How to Model a Clustered and Complex Data Service in Visual Composer for SAP Netweaver CE

Applies to:

Visual Composer for enhancement package 1 for SAP NetWeaver Composition Environment.

Summary

This article provides an understanding about clustered data services and describes how to model a Visual Composer application using a complex data structure.

Author: Netanel Slomianski

Company: SAP

Created on: 05 November 2008

Author Bio

Netanel Slomianski is a Visual Composer solution expert working for Visual Composer Solution Office group.

Table of Contents

1. What is a Clustered Data Service?

Data services can return data in different structures: A single record, a record set or a clustered structure. A clustered structure is a hierarchical structure that consists of single records and record set nodes that have a parent-child relationship between them.

This makes the readability of this data service more difficult and requires special handling methods in Visual Composer.

2. What is Normalization?

According to Wikipedia, normalization is “any process that makes something more normal, which typically means conforming to some regularity or rule, or returning from some state of abnormality” (http://en.wikipedia.org/wiki/Normalization).

To represent these kinds of clustered data services, special handling, called normalization, is performed in Visual Composer. Normalization is basically a transformation of the service’s data structure that enables simpler modeling of the used services.

Normalization serves the modeler by simplifying modeling while enabling use of even very complex data structures. This simplicity is most apparent in two places:

- The resulting model itself (its UI) is simpler to construct. After normalization, there are fewer levels of data structures to dive into.
- Dynamic expressions are simplified. Since field references are performed by referencing the entire path from the data structure root to the field, if the field is located a few levels deep, than referencing it (using it in dynamic expressions) becomes a painful chore. Dynamic expressions are widely used in Visual Composer when modeling applications, there is therefore a strong motivation to simplify them.

3. What Kind of Node Cardinalities are Available in SOA Services and How to Use Them in Visual Composer

There are a few types of node cardinalities in a SOA service:

- 0..1: You may have none or exactly one record in this data structure. This is an optional data structure.
- 1..1: You have exactly one record in this data structure.
- 0..n: You may have none or many records in this data structure.

When modeling with a SOA service, you will have all of the above node types.

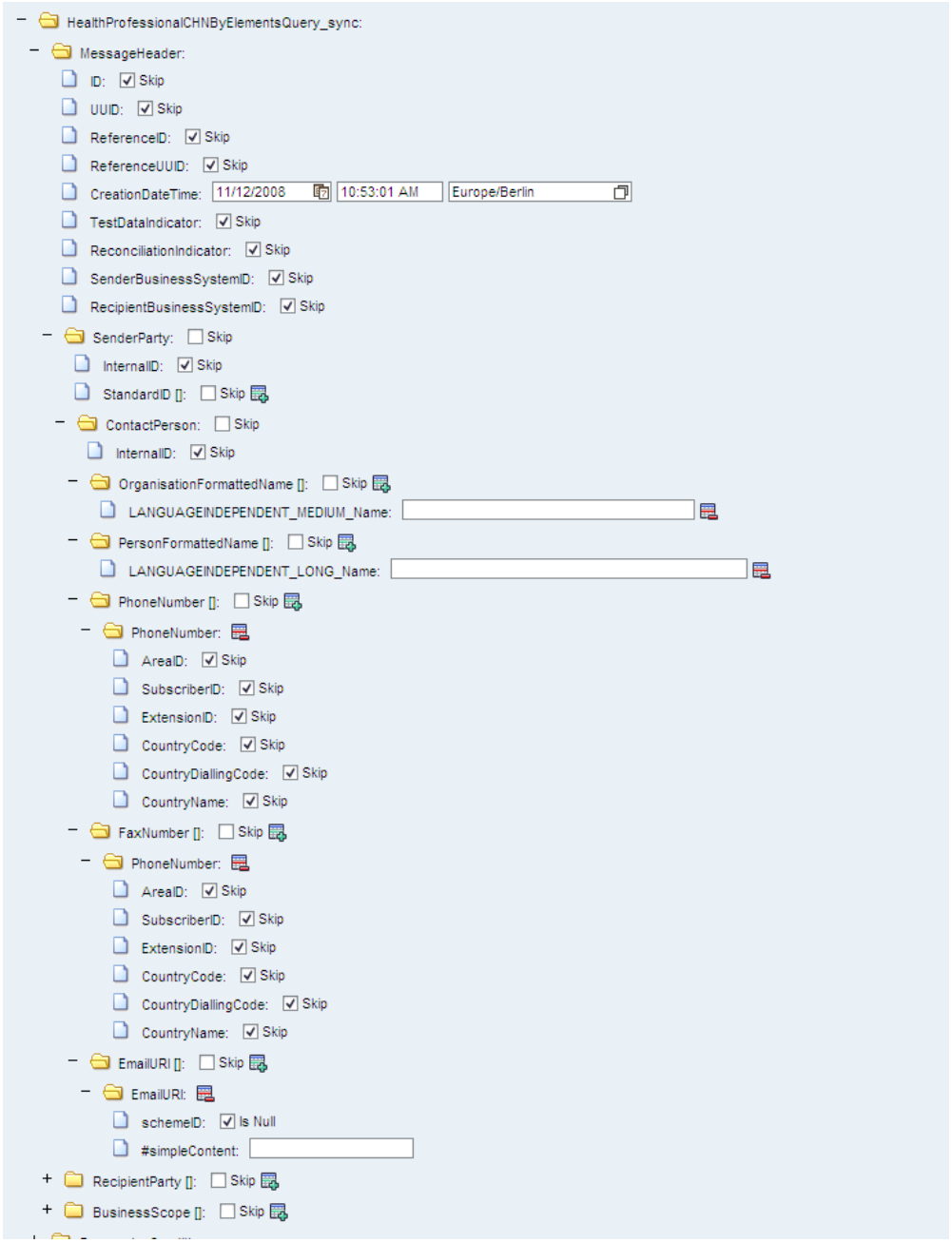
- A data service that has a node with cardinality of 0..1 in its input port will expect to get one or no records of data for this node. If you don’t intend to provide the data, you should map null values in this record.
- A data service that has a node with cardinality of 1..1 in its input port will expect to get a record of data for this node.
- A data service that has a node with cardinality of 0..n in its input port will expect to get a record set of data for this node, or no data at all (null value).

