caAdapter 4.3

Quick Start Guide

NATIONAL® CANCER INSTITUTE

Center for Biomedical Informatics
and Information Technology

April 30, 2009

Credits and Resources

caAdapter Development and Management Teams			
Development	***Quality Assurance***	***Documentation***	***Project and Product Management***
Eugene Wang[2]	Jyothsna Chilukuri[2]	Carolyn Kelley Klinger[3]	Sichen Lu[1]
Ki Sung Um[2]			Anand Basu[1]
Ye Wu[2]			Christo Andonyadis[1]
			Sharon Gaheen[2]
1. National Cancer Institute Center for Biomedical Informatics and Information Technology (NCI CBIIT)		2. Science Application International Corporation (SAIC)	3. Lockheed Martin Management System Designers

List	
caAdapter Users Discussion Forum	https://list.nih.gov/archives/caadapter_users-l.html

Application Support	
Application Support	http://ncicbsupport.nci.nih.gov/sw/ Telephone: 301-451-4384 Toll free: 888-478-4423

Contents

Illustrations

About This Guide

This section introduces you to the *caAdapter 4.3 Quick Start Guide*. Topics in this section include

- *Purpose* on this page
- *caAdapter is an open* source application that supports several types of mapping and transformation. It enables users, to map and transfer a Comma Separated Value (CSV) clinical data to an equivalent HL7 V3 format. It implements both a front end GUI and a back end engine. The fron end GUI supports drag-and-drop mapping between a source specification and a target specification,.The back end engine validates specifications and data, and transfer the CSV data into HL7 V3 message.

Using similar GUI and mapping features, caAdapter also enables users to map and convert an HL7 V2 messages to an HL7 V3.

This guide provides a brief overview to caAdapter 4.3.

- Organization of This Guide on this page
- *Text Conventions Used* on page 2

Purpose

caAdapter is an open source application that supports several types of mapping and transformation. It enables users, to map and transfer a Comma Separated Value (CSV) clinical data to an equivalent HL7 V3 format. It implements both a front end GUI and a back end engine. The fron end GUI supports drag-and-drop mapping between a source specification and a target specification,.The back end engine validates specifications and data, and transfer the CSV data into HL7 V3 message.

Using similar GUI and mapping features, caAdapter also enables users to map and convert an HL7 V2 messages to an HL7 V3.

This guide provides a brief overview to caAdapter 4.3.

Organization of This Guide

This guide includes the following chapters:

- *Chapter 1*, *CSV to HL7 V3 Mapping,* on page 3 explains how to use caAdapter to map and convert a CSV files to the HL7 V3 message format.
- *Chapter 2, Converting HL7 V2 to HL7 V3 Messages*, on page 24 explains how to use caAdapter to map and convert an HL7 V2 message to the HL7 V3 message format.

Text Conventions Used

This section explains conventions used in this guide. The various typefaces represent interface components, keyboard shortcuts, toolbar buttons, dialog box options, and text that you type.

Convention	***Description***	***Example***
Bold	Highlights names of option buttons, check boxes, drop-down menus, menu commands, command buttons, or icons.	Click **Search**.
URL	Indicates a Web address.	http://domain.com
text in SMALL CAPS	Indicates a keyboard shortcut.	Press ENTER.
text in SMALL CAPS + text in SMALL CAPS	Indicates keys that are pressed simultaneously.	Press SHIFT + CTRL.
Italics	Highlights references to other documents, sections, figures, and tables.	See *Figure 4.5*.
Italic boldface monospace type	Represents text that you type.	In the **New Subset** text box, enter ***Proprietary Proteins***.
Note:	Highlights information of particular importance.	**Note:** This concept is used throughout this document.
{ }	Surrounds replaceable items.	Replace {last name, first name} with the Principal Investigator's name.

Chapter 1 CSV to HL7 V3 Mapping

This chapter includes the following topics:

Introduction to CSV to HL7 V3 Mapping

caAdapter (http://trials.nci.nih.gov/projects/infrastructureProject/caAdapter) consists of several components that, via messaging standards, support data sharing at CBIIT (http://ncicb.nci.nih.gov) and/or cancer centers as part of the cancer Biomedical Informatics Grid (caBIG) (http://caBIG.nci.nih.gov) solution. The components include a core engine for building, parsing, and validating HL7 V3 messages via an API or web service, and a mapping tool for providing mapping and transformation services using an assortment of messaging standards or formats such as HL7 V2 and V3 and object and data models.

This chapter explains how to use caAdapter to map a comma-separated value (CSV) specification to an HL7 V3 message specification. It includes instructions on types of specification maps, kinds of specification mapping relationships and resulting output from specific mapping scenarios. To explain the mapping rules further, we will keep in mind the result of the transformation service, that is, the series of HL7 V3 messages, while discussing or illustrating the applications of various mapping rules.

For consistency, here we list a group of generally used terms that will be referenced in later context.

Term	***Explanation***
CSV Specification	A comma-separated value (CSV) specification that presents the segment-based structure of a series of CSV data files
Source Specification	In current context, it refers to the CSV Specification
HL7 V3 Message Specification	An XML-based specification that is derived from the HL7 .mif file for the selected message type, with customized clone and attribute layout as well as other pre-defined values on data type fields
Target Specification	In current context, it refers to the HL7 V3 Message Specification
Map Specification or Map File	An XML-based specification that records the mapping relationship between a source and target specification
CSV data file	A segment-based CSV data file that may conform to a certain CSV Specification hierarchically

Term	*Explanation*
CSV Specification	A comma-separated value (CSV) specification that presents the segment-based structure of a series of CSV data files
Source data file	In current context, it refers to CSV data file
HL7 V3 Message	The outcome of the transformation service, i.e. the XML messages that comply to HL7 V3 specification
Target Message	In current context, it refers to HL7 V3 Message

Hierarchical Tree Structure

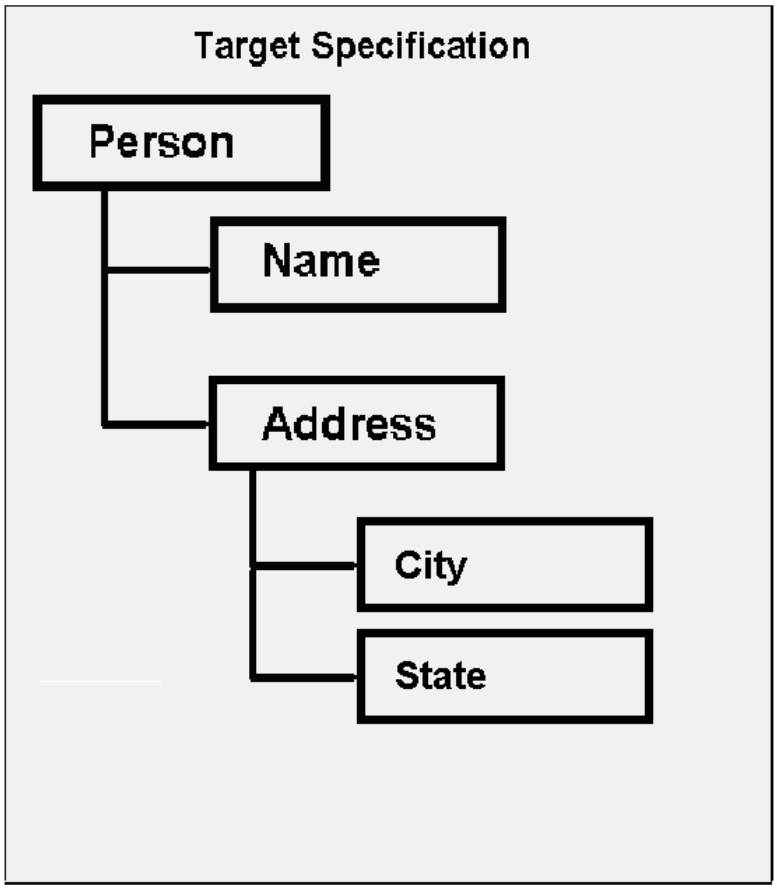

Figure 1. Hierarchical Tree Structure

Composite

A Composite is defined as a hierarchical structure that holds other hierarchical elements. As illustrated in *Figure 1. Hierarchical Tree Structure*, because the Person node holds the Name and Address nodes, it is a Composite structure. Similarly, because the Address node also holds the City and State nodes, it becomes a Composite structure as well. In general, a Composite could contain another Composite in its hierarchical structure.

