

TIBCO®Data Virtualization

Tutorial Guide

Version 8.0

Last Updated: November 7, 2018

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO and the TIBCO logo are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries

TIBCO, Two-Second Advantage, TIBCO Spotfire, TIBCO ActiveSpaces, TIBCO Spotfire Developer, TIBCO EMS, TIBCO Spotfire Automation Services, TIBCO Enterprise Runtime for R, TIBCO Spotfire Server, TIBCO Spotfire Web Player, TIBCO Spotfire Statistics Services, S-PLUS, and TIBCO Spotfire S+ are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2002-2018 TIBCO Software Inc. All rights reserved.

TIBCO Software Inc. Confidential Information

Contents

Preface	5
Product-Specific Documentation	5
How to Access TIBCO Documentation	6
How to Contact TIBCO Support	6
How to Join TIBCO Community	6
REST Tutorial	7
Understanding REST and the Tutorial Workflows	7
Basic REST Workflow	8
Overview of the ACT Example and Summary of Tasks	8
Creating Studio Resources for Use in this Tutorial	10
Creating Data Sources	11
Introspecting the Data Source	11
Creating the Vplaydata View	13
Creating the Procedures	14
Publishing the Procedures to a REST Service	17
Linking and Mapping Procedures to Specific REST Service URLs	19
Linking the playdataByID Procedure to an HTTP Verb	19
Mapping the Studio REST playdataByID URLs to Specific REST Services	21
Linking the insertPlaydata Procedure to an HTTP Verb	23
Mapping the Studio REST insertPlaydata URLs to Specific REST Services	25
Linking the deletePlaydata Procedures to an HTTP Verb	27
Mapping the Studio REST deletePlaydata URLs to Specific REST Services	28
Adding a Custom Parameter in an HTTP Header for a REST Request to a Published Method	30
Creating a Basic Transform Tutorial	33
Creating the Any-Any Transform	33
Adding a Query to the supplyChainV Transform	34
Linking Outputs for Your Transform	36
Executing and Testing Your Transform	37
Transform Table and XML Data Tutorials	39
Create an XML Definition Set for the Tutorial	39
Converting Relational to Hierarchical Data Tutorial	41
Adding the Views to Convert	41
Adding the Loops and Filters Used to Iterate and Select Data	42
Completing the Links Between Loop and out	44

Adding Loop2 to the Model.	46
Adding Loop3 to the Model.	47
Adding Links to Direct Data Flow	47
Executing the Transform	48
Converting XML to a View.	48
Creating the Transform and Adding the in Parameters	49
Adding Parameters to out	50
Adding the Loop and Iterators	52
Executing the XML2view Transform.	54
Transform XML Schemas Tutorial	57
Create the School.xml Data File for the Tutorial	57
Create an XML Data Source for the Tutorial	63
Create the GetStudents Transformation Procedure	64
Creating the ClassStudent Transformation	67
Adding and Connecting Source Operations	68
Completing the ClassStudent Transformation.	69
Transform Name and Value Pair Tutorial	71
Creating an XML Data Source for the Tutorial	72
Starting the NameValue Transformation Procedure	77
Creating the Resource to Loop Connections	78
Creating the Cursor Outputs	79
Connecting the Loop Operations to out	81
Connecting the Loop Operations to the Cast Operations	81
Allowing NULL Output Values	83
Executing and Testing Your Transform	84
Transform XML to Relational Tutorial	87
Creating the School.xml Data File for the Tutorial.	87
Creating an XML Data Source for the Tutorial	92
Creating the SchoolXML2Rel Transformation	93
Creating the SchoolXML2Rel View.	95
SAP Hana View Tutorial	97
Creating a TDV View using an SAP Hana Data Source	97
Using GROUP BY and COUNT with an SAP Hana Data Source.	97

Preface

Documentation for this and other TIBCO products is available on the TIBCO Documentation site. This site is updated more frequently than any documentation that might be included with the product. To ensure that you are accessing the latest available help topics, please visit:

- <https://docs.tibco.com>

Product-Specific Documentation

The following documents form the TIBCO® Data Virtualization(TDV) documentation set:

- *TIBCO TDV and Business Directory Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.
- TDV Installation and Upgrade Guide
- TDV Administration Guide
- TDV Reference Guide
- TDV User Guide
- TDV Security Features Guide
- Business Directory Guide
- TDV Application Programming Interface Guide
- TDV Tutorial Guide
- TDV Extensibility Guide
- TDV Getting Started Guide
- TDV Client Interfaces Guide
- TDV Adapter Guide
- TDV Discovery Guide
- TDV Active Cluster Guide
- TDV Monitor Guide
- TDV Northbay Example

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website mainly in the HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

Documentation for TIBCO Data Virtualization is available on <https://docs.tibco.com/products/tibco-data-virtualization-server>.

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <https://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking **Register** on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](https://community.tibco.com). For a free registration, go to <https://community.tibco.com>.

REST Tutorial

TDV supports REST (REpresentational State Transfer) for designing Web services. This tutorial focuses on the following standards used to implement a REST architectural style:

- HTTP
- URL

This tutorial includes the following topics:

- [Understanding REST and the Tutorial Workflows, page 7](#)
- [Creating Studio Resources for Use in this Tutorial, page 10](#)
- [Linking and Mapping Procedures to Specific REST Service URLs, page 19](#)
- [Adding a Custom Parameter in an HTTP Header for a REST Request to a Published Method, page 30](#)

Understanding REST and the Tutorial Workflows

Before working through specific examples using code and the Studio UI, it can be valuable to become familiar with some REST concepts. A REST service typically has the following characteristics:

- **Client-Server**—REST uses a pull-based style; in other words, the consuming components pull information about representations.
- **Stateless**—Each request from client to server contains all the information necessary to understand the request, and cannot take advantage of any stored context on the server.
- **Standard Interface**—All resources are accessed using an interface such as the HTTP verbs of GET, POST, PUT, and DELETE.
- **Named Resources**—Each resource is named using a URL.

This section includes these topics:

- [Basic REST Workflow, page 8](#)
- [Overview of the ACT Example and Summary of Tasks, page 8](#)

Basic REST Workflow

This section describes a typical REST development workflow.

The basic workflow associated with REST services

1. Identify all of the conceptual entities that you want to expose as services.
2. Create a URL to each resource.
3. Categorize your resources according to their HTTP verb (or method). HTTP verbs include:

Verb	Typical use
GET	Read operations
POST	Insert and update operations
PUT	Insert and update operations
DELETE	Delete operations

Resources accessed using GET should return a representation of the resource, but not modify it.

4. Specify the format of the response data.

Overview of the ACT Example and Summary of Tasks

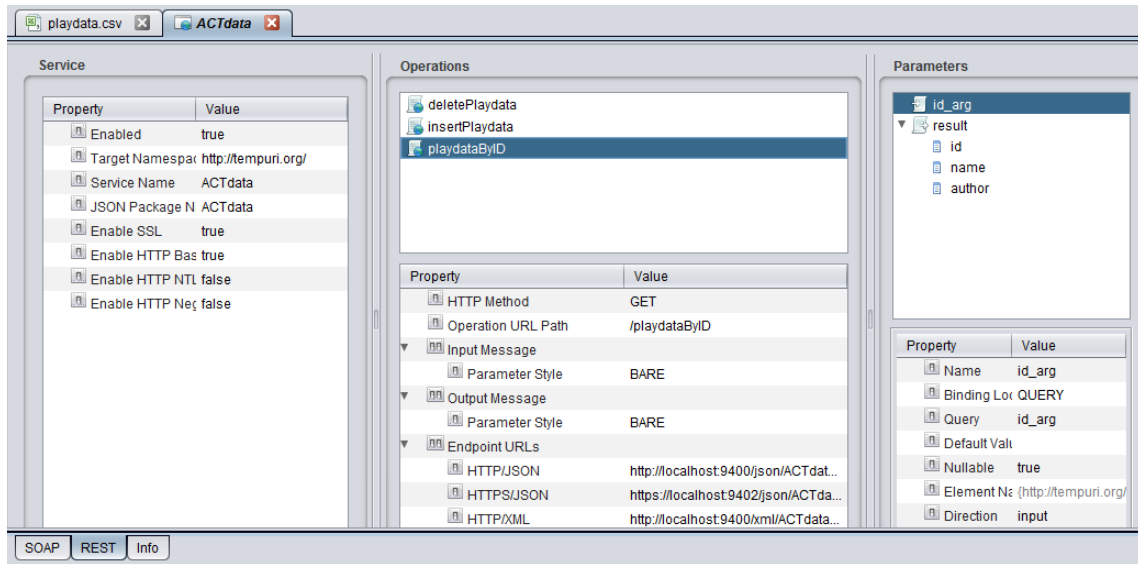
In this use case, you create a REST-based client application that helps the members of an actors’ guild called ACT manage and schedule their plays for the year. As part of this on-line system, members need to:

- Search for plays by their ID number (GET).
- Update author information (PUT).
- Create new plays (POST).
- Delete plays.

This tutorial guide helps you:

- Create play data in a CSV (comma-separated values) file
- Create the TDV SQL procedures that read, update, insert and delete data
- Understand how to create a REST client

TDV Studio deploys REST services to the TDV Web Service.



This tutorial pays special attention to the mapping of procedures to HTTP methods, and the construction of the necessary paths and arguments that make up the REST API for the sample application.

Note: This application is only an example of how you can set up a REST framework. The REST infrastructure in TDV is flexible and does not enforce any specific mapping or operation URL PATH structure.

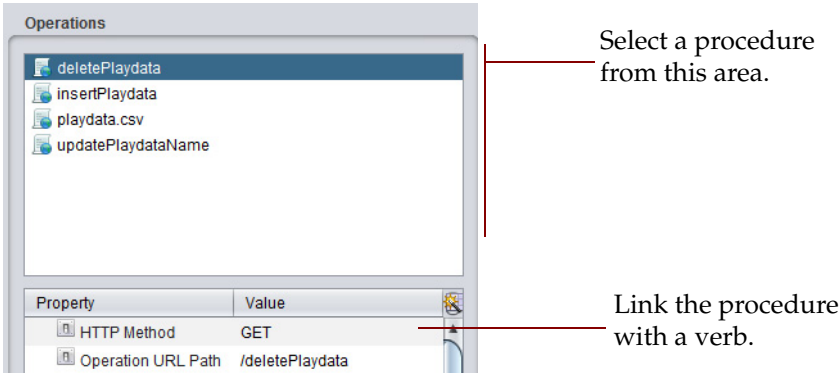
The base URL for REST services in TDV always includes the following derived components.

Protocol	Hostname	port	dataformat	WebContainerNameSpace
HTTP	localhost	9400	XML	ACTdata
HTTPS	localhost	9402	JSON	ACTdata

The basic required Studio tasks to set up REST

1. Create procedures in Studio for the REST client actions.
2. Publish the procedures and supporting resources to the Web service.

3. Create links between HTTP verbs within Studio and the Web service.



Each of the following verbs is associated with one or more of the procedures defined in Studio. After defining the procedures, you can publish them to the Web service and bind them to a verb. This tutorial focuses on the following procedure and verb associations.

HTTP Verb	SQLScript Procedure
GET	playdataByID
POST	insertPlaydata
DELETE	deletePlaydata

4. Map the REST client application to the URL from the Studio Web service.

Creating Studio Resources for Use in this Tutorial

These simple objects should take less than 30 minutes to create. (If you choose to use data that is already available on your system, you need to figure out what code and Studio resources to change for this to work.)

This section includes the following topics:

- [Creating Data Sources, page 11](#)
- [Introspecting the Data Source, page 11](#)
- [Creating the Vplaydata View, page 13](#)
- [Creating the Procedures, page 14](#)
- [Publishing the Procedures to a REST Service, page 17](#)

Creating Data Sources

For this example, you can replicate the data that ACT has by creating a simple Excel file with the following data and structure.

To create the data source used by the ACT example

1. Open Excel or a similar application.
2. Create the following rows of data, including the heading row with ID, name, and author:

ID	name	author
1	Hairspray	John Waters
2	St. Joan	George Bernard Shaw
3	King Lear	William Shakespeare
4	Skin of Our Teeth	Thornton Wilder
5	The Bald Soprano	Eugene Ionesco
6	A Streetcar Named Desire	Tennessee Williams
7	The Importance of Being Earnest	Oscar Wilde
8	Glengarry Glen Ross	David Mamet
9	The Threepenny Opera	Bertolt Brecht

3. Save the file as playdata.csv.

Introspecting the Data Source

This task creates a data source connection to the playdata.csv so that TDV users can access and manipulate the data.

To add the playdata.csv file to Studio

1. Open Studio.
2. Create a data sources folder under the Shared node.
3. Create a file-based folder under the data sources node.

4. Right-click filed based and select New Data Source.
5. Select the File Delimited data source adapter.
6. Click Next.
7. Assign the name “playdata” to the data source.

Provide the connection information for this File-Delimited data source.

Datasource Name

Name: playdata

Connection Information

Basic

☒ Local File System

Root Path: D:\source_info Browse...

☐ URL

(Eg. file:///Z:/Engineering/test.txt with disk drive name or file:///Filer/Engineering/test.txt without)

Character Set: iso-8859-1

Delimiter: ,

Text Qualifier: "

Starting Row: 1

☒ Has Header Row

☒ Ignore trailing delimiter

File Name Filter(s): *.csv,*.txt

<Back Create & Introspect Create & Close Cancel

8. Use the Browse button to navigate to the directory where you created the playdata.csv file.
9. Check the Has Header Row check box.
10. Click Create & Introspect.
11. Select the playdata.csv resource and click Next.
- In the summary of changes, make sure that playdata.csv is listed.
12. Click Finish.

13. Click OK.
14. Verify that the data source was added to your Studio tree and that you can view its contents within Studio.

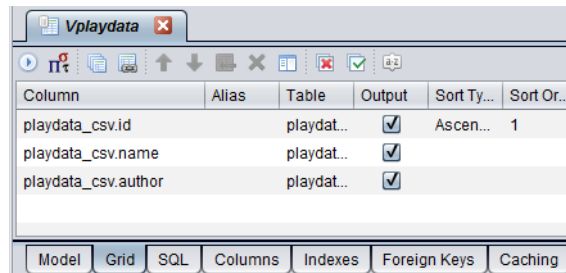
Creating the Vplaydata View

The procedures you create later will all use this view—a simple mapping from one view to one table. (The absence of transformations or joins within the exposed view means you can add to and update it.)

Note: It is typical to wrap even simple tables in a Studio view to decouple the services and clients from the physical connection.

To create the Vplaydata view

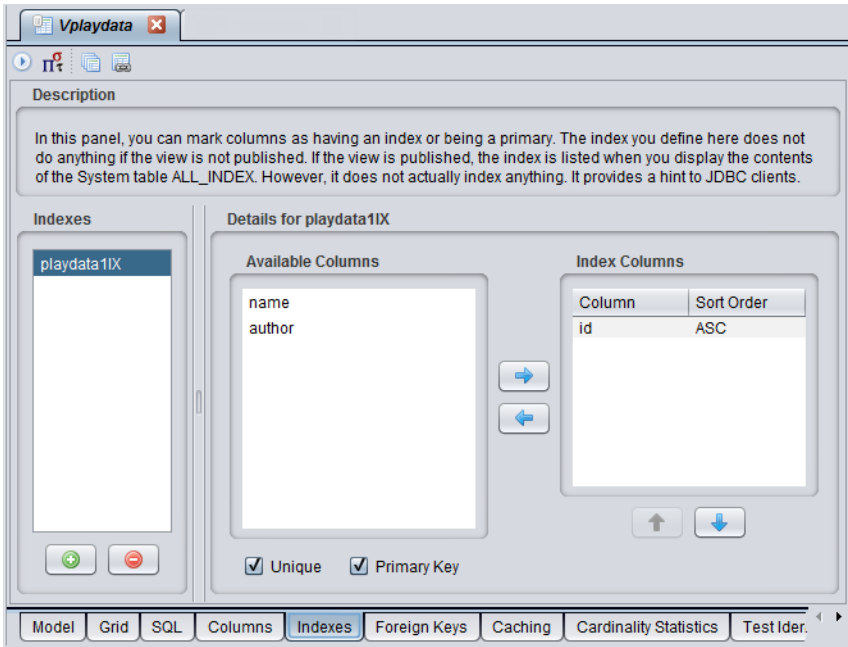
1. From the Studio resource tree, create the **playdata** folder under the Shared folder.
2. Right-click the playdata folder and select New View.
3. Give the view the name “Vplaydata” and click OK.
4. In the View editor for Vplaydata, drag the playdata.csv object onto the Model tab.
5. Select the Grid tab and set the following values.