4. How to Locate Fields in a Normalized Service

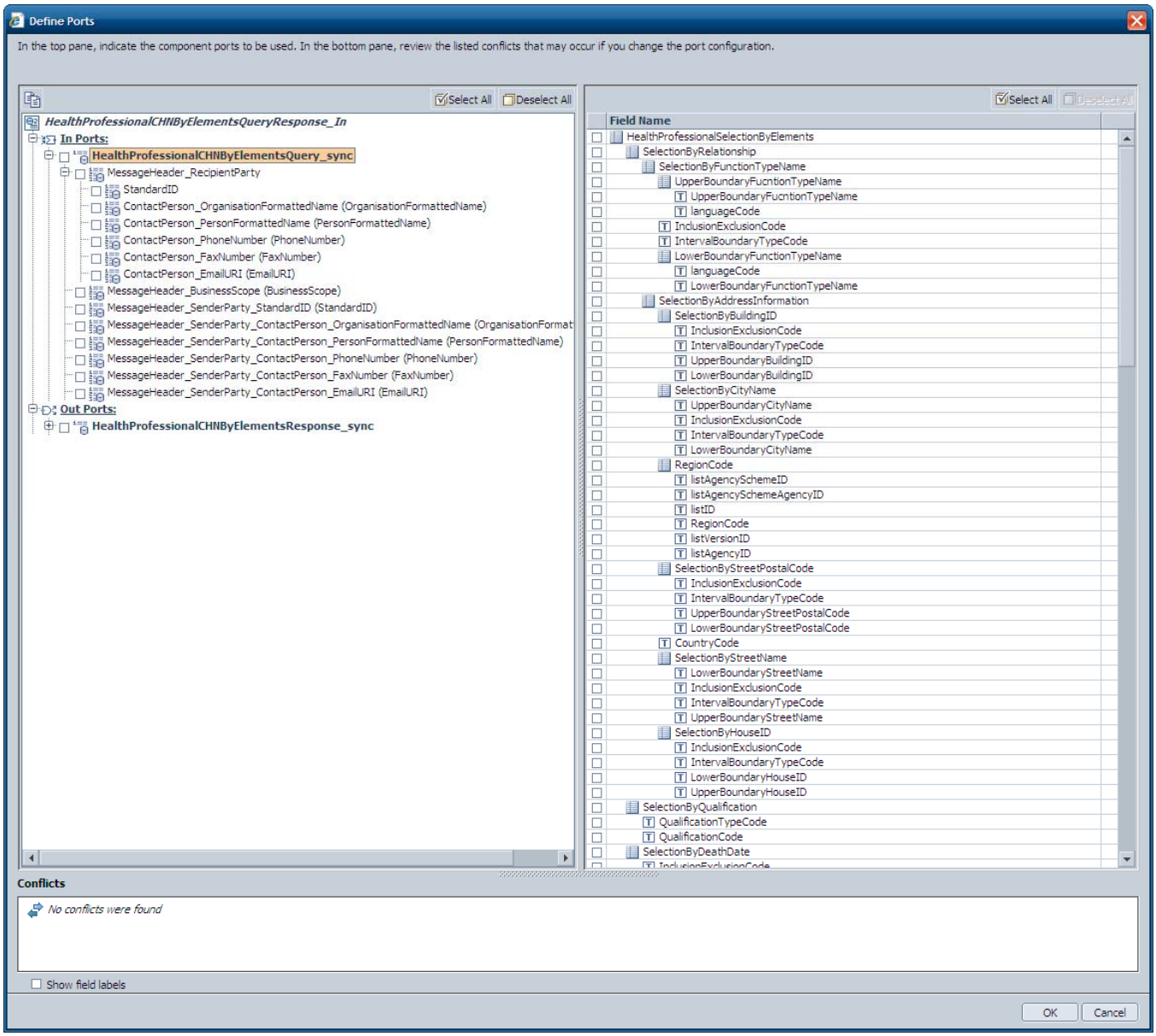
Normalization transforms the service’s data structure in such a way that the data structure can have a single record only in the root node of a structure. Single records (nodes with cardinality of 0..1 and 1..1) that were not originally located in the root node, will be moved up and merged with the closest parent record set if available. If the record does not have a record set parent in the data structure hierarchy, it will be moved to the root node.