Leaf

A Leaf is defined as a hierarchical structure that does not hold other hierarchical elements. As illustrated in *Figure 1. Hierarchical Tree Structure*, because the Name node does not have any other hierarchical structure beneath it, it is a Leaf node. In general, a leaf node belongs to one and only one Composite structure, while a composite structure could hold more than one Leaf nodes, such as the Address Composite.

Structural Relationship

Parent-Child Relationship

In a hierarchical tree structure, we refer to a node as a parent of another node if and only if the defined parent node hierarchically contains the other node. Similarly, we refer to the contained node is the child node of the aforementioned parent node.

For example, in the graphic below, under the source specification, the Organization node is the parent node of the Doctor node beneath it and the Doctor node is a child node of the Organization node.

In contrast, under the target specification, the Doctor node is the parent node of the Organization node beneath it.

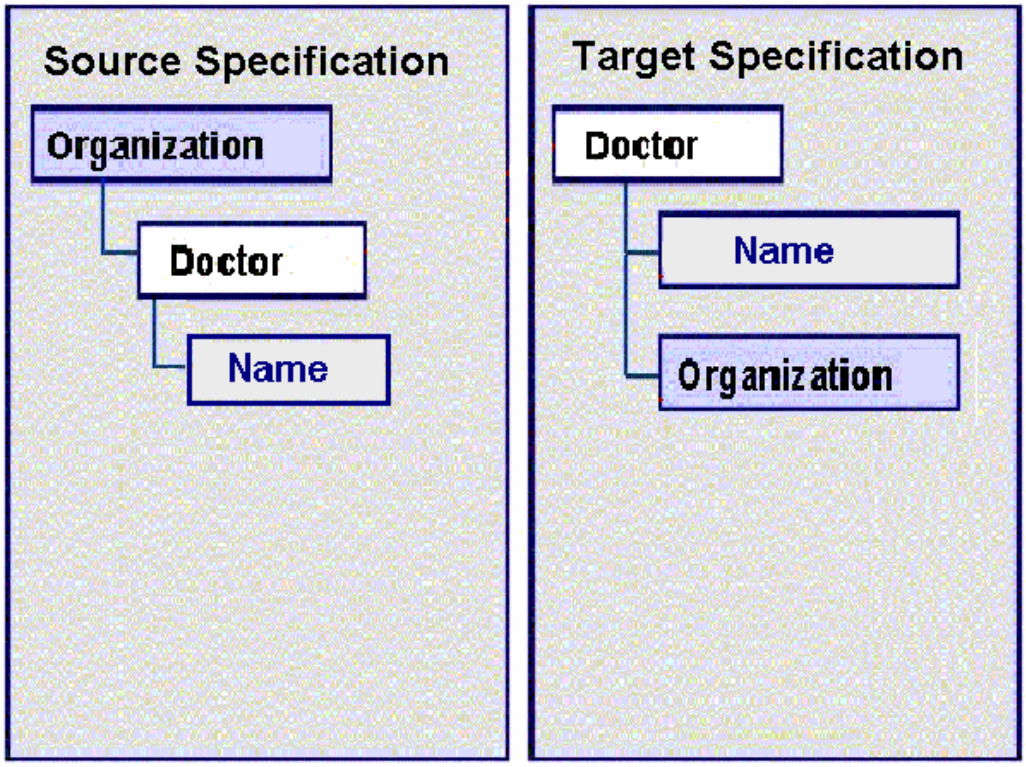

Figure 2. Example of a Parent Child Relationship

Sibling Relationship

Segments, fields, clones, attributes, and data type fields are all types of elements. In source and target specifications, sibling elements are the structures that share an immediate common parent. For example, in the figure below, a Doctor node has ID, Name, and Address as its child elements. The ID, Name and Address elements are collectively referred to as siblings under the context of the Doctor parent. In other words, a sibling relationship is always mentioned relative to a given parent element.

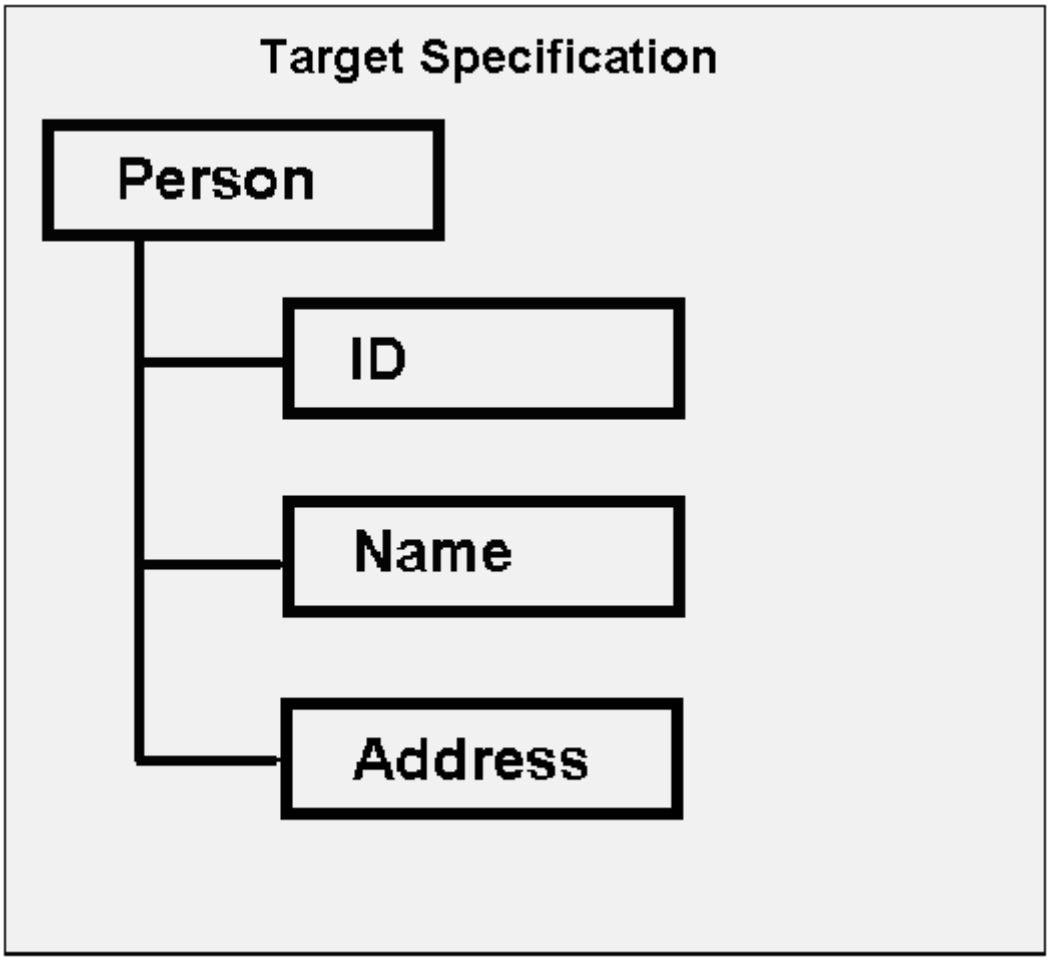

Figure 3. Sibling Relationship

Cardinality

Cardinality, also known as Multiplicity, defines the *number of* relationships between a parent and its child node, if any. For uniformity, we define the relationship from the parent node's point of view, but record the cardinality information on the child node side in a properties attribute. Cardinality only denotes the multiplicity relationship between a given node and its immediate parent node.

In the current release, cardinality is implied in the CSV source specification, while it is explicit in the HL7 V3 message specification. We will elaborate on this further after defining different types of cardinality relationships we may encounter in caAdapter.