6. Validate that the SQL tab contains the following.

```
SELECT
    playdata_csv.id,
    playdata_csv.name,
    playdata_csv.author
FROM
    /shared/datasources/filebased/playdata/"playdata.csv"
playdata_csv
```

7. Select the Indexes tab and set the values to match the following.



8. Save the definitions.

Creating the Procedures

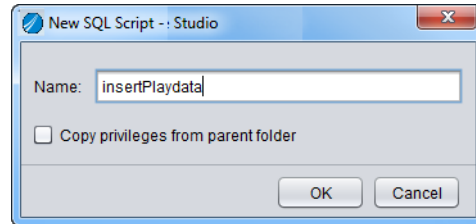
Now you create procedures within TDV to search, add to, and delete the play data.

Procedure	Used To
playdataByID	GET data
insertPlaydata	PUT data
deletePlaydata	DELETE data

Create the procedures needed to GET, PUT, and DELETE data

- 1. Open Studio.
- 2. For each of the three scripts, right-click to select New SQL Script from the Studio tree.

3. Name the scripts as follows:
 - playdataByID
 - insertPlaydata
 - deletePlaydata



4. Copy the appropriate script contents into the editor.

For the script

Copy the following into the editor

deletePlaydata

```
PROCEDURE deletePlaydata(
  IN pk VARCHAR(250),
  OUT response VARCHAR(250)
)
BEGIN
  INDEPENDENT TRANSACTION
    delete from /shared/playdata/Vplaydata
    where ID = pk;
    set response='row ' || pk || ' deleted';
  COMMIT;
END
```

For the script	Copy the following into the editor
----------------	------------------------------------

insertPlaydata	<pre> PROCEDURE insertPlaydata(IN id_val VARCHAR(250), IN name_val VARCHAR(250), IN author_val VARCHAR(250), OUT response VARCHAR(250)) BEGIN INDEPENDENT TRANSACTION DECLARE r ROW (ID VARCHAR(250)); DECLARE c CURSOR(ID VARCHAR(250)); set response='row ' id_val ' already EXISTS!'; OPEN c FOR SELECT cs.ID FROM /shared/playdata/Vplaydata cs WHERE cs.ID=id_val; FETCH c INTO r; IF NOT c.FOUND THEN INSERT INTO /shared/playdata/Vplaydata (ID,name,author) VALUES (id_val, name_val,author_val); set response='row ' id_val ' inserted'; END IF; COMMIT; END </pre>
----------------	---

For the script	Copy the following into the editor
----------------	------------------------------------

playdataByID	<pre> PROCEDURE playdataByID(IN id_arg INTEGER, OUT result CURSOR (ID INTEGER, name VARCHAR(32768), author VARCHAR(32768))) BEGIN DECLARE isnull varchar(12); set isnull=nvl2(id_arg,'' id_arg,'true'); IF isnull <> 'true' THEN OPEN result FOR SELECT id_arg ID, Vplaydata.name, Vplaydata.author FROM /shared/playdata/Vplaydata Vplaydata WHERE id_arg = Vplaydata.ID; ELSEIF isnull='true' THEN OPEN result FOR SELECT CAST(Vplaydata.ID as INTEGER) ID, Vplaydata.name, Vplaydata.author FROM /shared/playdata/Vplaydata Vplaydata; END IF; END </pre>
--------------	--

-
5. Save each procedure.

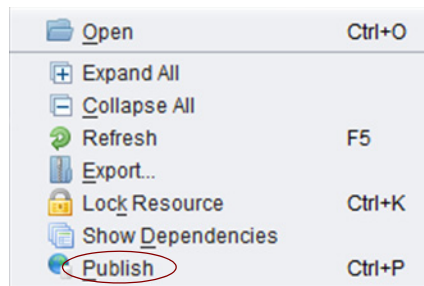
Publishing the Procedures to a REST Service

You publish TDV views and procedures as Web or database services to make them available to consuming applications—in this case, a REST client application.

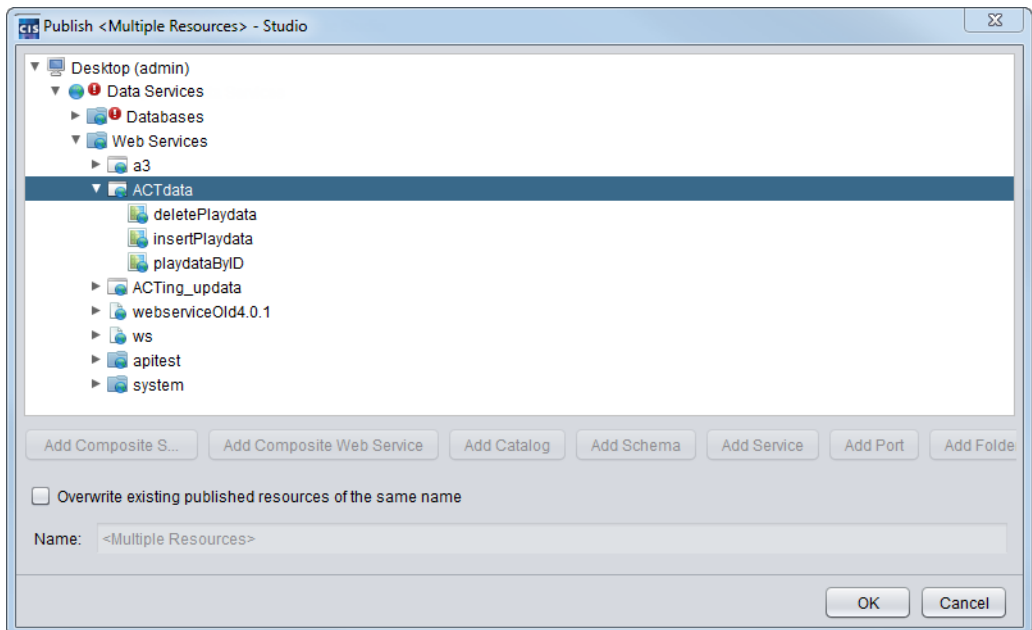
To publish the procedures to a REST service

1. Select the Web Services node under Data Services in the Studio resource tree.
2. Right-click to select New Composite Web Service.
3. For Data Service Name, enter a name for your Web service. For this example we use ACTdata.

4. Click OK.
5. Publish the following scripts to the Web service:
 - playdataById
 - insertPlaydata
 - deletePlaydata
 - a. Select all three of the scripts from the resource tree, and right-click.
 - b. Select Publish.



-
-
-
- c. Select the Web service to which you want the resource to be published.



-
-
-
-
- d. Click OK.

Linking and Mapping Procedures to Specific REST Service URLs

This section explains how to link each procedure to a specific HTTP verb and then how to map each URL for the procedure to your REST service client application.

HTTP defines verbs (or methods) to specify an action to perform on the data. You can link the logic that you have defined using other procedures within TDV to a specific HTTP verb. These verbs are trusted actions that HTTP allows to pass over the network.

SQL Script Procedure	HTTP Verb	Description of Action
playdataByID	GET	Requests the specified data. GET retrieves data and does nothing else.
insertPlaydata	POST	Submits data to be processed. The data is included in the body of the request.
deletePlaydata	DELETE	Deletes the specified data.

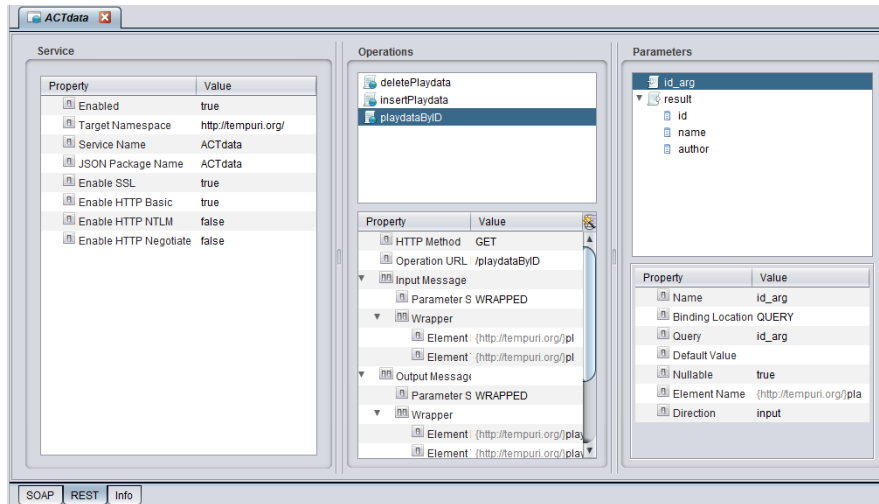
- This section includes the following topics:
- [Linking the playdataByID Procedure to an HTTP Verb, page 19](#)
 - [Mapping the Studio REST playdataByID URLs to Specific REST Services, page 21](#)
 - [Linking the insertPlaydata Procedure to an HTTP Verb, page 23](#)
 - [Mapping the Studio REST insertPlaydata URLs to Specific REST Services, page 25](#)
 - [Linking the deletePlaydata Procedures to an HTTP Verb, page 27](#)
 - [Mapping the Studio REST deletePlaydata URLs to Specific REST Services, page 28](#)

Linking the playdataByID Procedure to an HTTP Verb

In this task you link the playdataByID to the GET verb.

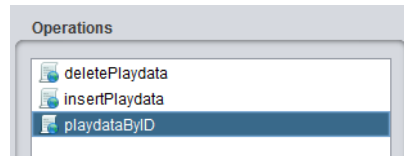
To link the playdataByID procedure to the GET verb

1. Open the ACTdata Web service.
2. Select the REST tab.

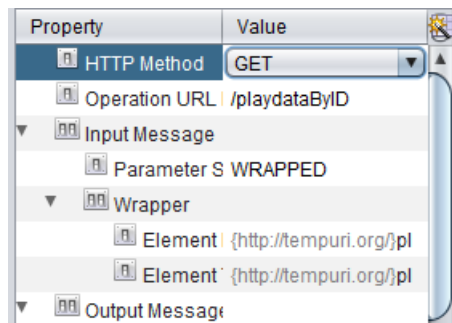


3. From the upper part of the Operations section, select the playdataByID procedure.

Note: You might need to close and reopen this window for it to appear in the list.

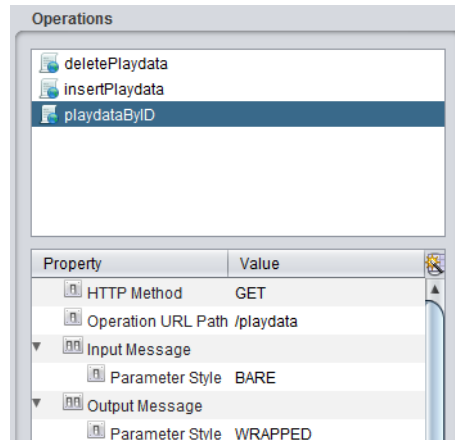


4. In the lower part of the Operations section, click in the Value field next to HTTP Method, and select the GET verb.



5. Edit the Operation URL Path so that it reads /playdata.

6. Change the input parameter style to BARE.



7. Save your changes.

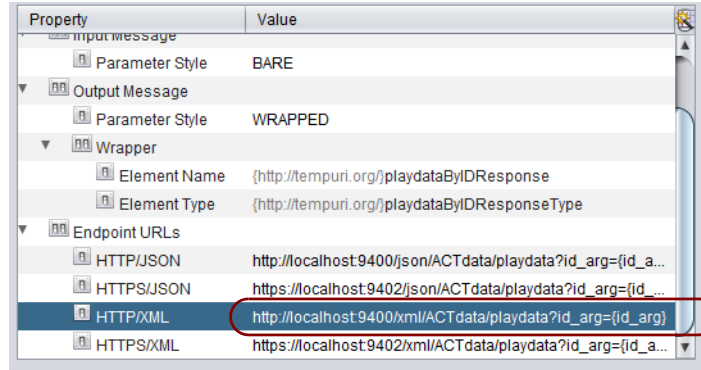
Mapping the Studio REST playdataByID URLs to Specific REST Services

Map a specific path for each REST service, paying particular attention to what is passed into each one and how. The following steps illustrate one method for connecting your client application with the REST service that you have created using Studio.

To map playdataByID

1. Open the ACTdata Web service.
2. Select the REST tab.
3. In the Operations portion of the screen, select the playdataByID procedure.

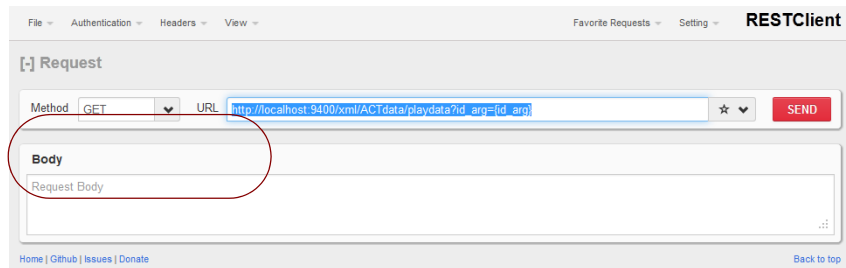
4. In the lower portion of the Operations area, double-click to select the value of the HTTP/XML endpoint.



5. Copy that endpoint URL into your REST client application development tool.
For example using the Firefox RESTClient, you can test the playdataById operation.
 - a. Make sure that the Method is set to GET.
 - b. Replace "{id_arg}" with 3.

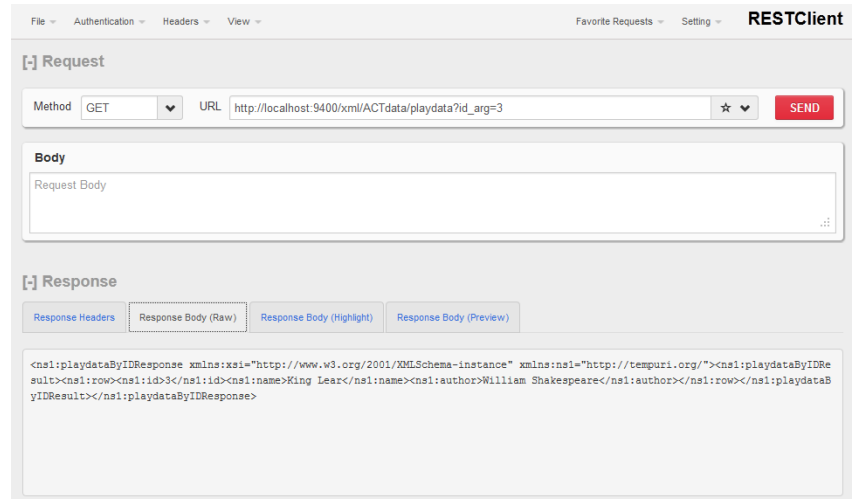
For an even cleaner URL path, try replacing “id_arg={id_arg}” with 3.

c. Click SEND.



d. Type the Studio username and password for the user with access to the REST service.

e. Select the Response Body (Raw) tab.



