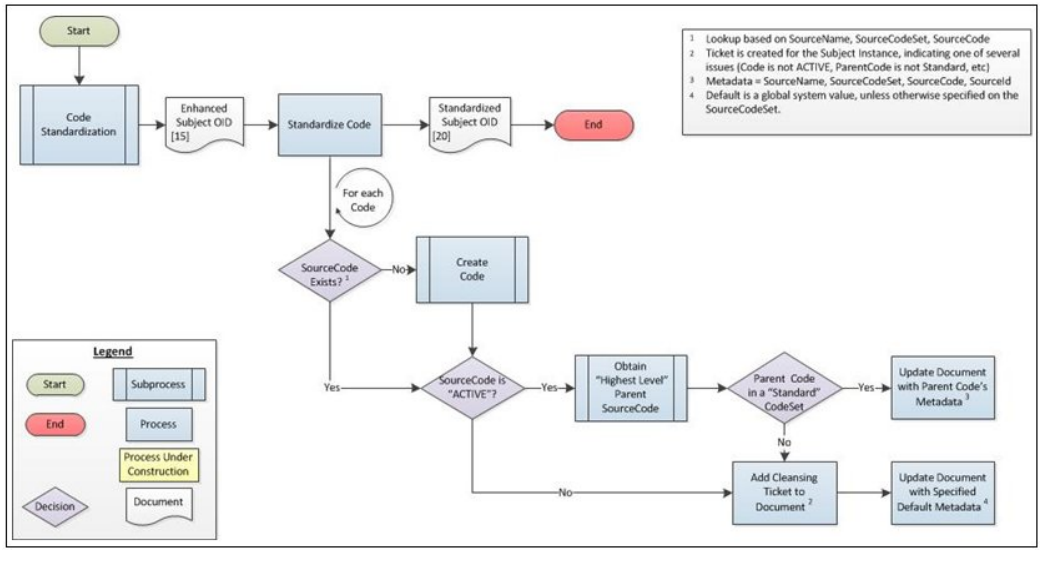
The *Code Standardization* subprocess is an extremely important step in Omni-Patient processing. The Cleanse, Match, and Merge rules, as well as Business Intelligence applications, require that the disparate values across Source Systems for the same data element concept (for example, Gender), be mapped into a single set of standard enumerated values.

Note: The Omni-Patient Server facilitates this process by providing the SourceCodeSet Interface Document Specification (IDS) to allow the integrator to pre-load these Source specific mapping.

The following is an example of a Code Standardization process.

Source	Source CodeSet	Source Code	Source	Standardized SourceCodeSet	Standardized Code	Standardized Description
CPSI	Gender	1	Omni	AdministrativeSex	F	Female
Cerner	Sex	f	Omni	AdministrativeSex	F	Female
Allscripts	GenderCode	Female	Omni	AdministrativeSex	F	Female
CPSI	Gender	2	Omni	AdministrativeSex	F	Male
Cerner	Sex	M	Omni	AdministrativeSex	F	Male
Allscripts	GenderCode	Male	Omni	AdministrativeSex	F	Male

Once preloaded, these SourceCodeSet and SourceCode values are used to validate codes that are submitted on Mastered or Transactional Subjects. The following process flow describes how these validation rules are applied to prepare the document for later use by the cleansing and matching services.



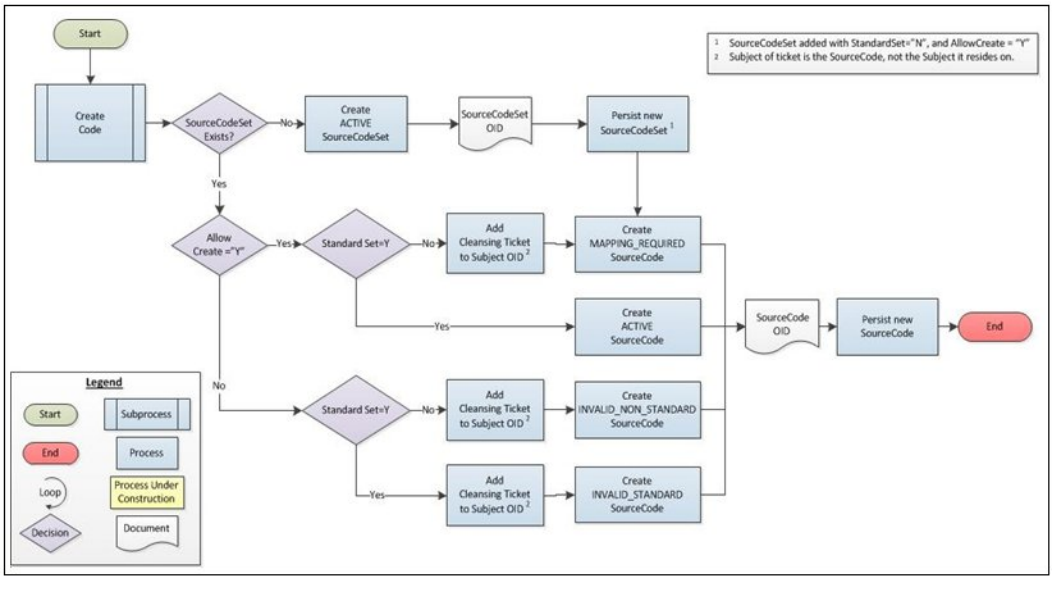
Although new code values may be created for a given SourceCodeSet during Subject OID processing, they are never mapped to the Standardized set, since it is impossible for the system to know this information.

Data Stewards are notified of this condition through Remediation Tickets, and may use Omni-Patient Management Central (OPMC) to help remediate the issue.

Create Code

When new code values are encountered during the processing of a subject, it may be necessary to create the code, and even the SourceCodeSet, if they do not currently exist in the system.

Omni-Patient uses two parameters of the SourceCodeSet (*AllowCreate* and *StandardSet*) to determine the status with which the code will be created, as well as whether remediation tickets are required, as shown in the following image.



The following table illustrates the status with which the resulting SourceCode will be created, given the AllowCreate and StandardSet parameters on its SourceCodeSet.

SourceCodeSet AllowCreate?	SourceCodeSet StandardSet?	Resulting Source Code Status	Remediation Ticket?	Business Scenario
Yes	No	MAPPING_REQUIRED	Yes	A new code has been added that requires mapping to a SourceCodeSet with StandardSet=Y.

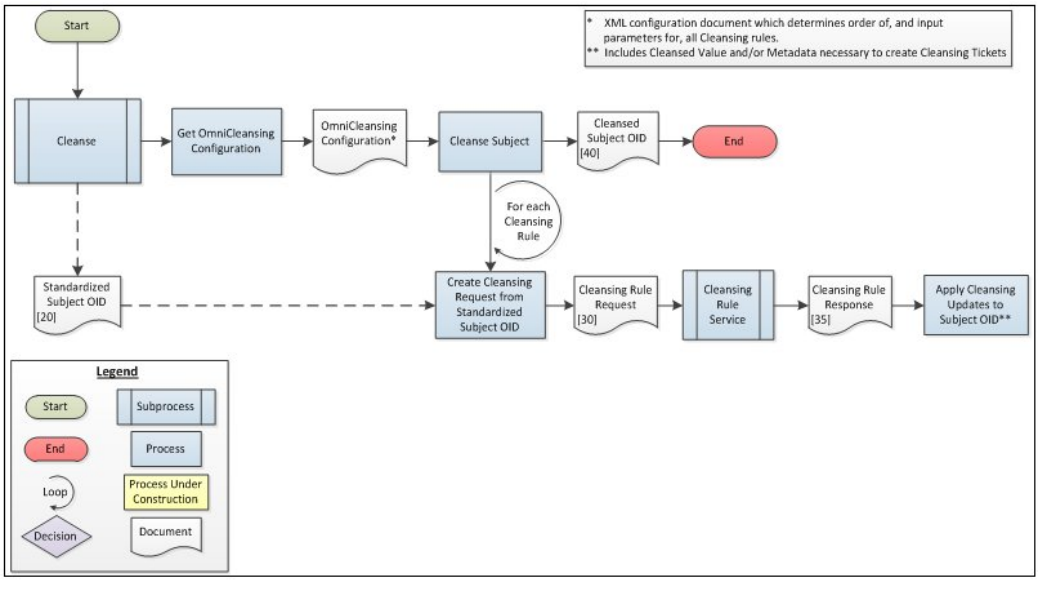
SourceCodeSet AllowCreate?	SourceCodeSet StandardSet?	Resulting Source Code Status	Remediation Ticket?	Business Scenario
Yes	Yes	ACTIVE	No	An active code that may be used immediately.
No	No	INVALID_NON_STANDARD	Yes	An attempt was made to add a new value to a CodeSet that does not allow creation. This will need to be remediated at the source (either through changing the value in the Subject or updating the SourceCodeSet.
No	Yes	INVALID_STANDARD	Yes	An attempt was made to add a new value to a CodeSet that does not allow creation. This will either need to be remediated at the source by changing the value in the Subject or mapping the code to a SourceCodeSet with StandardSet=Y.

Cleanse

The *Cleanse* subprocess interacts with the iWay Data Quality Suite by calling a set of customer-configured rules that help to ensure the accuracy, consistency, and completeness of the data that will reside in the EMIR.

Cleansed values and any other resulting metadata required to generate a ticket within a case in the Advanced Remediation tool are applied to the Cleansed Subject OID. At the end of the Cleanse process, the Subject OID is fully prepared and ready for further processing.

The following diagram illustrates the process flow that is followed by the *Cleanse* subprocess.



Process Subject

In this section:

Persist Instance

Process Match Response

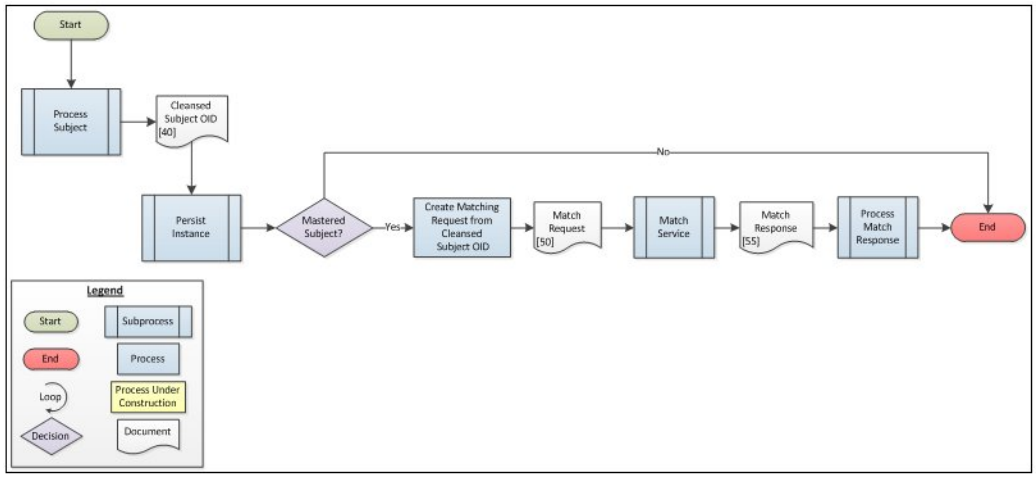
Persist Golden

Once the document has been *prepared*, it is now ready for processing and persistence to the Enterprise Mastered Information Repository (EMIR).