One-to-One

The one-to-one relationship, denoted as 1..1, implies that the one parent node contains only one child and conversely, one child belongs to only one parent node.

In Figure 2, under the target specification, the Name node may have a one-to-one relationship with the parent Doctor node. In other words, if the underlying data contained one Doctor data record that included more than one Name data record, it would cause a validation error between the specification and the data file.

One-to-Many

The one-to-many relationship, as denoted as 1..*, implies that the one parent node could contain one or more children and conversely, one or more children may belong to one given parent node.

In Figure 2 on page 5, under the **source specification**, the Organization node has a one-to-many relationship with the Doctor node. The definition implies that for one parent Organization node, it will contain minimally one Doctor node and may contain as many Doctor nodes as possible. In real life, it reflects the fact that a small clinic could have at least one doctor, while a large hospital may employ hundreds of doctors in different departments.

Cardinality in Source Specification

In the CSV source specification, cardinality is implied through the parent-child relationship between segments or between a segment and those fields underneath it. There are no numbers to enter or update for cardinality when defining a CSV specification. In fact, between fields and their parent segment, the cardinality is always one-to-one (1..1), that is, one field belongs to only one parent segment. From the aspect of parent and child segments, child segments are only needed when there may be a one-to-many relationship between data items. In other words, in the CSV source specification, the child segment has an implied one-to-many cardinality relationship to its parent segment. In the data file, child segments may appear zero, one, or many times for each occurrence of a parent segment.

For instance, if the parent segment (PAT) contains data about a patient and the child segment (MED) contains data about medications the patient is taking, a CSV specification may look like this:

```
PAT
  1: PatientId
  2: LastName
  3: FirstName
  4: MiddleName
  5: VisitDate
  MED
    1: AgentName
    2: DoseAmount
    3: DoseUOM
    4: DoseFrequency
    5: StartDate
```

And the data file may look like this:

```
PAT, 12345, Smith, John, Joseph, 20060123
MED, Ibuprofen, 200, mg, BID, 20060101
```

```
PAT, 23456, Green, Peter, David, 20060123
PAT, 34567, Parker, James, Robert, 20060123
MED, Ibuprofen, 200, mg, BID, 20060101
MED, Claritin, 100, mg, BID, 20051001
MED, Prevacid, 500, mg, BID, 20051201
```

This shows that Smith is taking one medication, Green is currently not taking any medications, and Parker is taking three.

Cardinality in Target Specification

Cardinality in an HL7 V3 message specification is inherited from the HL7 `.mif` file for the selected HL7 V3 message. Users of caAdapter cannot change this cardinality; they can only include or exclude optional portions of the message structure. For elements that are included in the target specification, the cardinality may be one-to-one or one-to-many.

Basic Mapping Structure

Mapping Types

Using caAdapter's CSV to HL7 V3 map specification, a user creates links between source fields and target data type fields and between source segments and target attributes or clones. Links between source fields and target data type fields represent data relationships. Links between segments and clones or attributes explicitly relate conceptual groups of data to each other for the purpose of applying cardinality rules.

From such a perspective, we have

- *Data mappings* specifying the source of the *content* and the element in the target message where it will go, and
- *Concept mappings* controling the number of occurrences of message elements in the target messages generated

From a structural perspective, the data mapping is also referred to as the “leaf-to-leaf” mapping, that is, a leaf node in the source tree (source field) to a leaf node in the target tree (data type field).

Concept mapping, in contrast, is also referred to as the “composite-to-composite” mapping, that is, from a source segment to a target clone or attribute. The application does not allow the user, however, to create cross mappings between “composite” and “leaf” elements, that is, map a source field to a target clone or attribute, or map a source segment to a target data type field.

When caAdapter generates HL7 V3 messages, four factors work together to govern the number of instances of message content to be rendered in the results. They are listed in order of consideration as follows:

1. The cardinality attribute of the given clone, attribute, or data type field;
2. Any explicit conceptual mapping to the given clone or attribute from the source segment;

3. If no explicit conceptual mapping is available, any implicit conceptual mapping to be derived by the given clone or attribute from the source segment, if an explicit mapping does not exist;
4. The data mapping to the given data type field from the source field;

caAdapter derives an implicit conceptual mapping from a source field's or segment's parent segment to a clone, attribute, or data type field's parent clone or attribute. For this to occur, two conditions must exist:

- Explicit data mappings between the aforementioned source segment or field and target clone, attribute, or data type field must exist.
- No explicit conceptual mapping activity can be defined to reflect the cardinality relationship between the source field's or segment's parent segment to the clone, attribute, or data type field's parent clone or attribute.

For example, the mapping between a child segment and child clone will imply the existence of an implicit conceptual mapping between their respective parents, if no such mapping is explicitly created. In other words, factor 3 is only effectively derived in the absence of factor 2.

In general, during transformation, caAdapter looks at each of data type field defined in the target specification to see if any field has been mapped to a field in the source specification. If mappings exist, caAdapter examines whether the given data type field's parent attribute or any of its parent attributes or clones above its parent attribute has any explicit conceptual mapping to the source field's parent segment in the source specification. If so, this conceptual mapping will be utilized as factor 2; otherwise, an implicit mapping relationship is derived between the given data type field's parent attribute and the given source field's parent segment in the source specification. Similar implicit mapping relationships will be derived up to the source root segment and the target root clone, or until the first explicit conceptual mapping between a source segment and a target clone or attribute along the path back to the source root segment and the target root clone.

After either implicit or explicit conceptual mapping is decided, the cardinality information of the given data type field is analyzed to help determine the number of occurrences of message elements in messages generated. If the value is 1..1, according to its definition, one and only one message element will be generated and no further computation is needed. If the value is 1..*, however, the cardinality from the implicit or explicit conceptual mapping will be referred to. If the cardinality value from the conceptual mapping is 1..1, it is obvious one and only one message element will be generated and will report warnings if more than one data record exist in the source data file. If the cardinality value from the conceptual mapping is 1..*, the transformation service will create one HL7 V3 message for each occurrence of the data in the source data file.

Since the one-to-one cardinality relationship (1..1) creates one of the strongest bonds between the parent and child nodes, and given the priority in applying mapping rules (see the rule 1), a beginner of this tool may expect several validation error messages regarding mismatched "Multiplicity" or "Cardinality" issues. This is because he or she may have mapped, for instance, a source segment that has one-to-many cardinality, to a target clone or attribute that has only one-to-one cardinality, and the underlying source data of the source specification include more than one data instance of the given source segment.

Therefore, when creating a map file, it is important to first understand the inner structure of the HL7 V3 specification, especially the cardinality definition between parent and child clones and attributes. After that, it is best to start with the concept level relationship before mapping data relationships.