6. Finish development of your REST client.

The exact steps that need to be done and the order in which they need to be done depends on what tool you have chosen.

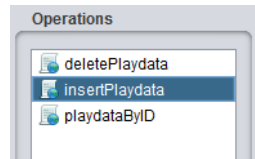
Linking the insertPlaydata Procedure to an HTTP Verb

This task links the insertPlaydata procedure to the POST HTTP verb. POST submits data to process and includes data in the body of the request.

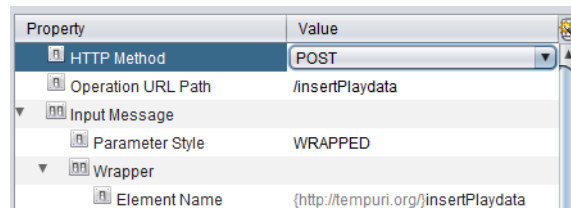
To link the insertPlaydata procedures to the POST verb

1. Open the ACTdata Web service.
2. Select the REST tab.
3. From the upper part of the Operations section, select the insertPlaydata procedure.

Note: You might need to close and reopen this window for it to appear in the list.

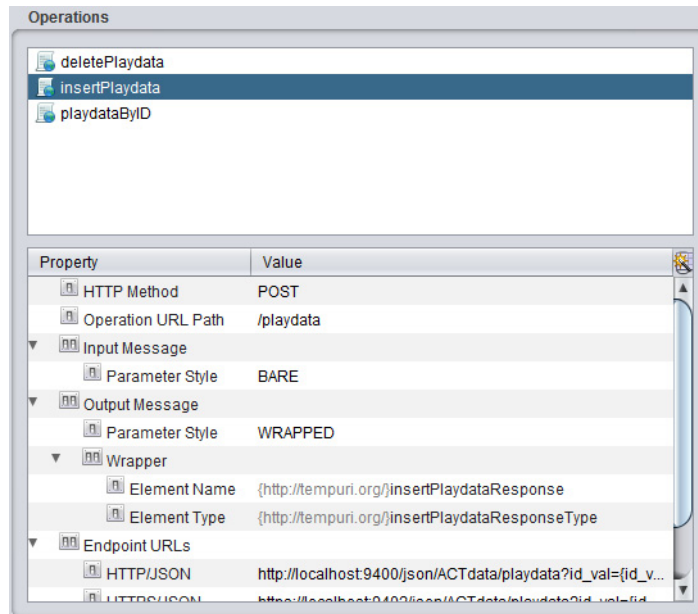


4. In the lower part of the Operations section, select the verb associated with the procedure.



5. Edit the Operation URL so that it reads /playdata.

6. Change the input parameter style to BARE.



7. Save your changes.

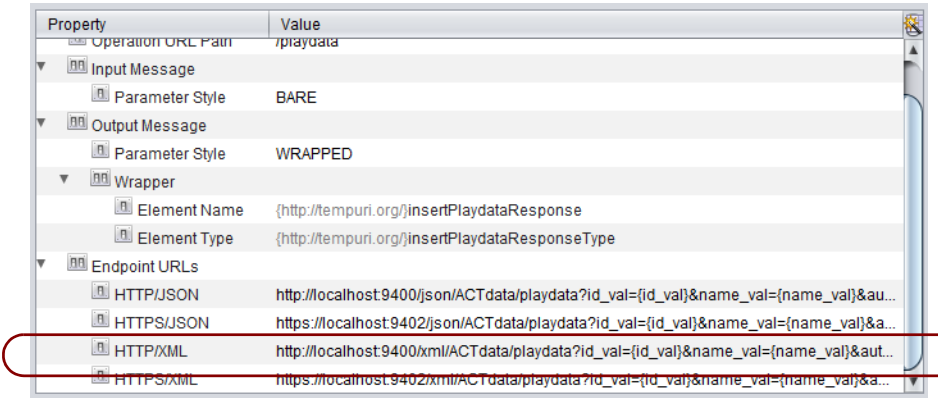
Mapping the Studio REST insertPlaydata URLs to Specific REST Services

Map a specific path for each REST service, paying special attention to what is passed into each one and how. For simplicity, this portion of the tutorial uses a Firefox browser plug-in to test that data can be updated using insertPlaydata.

To map insertPlaydata

1. Open the Studio Web service editor.
2. Select the REST tab.
3. In the Operations portion of the screen, select the insertPlaydata procedure.

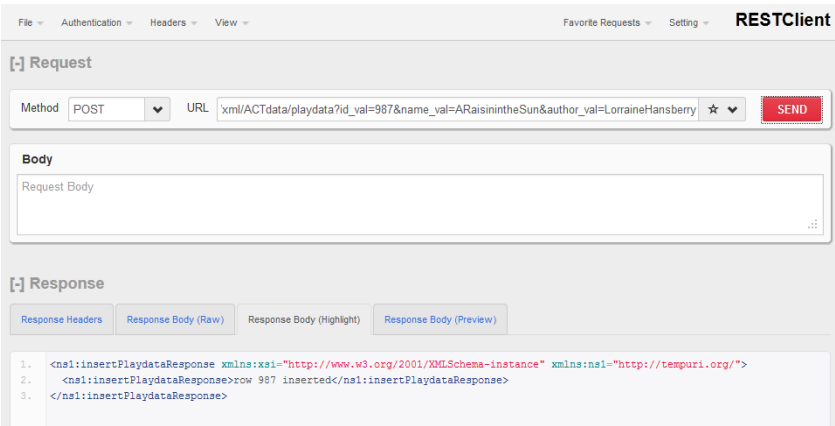
4. In the lower portion of the Operations area, double click to select the value of the HTTP/XML endpoint.



5. Copy that endpoint URL into your REST client application development tool.

For testing purposes, you can try the ACTdata Web service insertPlaydata operation using a plug-in from your browser.

- a. Paste the REST URL string into the browser RESTClient.
- b. Make sure that the Method is set to POST.
- c. Locate parameter values that are noted by curly brackets. This example replaces the values and submits the string as follows:
`http://localhost:9400/xml/ACTdata/playdata?id_val=987&name_val=ARaisinintheSun&author_val=LorraineHansberry`
- d. Click SEND.
- e. Type the Studio user name and password for the user who has access to the REST service.



6. Finish development of your REST client.

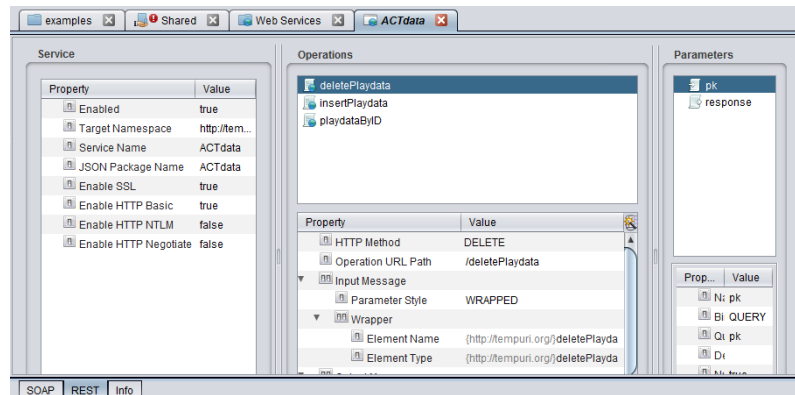
The exact steps depend on what tool you have chosen.

Linking the deletePlaydata Procedures to an HTTP Verb

This task describes how to link the deletePlaydata procedure to the DELETE HTTP verb to control the removal of data.

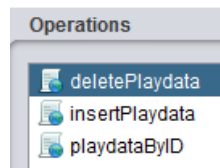
To link the deletePlaydata procedures to the DELETE verb

1. Open the ACTdata Web service to which you published your PLAYDATA procedures.
2. Select the REST tab.

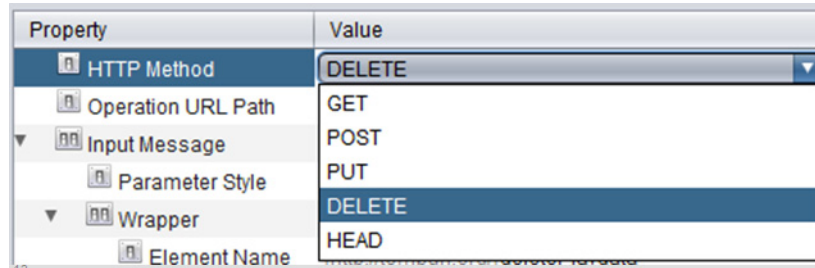


3. From the upper part of the Operations section, select the deletePlaydata procedure.

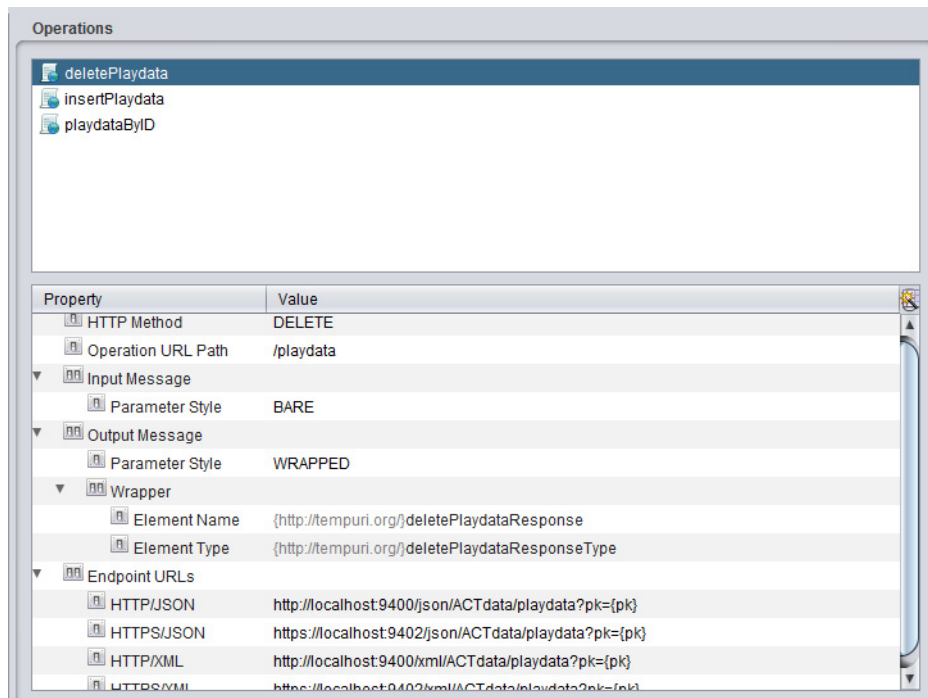
Note: You might need to close and reopen this window for it to appear in the list.



- In the lower part of the Operations section, select the DELETE verb.



- Edit the Operation URL so that it reads /playdata.
- Change the input parameter style to BARE.



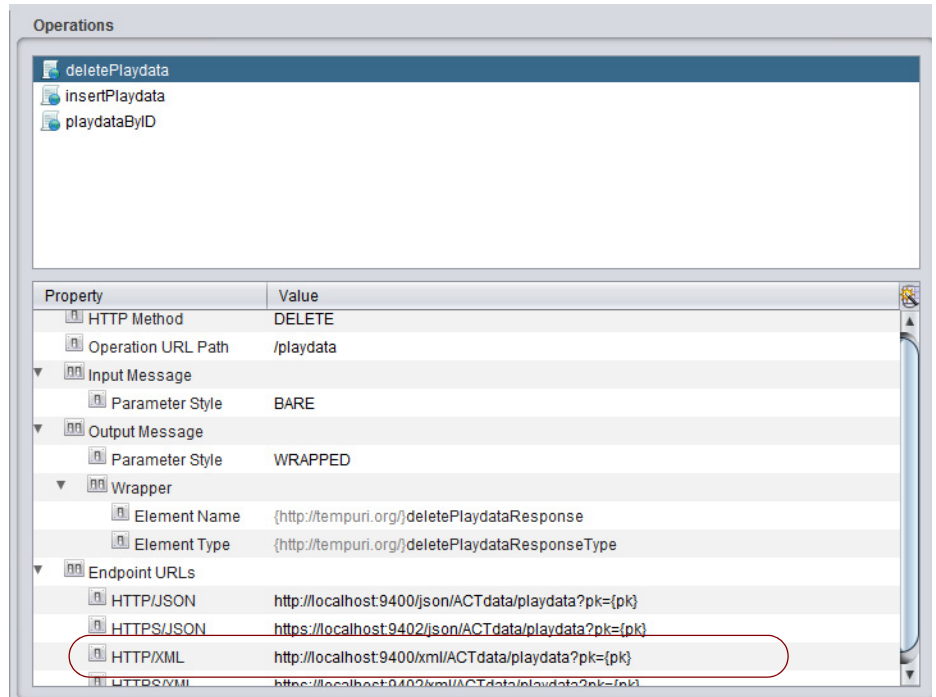
- Save your changes.

Mapping the Studio REST deletePlaydata URLs to Specific REST Services

Map a specific path for each REST service, being careful about what is passed into each one and how. The following steps show you one possible way to interface your client application with the REST service that you have created.

To map deletePlaydata

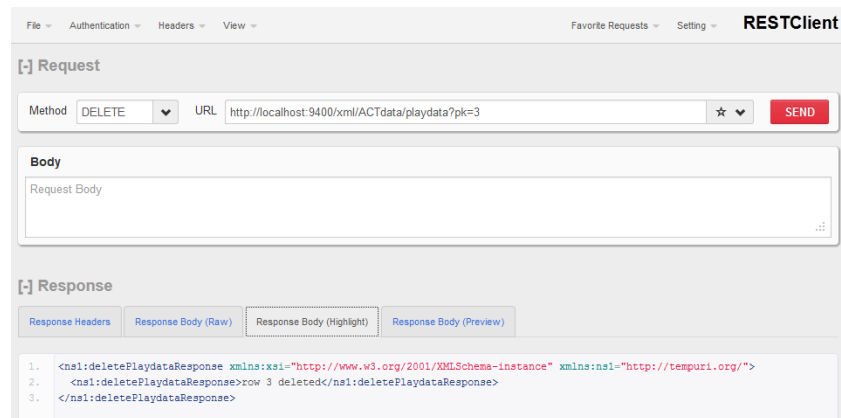
1. Open the Studio Web service editor.
2. Select the REST tab.
3. In the Operations portion of the screen, select the deletePlaydata procedure.
4. In the lower portion of the Operations area, double click to select the value of the HTTP/XML endpoint.



5. Copy that endpoint URL into your REST client application development tool.

For testing purposes, you can try the ACTdata Web service insertPlaydata operation using a plug-in from your browser.

- Paste the REST URL string into the browser RESTClient.
- Make sure that the Method is set to DELETE.
- Replace {pk} with 3.
- Click SEND.
- Type the Studio user name and password for the user with access to the REST service.



6. Finish development of your REST client.

The exact steps depend on the tool you have chosen.

Adding a Custom Parameter in an HTTP Header for a REST Request to a Published Method

For better security, you can add a custom parameter in an HTTP header for a REST request. This requires the ability to read the header of an incoming REST request in a TDV script. To read the parameter, you can wrap table or view in a script, or use a hook to read the values.

The following example uses a procedure to return the header property. It uses the TDV GetClientProperty API to get HTTP header properties per request from client.