All Subjects will attempt to persist the cleansed Subject Instance that results from the *Prepare Subject* subprocess. Mastered subjects (such as Patient, Provider, Facility, and so on) undergo further processing with the Match and Merge Services to create and/or update the Golden record, which is a set of tables representing the current state of the mastered subject.

Note: The output of the Match Service is a list of all affected Match Groups, defined as the set of instances that have been determined to represent the same Golden Instance.

The following diagram illustrates the process flow that is followed by the *Process Subject* subprocess.



Persist Instance

As stated earlier, the EMIR contains a table set to store the current state of each Subject Instance, as well as a history table equivalent, which stores row-level history for the Subject Instance tables.

Note: All code values stored on an instance reflect the value that came in from the Source System. Instance records do not store the standardized code value.

Omni-Patient has the following two requirements with regard to history tables:

- Omni-Patient must store a history record for each Object that has changed.
- Omni-Patient must store a history record for the Parents of each Object that has changed.

Note: A data change in one table does not necessarily result in a data change for all tables that make up the Subject Instance.

In order to satisfy the latter requirement, Omni-Patient stores a unique Transaction Id for each document that is processed. When a Table row is updated, its parent's Transaction Id is also modified, and a new history record is created for the parent. This behavior continues up the tree, but does not traverse back down to siblings.

Initial Load

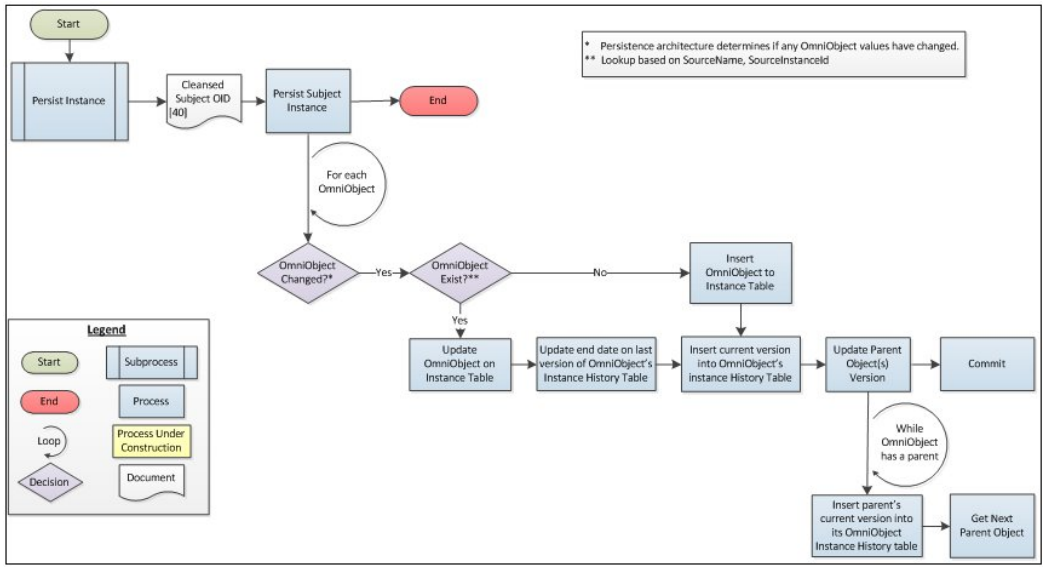
Table	ID	Parent ID	Txn ID	Hist_StartDate	Hist_EndDate
Patient_hist	Patient1		1	2013-01-01	
Person_hist	Person1	Patient1	1	2013-01-01	
PersonName_hist	PersonName1	Person1	1	2013-01-01	
PersonIdentifier_hist	PersonId1	Person1	1	2013-01-01	

Update of PersonName

Table	ID	Parent ID	Txn ID	Hist_StartDate	Hist_EndDate
Patient_hist	Patient1		1	2013-01-01	**2013-02-01**
Patient_hist	Patient1		**2**	**2013-02-01**	
Person_hist	Person1	Patient1	1	2013-01-01	**2013-02-01**
Person_hist	Person1	Patient1	**2**	**2013-02-01**	
PersonName_hist	PersonName1	Person1	1	2013-01-01	**2013-02-01**
PersonName_hist	PersonName1	Person1	**2**	**2013-02-01**	
PersonIdentifier_hist	PersonId1	Person1	1	2013-01-01	

Note: In the prior example where the update of the PersonName_hist table triggered an update for the Person_hist and Patient_hist tables, but did not update the PersonIdentifier_hist.

The following diagram illustrates the process flow that is followed by the Persist Instance process.



Process Match Response

After the set of Match Groups is returned by the Match Service on the Match Response, it must undergo further processing to create the merged Golden Record.

All Instances in the Match Group will be updated with the UEMID that represents that Match Group.

The last cleansed Subject OID is passed on the Merge Request for each instance in the Match Group.

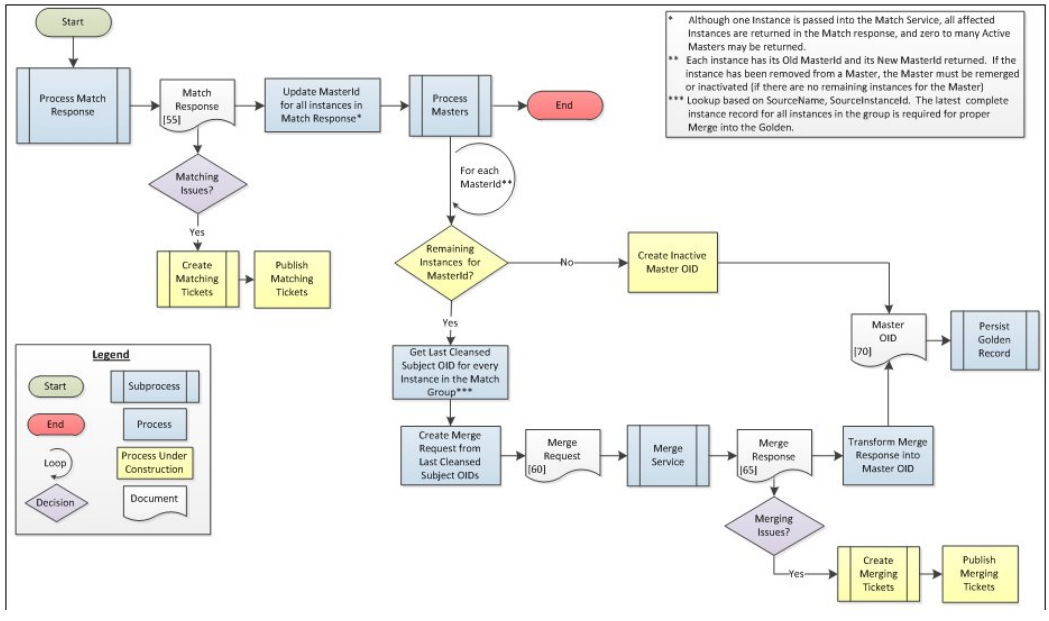
The Merge Service will execute its rules against the set of Instances to determine the content of the Golden Record, and communicates this back to the Omni-Patient Server on the Merge Response.

Note: Depending on the configuration of the Merge Rules, changes to an instance in the Match Group may not always result in changes to the Golden record.

Omni-Patient creates a Master Subject OID for internal use in creating the Golden Record.

If any issues are incurred along the way, tickets are created and published to Advanced Remediation.

The following diagram illustrates the process flow that is followed by the Process Match Response process.



Persist Golden

Similarly to the Instance tables, the Master tables containing the Golden record also have a history table equivalent that stores row-level history for the Subject Master tables.

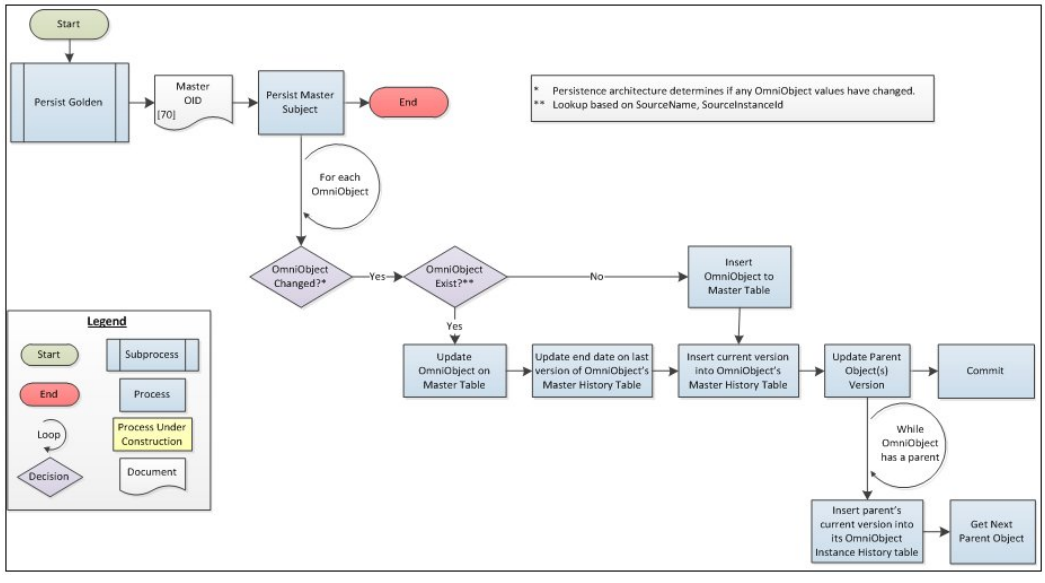
As stated earlier, the EMIR contains tables to store the current representation of the Subject Instance, as well as a history table equivalent that stores row-level history for the Subject Instance tables.

Note: All code values stored on a master reflect the Standardized code that was determined during the Code Standardization process. Master records do not generally store the value as passed by the SourceSystem.

The same history requirements that exist for the Instance, also exist for the Golden record.

- Omni-Patient must store a history record for each Object that has changed.
- Omni-Patient must store a history record for the Parents of each Object that has changed.

The following diagram illustrates the process flow that is followed by the Persist Golden process.



Remediation Request Handler (In Progress)

In this section:

Advanced Remediation allows the data steward to evaluate the results of the Cleansing and Matching rules for accuracy. There are times where the data steward needs to override data submitted from the Source System, or override the result of the Match to more accurately portray the *single version of the truth*.

The following outlines some of the supported functionality.

Manual Cleansing Overrides

- Allows a data steward to override null or incorrect values from the Source System to facilitate better accuracy in Cleanse, Match, and Merge processes.
- Must have the ability to Preview the results and Persist so that the Overrides can be applied to each subsequent update of the Subject OID.

Manual Matching Overrides

- Allows a data steward to resolve False Negative and False Positive matches that may occur in the Matching engine.
- Ability to manage a whitelist and blacklist of other instance records for any given instance record.
  - **Blacklist.** Regardless of what the match rules indicate, these records are not the same as me.
  - **Whitelist.** Regardless of what the match rules indicate, these records are the same as me.

Remediation Request Handler Overview

The Omni-Patient Server is required to process and persist the Manual Cleansing and Manual Matching Overrides submitted by the data steward during Advanced Remediation.