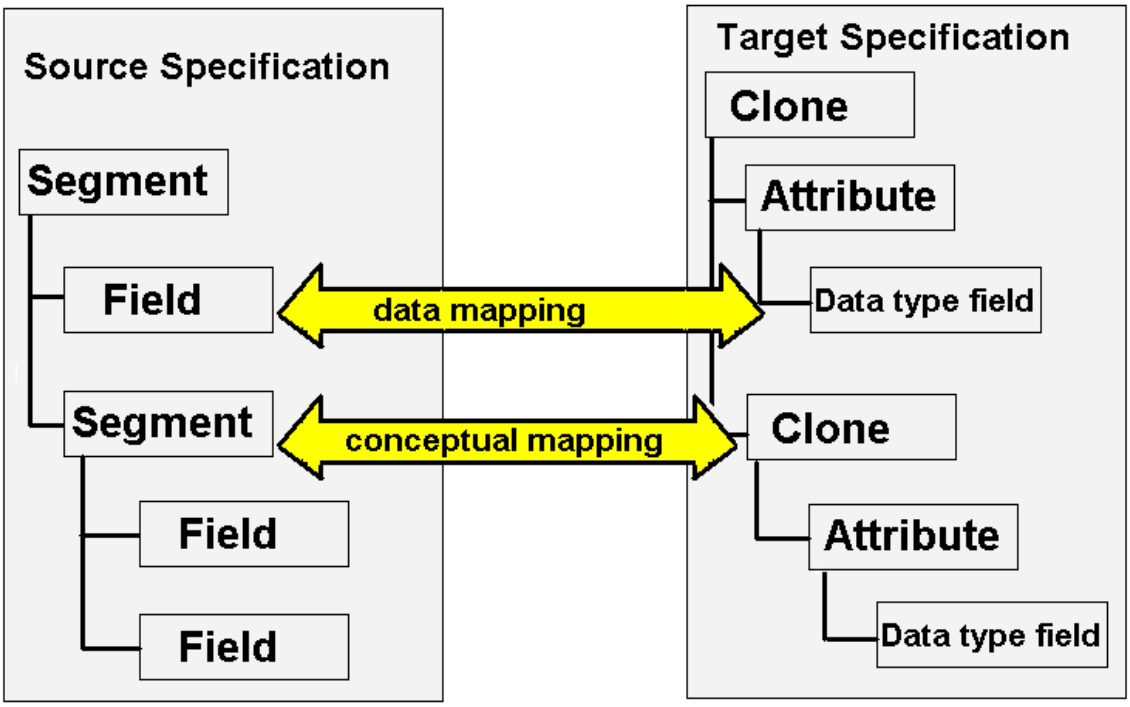

Figure 4. Mappings for Source to Target Specification

Source Specification

This chapter assumes the source data format is a comma-separated values (CSV) file. A comprehensive explanation of the CSV file format is in the *caAdapter 4.3 User's Guide*.

Briefly speaking, the CSV file format is a tabular data format that has fields separated by the comma character.

Example of a CSV file:

```
John, Doe, 5600 Fishers Lane, Rockville, MD, 20857
Jack, McGinnis, 456 Washington Blvd Suite 1000, Washington, DC,
20002
```

Each line is a segment containing a logical grouping of fields. Each segment may have one or more dependent child segments to handle one-to-many relationships between logical groups of data. To identify the segments for each logical record, segment identifiers are always the first element on each line.

Example of a CSV file:

```
PERS, John, Doe, 5600 Fishers Lane, Rockville, MD, 20857
PERSID, 2.16.840.1.113883.19.1, 12345
PERSID, 2.16.840.1.113883.19.1, 67890
```

```
PERS, Jack, McGinnis, 456 Washington Blvd Suite 1000, Washington,
DC, 20002
PERSID, 2.16.840.1.113883.19.1, 24680
```

In the above example, PERS is the segment identifier showing the root of each logical record. The PERSID segment identifiers show how one logical record can have many child records. caAdapter interprets this CSV file as two logical records of PERS, each with one-to-many PERSID segments.

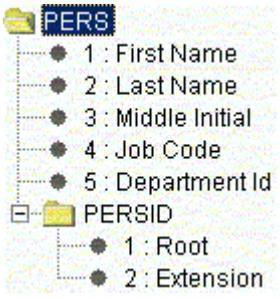

Figure 5. Example of the Parent-Child Relationship of Source Data

CSV source specification structures include a root segment with one or more child segments. A root segment and its child segments represent a single logical record. Each segment has one or more fields that represent elements of data. Segments may also contain one or more child segments that may also include child segments of their own. Mappings link both fields and segments to the XML elements in the target specification structure.

The CSV Specification tab allows users to reorder data fields, change structural relationships, and rename segments. Extra care should be taken when reordering segment structure and renaming segment names. You should ensure that the source specification segment structure and names match the data file segment structure and names. The validation action on source specification against a CSV data function is very useful for uncovering errors of this type.

Target Specification

This chapter assumes the target data format is an HL7 V3 XML message. An HL7 V3 specification is defined by the content of the HL7 .mif file for the target HL7 message type chosen by the user. A comprehensive explanation of the HL7 V3 specification and HL7 V3 message can be found in the *caAdapter 4.3 User's Guide*.

This guide uses a hypothetical example HL7 V3 message for illustration.

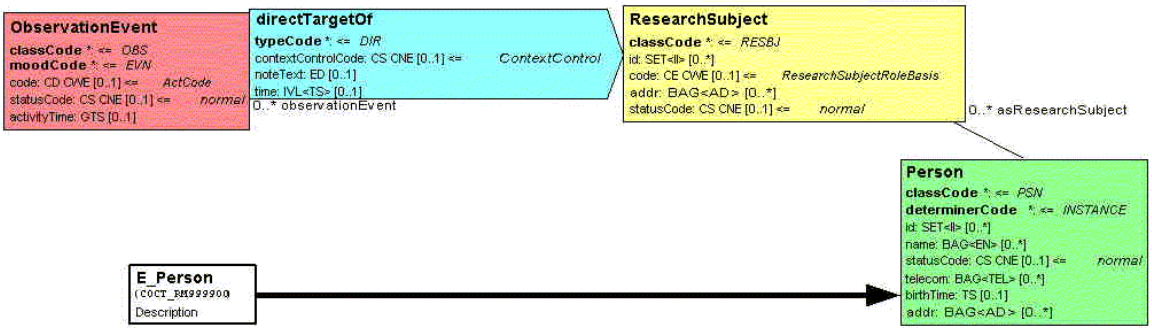

Figure 6. COCT_MT999900 Example Message

Target specification structures include the same elements as the message type. These elements include clones, attributes, and data type fields. Clones are classes derived from HL7 Reference Information Model (RIM) base classes. Each clone can have many child clones and many child attributes. An attribute is assigned a data type based on an independent specification of HL7 V3 data types.

Clones represent concept-level hierarchy. Each clone may be associated with one or more child clones that may also be associated to child clones of their own. Some clones are associated with Common Message Element Types (CMETs). These CMETs are a set of one or more clones that represent reusable concepts and may appear in multiple places in a message. In caAdapter, users can create mapping links from source segments and fields to target clones, attributes, and data types.

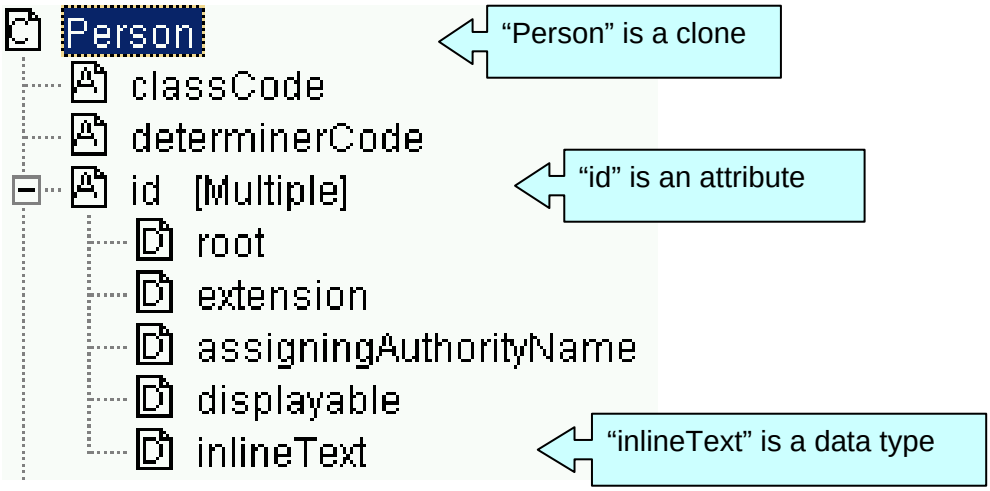

Figure 7. Parts of a Target Specification Structure

Defining Default Data

User-defined default values are constants assigned to data type fields. They are defined in the target specification, so they are pre-defined when it comes to the mapping.

The user-defined default value is not the same as the HL7 default value. HL7 default values originate in the HL7 V3 message type defintion and are pre-defined and unchangeable. User-defined default values allow users to assign values for attributes that may not be available from the source data. For example, if the root for all user IDs is common across the organization, this value can be entered in the target specification.

User-defined default values are assigned by using the HL7 V3 Specification tab. HL7 structural attributes and other elements that have their values fixed by the HL7 V3 standard cannot have default values defined.