To add the ability to read HTTP header fields

1. Create a procedure to return the header field. For example:

```
PROCEDURE readHeader(in param date, out OrderID integer, out
OrderDate TIMESTAMP, out contentType VARCHAR(255))
BEGIN
    CALL /lib/util/GetClientProperty('Contenttype123',
contentType);
    SELECT
        Orders.OrderID,
        Orders.OrderDate
    INTO OrderID, OrderDate
    FROM /shared/HttpHeader/ds_orders/tutorial/orders
    Orders
    WHERE Orders.OrderDate = param;
END
```

2. Publish the procedure to the Web Service.
3. Create a REST data source with HTTP/JSON Endpoint URLs. For example, the value of the Base URL is:
`http://DVBU-VDI-008:9400/json/HttpHeader004/readHeader?`
4. Scroll down the Basic tab to the Operations section.
5. Add a GET operation to the REST resource.
6. Add Header/Body Parameters. The Param Name must be same as the HTTP Header name, which is the first parameter of GetClientProperty. HTTP header names are case-insensitive.
7. Type Contenttype123 for the Parameter name.
8. Specify HTTP Header for the Location.
9. Save the data source.
10. Open the operation.
11. Execute the GET operation.
12. Input the parameters.
13. Review the results.
14. Save and close.
15. Use your client applications to access the REST service get operation that you have defined. Verify that the results are what you expect.

Creating a Basic Transform Tutorial

This topic steps you through the creation of a simple transformation that uses the sample data that is shipped with TDV. The steps to create and use this transform include:

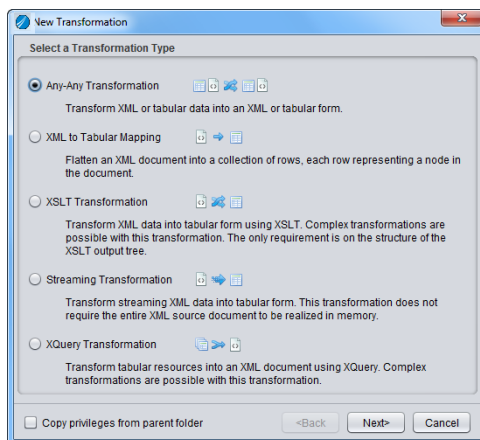
- [Creating the Any-Any Transform, page 33](#)
- [Adding a Query to the supplyChainV Transform, page 34](#)
- [Linking Outputs for Your Transform, page 36](#)
- [Executing and Testing Your Transform, page 37](#)

Creating the Any-Any Transform

Transforms, like procedures, are resources that once saved are available from the Studio resource tree.

To create the Any-Any transformation

1. Select a folder in the Studio resource tree.
2. Right-click and select **New Transformation**, or select File > New > Transformation.



3. Select **Any-Any Transformation**.
4. Click **Next**.

5. Type `supplyChainV`.
6. Click **Finish**.

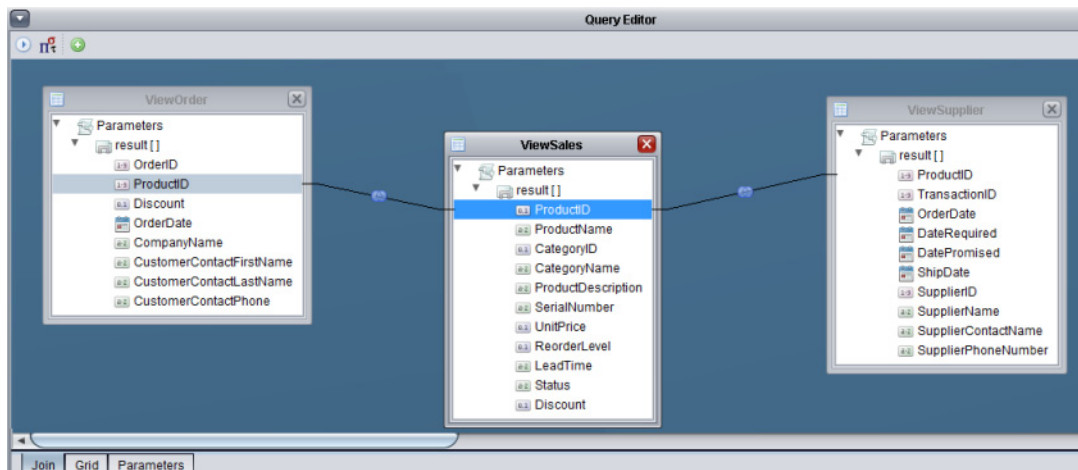
When you click **Finish**, the transformation is added to the resource tree, and the Transformation editor opens in the right pane.

Adding a Query to the supplyChainV Transform

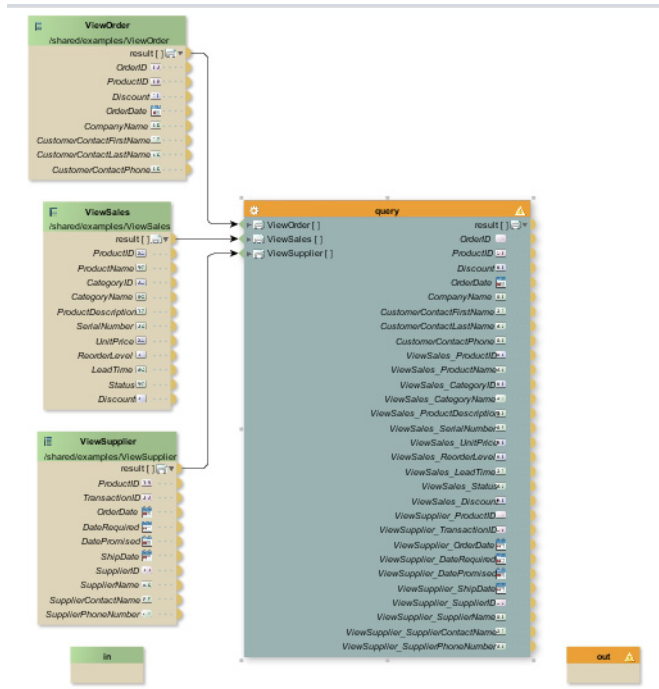
The query recreates the CompositeView resource, and is an easy way to explore how to add queries to your transform.

To create the supplyChainV transformation

1. Click and drag Query from the Transformation Editor palette.
2. Double-click the query operation that was added to the Model tab
3. From the Studio resource tree, expand Shared > Examples.
4. Click and drag the following views onto the Query Editor:
 - ViewOrder
 - ViewSales
 - ViewSupplier
5. Locate the ProductID column in each of the views.
6. Select and connect each of the ProductID columns. For example, your Query Editor should look like the following:



7. Close the Query Editor. Your changes are saved.
8. Your transform Model tab should now look like the following:

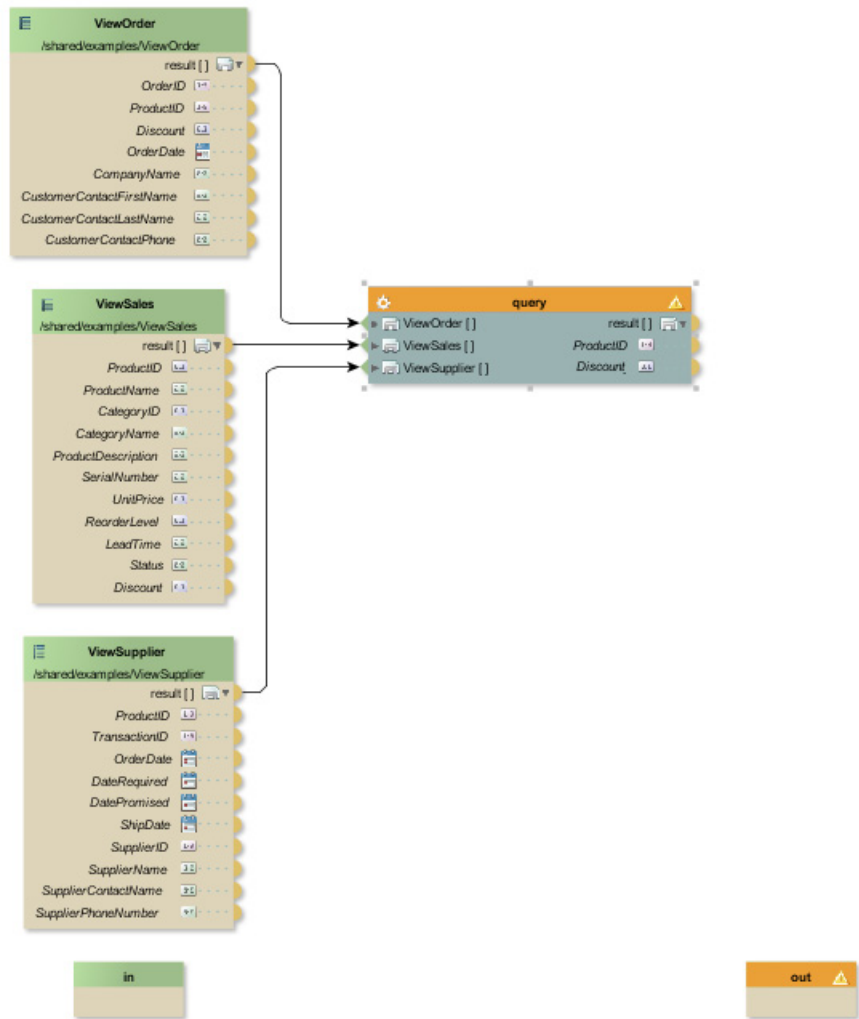


9. Save your transform.
10. Right-click the query operation that was added to the Model tab and select Edit.
11. Select the Grid tab.
12. Manipulate the fields so that they look like the following:

Query Editor						
Field	Alias	Output	Sort Type	Sort Order	Group By	Criteria
ViewOrder/OrderID	OrderID	<input type="checkbox"/>	UNSORTED		WHERE	
ViewOrder/ProductID	ProductID	<input checked="" type="checkbox"/>	UNSORTED		WHERE	
ViewOrder/Discount	Discount	<input checked="" type="checkbox"/>	UNSORTED		WHERE	
ViewOrder/OrderDate	OrderDate	<input type="checkbox"/>	UNSORTED		WHERE	

- a. Select the Output check for ProductID and Discount.
- b. Clear the Output check for all other fields.
- c. Close the Query Editor.

13. Validate that your query displays the ProductID and Discount outputs. For example, your transform model should now look like the following:




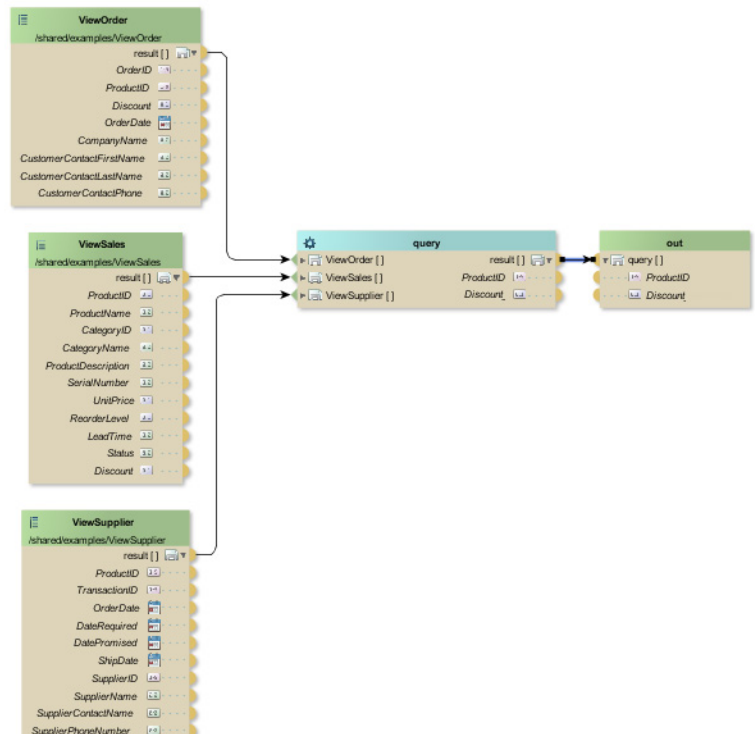
14. Save your transform.

Linking Outputs for Your Transform

The outputs of the Query operation need to be linked to the out for the transform.

To link the outputs from your transform

1. Make sure that Assign Link Mode () is selected.
2. Select the operation handle next to the result output on the query of your transform.
3. Click and drag the link to out.
4. Click Refresh Layout.
5. Validate that your Model tab now looks like the following:



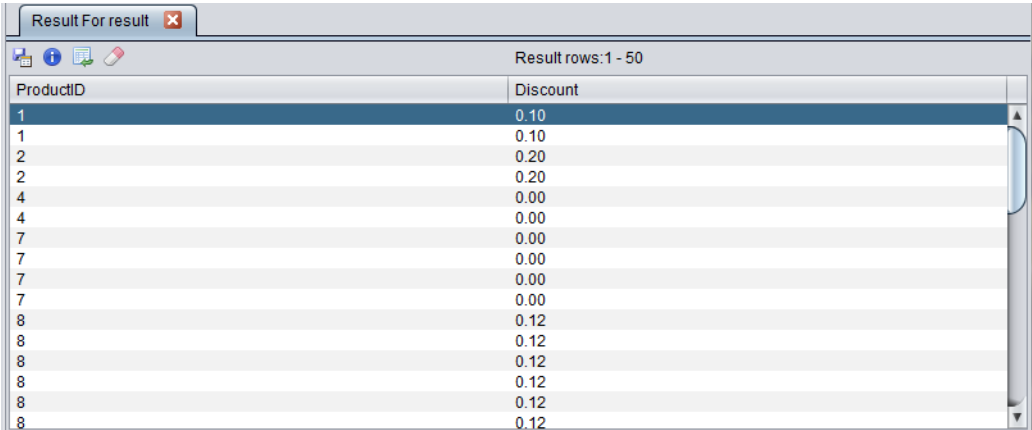
6. Save your transform.

Executing and Testing Your Transform

After designing your sample transform it is valuable to execute it to see what data it retrieves.

To execute and test the results of your transform

- 1. Select Execute.
- 2. The right-side of Studio splits and the Result panel opens.
- 3. Validate that your results look similar to the following:



ProductID	Discount
1	0.10
1	0.10
2	0.20
2	0.20
4	0.00
4	0.00
7	0.00
7	0.00
7	0.00
7	0.00
8	0.12
8	0.12
8	0.12
8	0.12
8	0.12
8	0.12

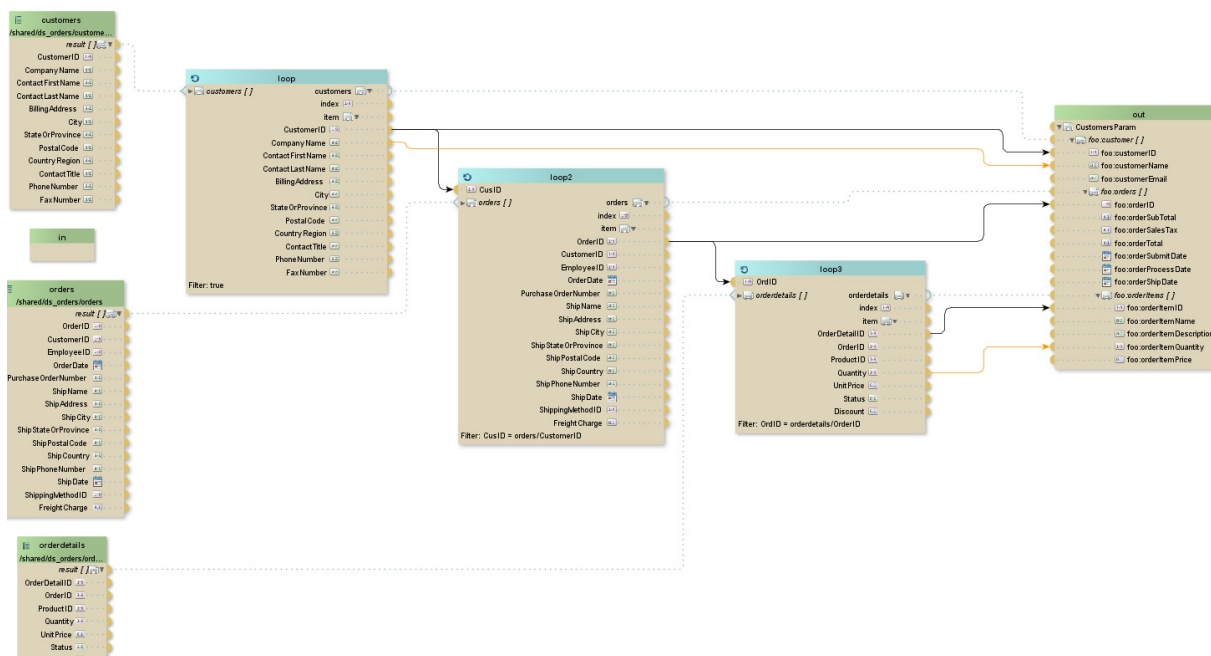
- 4. Save and close your transform.