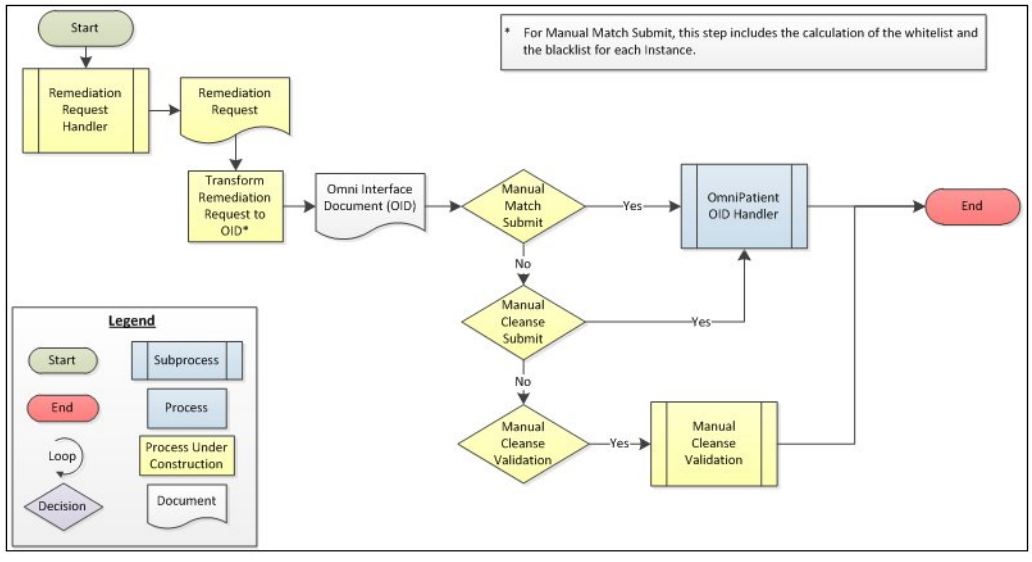
In addition, it must support the ability to execute the *Prepare Subject* subprocess, and communicate the result back to Advanced Remediation for the data steward to preview the results of a set of Manual Cleansing Overrides prior to submitting them.

The *Remediation Request Handler* subprocess transforms the request into the appropriate OID, and either routes to the Omni-Patient OID Handler, or to the *Manual Cleanse Validation* subprocess.

Notes:

- Manual Cleanse Validation is implemented as web service in order to provide synchronous response to Advanced Remediation. The Omni-Patient OID Handler processes the Manual Match Submit.
- The Omni-Patient OID Handler processes the Manual Match Submit and Manual Cleanse Submit asynchronously on an internal queue.

The following diagram illustrates a high level process flow that is followed by the Remediation Request Handler.



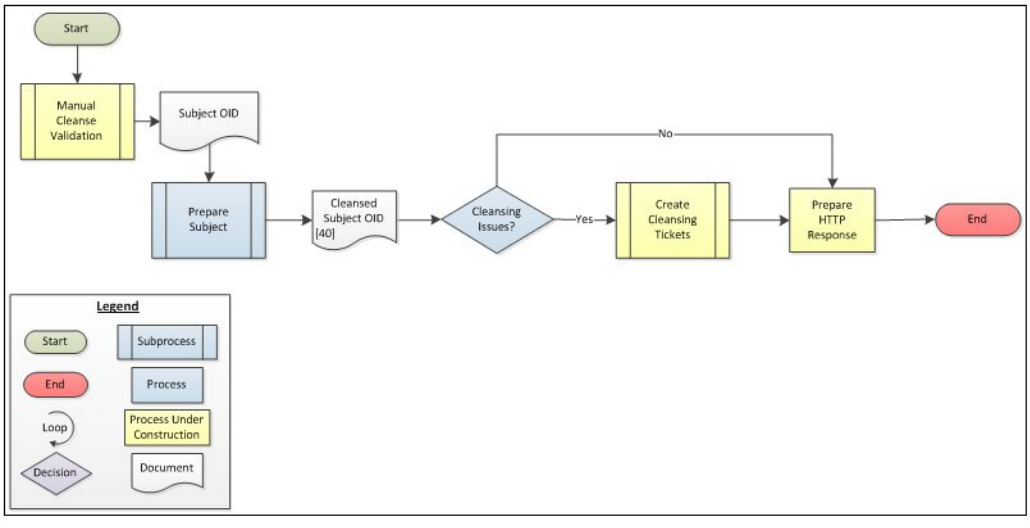
Manual Cleanse Validation (In Progress)

As stated previously, the Omni-Patient Server must support the ability to execute the *Prepare Subject* subprocess, and communicate the result back to Advanced Remediation for the data steward to preview the results of a set of Manual Cleansing Overrides prior to submitting them.

For a given Subject OID, the Manual Cleansing Overrides are applied to the document during OID Enhancement. Code Standardization is executed, and the current Cleansing rules are applied.

Any issues that result are communicated back on the web service response as a Cleansing ticket.

The following diagram illustrates a high level process flow that is followed by Manual Cleanse Validation.



2 Configuring Omni-Patient Server

This section describes configuration tasks related to Omni-Patient Server.

Topics:

- Starting and Stopping Omni-Patient Server
- Configuration Properties
- Debugging
- Error Folder
- Submitting IDS Documents
- Omni Codes
- Date Formats
- Cleansing
- Mastering

Starting and Stopping Omni-Patient Server

How to:

Start Omni-Patient Server on Windows

Each sub-system of Omni-Patient must be started separately. This is done to support distributed Omni-Patient architectures. There are convenience scripts available for Windows and Linux called start_all.bat/sh in the [OmniPatient Home] directory. This script will work if all sub-systems are run on the same machine.

Procedure: How to Start Omni-Patient Server on Windows

1. From a command prompt, navigate to the following directory:

   ```
   [OmniPatient]/iway7
   ```

2. Type the following command:

   ```
   >iway7.cmd base
   ```

3. From another command prompt, navigate to the following directory:

   ```
   [OmniPatient]/iway7
   ```

4. Type the following command:

   ```
   >iway7.cmd OmniPatient
   ```

5. From another command prompt, navigate to the following directory:

   ```
   [OmniPatient]/mdc/OMNI_DQ_services/bat
   ```

6. Type the following command:

   ```
   >start_online_service.bat
   ```

7. From another command prompt, navigate to the following directory:

   ```
   [OmniPatient]/mdc/OMNI_Mastering_services/bat
   ```

8. Type the following command:

   ```
   >start_online_service.bat
   ```

Configuration Properties

Configuration properties can be found in the Omnipatient.properties file, which is located in the following directory:

`[OmniPatient]/iway7/config/OmniPatient/resources`

The following table lists and describes these configuration properties:

Name	Default Value	Description
inputfolder.location	`_sreg(iwayhome)IDS`	Input folder location for the file listeners.
error.folder	`_sreg(iwayhome)error`	All messages that contain errors are delivered to this folder.
qa.on	`off`	Determines whether to debug messages, which will be written to the [iwayhome]/debug folder.
code.translation	`true`	Set value to *off* to stop source to Omni code translation.
ws.port	`8999`	Port for XML HTTP web service that takes IDS docs in the post body.
omnicode.load	`true`	Loads the Omni codes at startup.
ds.url	`jdbc:sqlserver://[hostname]:1433; databaseName=[dbname]`	Data source URL.
ds.username	`[user]`	Data source user name.
ds.password	`[password]`	Data source password.

Debugging

When enabled, Omni-Patient can emit debug documents to the following directory at certain points during processing:

`[OmniPatient]/iway7/debug`

This behavior can be enabled by setting the *qa.on* property value to *always*.

Error Folder

Until remediation features are added to Omni-Patient all documents that fall into an error state are emitted to an error folder. The following is the default location of this directory:

```
[OmniPatient Home]/iway7/error
```

This location can be configured by changing the *error.folder* property and restarting.

Submitting IDS Documents

In this section:

This section describes several techniques that can be used to submit IDS documents.

Using the File Listeners

In this section:

IDS documents can be submitted to Omni-Patient by copying them into one of the *in* folders under the following directory:

```
[OmniPatient Home]/iway7/IDS
```

IDS documents that contain more than one record need to have a root tag of *OmniPatientInterface* to be processed correctly.

Configuring the File Listeners

The root IDS in folder can be configured by setting the inputfolder.location property. The system will need to be restarted for changes to the file listeners to take effect.

Using the HTTP Service Interface

In this section:

Configuring the HTTP Service Interface

Omni-Patient contains an HTTP listener that can receive and process IDS documents. This is a simple HTTP POST service and is not a SOAP based web service. This service takes IDS documents one at a time and does not support multiple records in a single document.

Configuring the HTTP Service Interface

The service is hosted by default on port 8999, so local submissions would be sending an HTTP POST to `http://localhost:8999` with the IDS document as the body of the POST. The port that the service listens on can be modified by setting the *ws.port* property and restarting the server.

Note: The IDS document for the web service does work with the *OmniPatientInterface* root tag and only accepts one document at a time.

Omni Codes

In this section:

Loading Omni Codes

Loading Source Codes

Omni-Patient uses an internal code set to represent some data like Gender and Marital Status. Many source systems also use codes to represent this same data, so a translation system is provided by Omni to convert source codes to Omni Codes prior to storage. One of the first steps in an Omni implementation is to take the set of source codes and map them to Omni Codes.

Loading Omni Codes

The set of Omni Codes are loaded into the data store when the system starts. This occurs every time in an upsert fashion to support changes to the code set. This behavior can be controlled by changing the property omnicode.load to false.

Loading Source Codes

Source code to Omni Code translations prior to mastering and can be done with the following steps:

1. From the OmniCode Set.xsl file, determine the OmniCodes for which you wish to make translations.
2. Create an Omni-Patient IDS document based on the SourceCode.xsd. For example:

```
<OmniPatientInterface>
<SourceCodeSet>
<SourceName>SystemA</SourceName>
<CodeSetName>Demographics</CodeSetName>
<Description>Demographic codes </Description>
<SourceCodes>
<SourceCode>
<SourceName>SystemA </SourceName>
<SourceCode>F </SourceCode>
<Description>Female</Description>
<Note></Note>
<OmniCode>
<CodeSetName>0001 </CodeSetName>
<Code>Female</Code>
</OmniCode>
</SourceCode>
</SourceCodes>
</SourceCodeSet>
</OmniPatientInterface>
```

   Multiple *SourceCodeSet* and *SourceCode* elements can be added to the same document, so a full translation can be done using one document.

3. Start the Omni-Patient system.
4. Copy your translation IDS XML into the following directory:

```
[OmniPatient Home]/iway7/IDS/any/in
```

Date Formats

IDS documents have elements that contain date values. These elements contain attributes that describe the formatting used in the field. For example:

```
<DateOfBirth format="yyyy-MM-dd">1965-02-21</DateOfBirth>
```

The value of the format element must be a valid Java 1.6 java.text.SimpleDateFormatter format string. For more information, see the following API documentation:

http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html

Cleansing

In this section:

After enhancement with codes IDS documents are cleansed. This cleansing service is configurable through a configuration file. Cleansing can be disabled by commenting out all subjects. This file is located in the following directory and is called OmniCleansing.xml:

```
[OmniPatient Home]/iway7/config/OmniPatient/resource
```

A snippet of this file is shown below.

Turning Cleansing Off

To turn cleansing off, comment out the subject element that contains a name attribute matching the IDS root tag. For example comment out <*subject name="Patient"*> to stop cleansing of Patient IDS documents.

Configuring Cleansing

The cleansing of an IDS document is broken into sections and managed by the existence of a <routine> element. Each <routine> element under a root subject element is evaluated and the cleansing routine is executed based on the configuration specified in the child elements. The routines are executed in the order they exist in the configuration document. A cleansing routine is made up of the following elements:

- **<subject name="">.** The start of a new cleansing section. The name attribute should equal the name of the root element of the IDS. The name is case sensitive.
- **<routine name="">.** The root routine element. The name element is option.
- **<endpoint>.** This is the endpoint URL of the DQC cleansing web service.
- **<request>.** This is the name of the request as defined in the WSDL for the specific cleansing web service it is used to wrap the request body and define the soap action.
- **<iterate_xpath> (Optional).** An XPath to a collection to be cleansed. If included, multiple web service calls will be made and the request element input values will have the element XPath evaluated against each member of the collection.
- **<input>.** This collection contains the names of elements that make up the request body and the XPath that sets the values.