User-defined default values are overridden by values from a mapped data source. While required attributes are often populated with default values specified in the HMD, optional ones are usually only populated when a map is present for that data type field or when a user-defined default value is specified. The table below shows the expected behavior for attributes that are mapped to a non-null CSV value,

mapped but to a CSV value that turns out the be null, and unmapped data type fields.

	Mapped	***Mapped Null***	***UnMapped***
Optional	CSV Value	CSV Value	Not Populated
Required	CSV Value	Default Value	Default Value
Mandatory	CSV Value	Default Value	Default Value

Figure 8. User Defined Default Value Behavior

"Mandatory" means that the value may not be NULL, unless its container (clone, attribute, etc.) is NULL. "Required" means values must be supported and may be NULL.

Units of Measure

Some HL7 V3 attributes contain units of measure properties. These units of measure must match those specified in the Unified Code for Units of Measure (UCUM). The Unified Code for Units of Measure is a code system intended to include all units of measure being contemporarily used in international science, engineering, and business. For a complete list, see The Unified Code for Units of Measure at http://aurora.regenstrief.org/UCUM/ucum.html.

Types of Mapped Relationships

Simple Mapping Relationship

Each link is a mapped relationship between source and target hierarchical trees. The combined rules for these mapped relationships define what the generated HL7 V3 message XML will look like.

Most mappings link data fields from a single segment to data type fields of an attribute. This document refers to these relationships as simple relationships because source cardinality and target cardinality have a one-to-one relationship.

In a simple relationship, one field in the source data file represents one data type field of an attribute in the target XML message.

Parent-Child Inverted Relationship

In a hierarchical tree structure, we define a node as a parent of another node only if the defined parent node hierarchically contains the other node. Similarly, we define the contained node is the child node of the aforementioned parent node. For example, in the figure below, the Organization node of the source tree is the parent node of any Doctor node beneath it, and any Doctor node is a child node of the Organization node.

Similarly, both source and target specification structures have a parent concept with its child concepts.

In some cases, a child in the source system is mapped to a parent in the target system. At the same time, the parent of the child in the source system is mapped to a child of the parent in the target system.

For example, the source specification structure has an organization (parent) with one or more doctors (children). The target specification structure has doctors (parent) who are each associated with an organization (children). In this case, each doctor has the same organization. This document refers to this type of relationship as parent-child inverted relationship.

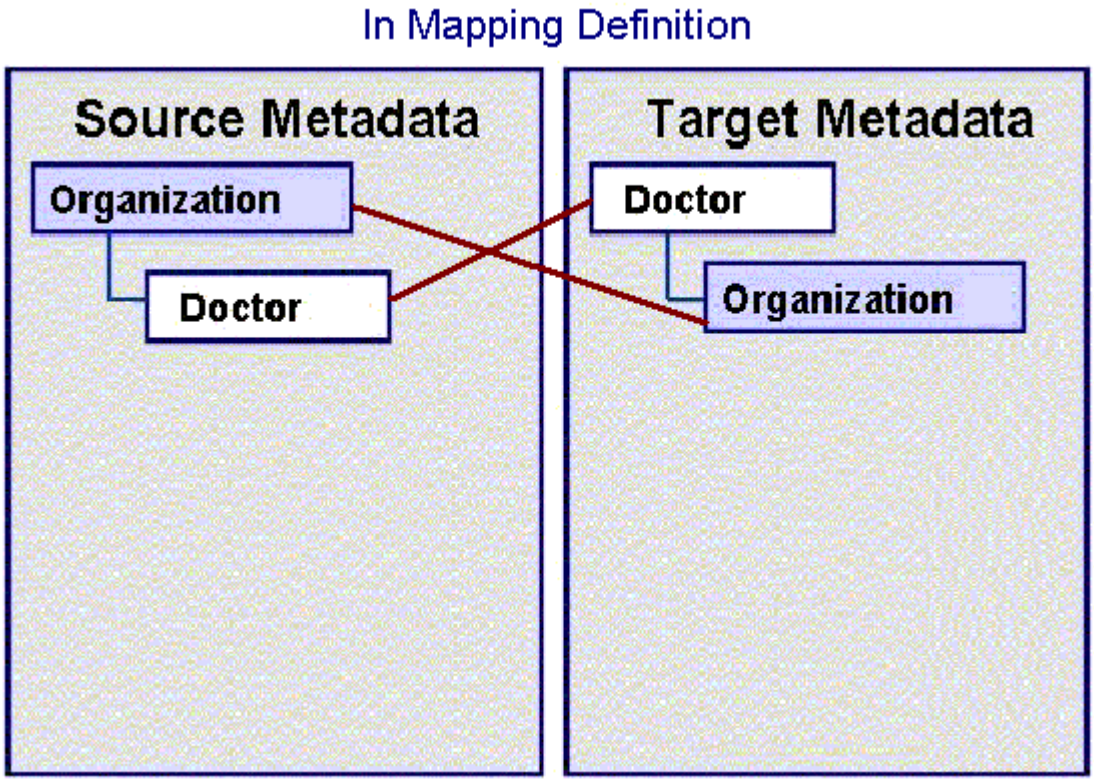

Figure 9. Example of a Parent-Child Inverted Relationship in a Mapping Definition

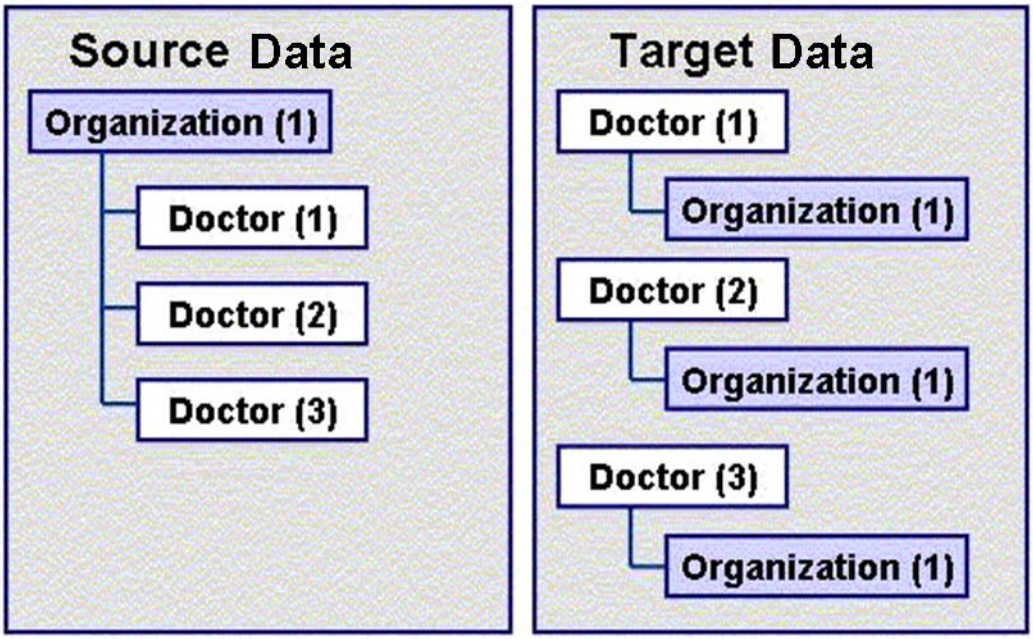

Figure 10. Example Result of a Parent-Child Inverted Relationship

In parent-child inverted mapping, the rules state that one parent source datum should be populated for each of the children.

For example, a single organization that has three doctors in the source data will appear as three doctors each with the same organization in the target data. The parent element in the source data, which is the organization in this case, is populated for each of the data type attributes in the target.

Siblings

In 0 Sibling, we defined the sibling relationship. With siblings, the mapping rule states that unless an explicit mapping is present for the common parent segment, one element of data in the source data file represents one data type attribute in the target file.