The outcome of normalization is that the clustered tree has no internal single records; i.e. after normalization, a single record can exist only in the root of the clustered tree structure, or under an array. All other internal data structures can be only record sets.

Let's take for example a SOA service that gets a clustered structure input. When you use the Web Service Navigator (<protocol>://<host>:<port>/wsnavigator) to invoke it, you see the following output:

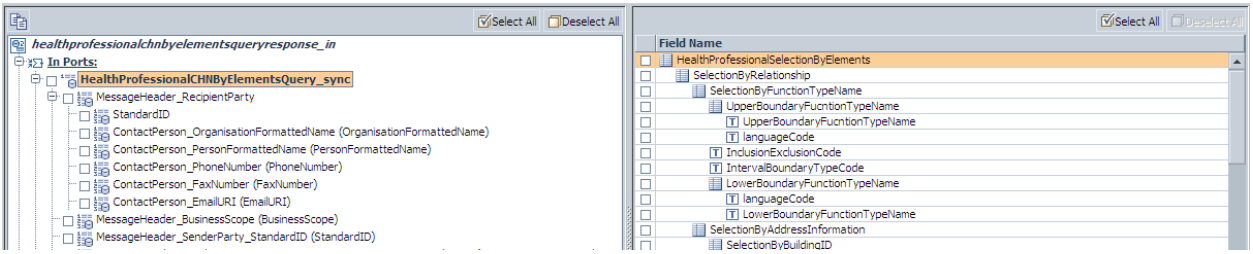


When you import the service to Visual Composer, you will see it normalized:

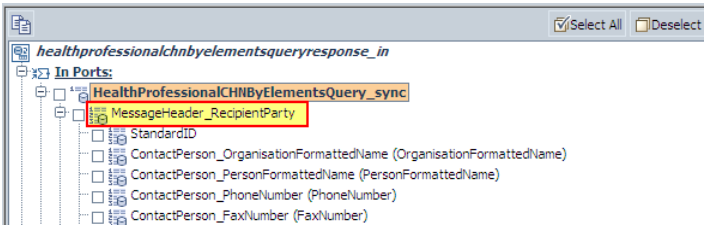


You'll notice that the structure of the data tree is different. The logic is:

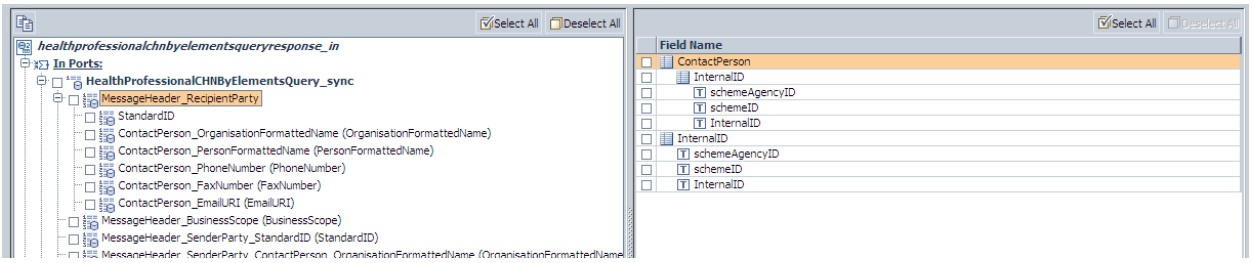
- *HealthProfessionalSelectionByElements* is a record of cardinality 1..1, the normalization process moves it under the root node of the tree, with all of its subnodes:



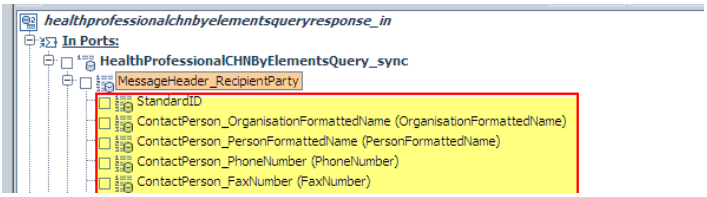
- Under the *MessageHeader* node, we have two record sets: *RecipientParty* and *BusinessScope*. The normalization process removes the *RecipientParty* node and unites it with *MessageHeader* in the form of <parent node>_<child node name>, i.e. *MessageHeader_RecipientParty*.



- The *RecipientParty* record set has the *InternalID* and *ContactPerson* records under it. This means that after normalization they are moved to their parent node. Note that their parent node is not *BusinessDocumentMessageHeaderParty* but *MessageHeader_RecipientParty*



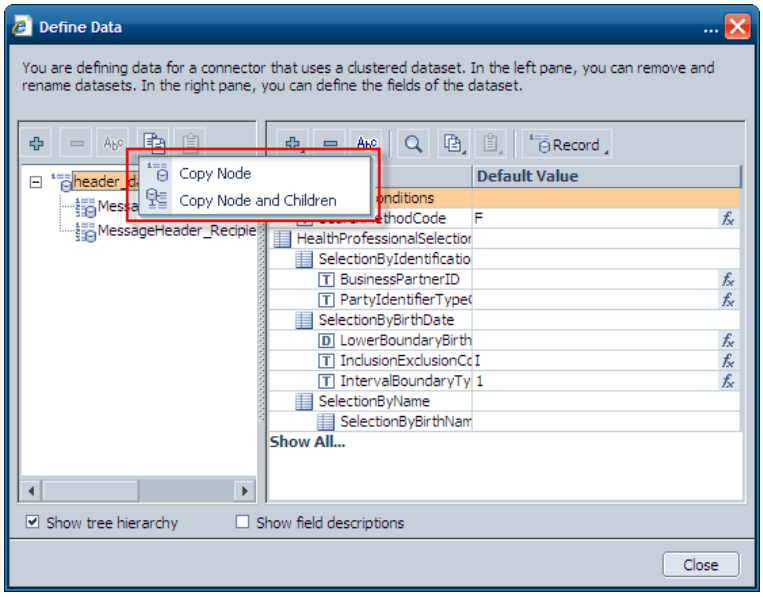
- Note that there are six record sets under the *MessageHeader* node. These record sets will also move under the *MessageHeader_RecipientParty* node with its subnodes which are all single records.