- **<element><name>.** The name of input element as defined in the WSDL for the cleansing web service. These elements must begin with in_. The in_ defines element that will return cleansed values.
- **<element><cleansed_name> (Optional).** The name of the element in the cleansing web service response containing the cleansed value. If omitted, the element name is assumed to be the value of <element><name> with the in_ replaced with std_.
- **<element><xpath>.** An XPath that returns a value from an element in the IDS document. When the <iterate_xpath> element is set, this XPath becomes relative to the root found by the iterate XPath.

Mastering

In this section:

Matching and merging can be configured by edit the OmniMaster.xml file located in the following directory:

`[OmniPatient]/iway7/config/OmniPatient/resource`

Turning Mastering Off

Mastering can be turned off by commenting out the <subject> element. For example, to turn off mastering for the Patient subject, comment out the <*subject name="Patient"*> element.

Configuring Mastering

The mastering of an IDS document is broken into merging and matching sections under the main subject element.

```
<subject name=""> - The start of a new mastering section. The name attribute should
equal the name of the root element of the IDS. The name is case sensitive.<matching>
 - Starts a new section defining the input elements for the matching service.
  <endpoint> - This is the endpoint URL of the cleansing web service.
  <request> - This is the name of the request as defined in the WSDL for the specific
 mastering web service it is used to wrap the request body and define the soap action.

  <input> - Starts the section defining the elements for the web service call.
   <element>
     <name> - The name of the element in the ws call
     <xpath> - Xpath to the value in the IDS document
   </element>
   <collection>
     <iterate_xpath> - Xpath to the collection
     <collection_name> - Element name used for the collection on the ws
     <member_name> - Element name for a member of the collection in the ws
     <element> - Same as element above but Xpaths are relative to the iterate_xpath
   </collection>
  </input>
</matching><merging> - Starts a new section defining the request and endpoint for the
 merging service.
     <endpoint> - This is the endpoint URL of the cleansing web service.
    <request> - This is the name of the request as defined in the WSDL for the specific
 mastering web service it is used to wrap the request body and define the soap action.
</merging>
```

3 Anatomy of Omni Interface Documents

This section describes the anatomy of Omni Interface Documents (OIDs).

Topics:

- Overview
- OmniPatientInterface Root Tag
- Understanding the Structure and Usage of OmniObjects
- Understanding the Structure and Usage of OmniElements
- Understanding the Structure and Usage of OmniGroups
- Understanding the Structure and Usage of OmniCollections

Overview

In this section:

Common Building Blocks

Omni Interface Documents (OIDs) are the means by which all data is loaded into Omni-Patient. These eXtensible Markup Language (XML) interface documents, implement an XML Schema Definition (.xsd) called an Interface Document Specification (IDS), which allow pertinent data to be communicated from various source systems into Omni-Patient.

Common Building Blocks

Omni Interface Documents (OIDs) use the following common constructs, defined in Interface Document Specifications, to exchange information about their business subject(s):

- OmniPatientInterface
- OmniObject
- OmniElements
- OmniGroup
- OmniCollection

OmniPatientInterface Root Tag

All Omni Interface Documents (OIDs) begin with the *OmniPatientInterface* root tag. The inclusion of this outer wrapper allows multiple Subjects to be submitted in the same OID.

The following is a basic structure of an OID:

```
<?xml version="1.0" encoding="UTF-8"?>
<OmniPatientInterface >
   <OmniObject />
   <OmniObject />
   <OmniObject />
</OmniPatientInterface >
```

Understanding the Structure and Usage of OmniObjects

In this section:

As depicted above, OmniObjects are the primary building blocks for any IDS. They contain child OmniElements, OmniGroups, and OmniCollections to further describe the OmniObject and its relationship to other OmniObjects. Every IDS is comprised of at least one OmniObject, known as the Subject of the IDS.

All OmniObjects have a number of common OmniElements and OmniCollections, as depicted in the following example:

```
<OmniObject version="Object Version Number">
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>UniqueKey</SourceInstanceId>
   <SourceInstanceIdName></SourceInstanceIdName>
   <ExtendedAttributes/>
   <SourceCreatedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
    10:12:32</SourceCreatedDate>
   <SourceCreatedBy>SourceCreatedBy</SourceCreatedBy>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-
    01</SourceModifiedDate>
   <SourceModifiedBy>SourceModifiedBy</SourceModifiedBy>
</OmniObject>
```

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

Element	Required?	Description
Version	Recommended	This value is only used when referring to a Subject, as defined above. It assists with backward compatibility and is designed to help Omni-Patient understand which version of a given IDS is being processed. By default, the assumption is that the Version is the latest Version installed at the customer site.

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.
SourceCreatedDate	Optional	DateTime that the record was created in the originating system.
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.
ExtendedAttributes	Optional	List of custom extensions to an OmniObject.

Interface Document Specifications are constructed to support insert or update operations, in whole or in part, for any OmniObject contained within a given OID. Therefore, the minimum data required for processing an OmniObject is its unique identifiers, SourceName and SourceInstanceId.

Although not required, it is strongly recommended to include the SourceModifiedDate with each OmniObject. The reason is that during initial load, history records for each instance may be loaded within minutes of one another. If the user would like to ask an "as of" query to see what the data looked like for an instance record at a certain point in time, the result would be severely constricted if basing off of the OmniModifiedDate, or the date on which the record was modified in Omni-Patient. In order to understand what the instance record for a given OmniObject may have looked like on a certain date in the Source system, the user should create "as of" queries using the SourceModifiedDate.

The minimum required implementation for an OmniObject is shown in the following example:

```
<OmniObject>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>UniqueKey</SourceInstanceId>
</OmniObject>
```

The minimum recommended implementation for an OmniObject is shown in the following example:

```
<OmniObject>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>UniqueKey</SourceInstanceId>
   <!-Facilitates "as of" queries against instance data. -->
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-
    01</SourceModifiedDate>
</OmniObject>
```

SourceName

The SourceName is used to define the source system from which the OmniObject originated. Therefore, it is not recommended to embed additional information, such as the transport method, in the SourceName.

Regardless of transport method, all records submitted from a single originating system shall have the same SourceName and SourceInstanceId. If they do not, an additional duplicate instance record is created and the history of the instance record is adversely impacted.

The recommended implementation for SourceName is shown in the following examples:

Initial Load

```
<OmniObject>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>UniqueKey</SourceInstanceId>
  <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</OmniObject>
```

Update

```
<OmniObject>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>UniqueKey</SourceInstanceId>
  <SourceModifiedDate format="yyyy-MM-dd">2013-01-02</SourceModifiedDate>
</OmniObject>
```

The non-recommended implementation for SourceName is shown in the following examples:

Initial Load

```
<OmniObject>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>UniqueKey</SourceInstanceId>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</OmniObject>
```

Update

```
<OmniObject>
   <!-SourceName is different than Initial Load. This will result in
    another physical instance record. -->
   <SourceName>TestSource-HL7</SourceName>
   <SourceInstanceId>UniqueKey</SourceInstanceId>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-02</SourceModifiedDate>
</OmniObject>
```

SourceInstanceId Selection

In this section:

Once a common SourceName is determined for both the initial load and on-going updates, it is essential that a repeatable SourceInstanceId be selected. It must not contain transient information, such as the message ID from an HL7 MSH segment, or a specific database row ID, which may change over time. This will potentially result in a significant number of additional unintended instance records.

The following sections will discuss guidelines for SourceInstanceId selection for two types of OmniObjects:

- Subject
- One-to-One OmniObject relationship

For more information on SourceInstanceId selection within a One-to Many OmniObject relationship, see the *OmniCollection* section.

For more information on SourceInstanceId selection for a One-to-One reference to another Subject, see the *OmniLink* section.

Subject

The SourceInstanceId for a subject should primarily consist of a commonly understood business key, such as a Medical Record Number (MRN) for the Patient Subject or National Provider Identifier (NPI) for the Provider Subject.

It is recommended that the SourceInstanceId for a Subject follow the pattern:

Subject<*Separator*>**BusinessKey**

While it is technically not a requirement to specify the Subject and Separator so long as the BusinessKey is provided, it is recommended that it be included for readability and debugging purposes.

The recommended implementation for SourceInstanceId is shown in the following example:

```
<Patient>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>Patient|12345678</SourceInstanceId>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Patient>
<Provider>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>Provider:4567890</SourceInstanceId>
<   SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Provider>
```

The non-recommended implementation for SourceInstanceId is shown in the following example:

```
<Patient>
   <SourceName>TestSource</SourceName>
   <!-The Message ID will change every time the Patient is sent via HL7.
    This will result in a new instance record every time the Patient is
    submitted.  History of change to the Patient will be improperly
    recorded.-->
   <SourceInstanceId>Patient|<MSH.MSG_ID></SourceInstanceId>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Patient>
<Provider>
   <!-If the DB is moved or reloaded, the DB.ROW_ID may no longer be valid.-
->
   <SourceName>TestSource</SourceName>
   <SourceInstanceId> <DB.ROW_ID></SourceInstanceId>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Provider>
```

One-to-One OmniObject Relationship

There are occasions where an IDS calls for a one-to-one relationship with another OmniObject that is not a subject which can be fed separately. In these cases, the OmniObject solely exists within the context of its parent.

An example of this is a Person OmniObject contained within a Patient Subject. Person may not be fed directly into Omni-Patient, but it is an OmniObject that appears within several Subjects. Since SourceInstanceIds are stored uniquely by OmniObject, it is recommended that the Person OmniObject use the same SourceInstanceId as its parent object (in this case Patient), for ease of traceability and debugging.

For this scenario, the SourceInstanceId should follow the pattern:

ParentObject.SourceInstanceId<*Separator*> **Object**<*Separator*>**ObjectKey**

The recommended implementation for SourceInstanceId is shown in the following example:

```
<Patient>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId>Patient|12345678</SourceInstanceId>
  <Person>
     <Person>
     <!-Same SourceInstanceId as parent helps assist with traceability.
      When viewing the Person table, it is possible to understand what
      object the Person is related to. -->
   <SourceName>TestSource</SourceName>
   <SourceInstanceId> Patient|12345678|Person|PersonKey</SourceInstanceId>

   <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>

     </Person>
  </Person>
     <SourceModifiedDate format="yyyy-MM-dd">2013-01-
01</SourceModifiedDate>
</Patient>
```

The non-recommended implementation for SourceInstanceId is shown in the following example:

```
<Patient>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId>Patient|12345678</SourceInstanceId>
 <Person>
  <Person>
   <!-When viewing the Person table, traceability is difficult. -->
   <SourceName>TestSource</SourceName>
   <SourceInstanceId> Person|PersonKey></SourceInstanceId>
  <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>

  </Person>
 </Person>
 <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Patient>
```

Extended Attributes

There are often scenarios where important customer data is not reflected in the IDS because it may not have as much relevance to the broader customer-base. To address these customer-specific extensions of Omni-Patient, capability has been provided for all OmniObjects to have an ExtendedAttributes OmniCollection.