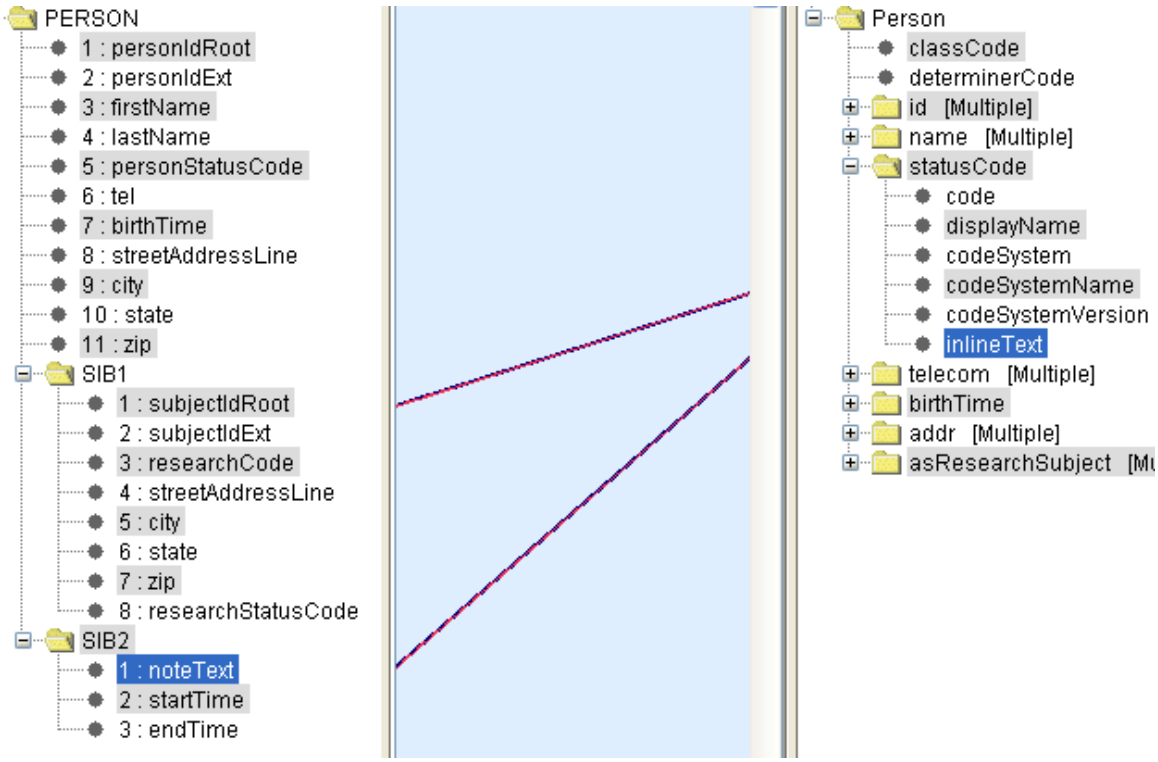

Figure 11. Implicit Mapping on Sibling Structure

For example, in the figure above, `PERSON.SIB1.subjectIdRoot` from the source is mapped to `Person.statusCode.codeSystemName` in the target, and `PERSON.SIB2.startTime` is mapped to `Person.statusCode.inlineText`.

According to the sibling rule, `Person.statusCode.codeSystemName` and `Person.statusCode.inlineText` from the target are siblings under the `Person.statusCode` context.

As an extension to the Sibling definition, we also consider that both `PERSON.SIB1.subjectIdRoot` and `PERSON.SIB2.startTime` from the source share the same mutual root at `PERSON`.

To apply the mapping rule stated above, unless there exists an explicit conceptual mapping between `PERSON` from the source and `Person.statusCode` in the target, three records are generated: that one represents `SIB1` data and the other two represent information from the `SIB2` data. In this case, the underlying CSV data must have the following value set.

Field Names	… …	… …	… …
PERSON	… …	… …	… …
Field Names	**subjectIdRoot**	… …	… …
SIB1	20030102	… …	… …
Field Names	… …	**startTime**	… …
SIB2	… …	B	… …
SIB2	… …	J	… …

The generated result contains three records. Note that the first record does not have a value for `inlineText`, while the second and third records do not have values for `codeSystemName`.

Field Names	… …	codeSystemName	… …	inlineText	… …
Record 1	… …	20030102	… …		… …
Record 2	… …		… …	B	… …
Record 3	… …		… …	J	… …

On the other hand, if there does exist an explicit mapping between `PERSON` from source and the `Person.statusCode` from the target, the cardinality definition of the common parent, that is, the `PERSON` element from the source, will have weight on the number of records being generated. For instance, if the cardinality of the `PERSON` element from the source is 1..1, one and only one record will be rendered in the result, no matter how many source data records of `SIB1` or `SIB2` exist, although a validation error will be reported on the mismatched cardinality because more than one data recordsfrom `SIB2` exists.

If the cardinality of the `PERSON` element from the source is 1..*, however, the actual number of data records in `SIB1` and `SIB2` in the real data file rules.

Conceptual Relationship

All mappings have an explicit or implicit conceptual relationship between source and target elements. An implicit relationship is defined by the nature of the link. For example, a child segment mapped to a child clone implies the existence of an implicit relationship between their common parents. The mapping tool also supports explicit mapping, where the user creates a link between a source segment and a target clone, or between a source segment and a target attribute. In explicit conceptual mapping, unless the target has a one-to-one (1..1) cardinality relationship with its parent, the cardinality from the source will help govern the number of records of the targeted node being generated, given its underlying children nodes' mapping structure.

Transformation Service

caAdapter uses the transformation service to determine target message structure based on mapping rules. These rules are based on the types of mapped relationship used in the .map file.

The Transformation Service builds the HL7 V3 XML message. It creates the message by traversing an object graph of HL7 clone objects. It then creates and populates these clone data objects based on the mapping rules.

Mapping Rules

In order for clones to be aggregated, a relationship needs to be established between the target clone and a source segment. The following rules define how these relationships are derived:

- The transformation service checks for conceptual mappings to the target clone in question. These are treated as having an explicit relationship between the target clone and the source segment.
- The transformation service checks for data mappings to the target clone in question. All source segments that are mapped to attributes of this target clone are treated as having an implicit relationship between the target clone and those source segment(s). Then the common parent of these segments is derived.
- The number of target clones created equals the number of corresponding implicit or explicit source segments.
- Child target clones will inherit the implicit or explicit source segments relationships from their parent clones. In these cases a single clone is created.

Mapping Functions

Functions are inline processes that are used to provide certain transformations to the data during map processing. For example, the concatenate function can combine two fields in the source data file into a single data type attribute.

Only fields from a single segment can be used as input to a function.

Functions can be chained together to offer a series of actions to one target attribute. In these cases all inputs to the chain of functions have to be from the same source segment.

Mapping Example

This section provides the procedure to transform a segment-based CSV data file to HL7 V3 mappings.

This example will

- Generate the segment-based CSV data from the sample output data;
- Generate CSV segment specification, a.k.a. SCS file, from the segment-based CSV data;
- Generate HL7 V3 specification, a.k.a. H3S file after some customizations of clone structure and filling in user default values, if any, from the pre-defined COCT_RM999900 HMD template;
- Create mapping specification, a.k.a. Map file, to design the mapping relationship between the SCS and H3S files, via the caAdapter mapping panel;
- Render the HL7 V3 messages given both the sample segment-based CSV output data and the mapping specification as inputs;

The sample output data is organized by patient record (`PERSON`). Each patient record (`PERSON`) participates as a research subject (`RESEARCHSUBJECT`) as the target (`DIRECTTARGET`) of an observation (`OBSERVATIONEVENT`).