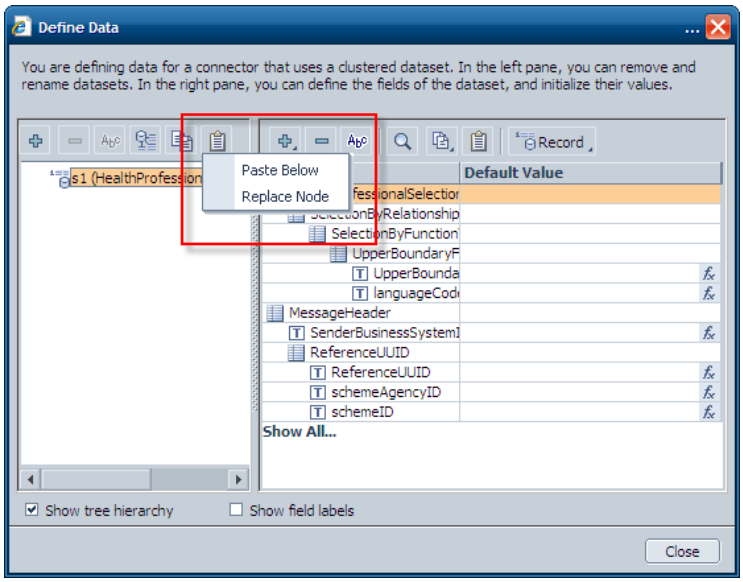
5. Streamlining Your Model's Data Storage Resources

Since SOA services (and complex services in general) have hundreds and sometimes thousands of fields in them, it is imperative to use only the fields required for the service operation when modeling. When you import a complex service to Visual Composer, choose only the fields you really need. Take into account that you will probably want to send this complex structure to other components in the model (which will make duplicates of it) and therefore cause very large memory usage at design time and runtime.

You can add fields to the data structure of the service later on. This can be done by right-clicking on the data service and choosing *Redefine Ports*. A dialog box is opened from which you can select the additional fields that you need. Once you added the additional fields to the input/output port, you need to have the matching fields added to the other objects that are connected to this service. You can either create the fields you need (which can be quite cumbersome since you can find yourself adding a whole lot of fields...) or copy and paste the fields. This can be done by creating a new element, by dragging a new data share from the service's port, selecting the specific fields you want to add and then copying the fields from it to all the other elements in the model.



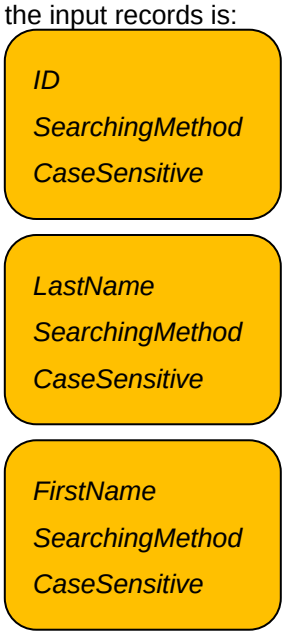
You can either choose *Copy Node* or *Copy Node and Children*. After you copy the relevant node, you can then open the object's *Define Data* dialog box and paste the node there. You can choose to paste the node below the selected node or replace the selected node.



6. How to Model an Input Form with 0..1 Node Cardinality

When you model a data service that has input records of 0..1 cardinality, the data service expects to receive null values for the records that are not used in the process. In the following example, we'll explain it further:

Let's assume you have a service that can search for an employee by first name/last name/ID. The structure of the input records is:



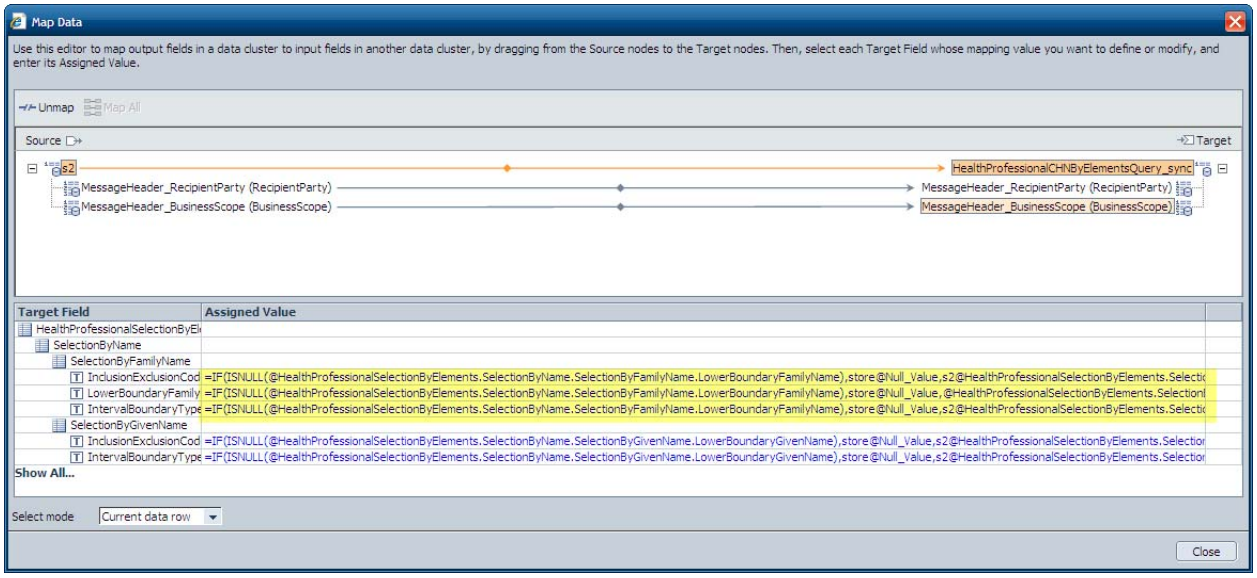
The modeler entered default values for the fields *SearchingMethod* and *CaseSensitive* and configured only the *ID* field to be visible. This is true for the other two input records (*LastName* and *FirstName*).

If the user chooses to search only by ID, the other two records' fields should be mapped to null, since the service expects a value to be entered to all fields of a record when at least one field has an input.

Note, that if the user did not enter anything in the form and the modeler did not map or set any fixed values, the runtime will implicitly send null to the service. This behavior has to be handled by the modeler only if the UI permits to enter values in different 0..1 nodes or if there are fixed values in the default values or initialization values.

In the next section, I'll explain exactly how to do this.

Note: A null value should be mapped to each field in the record, not only to one of the fields.



6.1 Passing Null Values to a SOA Service

SOA services require that fields used for inputting data that are not filled with any data, will get a null value when the service is executed. In Visual Composer, you need to explicitly set a null value to the field. Since currently Visual Composer does not have a designated keyword such as “NULL” in its language, this can be achieved by:

1. Creating a data store connector
2. Adding a new field of the corresponding field type. Do not assign any value to it. This will assign a null value to it.

After you create this field, you can map the value of the service’s input field to this field. Usually you will use a dynamic expression that will look like:

```
=IF(ISNULL(@FamilyName),store@Null_Value,@FamilyName)
```

Where:

`@FamilyName` is the family name input field in the form

`store@Null_Value` is the empty String field in the data store connector

This dynamic expression checks whether the user has filled anything in the *Family Name* input field. If not, the NULL value is mapped to the service’s input field; otherwise the actual form field value is mapped to the service input field.