Transform Table and XML Data Tutorials

This topic steps you through the creation of a simple transformation that uses the sample data that is shipped with TDV. The steps to create and use this transform include:

- [Create an XML Definition Set for the Tutorial, page 39](#)
- [Converting Relational to Hierarchical Data Tutorial, page 41](#)
- [Converting XML to a View, page 48](#)

The Transformation Editor model can accept an XML or WSDL source as input, and can generate a tabular mapping to the elements in the source schema. The goal for this tutorial is a transform that looks similar to the following:

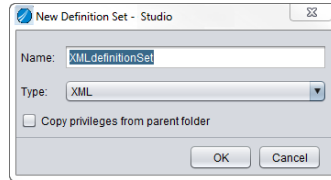


Create an XML Definition Set for the Tutorial

There is one Studio resource that you must create before starting this tutorial. It is a simple XML definition set. This procedure provides the code for you to copy and paste into the Studio editor to create the resource.

To create the XML definition set

1. Open Studio.
2. Select Shared on the Studio resource tree.
3. Right-click and select New Definition Set.
4. Type XMLdefinitionSet.
5. Select XML as the Type.



6. Click OK.

7. Copy and paste the following XML into the definition set editor:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ns1="foo" targetNamespace="foo"
attributeFormDefault="unqualified" elementFormDefault="qualified">
  <xs:element name="Customers" type="ns1:Customers"/>
  <xs:complexType name="Customer">
    <xs:sequence>
      <xs:element name="customerID" type="xs:integer"/>
      <xs:element name="customerName" type="xs:string"/>
      <xs:element name="customerEmail" type="xs:string"/>
      <xs:element minOccurs="0" maxOccurs="unbounded" name="orders"
type="ns1:Order"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="Order">
    <xs:sequence>
      <xs:element name="orderID" type="xs:integer"/>
      <xs:element name="orderSubTotal" type="xs:float"/>
      <xs:element name="orderSalesTax" type="xs:float"/>
      <xs:element name="orderTotal" type="xs:float"/>
      <xs:element name="orderSubmitDate" type="xs:date"/>
      <xs:element name="orderProcessDate" type="xs:date"/>
      <xs:element name="orderShipDate" type="xs:date"/>
      <xs:element minOccurs="0" maxOccurs="unbounded"
name="orderItems" type="ns1:OrderItem"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="OrderItem">
    <xs:sequence>
      <xs:element name="orderItemID" type="xs:integer"/>
      <xs:element name="orderItemName" type="xs:string"/>
      <xs:element name="orderItemDescription" type="xs:string"/>
      <xs:element name="orderItemQuantity" type="xs:int"/>
      <xs:element name="orderItemPrice" type="xs:float"/>
    </xs:sequence>
  </xs:complexType>
```