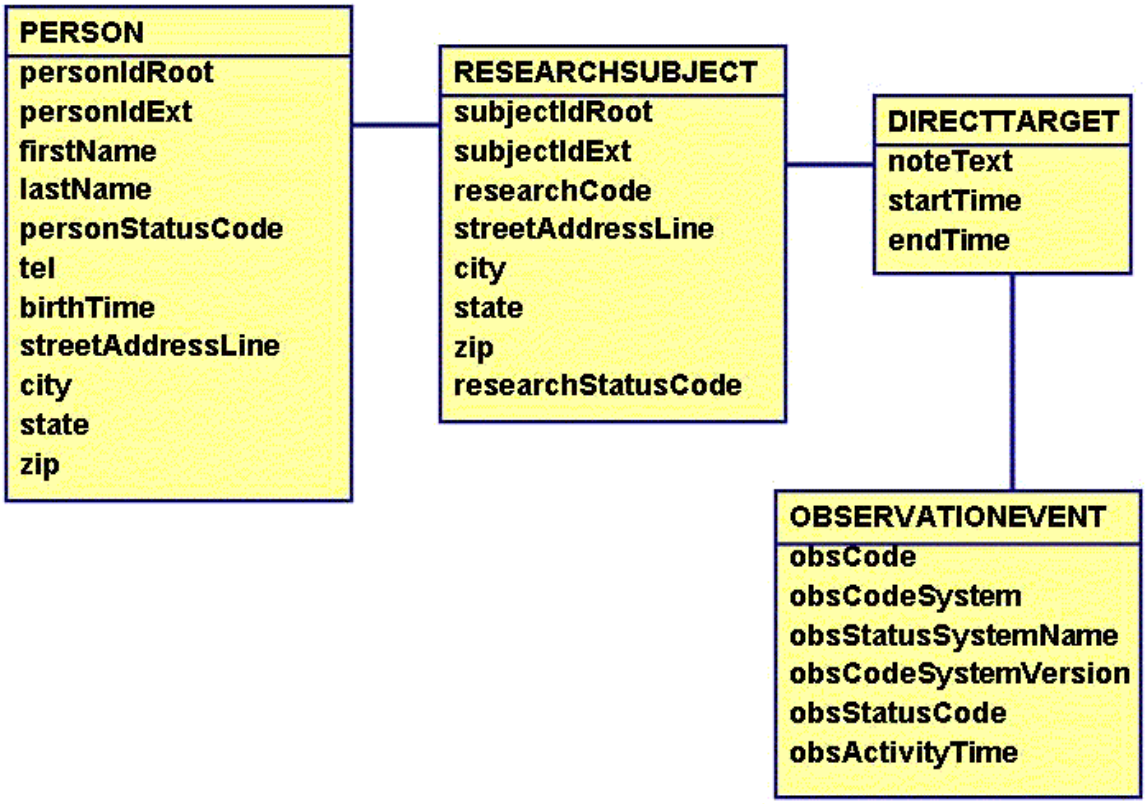

Figure 12. Sample Output Data

Using the sample output data, a CSV data file is generated. Each one-to-many relationship becomes a segment with the PERSON segment being the root.

```
PERSON, 2.16.840.1.1138..., 12345, John, Smith, 01, 800-555-9999, 20030102, 456
Washington Blvd, Washington, DC, 20002
RESEARCHSUBJECT, 2.16.840.1.113883.19.1, 12345, ABC123, 456 Washington Blvd,
Washington, DC, 20002, 01
DIRECTTARGET, This is target 1, 20030102, 20030704
OBSERVATIONEVENT, ABC123, 2.16.840.1.113883.5.4, caDSR, 3.01, 01, 20030607
DIRECTTARGET, This is target 2, 20030704, 20040214
OBSERVATIONEVENT, LMO456, 2.16.840.1.113883.5.4, caDSR, 3.01, 01, 20030823
DIRECTTARGET, This is target 3, 20050315, 20050926
OBSERVATIONEVENT, XYZ789, 2.16.840.1.113883.5.4, caDSR, 3.01, 01, 2005076
```

Figure 13. Sample Output CSV File

Source Specification

The CSV specification (`.scs` file) is created from the sample output CSV file.

Field names and segment structure are defined by use of the scs panel in caAdapter. Users can rename names of field and segment elements. Users can also modify parent-child relationships between segments and fields or between segments by dragging and dropping source segments or fields onto the targeted segment location. In addition, users may reshuffle the order of field elements under the same segment to create alternative data structure layouts. These structural relationships (i.e. parent-child, siblings, etc.) are essential because combined with actual CSV data, they contribute to how the transformation service interprets mapping relationships when generating the target HL7 V3 messages.

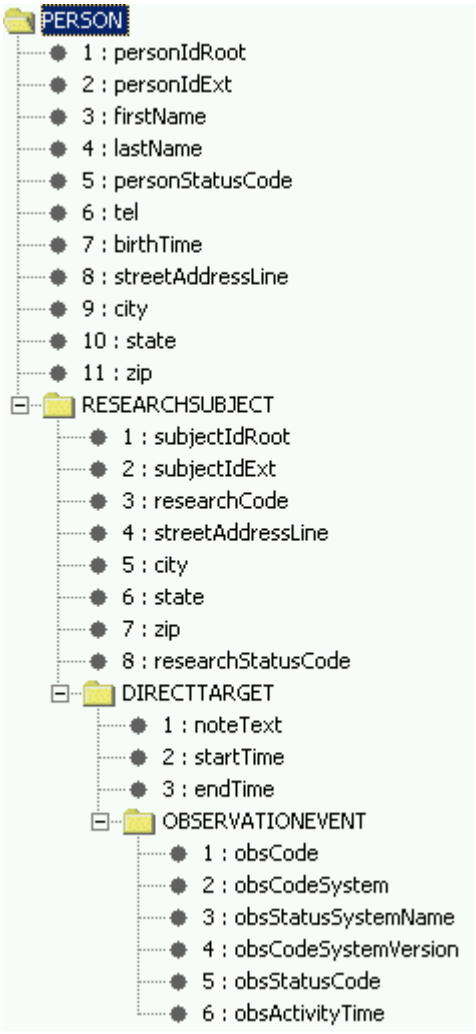

Figure 14. Example of CVS Source Specification

In our example, each of the source segments is a child of the segment that came before it.

Target Specification

The example target specification (the .H3S file), includes default values defined for some fields or default clone layout per generic message type. For example, `observationEvent` code has a default `codeSystem` already applied. The `codeSystem` is the same for all messages created with this h3s specification file. Multiple instances of clones or data types could also be added to the target specification.

Mappings

Each transformation from the source specification (SCS file) to the target HL7 V3 specification (H3S file) in this example requires an explicit mapping relationship. In the example, an explicit map exists between `DIRECTTARGET` and the HL7 V3 clone of `asResearchSubject`.

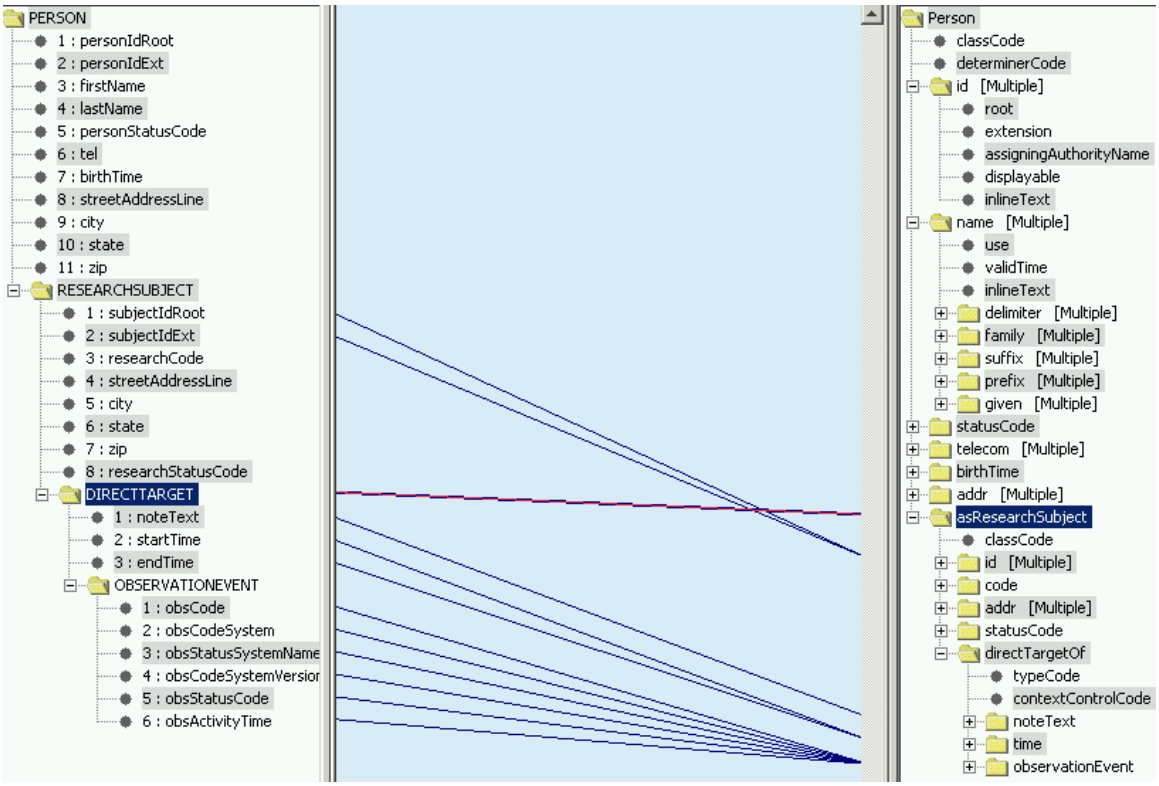

Figure 15. Example Mapping

The rest of the mapping is simple one-to-one maps between the source specification and target specification.