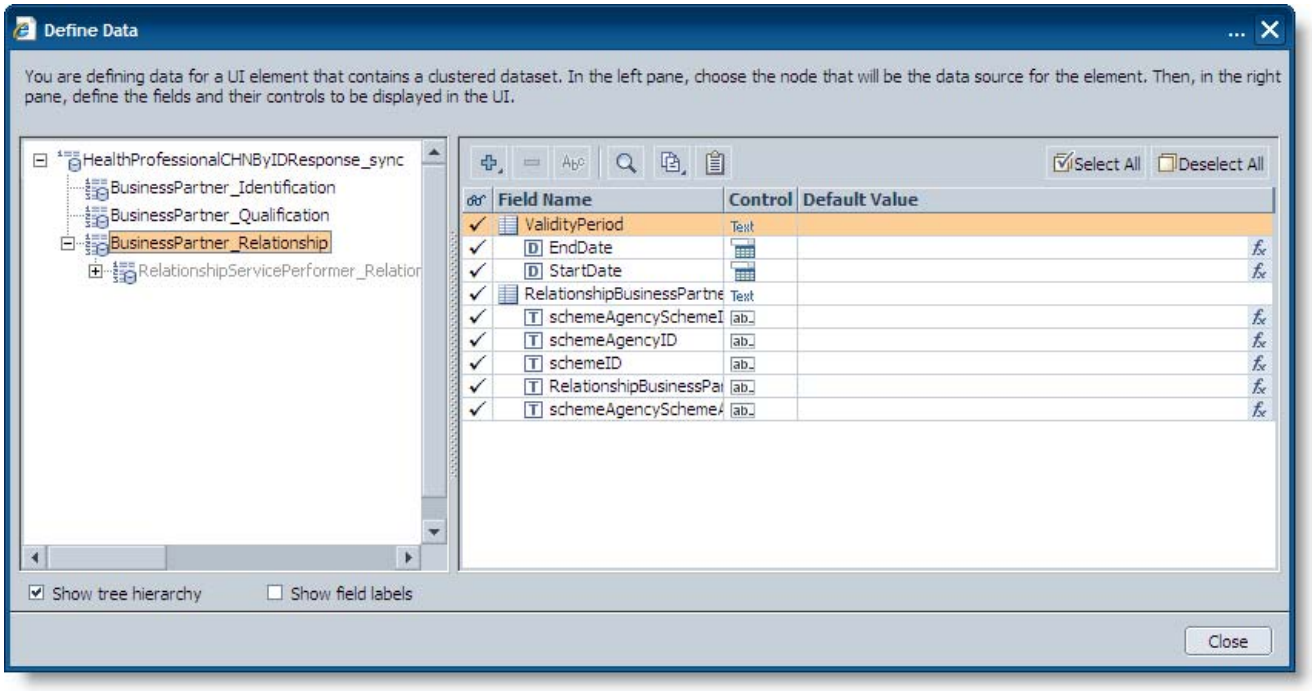
7. How to Display Data from Different Hierarchies in the Complex Structure

Visual Composer creates UI for data structures according to their cardinality:

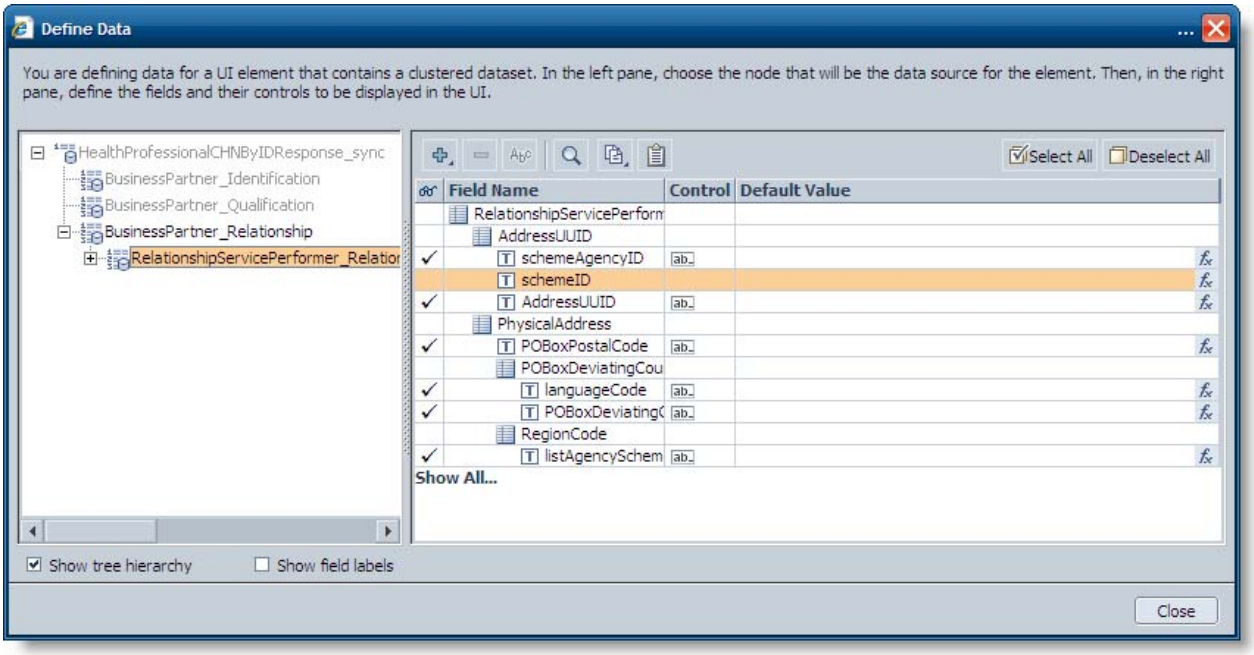
- For a record – a form is created.
- For a record set – a table is created.