The data elements of an ExtendedAttribute are shown in the following example:

```
<OmniObject>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId>UniqueKey</SourceInstanceId>
  <ExtendedAttributes>
    <ExtendedAttribute>
       <SourceName>TestSource</SourceName>
       <SourceInstanceId>UniqueKey:ExtendedAttribute:XyzDate
       </SourceInstanceId>
       <Name>XyzDate</Name>
       <Type>Date</Type>
       <Format>yyyy-MM-dd</Format>
       <Value>2000-01-01</Value>
       <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
        10:12:32</SourceModifiedDate>
    </ExtendedAttribute>
  </ExtendedAttributes>
  <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
   10:12:32</SourceModifiedDate>
</OmniObject>
```

The following table lists and describes the supported data elements of an ExtendedAttribute.

Element	Required?	Description
Name	Required	Defines the name of the Extended Attribute. It should be thought of as the equivalent of a DataElement name.
Type	Required	The data type of the named Extended Attribute.
Format	Optional	Allows the integrator to provide further specifics on how the Value is represented. A common use of this attribute would be in providing the Format when the Type="Date".
Value	Required	The value of the named Extended Attribute.

Use of these attributes should be judicious. A good rule of thumb for determining whether to add an ExtendedAttribute is if the data element falls into one of the following scenarios:

- There is a desire to Cleanse and/or Match and Merge the data element.
- It is desirable to maintain the history of that element with respect to other data that is being submitted for that OmniObject.

If the data element is being used strictly for Business Intelligence (BI) purposes (not used in Cleanse, Match, or Merge processes), another technique may be considered. The data element can be incorporated directly into a Reporting Data Mart (RDM) that is comprised of Omni-Patient and non-Omni-Patient source data, effectively bypassing the Omni-Patient repository.

Understanding the Structure and Usage of OmniElements

In this section:

OmniElements are the mechanism by which data elements of an OmniObject are communicated to Omni-Patient. Outside of the complex data types noted below, OmniElements implement common XML schema data types, as defined in by W3C's XML Schema Part 2: Datatypes Second Edition, which can be referenced at:

http://www.w3.org/TR/xmlschema-2/

Omni-Patient provides the following additional constructs to facilitate data submission through IDS:

- OmniDate
- OmniLink
- SourceCodeType

OmniDate

OmniDate is the mechanism by which any date or date/time value may be communicated to Omni-Patient. The OmniDate provides an optional format attribute, which allows the integrator to specify how the Date and Time is represented, using pattern letters as defined in the Java SimpleDateFormat.

For more information on SimpleDateFormat, see:

http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html

By default, Omni-Patient assumes the following pattern:

```
yyyy-MM-dd'T'HH:mm:ss
```

The recommended implementation is shown in the following example:

```
<Patient>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId>Patient|12345678</SourceInstanceId>
  <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Patient>
<Patient>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId>Patient|12345678</SourceInstanceId>
  <SourceModifiedDate>2013-01-01T11:30:00</SourceModifiedDate>
</Patient>
```

The non-recommended implementation is shown in the following example:

```
<Patient>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId>Patient|12345678</SourceInstanceId>
  <SourceModifiedDate >2013-01-01</SourceModifiedDate>
</Patient>
<Patient>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId>Patient|12345678</SourceInstanceId>
  <SourceModifiedDate format="yyyy-MM-dd">13-01-01</SourceModifiedDate>
</Patient>
```

OmniLink

In this section:

An OmniLink is simply a one-to-one reference from a data element inside one IDS to the Subject of another IDS. The reference will only contain the data elements required to uniquely identify the Subject, which in most cases will be the SourceName and SourceInstanceId.

SourceInstanceId Selection for an OmniLink

A practical example of an OmniLink is the Patient Subject referencing its Preferred Provider. In this case, the referenced Subject must be specified with the same SourceName and SourceInstanceId that it has when submitted on its own.

In the example below, it is assumed that the Provider had been loaded separately (additional OmniElements were omitted for brevity). In order to reference the same Provider from a Patient OID, the only requirement is to use the same SourceName and the SourceInstanceId used by the previously submitted Provider. If the SourceName and/or SourceInstanceId are referred to in a different manner than when it was originally submitted, it will result in a new incomplete Provider instance.

The recommended implementation is shown in the following example:

```
<!-- Pre-loaded Provider -->
<Provider>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId>Provider:4567890</SourceInstanceId>
 <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Provider>
<Patient>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId>Patient|12345678</SourceInstanceId>
 <PreferredProvider>
  <Provider>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>Provider: 4567890</SourceInstanceId>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
  </Provider>
 </PreferredProvider>
 <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Patient>
```

The non-recommended implementation is shown in the following example:

```
<!-- Pre-loaded Provider -->
<Provider>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId>Provider|4567890</SourceInstanceId>
 <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Provider>
<Patient>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId>Patient|12345678</SourceInstanceId>
 <PreferredProvider>
  <Provider>
   <!-Same logical Provider as above, but a mismatched SourceName and
SourceInstanceId. This results in a new, sparsely populated instance record
being created for this Provider.
   -->
   <SourceName>TestSource2</SourceName>
   <SourceInstanceId>Patient|12345678|Provider|4567890</SourceInstanceId>
  </Provider>
 </PreferredProvider>
 <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Patient>
```

SourceCodeType

In this section:

SourceCodeType is the mechanism by which enumerated values from various source systems can be communicated to Omni-Patient.

Background

Key definitions in the ISO/IEC 11179 metadata registry standard provide important background for understanding how codified values are handled within Omni-Patient:

- **data element concept.** A concept that can be represented in the form of a data element, described independently of any particular representation.
- **value domain.** Set of permissible data values for a data element or data element concept.
- **permissible values.** Expression of a value meaning allowed in a specific value domain.
- **enumerated value domain.** Value domain that is specified by a list of all its permissible data values, and their value meanings.

A concrete example, such as the handling of a Country Code, might better explain how to apply each of these technical metadata terms:

- data element concept: Country Code
- value domain: ISO 3166-1 - Country Codes.
- permissible values for Country Code as an enumerated value domain:
  - AD, Andorra
  - AE, United Arab Emirates
  - AF, Afghanistan

- ....
- ZW, Zimbabwe

In this particular case, the Country Code is classified as a data element concept because its enumerated value domain may be used for multiple data element representations in the system (i.e. Person.Citizenship, PersonAddress.Country, OrganizationAddress.Country).

Omni-Patient uses the ideas expressed above to define the following equivalent terms:

- **code set.** A specific representation of an enumerated value domain (for example, CountryCodes), containing a list of permissible values.
- **code.** Permissible value in the enumerated value domain for the code set.
- **description.** The text-based value meaning for a code value in the enumerated value domain for the code set.

With these background definitions in mind, it is possible to see how they are applied to the structure of a SourceCodeType.

Structure of a SourceCodeType

Any data element defined as a SourceCodeType will have the following common structure, as defined in the Interface Document Specification.

```
<SourceCode  sourceName="owner" sourceCodeSet="codeSetName"
sourceCode="Code">Description</SourceCode>
```

The structural components of a SourceCodeType are listed and described in the following table.

Element	Required?	Description
SourceName	Required	The SourceName is used to define the owner of the SourceCodeSet. By default, the value is assumed to be the SourceName of the parent OmniObject. If mapping directly to a SourceCodeSet provided by Omni, it should be explicitly populated with the reserved word "OMNI".
SourceCodeSet	Required	The descriptor given to any enumerated value domain in Omni-Patient. SourceCodeSet descriptors must be unique by SourceName.

Element	Required?	Description
SourceCode	Required	The specific enumerated value in the SourceCodeSet designated for this SourceCode.
<NodeValue>	Optional	The node value of a SourceCodeType represents SourceCodeDescription, or the text-based value meaning of the code value specified by the SourceCode.

CASE_SENSITIVE LOOKUP

A practical example of a SourceCodeType may look like the following:

```
<Gender sourceName="TestSource" sourceCodeSet="genderCodeSet"
sourceCode="f">Female</Gender>
```

In the overwhelming majority of cases, the SourceName attribute of the SourceCodeType will be the same value as the required SourceName data element of its parent OmniObject. For the sake of brevity on the large volume of data elements designated as a SourceCodeType, the following alternate representation may be more commonly utilized.

```
<Gender sourceCodeSet="genderCodeSet" sourceCode="f">Female</Gender>
```

This technique assumes that the sourceName is obtained from the parent OmniObject of the SourceCodeType, as depicted in the following example.

```
<Person>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId>Patient|12345678|Person|PersonKey</SourceInstanceId>
  <!--The SourceName attribute of the Gender below is assumed to be
   equivalent to the SourceName data element of the parent Person
   OmniObject.-->
  <Gender sourceCodeSet="genderCodeSet" sourceCode="1">Female</Gender>
  <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</Person>
```

Mapping Directly to a Specified SourceCodeSet

There are often scenarios where Omni-Patient expects a SourceCode for a particular data element, but the SourceSystem may not have a codified value. A common example of this may be the Medical Record Number (MRN), which can be represented in some source systems as a column on a database, but is expected as an IdentifierType in Omni-Patient.

In cases such as these, it is acceptable to map directly to another specified SourceCodeSet, such as those supplied by Omni-Patient:

```
<PersonIdentifier>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId> Patient|12345678|Person|PersonKey|PersonIdentifier|MR
  </SourceInstanceId>
  <!-The SourceName attribute of the IdentifierType below can explicitly
  reference the IdentifierType SourceCodeSet (0203) provided by
  OmniPatient.-->
  <IdentifierType sourceName="OMNI" sourceCodeSet="0203" sourceCode="MR">
  </IdentifierType>
  <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
</PersonIdentifier>
```

Loading a SourceCodeSet

A SourceCodeSet represents a named, enumerated value domain in Omni-Patient. CodeSetNames must be unique by SourceName. All SourceCodeSets implement the following common structure.

```
<SourceCodeSet>
   <SourceName>TestSource</SourceName>
   <CodeSetName>UniqueCodeSetName</CodeSetName>
   <Description>A further description of the Code Set Name</Description>
   <SourceCodes/>
   <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
    10:12:32</SourceModifiedDate>
</SourceCodeSet>
```

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.
CodeSetName	Required	The descriptor given to any enumerated value domain in Omni-Patient. SourceCodeSet descriptors must be unique by SourceName. Please note that this value is assumed to be the equivalent of a SourceInstanceId, and it is used in conjunction with the SourceName to uniquely identify the SourceCodeSet.

Element	Required?	Description
Description	Optional	A more verbose business description of the SourceCodeSet.
SourceCodes	Optional	List of SourceCode objects that represent the enumerated value domain for the SourceCodeSet.

Specifying SourceCodes

The SourceCodes OmniCollection in the SourceCodeSet IDS provides the mechanism for communicating an enumerated value domain, as well as its mapping to another standardized SourceCodeSet. The SourceCodes OmniCollection assumes a list of SourceCode OmniObjects, which implement the following common structure.

```
<SourceCode>
  <SourceName>TestSource</SourceName>
  <SourceCode> EnumeratedValue </SourceCode>
  <Description> The text-based value meaning of the code value specified by
  the SourceCode.</Description>
  <ParentCode>
    <SourceCode>
      <SourceName>TestSource</SourceName>
      <CodeSetName> TestSourceCodeSetName</CodeSetName>
      <SourceCode>EnumeratedValue</SourceCode>
    </SourceCode>
  </ParentCode>
  <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
   10:12:32</SourceModifiedDate>
</SourceCode>
```

Element	Required?	Description
SourceName	Required	Defines the source system from which the SourceCode OmniObject originated. It is used in conjunction with the SourceCode data element to uniquely identify the SourceCode OmniObject.

