QlikView

QlikView Expressor Extensions SDK Tutorial: Enabling Schema-from-Type

QlikView Expressor Version 3.9

Newton, Massachusetts, April 2013

Authored by James Siwila

This tutorial exercise demonstrates how to develop an Extension Object Schema Descriptor that allows a Schema to be created from a Composite Type. This tutorial builds on exercises in the online help for QlikView Expressor 3.9. Before attempting to complete this tutorial, you should have a working knowledge of the Extensions SDK and complete at least the first three tutorial exercises in the online help.

This exercise uses the File Connection Descriptor, Object Schema Descriptor, Read Operator Descriptor (Schema & Connection), and the Datascript Modules developed in the exercise titled:

3. Read Operator with Schema Exercise

The Object Schema Descriptor constructed in that exercise supports reading a CSV file and creating the Schema fields from that file's data. When data is written out at the end of a Dataflow, the Schema required must be based on data that has been restructured in the Dataflow. The most effective way to create the required Schema is to base it on the Composite Type that represents the data when it reaches the Write operator.

QlikView Expressor's Delimited and Table Schemas can be created from a Composite Type. To enable that capability in an Extension Schema, you set the appropriate option in the Object Schema Descriptor and write a CreateFromType function for the Schema.

In this exercise, we use the CsvSchemaDescriptor from *3. Read Operator with Schema Exercise* and its associated Datascript Module.

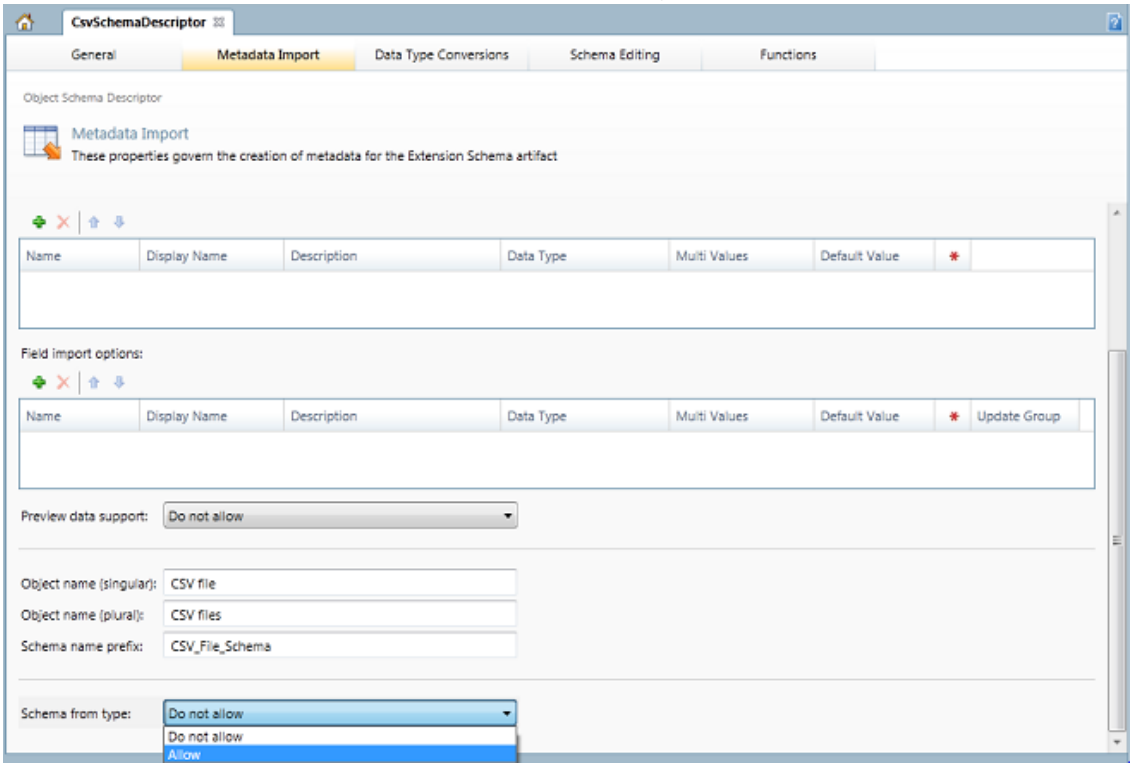
1. Modify the Object Schema Descriptor

1. Open the Schema Descriptor named "CsvSchemaDescriptor."

   This is the Object Schema Descriptor created in *3. Read Operator with Schema Exercise* in the QlikView Expressor 3.9 online help.