Visual Composer will create the UI for a data service output port according to the data nesting structure. Let’s use the following example to understand it better.

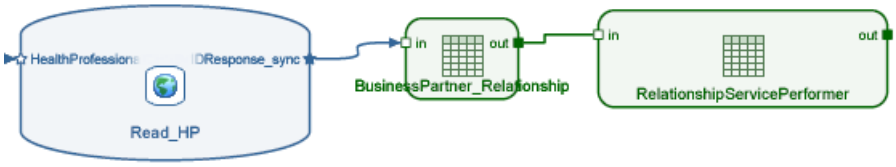
After dragging a table view from an output port, here are the available nodes:



Note that a table UI element can show a single node of data. When displaying a hierarchy of nodes, skipping an array node is not permitted. This is the reason that the BusinessPartner_Relationship's subnode cannot be selected, since it's a record set within a record set. For displaying such a hierarchy, another table should be dragged from this one. After dragging it, the *Define Data* dialog box will look like this:



On the Design board, the model looks like this:



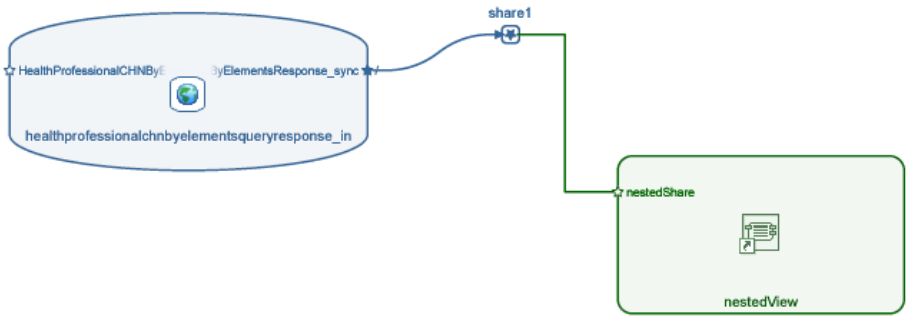
There are situations when you have a complex data service such as this with several nested record sets and you only want one table that will show data from a certain table from up the structure. You can achieve this by creating a control with a virtual field that references a value of the parent table/form. Once you do this, you either set the visibility condition of the parent table/form to true or false (according to your needs).

8. How to Model a SOA Application with Multiple Components

When modeling a complex application with several forms, tables and UI elements, it is usually a good idea to separate it into components that hold relevant data together. This means you will need to transfer data from one component to another. You can do this by “loading” the data onto a signal event or by using the Data Share connector in Visual Composer.

You can connect the data service’s output port to a data share connector. A data share connector is a non-UI element that can hold complex (as well as simple) data structures. Its *Scope* attribute can be *Private* or *Public* (*Private* means it can be accessed only by elements within the same component, and *Public* means it can be accessed from nested Visual Composer components or other composition tool components). After you connect the data service and the data share connector, an automatically created corresponding data structure is created in the data share connector with automatic mapping of the values from the data service to the data share connector.

You can then use the values that are stored in the data share in any place that you can use dynamic expressions. You can also connect it to another data share connector in a referenced composite view. This will create a binding connection between the two data share connectors that will cause them to update each other. Note that the data share scope in the nested composite view should be set to public if you want to access it from the parent composite view. Remember to copy the data structure from the data share connector in the parent composite view and paste it in the data share connector in the nested composite view. This will ensure you get perfect data binding. After you do that, you can right-click on the nested composite view, choose Redefine Ports, select the data share connector and connect the parent composite view's data share connector to it.



Related Content

Visual Composer 7.1 Development Guidelines and Best Practices

Visual Composer 7.1 (CE) Page on SDN

Visual Composer Naming Conventions

Copyright