Generating the HL7 V3 Messages

After the map file is created, users can generate an HL7 V3 message in caAdapter to render the HL7 V3 message instances for the mapped HL7 V3 specification. Following is the example HL7 V3 message.

```
<?xml version="1.0" encoding="UTF-8"?>
<COCT_MT999900.Person classCode="PSN" determinerCode="INSTANCE">
  <asResearchSubject classCode="RESBJ">
     <id root="2.16.840.1.113883.19.1" extension="12345"/>
     <code/>
     <statusCode/>
     <directTargetOf typeCode="DIR">
        <noteText>This is target 1</noteText>
        <time>
           <low value="20030102"/>
           <high value="20030704"/>
        </time>
```

```
         <observationEvent classCode="OBS" moodCode="EVN">
            <code      code="ABC123"      codeSystem="2.16.840.1.113883.5.4"
codeSystemName="caDSR" codeSystemVersion="3.01"/>
            <statusCode code="01" codeSystem="2.16.840.1.113883.19.1"/>
            <activityTime value="20030607"/>
         </observationEvent>
      </directTargetOf>
   </asResearchSubject>
   <asResearchSubject classCode="RESBJ">
      <id root="2.16.840.1.113883.19.1" extension="12345"/>
      <code/>
      <statusCode/>
      <directTargetOf typeCode="DIR">
         <noteText>This is target 2</noteText>
         <time>
            <low value="20030704"/>
            <high value="20040214"/>
         </time>
         <observationEvent classCode="OBS" moodCode="EVN">
            <code      code="LMO456"      codeSystem="2.16.840.1.113883.5.4"
codeSystemName="caDSR" codeSystemVersion="3.01"/>
            <statusCode code="01" codeSystem="2.16.840.1.113883.19.1"/>
            <activityTime value="20030823"/>
         </observationEvent>
      </directTargetOf>
   </asResearchSubject>
   <asResearchSubject classCode="RESBJ">
      <id root="2.16.840.1.113883.19.1" extension="12345"/>
      <code/>
      <statusCode/>
      <directTargetOf typeCode="DIR">
         <noteText>This is target 3</noteText>
         <time>
            <low value="20050315"/>
            <high value="20050926"/>
         </time>
         <observationEvent classCode="OBS" moodCode="EVN">
            <code      code="XYZ789"      codeSystem="2.16.840.1.113883.5.4"
codeSystemName="caDSR" codeSystemVersion="3.01"/>
            <statusCode code="01" codeSystem="2.16.840.1.113883.19.1"/>
            <activityTime value="2005083"/>
         </observationEvent>
      </directTargetOf>
   </asResearchSubject>
</COCT_MT999900.Person>
```

Conclusion

The example shows the steps to generate HL7 V3 messages from a segment-based CSV data file. To build the mapping file, users need to have both the SCS and H3S files available. The SCS and H3S files could be created and modified by utilizing the SCS and H3S modules, respectively, in the caAdapter application. Once the map files have been created, HL7 V3 message instances will be generated via HL7 V3 message module for the mapped HL7 V3 specification.

Chapter 2 Converting HL7 V2 to HL7 V3 Messages

This chapter provides instructions on using caAdapter to map and convert an HL7 V2 message to the HL7 V3 message format.

Topics in this chapter include:

Methods for Converting HL7 V2 Messages

You can convert an HL7 V2 message to an HL7 V3 message using XML-formatted HL7 V2 resources, which are available as a free download at the HL7 home page, http://www.hl7.org. HL7 V2 messages can be directly converted to HL7 V3 messages without any pre-processing. Since this method does not validate HL7 V2 syntax and vocabulary, you would achieve the best results if you use another tool to validate and filter the source HL7 V2 message.

The current V2-related functions in the caAdapter GUI menus and all the explanations in this chapter use the method described above. If you need to use the legacy method, refer to earlier versions of the caAdapter User's Guide or contact Application Support.

Understanding Mapping and Transformation

When you convert an HL7 V2 message to HL7 V3 format, an SCS file is not required. You must only select the HL7 V2 version and message type in the mapping panel.

The process of transforming HL7 V2 messages to the HL7 V3 message format has the following three steps.

1. Generate an H3S file for the target HL7 V3 message (see *Step 1: Generating the H3S File*, below)
2. Create a .map file between the source HL7 V2 message type and the target H3S file (see *Step 2: Creating the .map file* on page 25)
3. Transform and generate a V3 message from the source HL7 V2 message file using the .map file (see *Step 3: Transforming a Message from V2 to V3* on page 25)

Step 1: Generating the H3S File

Generating an H3S file for HL7 V2 to HL7 V3 mapping follows the same process as explained in the *caAdapter 4.3 User's Guide*.

Knowing which V3 message type is appropriate for a given V2 message type is extremely important.

Proceed to Step 2: Creating the .map file, below.

Step 2: Creating the .map file

To create the .map file

1. Select **New** > **HL7 V2 To HL7 V3 Mapping and Transformation Service** > **New HL7 V2 to HL7 V3 Map Specificatio**n. The main mapping panel appears.
2. Click the **Open Source** button. The HL7 V2 Schema Selection dialog box appears.
3. Select the HL7 V2 message version and schema. Click **OK**. The HL7 V2 message tree appears on the left side of the window.

For more information on working with mapping files, see the *caAdapter 4.3 User's Guide*.

Step 3: Transforming a Message from V2 to V3

When you transform a message from HL7 V2 to HL7 V3, your input file type is .hl7. You must save your source HL7 V2 message as an .hl7 file before transforming it.

To transform a message from HL7 V2 to HL7 V3

1. Select **New** > **HL7 V2 to HL7 V3 Mapping and Transformation Service** > **New HL7 V2 to HL7 V3 Message**. The New HL7 V2 to HL7 V3 Message dialog box appears.
2. In the Data File field, click **Browse**. The Open dialog box appears. Locate an .hl7 file and then click **OK**. The New HL7 V2 to HL7 V3 Message dialog box appears.
3. In the Map Specification field, click **Browse**. The Open dialog box appears. Locate a .map file and then click **OK**. The New HL7 V2 to HL7 V3 Message dialog box appears.
4. In the Result File field, click **Browse**. The Open dialog box appears. Name a .zip file and then click **OK**. The New HL7 V2 to HL7 V3 Message dialog box appears.
5. Click **OK**.

Using the API for HL7 V2 to HL7 V3 Transformation

You can use the caAdapter API to transform HL7 V2 messages to HL7 V3 format.

Earlier versions of caAdapter required two steps to transfer HL7 V2 messages to HL7 V3. First the messages had to be converted to CSV using KNU (Kyungpook National University) V2 parser engine. This process includes V2 message syntax and vocabulary validating, and can process segments such as ‘OBX’. In the second step, caAdapter transformed the CSV file to an HL7 V3 message. If you have a legacy application using this earlier process, legacy APIs are included in the caAdapter distribution.