Element	Required?	Description
SourceCode	Required	The specific enumerated value in the SourceCodeSet designated for this SourceCode. SourceCodes must be unique within a SourceCodeSet. Please note that this value is assumed to be the equivalent of a SourceInstanceId, and it is used in conjunction with the SourceSourceCodeSet to uniquely identify the SourceCode.
Description	Optional	The text-based value meaning of the code value specified by the SourceCode.
ParentCode	Optional	An OmniLink to a standardized SourceCode within another SourceCodeSet.

Code Standardization

Prior to executing common Cleanse, Match, and Merge rules for mastering, it is essential that code values from disparate systems be unified into a common codeset. However, from a data lineage perspective, it is also imperative to understand the value as it was sent by the source system. The process of translating a SourceCode into a unified standard is called Code Standardization. For more information on the Code Standardization process, please see [Document Reference]. The following is a practical example of two SourceCodeSets, each from different SourceNames, which are unified on the 0001 Administrative Sex codeset supplied by Omni-Patient.

The following is a Code Standardization example.

```
<SourceCodeSet>
  <SourceName>CPSI</SourceName>
  <CodeSetName>GenderCodeSet-CPSI</CodeSetName>
  <Description>Gender Code representation from CPSI</Description>
  <SourceCodes>
    <SourceCode>
      <SourceName>CPSI</SourceName>
      <SourceCode> 1</SourceCode>
      <Description>Female</Description>
      <ParentCode>
        <SourceCode>
           <SourceName>OMNI</SourceName>
           <CodeSetName> 0001</CodeSetName>
           <SourceCode>f</SourceCode>
        </SourceCode>
      </ParentCode>
      <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
      10:12:32</SourceModifiedDate>
        </SourceCode>
        <SourceCode>
           <SourceName>CPSI</SourceName>
           <SourceCode> 2</SourceCode>
           <Description>Male</Description>
           <ParentCode>
             <SourceCode>
               <SourceName>OMNI</SourceName>
               <CodeSetName> 0001</CodeSetName>
               <SourceCode>m</SourceCode>
             </SourceCode>
           </ParentCode>
           <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
            10:12:32</SourceModifiedDate>
            </SourceCode>
 </SourceCodes>
 <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
  10:12:32</SourceModifiedDate>
</SourceCodeSet>
```

```
<SourceCodeSet>
  <SourceName>Cerner</SourceName>
  <CodeSetName>GenderCodeSet-Cerner</CodeSetName>
  <Description>Gender Code representation from Cerner</Description>
  <SourceCodes>
    <SourceCode>
       <SourceName>Cerner</SourceName>
       <SourceCode> female</SourceCode>
       <Description>Female</Description>
       <ParentCode>
         <SourceCode>
           <SourceName>OMNI</SourceName>
           <CodeSetName> 0001</CodeSetName>
           <SourceCode>f</SourceCode>
         </SourceCode>
       </ParentCode>
        <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
        10:12:32</SourceModifiedDate>
       </SourceCode>
       <SourceCode>
          <SourceName>Cerner</SourceName>
          <SourceCode> male</SourceCode>
          <Description>Male</Description>
          <ParentCode>
             <SourceCode>
              <SourceName>OMNI</SourceName>
              <CodeSetName> 0001</CodeSetName>
              <SourceCode>m</SourceCode>
            </SourceCode>
          </ParentCode>
          <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
           10:12:32</SourceModifiedDate>
           </SourceCode>
  </SourceCodes>
  <SourceModifiedDate format="yyyy-MM-dd hh:mm:ss">2000-01-01
  10:12:32</SourceModifiedDate>
</SourceCodeSet>
```

A summarization of the result of loading these two SourceCodeSets is described in the following table.

Source	SourceCodeSet	Source Code	Parent Source	Parent Source Code Set	Parent Source Code	Parent Description
CPSI	GenderCodeSet-CPSI	1	OMNI	0001	F	Female

Source	SourceCodeSet	Source Code	Parent Source	Parent Source Code Set	Parent Source Code	Parent Description
Cerner	GenderCodeSet-Cerner	female	OMNI	0001	F	Female
CPSI	GenderCodeSet-CPSI	2	OMNI	0001	M	Male
Cerner	GenderCodeSet-Cerner	Male	OMNI	0001	M	Male

During the Code Standardization phase of Omni-Patient processing, the values supplied by CPSI and Cerner will be translated into the parent equivalent. The SourceCode will be associated with the instance record, and the Parent SourceCode will be populated on the golden record after Cleansing, Matching, and Merging rules have been applied.

In this case, both a value of '1' sent from CPSI, and a value of 'female' sent by Cerner, will result in a value of 'F' on the Golden Record to represent 'Female'.

Extending the SourceCodeSets supplied by Omni-Patient

At startup, Omni-Patient loads a series of SourceCodeSets that are maintained and shipped with the product. These codesets use the reserved word 'OMNI' in the SourceName. These SourceCodeSets are primarily used as ParentCodes that unify disparate values from various source systems. Given the unique nature of each customer's business, there may be a need to extend one or more of these enumerated value domains by submitting an update to the appropriate SourceCodeSet with additional customer-specific values.

Important: Default Cleanse, Match, and Merge rules that ship with the product are based on values in the SourceCodeSets provided by Omni. These rules must be customized in the customer environment should they need to take into account extended values in an Omni SourceCodeSet.

The following is a sample Omni SourceCodeSet.

```
<SourceCodeSet>
   <SourceName>OMNI</SourceName>
   <CodeSetName>0001</CodeSetName>
   <Description>AdministrativeSex</Description>
   <SourceCodes>
     <SourceCode>
       <SourceCode> A</SourceCode>
       <Description>Ambiguous</Description>
     </SourceCode>
    <SourceCode>
       <SourceCode> F</SourceCode>
       <Description>Female</Description>
    </SourceCode>
    <SourceCode>
       <SourceCode> M</SourceCode>
       <Description>Male</Description>
    </SourceCode>
  </SourceCodes>
</SourceCodeSet>
```

The following is a custom Omni SourceCodeSet extension.

```
<SourceCodeSet>
   <SourceName>OMNI</SourceName>
   <CodeSetName>0001</CodeSetName>
   <SourceCodes>
   <SourceCode>
        <!-Customer-specific extension of the Enumerated value domain. -->
      <SourceCode> CustomerCode</SourceCode>
      <Description>Customer Code Description</Description>
   </SourceCode>
  </SourceCodes>
</SourceCodeSet>
```

SourceCodeSet Processing Considerations

A parent SourceCodeSet is one that is referred to by the ParentCode of another SourceCodeSet. All SourceCodeSets should be processed in the following order:

1. SourceCodeSets provided by Omni.
2. Customer extensions of the SourceCodeSets provided by Omni.
3. All other parent SourceCodeSets.
4. All other customer-specific SourceCodeSets.

All parent SourceCodeSets must be loaded before they are referred to in an OmniLink. The SourceCodeSets provided by Omni are always loaded first during startup. It is essential that any extensions to the SourceCodeSets provided by Omni-Patient be run subsequent to the initial load of these SourceCodeSets. This ensures that the values supplied by Omni-Patient will never overwrite the custom extensions created by the customer.

Understanding the Structure and Usage of OmniGroups

An OmniGroup is simply a node that wraps other OmniElements that are logically similar to promote readability. In the following example, the Military OmniGroup logically groups several like fields together:

```
<Person >
   <SourceName>TestSystem</SourceName>
   <SourceInstanceId>Patient:12345678:Person:PersonKey</SourceInstanceId>
   <Military>
      <VeteranStatus sourceCode="Reserves"
      sourceCodeSet="VeteranStatusCodes">Reserves </VeteranStatus>
      <Branch sourceCode="Army" sourceCodeSet="MilitaryBranchCodes">United
      States Army</Branch>
     <Category sourceCode="A14"
       sourceCodeSet="MilitaryCategoryCodes">A14</Category>
     <Component sourceCode="AR"
      sourceCodeSet="MilitaryComponentCodes">Army</Component>
     <PayGrade>E-7</PayGrade>
      <Rank sourceCode="E7" sourceCodeSet="MilitaryRankCodes">Enlisted
       Level 7</Rank>
     <UnitId>platoon</UnitId>
   </Military>
</Person >
```

Understanding the Structure and Usage of OmniCollections

In this section:

Operation Attribute (Future Functionality)

Clearing an OmniCollection

CollectionItem

An OmniCollection is a standard mechanism for depicting one-to-many relationships from one OmniObject to another. An OmniCollection contains a list of specific OmniObjects, also known as CollectionItems. The OmniCollection does not directly implement OmniElements or OmniGroups.

The following example shows the structure of an OmniCollection:

```
<!-OmniObjects defines the OmniCollection. -->
<OmniObjects operation="<processing instruction>">
  <!-Each OmniObject is a CollectionItem. -->
  <OmniObject >
     <SourceName>TestSource</SourceName>
     <SourceInstanceId>UniqueKey1</SourceInstanceId>
     <SourceModifiedDate format="yyyy-MM-dd">2013-01
      01</SourceModifiedDate>
  </OmniObject>
  <OmniObject >
     <SourceName>TestSource</SourceName>
     <SourceInstanceId>UniqueKey2</SourceInstanceId>
     <SourceModifiedDate format="yyyy-MM-dd">2013-01-
      01</SourceModifiedDate>
  </OmniObject>
</OmniObject>
```

Operation Attribute (Future Functionality)

The operation attribute on an OmniCollection provides a processing instruction to Omni-Patient. The valid values for this attribute are described as follows:

- **"replace"**

  Instructs Omni-Patient to delete all prior CollectionItems and insert the new ones provided.

- **"merge"**

  Instructs Omni-Patient to use a SQL Merge, or "upsert", for the new CollectionItems provided. If the CollectionItem exists on the database, it will be updated. If it does not currently exist, it will be inserted. Any previously loaded CollectionItem that does not appear in the new list will remain unchanged on the database. This is considered the default operation.

Clearing an OmniCollection

During an update operation, it may become necessary to clear the contents of an OmniCollection. The technique that is used to communicate the clearing of an OmniCollection is to send an empty OmniCollection node, as shown in the following example.

```
<!-An empty OmniObject will clear the contents of an OmniCollection. -->
<OmniObjects />
```

CollectionItem

In this section:

As stated earlier, a CollectionItem is simply one of the OmniObject children of an OmniCollection. Since a CollectionItem is an OmniObject, the choice of SourceInstanceId presents some unique considerations.

The following is an example of a CollectionItem.

```
<Patient>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId>Patient|12345678</SourceInstanceId>
  <Person>
    <Person>
       <!-Same SourceInstanceId as parent helps assist with traceability. -->
     <SourceName>TestSource</SourceName>
     <SourceInstanceId> Patient|12345678</SourceInstanceId>
     <SourceModifiedDate format="yyyy-MM-dd">2013-01-
      01</SourceModifiedDate>
     <!- PersonIdentifiers is an OmniCollection. -->
     <PersonIdentifiers>
       <!- PersonIdentifier is a CollectionItem. -->
       <PersonIdentifier>
         <SourceName>TestSource</SourceName>
         <SourceInstanceId>
          Patient|12345678|<Discriminator1></SourceInstanceId>
         <Type sourceName="OMNI" sourceCode="MR" sourceCodeSet="0001"/>
         <Value>12345678</Value>
         <SourceModifiedDate format="yyyy-MM-dd">
          2013-01-01</SourceModifiedDate>
       </PersonIdentifier>
       <!- PersonIdentifier is a CollectionItem. -->
       <PersonIdentifier>
         <SourceName>TestSource</SourceName>
         <SourceInstanceId>
          Patient|12345678|<Discriminator2></SourceInstanceId>
         <Type sourceName="OMNI" sourceCode="SS" sourceCodeSet="0001"/>
         <Value>123-45-6789</Value>
         <SourceModifiedDate format="yyyy-MM-dd">
         2013-01-01</SourceModifiedDate>
       </PersonIdentifier>
    </PersonIdentifiers>
   </Person>
  </Person>
</Patient>
```

SourceInstanceId Selection

In order to uniquely identify a CollectionItem in Omni-Patient, it is recommended to identify the string of parent OmniObjects all the way up to the Subject, in addition to some type of discriminator that uniquely identifies one CollectionItem from another.