2. Select the General tab.
3. Change the Description to "Create a Schema to read from or write to a CSV file."
4. Change the Description (Home Page) to "This Schema reads data from or writes data to a CSV file."
5. Select the Metadata Import tab.

6. Select “Allow” from the drop-down list for the **Schema from type** field.



7. Select the Functions tab.
   Note that a new function is listed in the Extension Function column: `createFromType`.

8. Select the `createFromType` function in the list of functions.
   This function was added when you set the **Schema from Type** to Allow.

9. Use the **Create Function in** drop-down menu to create a function for `createFromType` in the `CsvReadMetadata_DSM` Datascript Module that contains the other functions for this Descriptor.

10. Save the Descriptor.

2. Write the CreateFromType Function

1. Use the **Go To Function** button on the Function tab of the Datascript Editor to open the `CsvReadMetadata_DSM` Datascript Module.

2. Edit the `CreateFromType` function.

```
Function CreateFromType(compositeType)
local schema=
{
 fields={},
 sourceName=compositeType.Name,
 customizable=true
}
for i, typeField in ipairs(compositeType.Attributes) do
  local field={}
  field.name=typeField.Name
  field.attrName=typeField.Name
  field.allownulls=typeField.IsNullable
  field.fieldSpecial={}
  field.dataTypeName='CSV_STRING'
  --set the size if there is a MaxLength constraint in an attribute.
  --MaxLength is used only for the string data type.
  field.size=typeField.Type.Constraints.MaxLength or 0
  field.customizable=true
```

```
    table.insert(schema.fields,field)
end
return 0, schema
end
```

3. Save the Datascript Module.

3. Create a Composite Type

In the first tutorial exercise in the online help (*1. A Basic Hello World Exercise*), we created a Project in which to test the Extension Descriptors and Datascript Modules. A Project is used for tests rather than the Extension Library because you want to exclude the test objects like Composite Types and Dataflows from the Extension Package. Everything in the Library is included in the Package, but Projects are not included.

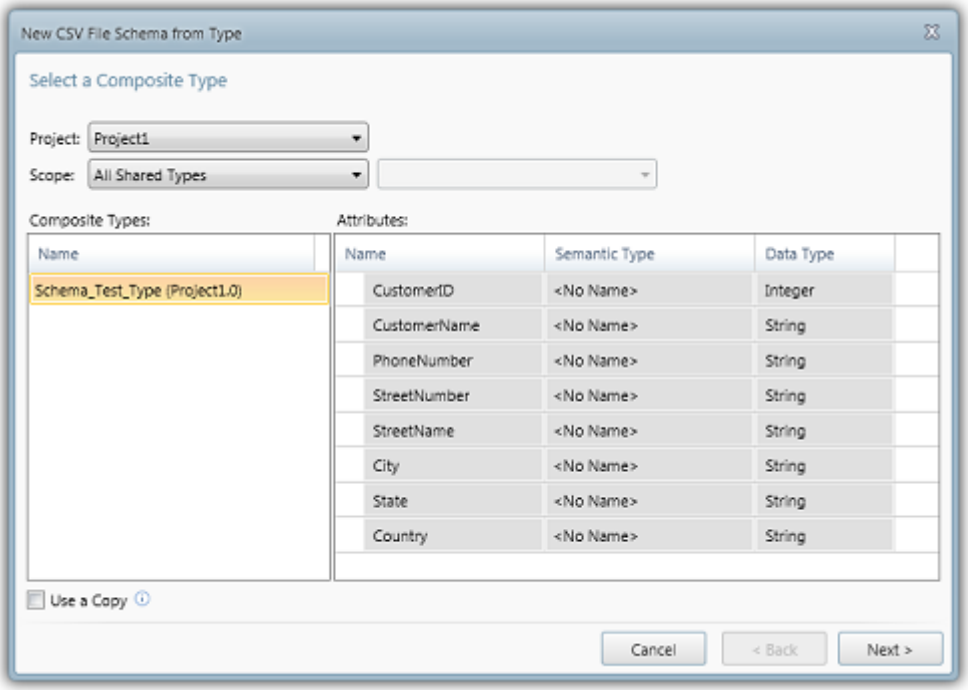
1. In Project1 in the Hello_World_Sample Workspace, create a new Composite Type.
2. Name the new Composite Type "Schema_Test_Type."
3. Add the following Attributes to the Composite Type:

   CustomerID (integer)
   CustomerName (string)
   PhoneNumber (string)
   StreetNumber (string)
   StreetName (string)
   City (string)
   State (string)
   Country (string)

4. Save the Composite Type.

4. Create a Schema from the Composite Type

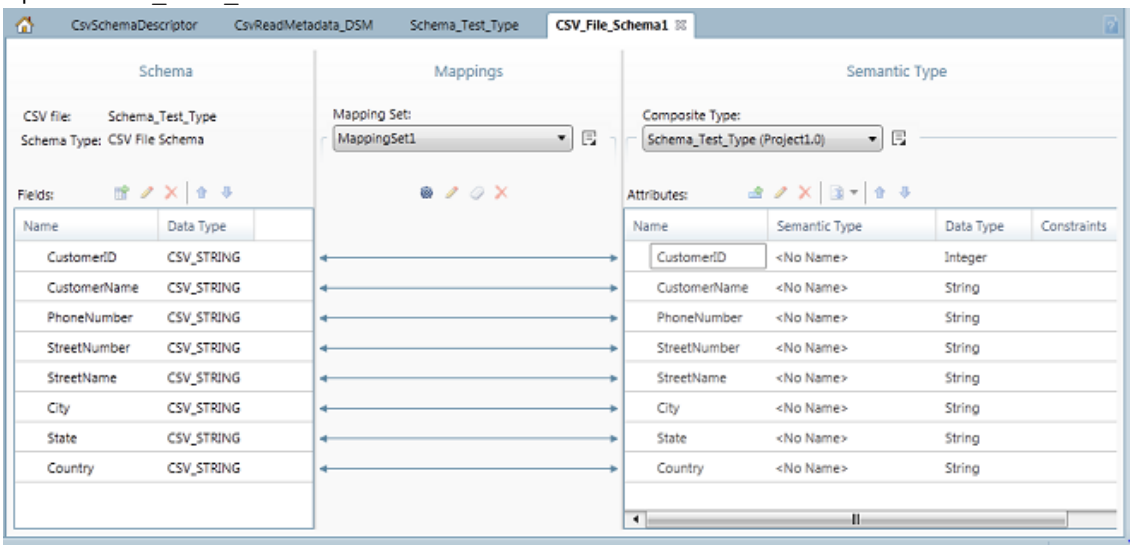
1. In Project1 in the Hello_World_Sample Workspace, right-click on the Schemas folder and select **New>CSV File Schema>From Composite Type**.
2. Review the **New CSV File Schema from Type** dialog box.



If the Schema_Test_Type is the only Shared Composite Type in Project1, it is displayed in the **New CSV File Schema from Type** dialog box. Or it can be selected from a list of Shared Composite Types. But if the dialog box comes up successfully, that confirms that the `CsvSchemaDescriptor` Artifact Descriptor and the `CsvReadMetadata_DSM` Datascript Module are correct.

3. Save the Schema as `CSV_File_Schema1`.

4. Open the `CSV_File_Schema1` Schema.



Note that the CustomerID field has a data type of CSV_STRING. That is because `CsvSchemaDescriptor` uses a File Connection Descriptor for its Connection. Also, on the Data Type Conversions tab, the `CsvSchemaDescriptor` lists integer as a Supported Internal Data Type for CSV_STRING, so it can be mapped to an Attribute with the integer data type.