```

<xs:complexType name="Customers">
  <xs:sequence>
    <xs:element minOccurs="0" maxOccurs="unbounded"
name="customer" type="ns1:Customer"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>

```

8. Save the definition set.

Converting Relational to Hierarchical Data Tutorial

Relational data is typically saved as tables or views in a relational database. Hierarchical data is typically saved as XML data, and sometimes as tables or views in a hierarchical database such as IMS.

This tutorial uses three of the tables shipped with TDV from the Examples project (customers, orders, and orderdetails) to produce an XML document result.

- [Adding the Views to Convert, page 41](#)
- [Adding the Loops and Filters Used to Iterate and Select Data, page 42](#)
- [Completing the Links Between Loop and out, page 44](#)
- [Adding Loop2 to the Model, page 46](#)
- [Adding Loop3 to the Model, page 47](#)
- [Adding Links to Direct Data Flow, page 47](#)
- [Executing the Transform, page 48](#)

Adding the Views to Convert

Add three of the tables shipped with TDV from the Examples project (customers, orders, and orderdetails).

To convert relational to XML data

1. Open Studio.
2. In the Studio resource tree, select Shared.
3. Right-click and select New Transformation.
4. Select Any-Any Transformation and click Next.
5. Type TBL2XML in the Transformation Name field.

6. Click Finish.

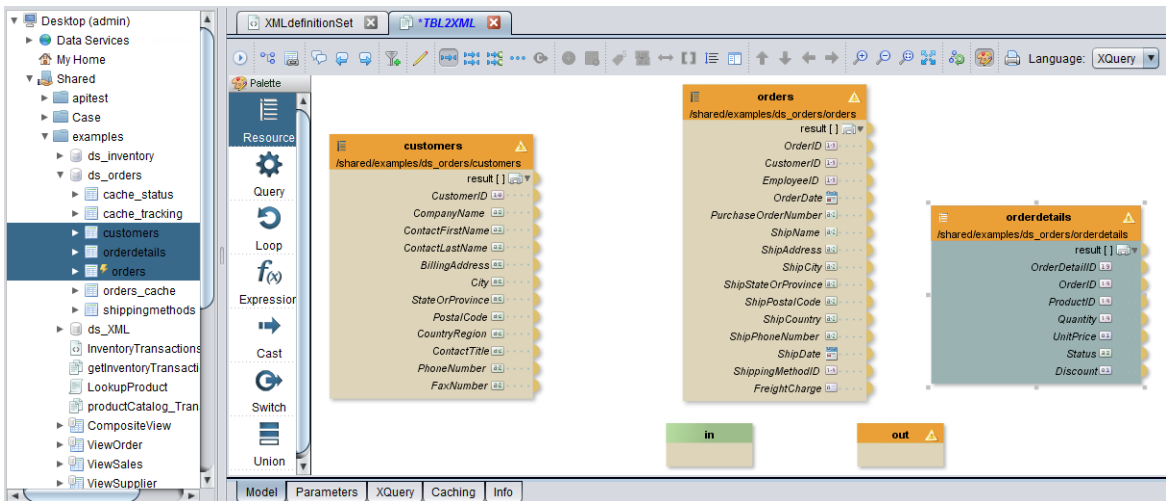
The Transformation Editor opens with the Model tab displayed.

7. From the Studio resource tree, expand Shared/examples.

8. Expand Shared/examples/ds_orders.

9. Select the following views and drag them from the resource tree onto the Transformation Editor Model tab:

- customers
- orders
- orderdetails



10. Save your transform.

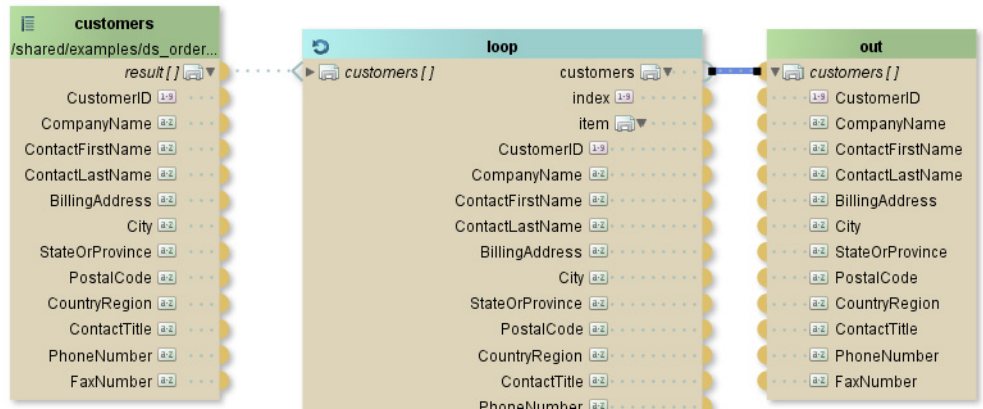
Adding the Loops and Filters Used to Iterate and Select Data

This example makes use of programmatic loops to iterate through and select the data to be transformed.

To add loops and filters

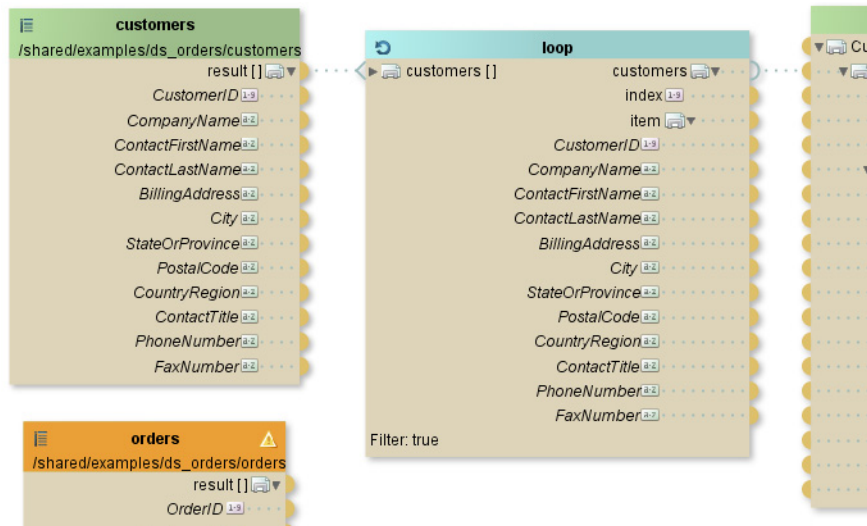
1. Select the Loop Link Mode. Alternatively hold SHIFT key while creating a link (Shift+Minus) icon at the top of the editor.
2. Select the operation handle next to the result column that is listed in customers and drag the line to out.

3. Save the transform.



4. Delete the connection between loop and out.
 5. On the out container, select the operation handle next to customers and right-click.
 6. Select Delete "customers".
 7. Click out to select it.
 8. Right-click out and select Add Parameter.
 9. Select Browse.
 10. Browse to XMLdefinitionSet.
 11. Select Type in the Show drop down list on the right.
 12. Select Customers from that list.
 13. Click OK.
- By default, the XML hierarchy is collapsed.
14. Click the arrow next to foo:customer, then foo:orders, and foo:orderitems to expand the hierarchy.
 15. Click the Assign Link Mode. Alternatively hold CTRL key while creating a link (CTRL+Minus) icon at the top of the editor.
 16. From loop, click in the operation handle next to customers and drag the line to foo:customer.

17. Save your transform.

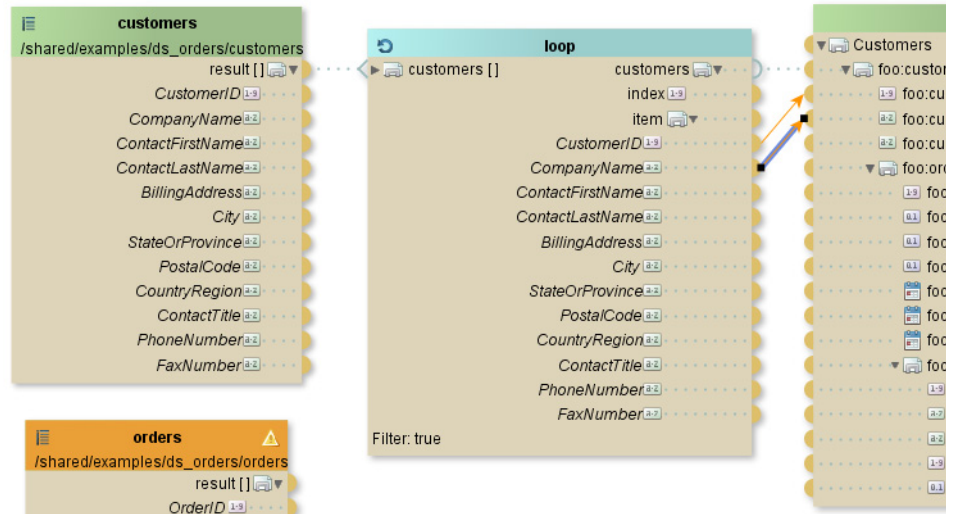


Completing the Links Between Loop and out

To complete loop links

1. Click the Assign Link Mode icon at the top of the editor.
2. Link CustomerID to foo:customerID.

3. Link CompanyName to foo:CustomerName.

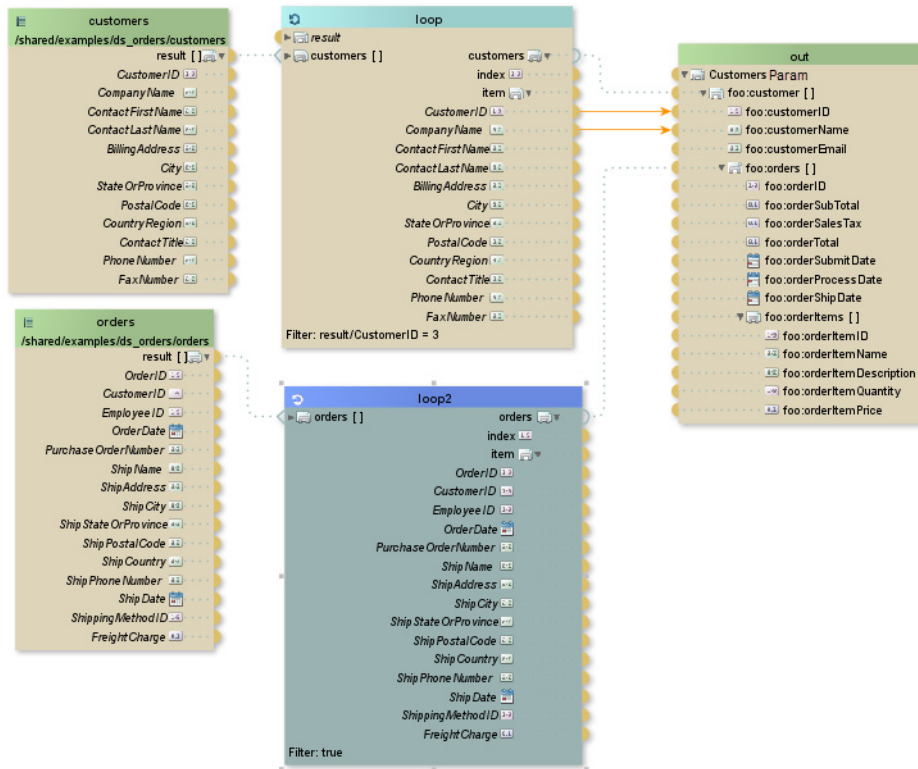


4. Save your transform.

Adding Loop2 to the Model

To add loop2

1. Select the Loop Link Mode.
2. Click in the operation handle next to the result column of the orders table and connect it to the foo:orders column of out.



3. Double-click loop2.
4. In the field, type the following:
CusID = orders/CustomerID
5. Click **Close**.
6. Save your transform.

Adding Loop3 to the Model

To add loop3

- 1. Select the Loop Link Mode.
- 2. Link orderdetails result to foo:orderItems.
- 3. Double-click loop3.
- 4. In the field, type the following:
OrdID = orderdetails/OrderID
- 5. Save your transform.

Adding Links to Direct Data Flow

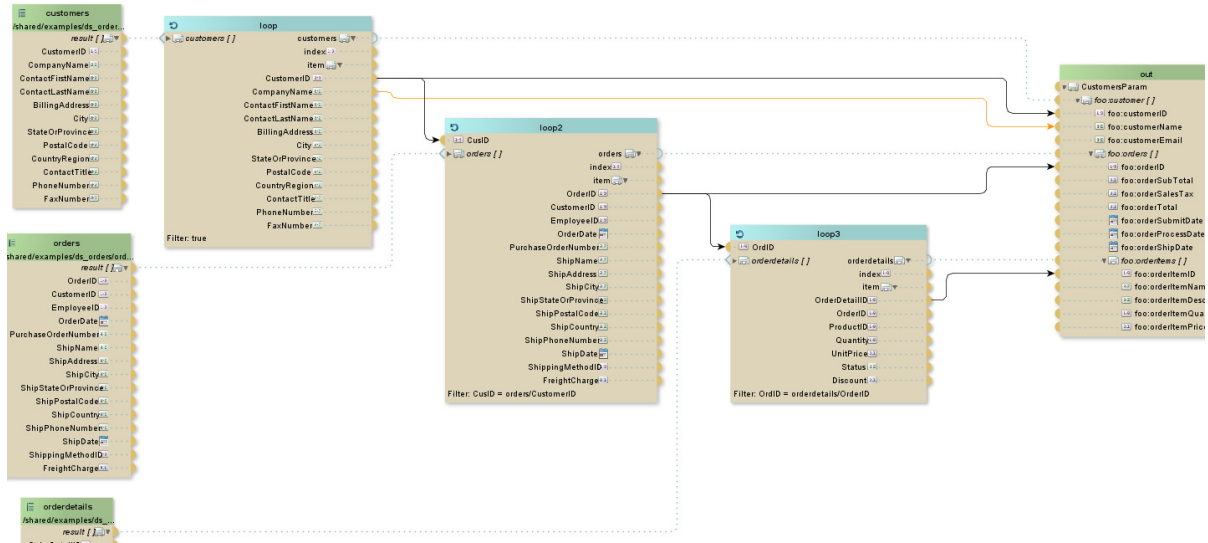
Linking data on certain columns helps direct the flow of data and help select the data for transformation.

To add links

- 1. Open your transform.
- 2. Validate and or create links between the following From and to columns.

From Operator	From Column	To Operator	To Column
loop	customers	out	out.foo.customer
	CustomerID	out	out.foo.customerID
	CompanyName	out	foo.customerName
	CustomerID	Loop2	CusID
loop2	orders	out	out.foo.orders []
	OrderID	out	out.foo.orderID
	OrderID	loop3	loop3.OrdID
loop3	orderdetails	out	foo.orderitems []
	OrderDetailID	out	foo.orderitemID

3. Save your transform.



Executing the Transform

When you execute a transform the convert relational data into XML structured hierarchical data, you can use the results to get an idea of what would be produced at runtime.

To execute the transform

1. Execute your transform to review the generated XML code. (Click the execute button.)
2. Click Details on the Results panel to view the Value window.
3. Click OK to close the Value window when you have finished.
4. Save and close your transform.

Converting XML to a View

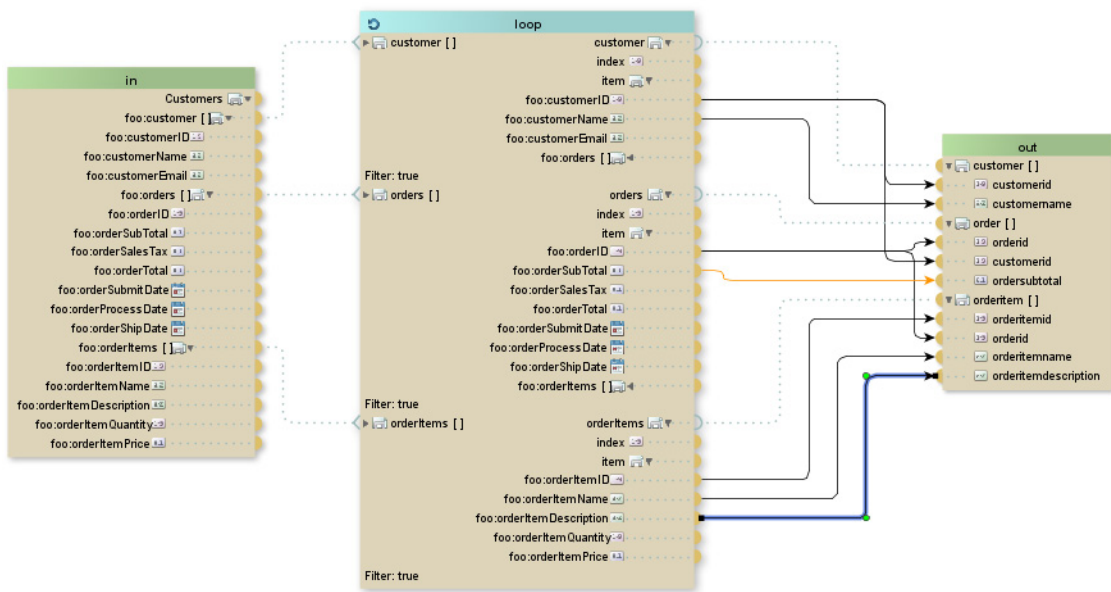
This portion of the tutorial converts the XMLdefinitionset to 3 views. Typically, this type of transformation is useful when data is generated through WSDL or other Web application based XML. The data accumulated in XML or WSDL files can be converted to tabular format and then consumed by other database analysis tools to pattern customer behavior.

Your eventual implementation might need to be more complex than the example used in this tutorial, but it should give you a solid foundation to understand how the Transformation Editor can be used to achieve your goals.

The following topics are included in this tutorial:

- [Creating the Transform and Adding the in Parameters, page 49](#)
- [Adding Parameters to out, page 50](#)
- [Adding the Loop and Iterators, page 52](#)
- [Executing the XML2view Transform, page 54](#)

The transformation model that you will create in this tutorial is shown in the following:



Creating the Transform and Adding the in Parameters

To convert XML formatted data to a relational database view

1. Open Studio.
2. Select Shared on the Studio resource tree.
3. Right-click and select New Transformation.
4. Select Any-Any Transformation.

5. Type XML2view.
6. Click OK.
7. On the Model tab of the Transformation Editor, click to select in.
8. Right-click and select Add Parameter.
9. Select Browse.
10. In the Add Definition Type dialog, navigate to and select XMLdefinitionSet.
11. Select Customers from the list on the right (under Show Element).
12. Click OK.

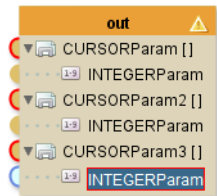
You can click the arrows in the hierarchy to expand and see all of the elements of the XML that you just added to the transform.

Adding Parameters to out

This portion of the tutorial demonstrates two different ways to add and edit parameters.

To add the necessary parameters to out

1. Open your transform.
2. Click to select out.
3. Right-click and select Add Parameter.
4. Select Complex > CURSOR.
5. In the operation handle to the left of CURSORParam, right-click and select Add Parameter.
6. Select Integer > INTEGER.
7. Repeat until you have 3 cursor parameters with integer subparameters.

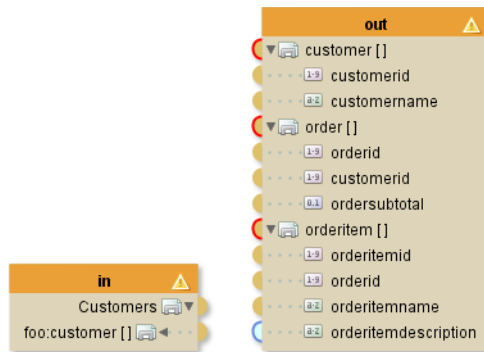


8. Save your transform.
9. Select the CURSORParam row.
10. Right-click and select Rename.

- 11. Type customer.
- 12. Select the CURSORParam2 row.
- 13. Right-click and select Rename.
- 14. Type order.
- 15. Select the CURSORParam3 row.
- 16. Right-click and select Rename.
- 17. Type orderitem.
- 18. Add and rename subparameters until your have parameters with the following structure:

Parent Parameter	subparameter	Data Type
customer		CURSOR
	customerid	INTEGER
	customername	VARCHAR
order		CURSOR
	orderid	INTEGER
	customerid	INTEGER
	ordersubtotal	FLOAT
orderitem		CURSOR
	orderitemid	INTEGER
	orderid	INTEGER
	orderitemname	VARCHAR
	orderitemdescription	VARCHAR

19. Save your transform.

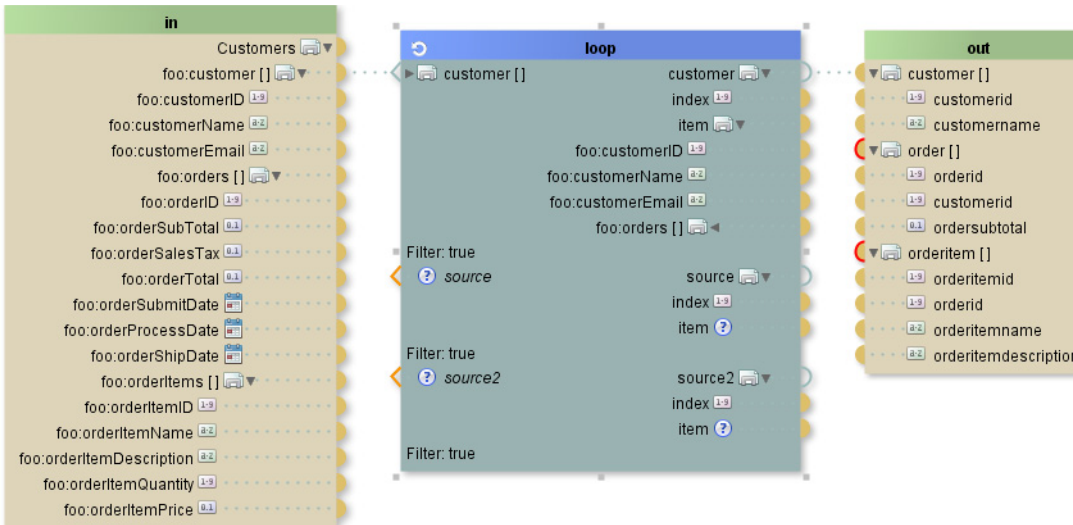


Adding the Loop and Iterators

To add the necessary loop and iterators for the tutorial

1. Select the Loop Link Mode.
2. Click in the operation handle next to the foo:customer column of in and connect it to customer [] .
3. Save your transform.
4. Double-click loop.
5. Click the green plus icon to add a Loop Iterator.
6. Click the green plus icon one more time.
7. Click Close.

8. Save your transform.



9. Validate and or create links between the following From and To columns.

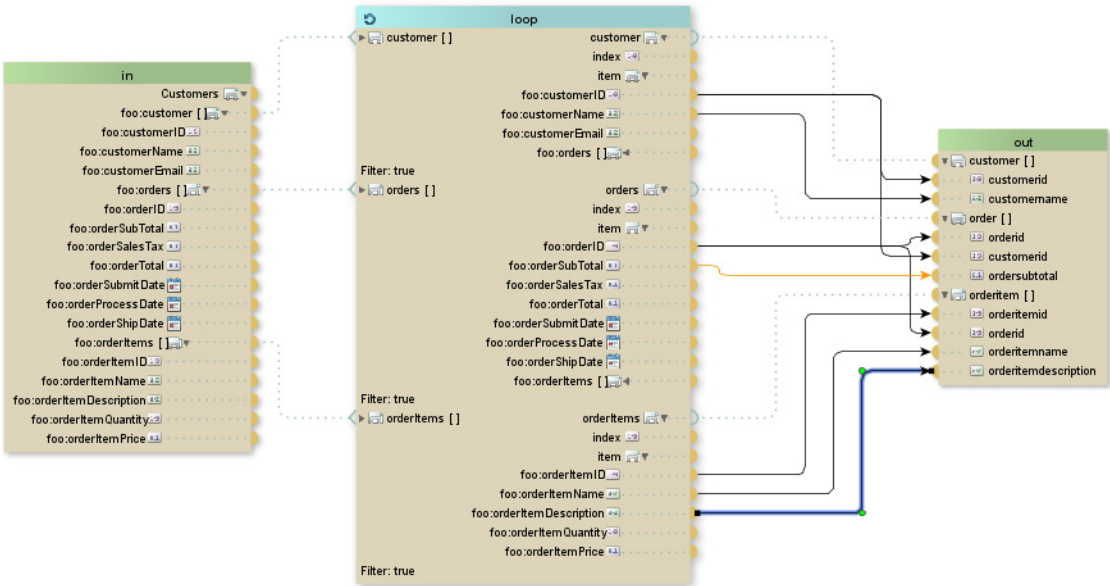
From in	To loop
foo.Customer	foo.customer
foo.orders	source
foo.orderitems	source2

10. Validate and or create links between the following From and To columns.

From loop	To out
customer	customer
foo.customerID	customer : customerid
foo.customerID	order : customerid
foo:customerName	customer : customername
orders	order
foo:orderID	order : orderid

From loop	To out
foo:orderId	orderitem : orderid
foo.orderSubTotal	order : ordersubtotal
orderitems	orderitem
foo:orderItemID	orderitem : orderitemid
foo:orderItemName	orderitem : orderitemname
foo:orderItemDescription	orderitem : orderitemdescription

11. Save your transform.



Executing the XML2view Transform

For this portion of the tutorial, you will be provided with the XML data to copy and paste into the parameter input window so that you can validate how the transform will perform when given data.

To execute the XML2TBL transform

1. Open your transform and click Execute.
2. Copy the following XML into the Input Values for XML2view window:

```
<?xml version="1.0" encoding="utf-8"?>
<CustomersParam xmlns:foo="foo">
  <foo:customer>
    <foo:customerID>1</foo:customerID>
    <foo:customerName>Able Computing</foo:customerName>
    <foo:customerEmail>compositesw.com</foo:customerEmail>
    <foo:orders>
      <foo:orderID>2</foo:orderID>
      <foo:orderSubTotal>232.6</foo:orderSubTotal>
      <foo:orderSalesTax>99.9</foo:orderSalesTax>
      <foo:orderTotal>88.34</foo:orderTotal>
      <foo:orderSubmitDate>2013-1-6</foo:orderSubmitDate>
      <foo:orderProcessDate>2013-3-9</foo:orderProcessDate>
      <foo:orderShipDate>2013-4-5</foo:orderShipDate>
      <foo:orderItems>
        <foo:orderItemID>1009</foo:orderItemID>
        <foo:orderItemName>ldap</foo:orderItemName>
        <foo:orderItemDescription>this is a
computer.</foo:orderItemDescription>
        <foo:orderItemQuantity>much good</foo:orderItemQuantity>
        <foo:orderItemPrice>99.9</foo:orderItemPrice>
      </foo:orderItems>
    </foo:orders>
  </foo:customer>
</CustomersParam>
```

3. Paste it into the Customers field.
4. Click OK to display the Result panel.
5. Click Display to view the data.

Transform XML Schemas Tutorial

This topic steps you through the creation of a transformation that mutates an XML source to another XML hierarchy based on how the data is navigated.

For example, using the following structure of Students, Teachers and Classes, you can have:

- Students with their classes and each teacher
- Teachers with their classes and the students in each class
- Classes with their students and teachers

The steps to create and use this transform include:

- [Create the School.xml Data File for the Tutorial, page 57](#)
- [Create an XML Data Source for the Tutorial, page 63](#)
- [Create the GetStudents Transformation Procedure, page 64](#)
- [Creating the ClassStudent Transformation, page 67](#)
- [Adding and Connecting Source Operations, page 68](#)
- [Completing the ClassStudent Transformation, page 69](#)

Create the School.xml Data File for the Tutorial

The XML file contain data for the execution of the transform is required for this tutorial. This procedure provides the code for you to copy and paste into the Studio editor to create the simple XML data file.

To create the School.xml data file

1. Open your favorite text editor.
2. Copy and paste the following XML code into the text editor.