The recommendation is that the SourceInstanceId for a CollectionItem follow the pattern:

```
ParentObject.SourceInstanceId<Separator> Object<Separator> CollectionItemDiscriminator
```

The recommended implementation of SourceInstanceId for a CollectionItem is shown in the following example:

```
<Patient>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId> Patient|12345678</SourceInstanceId>
 <Person>
  <Person>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId> Patient|12345678|Person|ABC</SourceInstanceId>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
   <PersonIdentifiers>
    <PersonIdentifier>
     <SourceName>TestSource</SourceName>
     <!- Concatenate the Discriminator to the Patient's SourceInstanceId.-->
     <SourceInstanceId>
Patient|12345678|Person|ABC|PersonIdentifier|<Discriminator1>
     </SourceInstanceId>
     <Type sourceName="OMNI" sourceCode="MR" sourceCodeSet="0001"/>
     <Value>12345678</Value>
     <SourceModifiedDate format="yyyy-MM-dd">
     2013-01-01</SourceModifiedDate>
    </PersonIdentifier>
   </PersonIdentifiers>
  </Person>
 </Person>
</Patient>
```

The non-recommended implementation of SourceInstanceId for a CollectionItem is shown in the following example:

```
<Patient>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId> Patient|12345678</SourceInstanceId>
 <Person>
  <Person>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId> Patient|12345678|Person|ABC</SourceInstanceId>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-01</SourceModifiedDate>
   <PersonIdentifiers>
    <PersonIdentifier>
     <SourceName>TestSource</SourceName>
     <!- Without the Parent Object's SourceInstanceId, there is nothing to
          differentiate Discriminator1 across multiple Patients. -->
```

```
    <SourceInstanceId>
 PersonIdentifier|<Discriminator1>
    </SourceInstanceId>
    <Type sourceName="OMNI" sourceCode="MR" sourceCodeSet="0001"/>
    <Value>12345678</Value>
    <SourceModifiedDate format="yyyy-MM-dd">
    2013-01-01</SourceModifiedDate>
   </PersonIdentifier>
  </PersonIdentifiers>
 </Person>
</Person>
</Patient>
<Patient>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId> Patient|23456789</SourceInstanceId>
 <Person>
  <Person>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId> Patient|23456789|Person|ABC</SourceInstanceId>
   <SourceModifiedDate format="yyyy-MM-dd">2013-01-   01</SourceModifiedDate>
   <PersonIdentifiers>
    <PersonIdentifier>
     <SourceName>TestSource</SourceName>
     <!- Without the Parent Object's SourceInstanceId, there is nothing to
              differentiate Discriminator1 across multiple Patients. The      result
 is that this identifier will be repeatedly updated across all Patients.  -->
     <SourceInstanceId>
      PersonIdentifier|<Discriminator1>
     </SourceInstanceId>
     <Type sourceName="OMNI" sourceCode="MR" sourceCodeSet="0001"/>
     <Value>23456789</Value>
     <SourceModifiedDate format="yyyy-MM-dd">
     2013-01-01
     </SourceModifiedDate>
    </PersonIdentifier>
   </PersonIdentifiers>
  </Person>
 </Person>
</Patient>
```

CollectionItem Discriminator Selection

The selection of the CollectionItem Discriminator depends on how the CollectionItem is implemented in the SourceSystem.

Often there will be a CollectionItem Type that will help differentiate one CollectionItem from another. In the cases where one and only one record of a certain CollectionItem Type is expected, the value of the Type code can be utilized as the Discriminator.

Note: The following examples have been reduced to relevant data elements for brevity.

The recommended implementation of a CollectionItem Discriminator selection is shown in the following example:

```
<Person>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId> Patient|12345678|Person|ABC</SourceInstanceId>
 <PersonIdentifiers>
  <PersonIdentifier>
   <SourceName>TestSource</SourceName>
   <!- When there is one and only one of each sourceCode value is expected    for the
 Type, the sourceCode value may be used as the Discriminator.
   Parent Obect's SourceInstanceId = Patient|12345678|Person|ABC
             Object = PersonIdentifier
             Discriminator = MR
    -->
   <SourceInstanceId>
   Patient|12345678|Person|ABC|PersonIdentifier|MR
   </SourceInstanceId>
   <!- The sourceCode value below is used as the Discriminator in the
SourceInstanceId -->
   <Type sourceName="OMNI" sourceCode="MR" sourceCodeSet="0001"/>
   <Value>12345678</Value>
  </PersonIdentifier>
  <PersonIdentifier>
   <SourceName>TestSource</SourceName>
   <!- When there is one and only one of each sourceCode value is expected    for the
 Type, the sourceCode value may be used as the Discriminator.
   Parent Obect's SourceInstanceId = Patient|12345678|Person|ABC
             Object = PersonIdentifier
             Discriminator = SS
    -->
   <SourceInstanceId>
Patient|12345678|Person|ABC|PersonIdentifier|SS</SourceInstanceId>
   <!- The sourceCode value below is used as the Discriminator in the
SourceInstanceId -->
   <Type sourceName="OMNI" sourceCode="SS" sourceCodeSet="0001"/>
   <Value>123-45-6789</Value>
  </PersonIdentifier>
 </PersonIdentifiers>
</Person>
```

The non-recommended implementation of a CollectionItem Discriminator selection is shown in the following example:

```
<Person>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId> Patient|12345678|Person|ABC</SourceInstanceId>
 <PersonIdentifiers>
  <PersonIdentifier>
  <SourceName>TestSource</SourceName>
  <!- Lack of a Discriminator in the SourceInstanceId will result in only   one
PersonIdentifier being created. It will get consistently over-written with each
subsequent identifier. -->
  <SourceInstanceId> Patient|12345678|Person|ABC|PersonIdentifier   </SourceInstanceId>

  <Type sourceName="OMNI" sourceCode="MR" sourceCodeSet="0001"/>
  <Value>12345678</Value>
 </PersonIdentifier>
  <PersonIdentifier>
   <SourceName>TestSource</SourceName>
   <!- Lack of a Discriminator in the SourceInstanceId will result in only   one
PersonIdentifier
   being created. It will get consistently over-written with   each subsequent
identifier. -->
   <SourceInstanceId> Patient|12345678|Person|ABC |PersonIdentifier
</SourceInstanceId>
   <Type sourceName="OMNI" sourceCode="SS" sourceCodeSet="0001"/>
   <Value>123-45-6789</Value>
  </PersonIdentifier>
 </PersonIdentifiers>
</Person>
```

Many times, the CollectionItem Type alone may not function as enough of a discriminator. For example, if the source system allows for multiple PersonNames with the type Alias, an additional value must be selected to further distinguish the two Alias records. If there is another business value discriminator that can be used, it's best. However, if that is unavailable, a numeric order may be concatenated, so long as it is consistently repeatable for the same logical record(s).

The recommended implementation is shown in the following example:

```
<Person>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId> Patient|12345678|Person|ABC</SourceInstanceId>
 <PersonNames>
  <PersonName>
   <SourceName>TestSource</SourceName>
   <!- When more than one of a each sourceCode value is expected for the    Type, an
 additional value is necessary on the Discriminator to further    distinguish the
records.

   Parent Obect's SourceInstanceId = Patient|12345678|Person|ABC
             Object = PersonName
             Discriminator = A1
    -->
   <SourceInstanceId>
   Patient|12345678|Person|ABC|PersonName|A1
   </SourceInstanceId>
   <Type sourceName="OMNI" sourceCode="A" sourceCodeSet="0200"/>
   <FullName>John Doe</FullName>
  </PersonName>
  <PersonName>
   <SourceName>TestSource</SourceName>
   <!- When more than one of a each sourceCode value is expected for the    Type, an
 additional value is necessary on the Discriminator to further    distinguish the
records.

   Parent Obect's SourceInstanceId = Patient|12345678|Person|ABC
             Object = PersonName
             Discriminator = A2
    -->
   <SourceInstanceId> Patient|12345678|Person|ABC|PersonName|A2</SourceInstanceId>
   <Type sourceName="OMNI" sourceCode="A" sourceCodeSet="0001"/>
   <FullName>John Q Public</FullName>
  </PersonName>
 </PersonNames>
</Person>
```

```
<!- In the following scenario, a PersonName update for "John Q. Public" (as represented
 by the "A2" Discriminator) is
sent as an update.  In this case, "John Quincy Public" updates the "John Q Public"
record, resulting in one record for "John Quincy Public" and another for "John Doe",
 sent previously above.   -->
<Person>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId> Patient|12345678|Person|ABC </SourceInstanceId>
  <PersonNames>
    <PersonName>
      <SourceName>TestSource</SourceName>
     <!- Update of the John Q Public record above -->
      <SourceInstanceId>
Patient|12345678|Person|ABC|PersonName|A2</SourceInstanceId>
      <Type sourceName="OMNI" sourceCode="A" sourceCodeSet="0001"/>
      <FullName>John Quincy Public</FullName>
    </PersonName>
  </PersonNames>
</Person>
```