```
<?xml version="1.0" encoding="UTF-8"?>
<Classes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://compositesw.com/transform/example/school">
  <Class>
    <CourseName>BIO-101</CourseName>
    <CourseDescription>Intro to biology</CourseDescription>
    <Department>Science</Department>
    <CreditNumber>1101</CreditNumber>
    <Teacher>
```

```

        <TeacherName>Bob Darwin</TeacherName>
        <TeacherEmail>bdarwin@school.com</TeacherEmail>
    </Teacher>
    <Student><StudentName>Evelina
Ruffino</StudentName><StudentEmail>ERuffino@school.com</StudentEmail></Student>
    <Student><StudentName>Krystal
Deeter</StudentName><StudentEmail>KDeeter@school.com</StudentEmail></Student>
    <Student><StudentName>Temika
Kanter</StudentName><StudentEmail>TKanter@school.com</StudentEmail></Student>
    <Student><StudentName>Zina
Borey</StudentName><StudentEmail>ZBorey@school.com</StudentEmail></Student>
    <Student><StudentName>Eugenio
Bossard</StudentName><StudentEmail>EBossard@school.com</StudentEmail></Student>
    <Student><StudentName>Bess
Zucker</StudentName><StudentEmail>BZucker@school.com</StudentEmail></Student>
    <Student><StudentName>Gisele
Landsman</StudentName><StudentEmail>GLandsman@school.com</StudentEmail></Student>
    <Student><StudentName>Lorretta
Strawn</StudentName><StudentEmail>LStrawn@school.com</StudentEmail></Student>
    <Student><StudentName>Juanita
Hamby</StudentName><StudentEmail>JHamby@school.com</StudentEmail></Student>
    <Student><StudentName>Fredda
Nakagawa</StudentName><StudentEmail>FNakagawa@school.com</StudentEmail></Student>
    <Student><StudentName>Fletcher
Sigg</StudentName><StudentEmail>FSigg@school.com</StudentEmail></Student>
    <Student><StudentName>Latrina
Rumore</StudentName><StudentEmail>LRumore@school.com</StudentEmail></Student>
    <Student><StudentName>Neg
Wisener</StudentName><StudentEmail>NWisener@school.com</StudentEmail></Student>
    <Student><StudentName>Hwa
Speegle</StudentName><StudentEmail>HSpeegle@school.com</StudentEmail></Student>
    <Student><StudentName>Mariella
Stonerock</StudentName><StudentEmail>MStonerock@school.com</StudentEmail></Student>
    <Student><StudentName>Evan
Pagaduan</StudentName><StudentEmail>EPagaduan@school.com</StudentEmail></Student>
    <Student><StudentName>Gertrud
Millman</StudentName><StudentEmail>GMillman@school.com</StudentEmail></Student>
    <Student><StudentName>Bill
Baynard</StudentName><StudentEmail>BBaynard@school.com</StudentEmail></Student>

```

```

        <Student><StudentName>Neal
Sheilds</StudentName><StudentEmail>NSheilds@school.com</StudentEmail></Student>
        <Student><StudentName>Emery
Galles</StudentName><StudentEmail>EGalles@school.com</StudentEmail></Student>
    </Class>
    <Class>
        <CourseName>BIO-201</CourseName>
        <CourseDescription>Evolution</CourseDescription>
        <Department>Science</Department>
        <CreditNumber>1201</CreditNumber>
        <Teacher>
            <TeacherName>Bob Darwin</TeacherName>
            <TeacherEmail>bdarwin@school.com</TeacherEmail>
        </Teacher>
        <Student><StudentName>Leeann
Trimpe</StudentName><StudentEmail>LTrimpe@school.com</StudentEmail></Student>
        <Student><StudentName>Georgeann
Sones</StudentName><StudentEmail>GSones@school.com</StudentEmail></Student>
        <Student><StudentName>Chrissy
Loiselle</StudentName><StudentEmail>CLoiselle@school.com</StudentEmail></Student>
        <Student><StudentName>Natalya
Fife</StudentName><StudentEmail>NFife@school.com</StudentEmail></Student>
        <Student><StudentName>Kandi
Markwell</StudentName><StudentEmail>KMarkwell@school.com</StudentEmail></Student>
        <Student><StudentName>Tommie
Weller</StudentName><StudentEmail>TWeller@school.com</StudentEmail></Student>
        <Student><StudentName>Elin
Schimpf</StudentName><StudentEmail>ESchimpf@school.com</StudentEmail></Student>
        <Student><StudentName>Spring
Luke</StudentName><StudentEmail>SLuke@school.com</StudentEmail></Student>
        <Student><StudentName>Devona
Bertelsen</StudentName><StudentEmail>DBertelsen@school.com</StudentEmail></Student>
        <Student><StudentName>Celinda
Chaplin</StudentName><StudentEmail>CChaplin@school.com</StudentEmail></Student>
        <Student><StudentName>Annamarie
Eustice</StudentName><StudentEmail>AEustice@school.com</StudentEmail></Student>
        <Student><StudentName>Ambrose
Keating</StudentName><StudentEmail>AKeating@school.com</StudentEmail></Student>
        <Student><StudentName>Kiley
Cranor</StudentName><StudentEmail>KCranor@school.com</StudentEmail></Student>
    </Class>

```

```

        <Student><StudentName>Jani
Gibeault</StudentName><StudentEmail>JGibeault@school.com</StudentE
mail></Student>
        <Student><StudentName>Willian
Gochenour</StudentName><StudentEmail>WGochenour@school.com</Studen
tEmail></Student>
        <Student><StudentName>Jeanna
Marmolejo</StudentName><StudentEmail>JMarmolejo@school.com</Studen
tEmail></Student>
        <Student><StudentName>Norah
Revelle</StudentName><StudentEmail>NRevelle@school.com</StudentEma
il></Student>
        <Student><StudentName>Claud
Lares</StudentName><StudentEmail>CLares@school.com</StudentEmail><
/Student>
        <Student><StudentName>Lacy
Bossert</StudentName><StudentEmail>LBossert@school.com</StudentEma
il></Student>
        <Student><StudentName>Ernesto
Dangelo</StudentName><StudentEmail>EDangelo@school.com</StudentEma
il></Student>
    </Class>
    <Class>
        <CourseName>CSC-450</CourseName>
        <CourseDescription>Compilers I</CourseDescription>
        <Department>Computer Science</Department>
        <CreditNumber>2450</CreditNumber>
        <Teacher>
            <TeacherName>Chuck Babbage</TeacherName>
            <TeacherEmail>cbabbage@school.com</TeacherEmail>
        </Teacher>
        <Student><StudentName>Fredda
Nakagawa</StudentName><StudentEmail>FNakagawa@school.com</StudentE
mail></Student>
        <Student><StudentName>Fletcher
Sigg</StudentName><StudentEmail>FSigg@school.com</StudentEmail></S
tudent>
        <Student><StudentName>Latrina
Rumore</StudentName><StudentEmail>LRumore@school.com</StudentEmail
></Student>
        <Student><StudentName>Nan
Wisener</StudentName><StudentEmail>NWisener@school.com</StudentEma
il></Student>
        <Student><StudentName>Hwa
Speegle</StudentName><StudentEmail>HSpeegle@school.com</StudentEma
il></Student>
        <Student><StudentName>Mariella
Stonerock</StudentName><StudentEmail>MStonerock@school.com</Studen
tEmail></Student>
        <Student><StudentName>Evan
Pagaduan</StudentName><StudentEmail>EPagaduan@school.com</StudentE
mail></Student>
        <Student><StudentName>Gertrud
Millman</StudentName><StudentEmail>GMillman@school.com</StudentEma
il></Student>

```

```

        <Student><StudentName>Bill
Baynard</StudentName><StudentEmail>BBaynard@school.com</StudentEmail></Student>
        <Student><StudentName>Neal
Sheilds</StudentName><StudentEmail>NSheilds@school.com</StudentEmail></Student>
        <Student><StudentName>Emery
Galles</StudentName><StudentEmail>EGalles@school.com</StudentEmail></Student>
        <Student><StudentName>Ambrose
Keating</StudentName><StudentEmail>AKeating@school.com</StudentEmail></Student>
        <Student><StudentName>Kiley
Cranor</StudentName><StudentEmail>KCranor@school.com</StudentEmail></Student>
        <Student><StudentName>Jani
Gibeault</StudentName><StudentEmail>JGibeault@school.com</StudentEmail></Student>
        <Student><StudentName>Willian
Gochenour</StudentName><StudentEmail>WGochenour@school.com</StudentEmail></Student>
        <Student><StudentName>Jeanna
Marmolejo</StudentName><StudentEmail>JMarmolejo@school.com</StudentEmail></Student>
        <Student><StudentName>Norah
Revelle</StudentName><StudentEmail>NRevelle@school.com</StudentEmail></Student>
        <Student><StudentName>Claud
Lares</StudentName><StudentEmail>CLares@school.com</StudentEmail></Student>
        <Student><StudentName>Lacy
Bossert</StudentName><StudentEmail>LBossert@school.com</StudentEmail></Student>
        <Student><StudentName>Ernesto
Dangelo</StudentName><StudentEmail>EDangelo@school.com</StudentEmail></Student>
    </Class>
    <Class>
        <CourseName>ECON-323</CourseName>
        <CourseDescription>Macro Supply and
Demand</CourseDescription>
        <Department>Economics</Department>
        <CreditNumber>3323</CreditNumber>
        <Teacher>
            <TeacherName>Scrooge Duck</TeacherName>
            <TeacherEmail>sduck@school.com</TeacherEmail>
        </Teacher>
        <Student><StudentName>Evelina
Ruffino</StudentName><StudentEmail>ERuffino@school.com</StudentEmail></Student>
        <Student><StudentName>Krystal
Deeter</StudentName><StudentEmail>KDeeter@school.com</StudentEmail></Student>
        <Student><StudentName>Temika
Kanter</StudentName><StudentEmail>TKanter@school.com</StudentEmail></Student>
    </Class>

```