The non-recommended implementation is shown in the following example:

```
<Person>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId> Patient|12345678|Person|ABC </SourceInstanceId>
  <PersonNames>
    <PersonName>
      <SourceName>TestSource</SourceName>
      <!- When more than one of a each sourceCode value is expected for the Type, the
 sourceCode value may not be solely used as the Discriminator.  It will result in a
single record being created for each sourceCode value ('A') in this case. This record

     will get consistently over-written with each subsequent identifier.

Parent Obect's SourceInstanceId = Patient|12345678|Person|ABC
          Object = PersonName
          Discriminator = A
 -->
```

```
      <SourceInstanceId>
Patient|12345678|Person|ABC|PersonName|A</SourceInstanceId>
      <!- The sourceCode value below is used as the Discriminator in the
SourceInstanceId -->
      <Type sourceName="OMNI" sourceCode="A" sourceCodeSet="0200"/>
      <FullName>John Doe</FullName>
    </PersonName>
    <PersonName>
    <SourceName>TestSource</SourceName>
    <!- When more than one of a each sourceCode value is expected for the Type, the
sourceCode value may not be solely used as the Discriminator.  It will result in a
single record being created for each sourceCode value.  This record will get
consistently over-written with each subsequent identifier.
Parent Obect's SourceInstanceId = Patient|12345678|Person|ABC
          Object = PersonName
          Discriminator = A (same as the PersonName record above)
 --><SourceInstanceId> Patient|12345678|Person|ABC|PersonName|A
</SourceInstanceId>
   <!- The sourceCode value below is used as the Discriminator in the SourceInstanceId
 -->
    < Type sourceName="OMNI" sourceCode="A" sourceCodeSet="0001"/>
      <FullName>John Q Public</FullName>
      </PersonName>
  </PersonNames>
</Person>
<!- In the following scenario, a PersonName update for "John Q. Public" is sent with
 a different Discriminator  the second time through ("A1" vs. "A2").  In this case,
"John Quincy Public" updates the "John Doe" record, rather than the intended "John Q.
 Public" record, resulting in one record for "John Quincy Public" and another for
"John Q. Public". The "John Doe" record will only exist in the history tables for the
 "John Quincy Public" record. -->
<Person>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId> Patient|12345678|Person|ABC </SourceInstanceId>
  <PersonNames>
      <PersonName>
        <SourceName>TestSource</SourceName>
        <!- When more than one of a each sourceCode value is expected for the Type,
an additional value is necessary on the Discriminator to further distinguish the
records. -->
        <SourceInstanceId>
Patient|12345678|Person|ABC|PersonName|A1</SourceInstanceId>
        <Type sourceName="OMNI" sourceCode="A" sourceCodeSet="0200"/>
        <FullName>John Doe</FullName>
      </PersonName>
      <PersonName>
        <SourceName>TestSource</SourceName>
        <!- When more than one of a each sourceCode value is expected for the Type,
```

```
an additional value is necessary on the Discriminator to further distinguish the
records. -->
        <SourceInstanceId>
Patient|12345678|Person|ABC|PersonName|A2</SourceInstanceId>
        <Type sourceName="OMNI" sourceCode="A" sourceCodeSet="0001"/>
        <FullName>John Q Public</FullName>
      </PersonName>
    </PersonNames>
  </Person>
<Person>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId> Patient|12345678|Person|ABC </SourceInstanceId>
  <PersonNames>
    <PersonName>
      <SourceName>TestSource</SourceName>
      <!- This update is sent with "John Doe's" Discriminator.  This will have the
unintended effect of updating the "John Doe" record (A1), rather than the "John Q
Public"
      record (A2). -->
     <SourceInstanceId> Patient|12345678|Person|ABC| PersonName|A1</SourceInstanceId>

      <Type sourceName="OMNI" sourceCode="A" sourceCodeSet="0001"/>
      <FullName>John Quincy Public</FullName>
    </PersonName>
  </PersonNames>
</Person>
```

In some cases, the CollectionItem does not have a Type, but there may be other data element(s) that serve as a discriminator for the CollectionItem. In the example below, the ProviderPracticeSpecialties are differentiated by the Facility at which the Specialty is practiced.

The recommended implementation is shown in the following example:

```
<Provider>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId> Provider |12345678</SourceInstanceId>
 <ProviderSpecialties>
  <ProviderSpecialty>
   <SourceName>TestSource</SourceName>
   <!- When more than one of a each sourceCode value is expected for the Type, an
additional value is necessary on the Discriminator to further distinguish the records.

   <SourceInstanceId>
      Provider |12345678|ProviderSpecialty|261QA1903X
   </SourceInstanceId>
```

```
   <SpecialtyType sourceName="OMNI" sourceCode=" 261QA1903X" sourceCodeSet="0454"/>
   <PracticingSpecialties>
    <ProviderPracticeSpecialty>
     <SourceName>TestSource</SourceName>
     <!- When there is noType, another data element value can serve
as the Discriminator for the record.

Parent Obect's SourceInstanceId = Provider|12345678|ProviderSpecialty|261QA1903X
          Object = ProviderPracticeSpecialty
          Discriminator = Facility1
 -->
     <SourceInstanceId>
Provider|12345678|ProviderSpecialty|261QA1903X|ProviderPracticeSpecialty|Facility1
     </SourceInstanceId>
     <!- This is an OmniLink to the Facility at which the
         Provider practices this specialty.   The Facility data element
         contains a reference to a Facility OmniObject.-->
     <Facility>
      <Facility>
       <SourceName>TestSource</SourceName>
       <!- The Facility at which the Provider  practices this
         specialty is used as the Discriminator.-->
       <SourceInstanceId>
       Facility:Facility1
       </SourceInstanceId>
      </Facility>
     </Facility>
    </ProviderPracticeSpecialty>
   </PracticingSpecialties >
  </ ProviderSpecialty>
 </ ProviderSpecialties >
</ Provider >
<!- In the following scenario, an attempt is made to use an Alphabetic sort order of
 the Facility Name as a Discriminator.
        This is not recommended if another business key is available because it may
 be difficult to replicate when new
       records are introduced.  In the example Facility "B" is inserted after Facility
 "A" and Facility "C" were loaded
        previously.  This changes the sort order and makes history a little more
difficult to understand. -->
```

The non-recommended implementation is shown in the following example:

```
<Provider>
 <SourceName>TestSource</SourceName>
 <SourceInstanceId> Provider |12345678</SourceInstanceId>
 <ProviderSpecialties>
  <ProviderSpecialty>
   <SourceName>TestSource</SourceName>
   <SourceInstanceId>
   Provider |12345678|ProviderSpecialty| 261QA1903X
   </SourceInstanceId>
  <SpecialtyType sourceName="OMNI" sourceCode=" 261QA1903X"   sourceCodeSet="0454"/>

   <PracticingSpecialties>
    <ProviderPracticeSpecialty>
     <SourceName>TestSource</SourceName>
     <!- When there is noType, another data element value can serve
     as the Discriminator for the record.

    Parent Obect's SourceInstanceId = Provider|12345678|ProviderSpecialty|261QA1903X

          Object = ProviderPracticeSpecialty
          Discriminator = 1
      -->
     <SourceInstanceId>
Provider|12345678|ProviderSpecialty|261QA1903X|ProviderPracticeSpecialty|1
     </SourceInstanceId>
     <!- This is an OmniLink to the FacilityLocation at which the
         Provider practices this specialty.   The Facility data element
         contains a reference to a Facility OmniObject.-->
     <Facility>
      <Facility>
       <SourceName>TestSource</SourceName>
       <!- The Facility at which the Provider  practices this
         specialty is used as the Discriminator.-->
       <SourceInstanceId>
       Facility:FacilityA
       </SourceInstanceId>
      </Facility>
     </Facility>
     <SourceModifiedDate format="yyyy-MM-dd">
     2013-01-01
     </SourceModifiedDate>
    </ProviderPracticeSpecialty>
    <ProviderPracticeSpecialty>
     <SourceName>TestSource</SourceName>
```

```
     <!- When there is noType, another data element value can serve
as the Discriminator for the record.

Parent Obect's SourceInstanceId = Provider|12345678|ProviderSpecialty|261QA1903X
          Object = ProviderPracticeSpecialty
          Discriminator = 2
 -->
     <SourceInstanceId>
Provider|12345678|ProviderSpecialty|261QA1903X|ProviderPracticeSpecialty|2
     </SourceInstanceId>
     <Facility>
      <Facility>
       <SourceName>TestSource</SourceName>
       <SourceInstanceId>
       Facility:FacilityC
       </SourceInstanceId>
      </Facility>
     </Facility>
     <SourceModifiedDate format="yyyy-MM-dd">
     2013-01-01
     </SourceModifiedDate>
    </ProviderPracticeSpecialty>
   </PracticingSpecialties >
  </ ProviderSpecialty>
 </ ProviderSpecialties >
</ Provider >
<!- Update of the ProviderPracticeSpecialties, inserting Facility "B" as a new Facility
 at which the Specialty is being
         practiced by the Provider (in addition to Facility "A" and Facility "C").-->

 <Provider>
  <SourceName>TestSource</SourceName>
  <SourceInstanceId> Provider |12345678</SourceInstanceId>
  <ProviderSpecialties>
   <ProviderSpecialty>
    <SourceName>TestSource</SourceName>
    <SourceInstanceId>
    Provider |12345678|ProviderSpecialty| 261QA1903X
    </SourceInstanceId>
    <SpecialtyType sourceName="OMNI" sourceCode=" 261QA1903X" sourceCodeSet="0454"/>
    <PracticingSpecialties>
     <ProviderPracticeSpecialty>
     <SourceName>TestSource</SourceName>
     <SourceInstanceId>
Provider|12345678|ProviderSpecialty|261QA1903X|ProviderPracticeSpecialty|1
     </SourceInstanceId>
     <Facility>
```

```
      <Facility>
       <SourceName>TestSource</SourceName>
       <SourceInstanceId>
       Facility:FacilityA
       </SourceInstanceId>
      </Facility>
     </Facility>
     <SourceModifiedDate format="yyyy-MM-dd">
     2013-02-01
     </SourceModifiedDate>
    </ProviderPracticeSpecialty>
    <ProviderPracticeSpecialty>
     <SourceName>TestSource</SourceName>
     <SourceInstanceId>
Provider|12345678|ProviderSpecialty|261QA1903X|ProviderPracticeSpecialty|2
     </SourceInstanceId>
     <Facility>
      <Facility>
       <SourceName>TestSource</SourceName>
       <SourceInstanceId>
       Facility:FacilityB
       </SourceInstanceId>
      </Facility>
     </Facility>
     <SourceModifiedDate format="yyyy-MM-dd">
     2013-02-01
     </SourceModifiedDate>
    </ProviderPracticeSpecialty>
    <ProviderPracticeSpecialty>
     <SourceName>TestSource</SourceName>
     <SourceInstanceId>
Provider|12345678|ProviderSpecialty|261QA1903X|ProviderPracticeSpecialty|3
     </SourceInstanceId>
     <Facility>
     <Facility>
      <SourceName>TestSource</SourceName>
      <SourceInstanceId>
      Facility:FacilityC
      </SourceInstanceId>
     </Facility>
    </Facility>
    <SourceModifiedDate format="yyyy-MM-dd">
    2013-02-01
    </SourceModifiedDate>
   </ProviderPracticeSpecialty>
  </PracticingSpecialties >
 </ ProviderSpecialty>
```

```
</ ProviderSpecialties >
</ Provider >
```

A summarization of the history records, as a result of loading these ProviderPracticeSpecialties, is indicated below. An unpopulated EndDate in a history table indicates the current record. Please note in the update that Facility "C" changes from being referenced by Discriminator "2" to Discriminator "3". In order to understand how long the Provider has been practicing the Specialty at Facility "C", one would need to calculate across multiple objects.

Initial Load

Object + Discriminator	Facility	Hist_StartDate	Hist_EndDate
ProviderPracticeSpecialty|1	A	2013-01-01	
ProviderPracticeSpecialty|2	C	2013-01-01	

Update

Object + Discriminator	Facility	Hist_StartDate	Hist_EndDate
ProviderPracticeSpecialty|1	A	2013-01-01	2013-02-01
ProviderPracticeSpecialty|1	A	2013-01-01	
ProviderPracticeSpecialty|2	C	2013-01-01	2013-02-01
ProviderPracticeSpecialty|2	B	2013-01-01	
ProviderPracticeSpecialty|3	C	2013-01-01	

Reader Comments

In an ongoing effort to produce effective documentation, the Technical Content Management staff at Information Builders welcomes any opinion you can offer regarding this manual.

Please share your suggestions for improving this publication and alert us to corrections. Identify specific pages where applicable. You can contact us through the following methods:

Mail:	Technical Content Management Information Builders, Inc. Two Penn Plaza New York, NY 10121-2898
Fax:	(212) 967-0460
Email:	*books_info@ibi.com*
Website:	*http://documentation.informationbuilders.com/connections.asp*

Name:_____

Company:_____

Address:_____

Telephone:_____ Date:_____

Email:_____

Comments:

Reader Comments