```

        <Student><StudentName>Zina
Borey</StudentName><StudentEmail>ZBorey@school.com</StudentEmail><
/Student>
        <Student><StudentName>Eugenio
Bossard</StudentName><StudentEmail>EBossard@school.com</StudentEma
il></Student>
        <Student><StudentName>Bess
Zucker</StudentName><StudentEmail>BZucker@school.com</StudentEmail
></Student>
        <Student><StudentName>Gisele
Landsman</StudentName><StudentEmail>GLandsman@school.com</StudentE
mail></Student>
        <Student><StudentName>Lorretta
Strawn</StudentName><StudentEmail>LStrawn@school.com</StudentEmail
></Student>
        <Student><StudentName>Juanita
Hamby</StudentName><StudentEmail>JHamby@school.com</StudentEmail><
/Student>
        <Student><StudentName>Fredda
Nakagawa</StudentName><StudentEmail>FNakagawa@school.com</StudentE
mail></Student>
        <Student><StudentName>Fletcher
Sigg</StudentName><StudentEmail>FSigg@school.com</StudentEmail></S
tudent>
        <Student><StudentName>Kiley
Cranor</StudentName><StudentEmail>KCranor@school.com</StudentEmail
></Student>
        <Student><StudentName>Jani
Gibeault</StudentName><StudentEmail>JGibeault@school.com</StudentE
mail></Student>
        <Student><StudentName>Willian
Gochenour</StudentName><StudentEmail>WGochenour@school.com</Studen
tEmail></Student>
        <Student><StudentName>Jeanna
Marmolejo</StudentName><StudentEmail>JMarmolejo@school.com</Studen
tEmail></Student>
        <Student><StudentName>Norah
Revelle</StudentName><StudentEmail>NRevelle@school.com</StudentEma
il></Student>
        <Student><StudentName>Claud
Lares</StudentName><StudentEmail>CLares@school.com</StudentEmail><
/Student>
        <Student><StudentName>Lacy
Bossert</StudentName><StudentEmail>LBossert@school.com</StudentEma
il></Student>
        <Student><StudentName>Ernesto
Dangelo</StudentName><StudentEmail>EDangelo@school.com</StudentEma
il></Student>
    </Class>
</Classes>

```

3. Save the file with the name `school.xml`.

Create an XML Data Source for the Tutorial

There are some Studio resources that you must create before starting this tutorial.

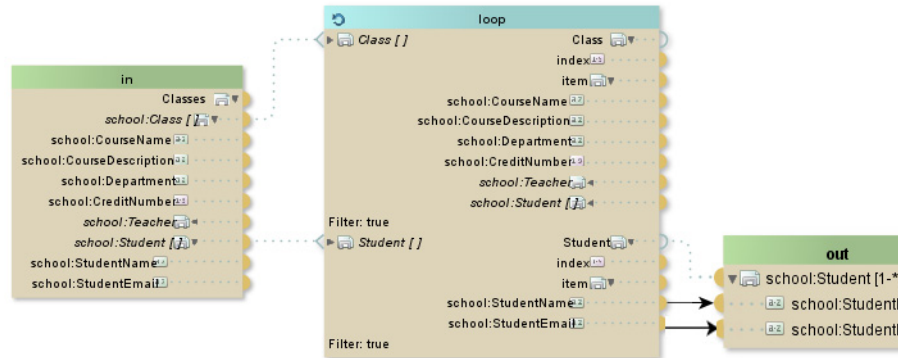
To create the XML data source

1. Open Studio.
2. Select Shared on the Studio resource tree.
3. Right-click and select New Data Source.
4. Select File-XML.
5. Click **Next**.
6. Type `school`.
7. Browse to the directory location where you saved the `school.xml` data file.
8. Click **OK**.
9. Click Create & Introspect.
10. Select the school.xml file.
11. Click **Next**.
12. Click **Finish**.
13. When introspection is complete, click **OK**.

The XML definition set is created for you.

Create the GetStudents Transformation Procedure

The GetStudents transformation is a simple procedure that will get used as part of a larger XML schema mutation transformation. The following graphic shows the goal for this task:



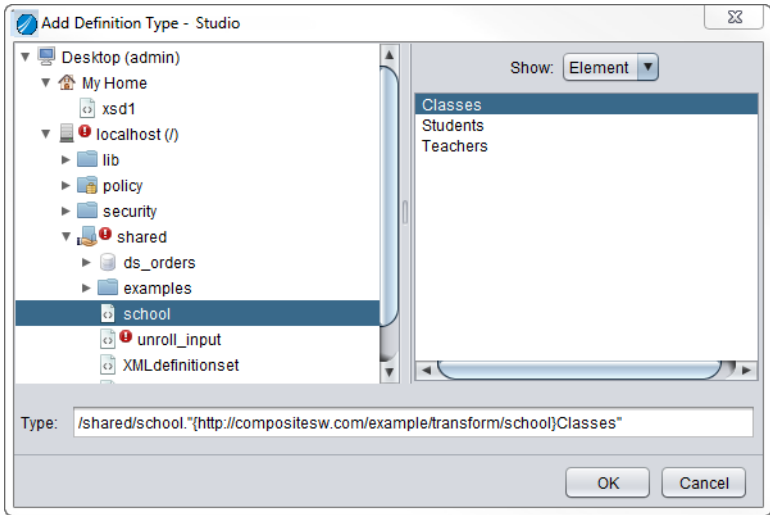
To create the GetStudents transform

1. Open Studio.
2. In the Studio resource tree, select Shared.
3. Right-click and select New Transformation.
4. Select Any-Any Transformation and click **Next**.
5. Type GetStudents in the Transformation Name field.
6. Click **Finish**.

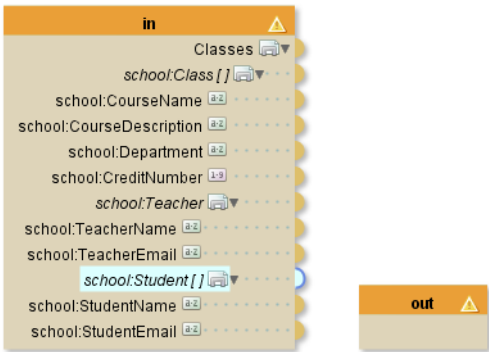
The Transformation Editor opens with the Model tab displayed.

7. Click the in container.
8. Right-click and select Add Parameter.
9. Select Browse.
10. Browse to the schoolDefinitions definition set.

11. Select Classes.

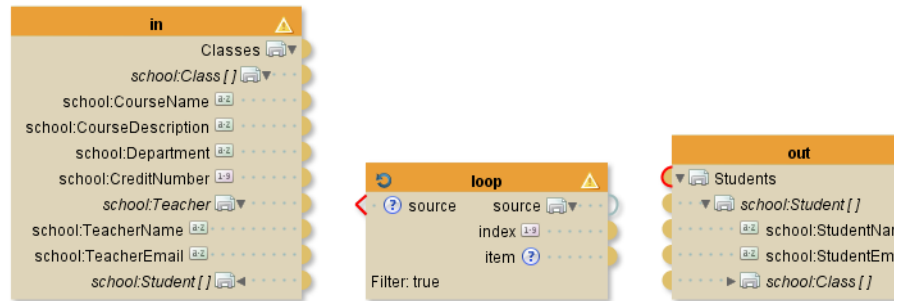


12. Click OK.



13. Save your transform.

14. From the palette, click and drag Loop onto the editor.

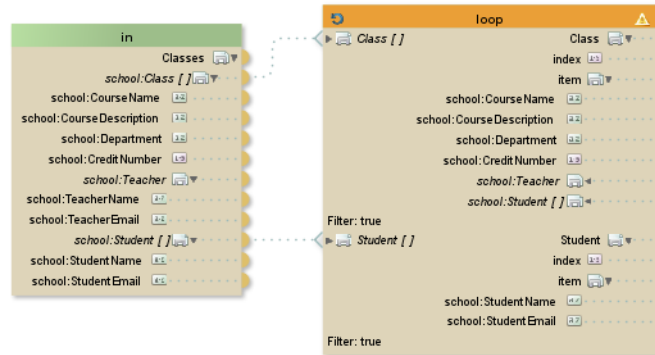


15. Draw a connection from school:Class to source on loop.

16. Click loop to make sure it is selected.

17. Right-click and select Add Source.

18. Draw a connection from school:Student to source on loop.



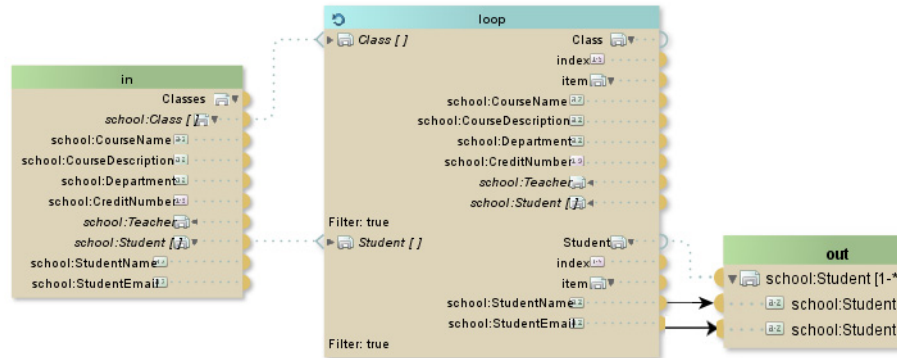
19. From loop draw a connection from Student to out.

Out will be populated with an array for the Student element.

20. Connect loop :school:StudentName to out: school:StudentName.

21. Connect loop :school:StudentEmail to out: school:StudentEmail

22. Save the transform.



Creating the ClassStudent Transformation

This task creates the transformation container for this tutorial.

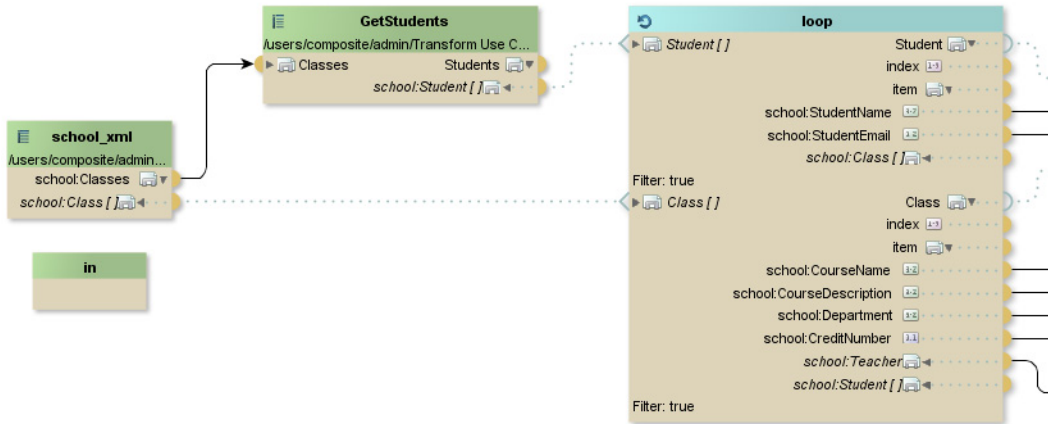
To create the ClassStudent transform

1. Open Studio.
2. In the Studio resource tree, select Shared.
3. Right-click and select New Transformation.
4. Select Any-Any Transformation and click **Next**.
5. Type ClassStudent in the Transformation Name field.
6. Click **Finish**.

The Transformation Editor opens with the Model tab displayed.

Adding and Connecting Source Operations

We need to bring in the XML file with its original hierarchy before we can transform it. The GetStudents simple transformation procedure also helps define the more complex transformation that we are building in this use case. The following partial transform is the goal for this task:

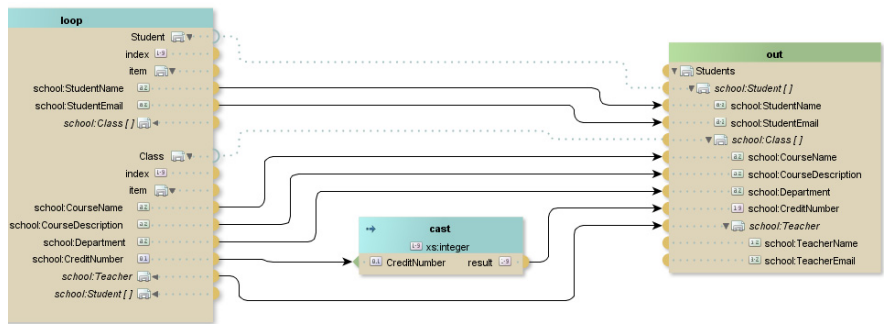


To add and connect the source operations

1. Open your transform.
2. From the Studio resource tree, select and drag the following resources onto the Transformation Editor:
 - school.xml
 - GetStudents
3. From the palette, add a loop to the editor.
4. Select the loop and add a new source.
5. Connect school:Classes to Classes in Get Students.
6. Connect school:Class to one of the sources in the loop.
7. Connect school:Student from GetStudents to the other source in the loop.
8. Save your transform.

Completing the ClassStudent Transformation

In this task the data type of one parameter needs to be transformed using the cast function and the parameters must be connected to out.



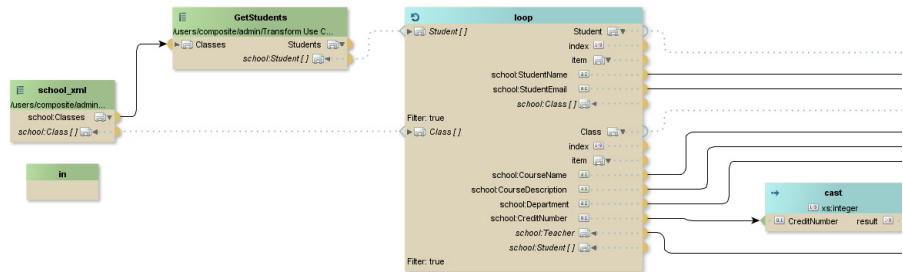
To complete the ClassStudent transform

1. Open you transform.
2. From the palette, add the cast operation to the editor.
3. Connect Student to the out container.
This creates the hierarchy that you need in the out container.
4. Connect the following loop and out parameters:

loop	out
school:StudentName	school:StudentName
school:StudentEmail	school:StudentEmail
Class	school:Class
school:CourseName	school:CourseName
school:CourseDescription	school:CourseDescription
school:Department	school:Department
school:Teacher	school:Teacher

5. Connect school:CreaditNumber to the source parameter in cast.
6. Connect result to school:CreditNumber in out.

7. Double-click cast.
8. In the Type field, select Integer > xs:integer.
9. Click OK.
10. Save your transform.



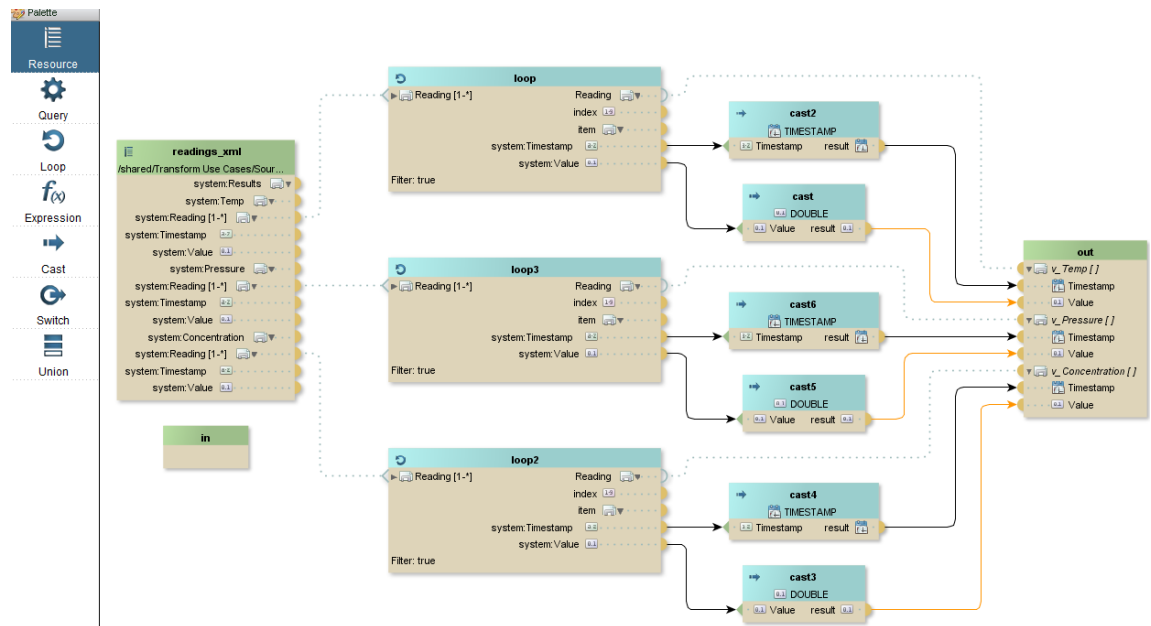
Transform Name and Value Pair Tutorial

This topic steps you through the creation of a transformation that puts XML into a table so it is easier to analyze.

The steps to create and use this transform include:

- [Creating an XML Data Source for the Tutorial, page 72](#)
- [Starting the NameValue Transformation Procedure, page 77](#)
- [Creating the Resource to Loop Connections, page 78](#)
- [Connecting the Loop Operations to out, page 81](#)
- [Connecting the Loop Operations to the Cast Operations, page 81](#)
- [Allowing NULL Output Values, page 83](#)

The goal of this tutorial is a transform similar to the following:



Creating an XML Data Source for the Tutorial

There is one Studio resource that you must create before starting this tutorial. It is a simple XML data source. This procedure provides the code for you to copy and paste into a text file.

To create the XML data source

1. Open a text editor of your choice.

2. Copy and paste the following into the editor:

```
<?xml version="1.0" encoding="UTF-8"?>
<Results xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://compositesw.com/transform/example/system"

xsi:schemaLocation="http://compositesw.com/transform/example/system
file:///d:/data/transform/xml/systemReadings.xsd">
  <Temp>

  <Reading><Timestamp>2013-04-17T18:00:13-09:00</Timestamp><Value>40
.69</Value></Reading>

  <Reading><Timestamp>2013-04-17T19:00:13-09:00</Timestamp><Value>41
.41</Value></Reading>

  <Reading><Timestamp>2013-04-17T20:00:13-09:00</Timestamp><Value>33
.89</Value></Reading>

  <Reading><Timestamp>2013-04-17T21:00:13-09:00</Timestamp><Value>25
.30</Value></Reading>

  <Reading><Timestamp>2013-04-17T22:00:13-09:00</Timestamp><Value>21
.45</Value></Reading>

  <Reading><Timestamp>2013-04-17T23:00:13-09:00</Timestamp><Value>25
.70</Value></Reading>

  <Reading><Timestamp>2013-04-18T00:00:13-09:00</Timestamp><Value>23
.01</Value></Reading>

  <Reading><Timestamp>2013-04-18T01:00:13-09:00</Timestamp><Value>22
.27</Value></Reading>

  <Reading><Timestamp>2013-04-18T02:00:13-09:00</Timestamp><Value>15
.24</Value></Reading>

  <Reading><Timestamp>2013-04-18T03:00:13-09:00</Timestamp><Value>20
.41</Value></Reading>

  <Reading><Timestamp>2013-04-18T04:00:13-09:00</Timestamp><Value>13
.96</Value></Reading>
```



```

<Reading><Timestamp>2013-04-18T05:00:13-09:00</Timestamp><Value>7.
37</Value></Reading>

<Reading><Timestamp>2013-04-18T06:00:13-09:00</Timestamp><Value>14
.71</Value></Reading>

<Reading><Timestamp>2013-04-18T07:00:13-09:00</Timestamp><Value>12
.30</Value></Reading>

<Reading><Timestamp>2013-04-18T08:00:13-09:00</Timestamp><Value>2.
46</Value></Reading>

<Reading><Timestamp>2013-04-18T09:00:13-09:00</Timestamp><Value>5.
45</Value></Reading>

<Reading><Timestamp>2013-04-18T10:00:13-09:00</Timestamp><Value>10
.34</Value></Reading>

<Reading><Timestamp>2013-04-18T11:00:13-09:00</Timestamp><Value>19
.02</Value></Reading>

<Reading><Timestamp>2013-04-18T12:00:13-09:00</Timestamp><Value>14
.26</Value></Reading>

<Reading><Timestamp>2013-04-18T13:00:13-09:00</Timestamp><Value>16
.90</Value></Reading>

<Reading><Timestamp>2013-04-18T14:00:13-09:00</Timestamp><Value>13
.40</Value></Reading>

<Reading><Timestamp>2013-04-18T15:00:13-09:00</Timestamp><Value>10
.73</Value></Reading>

<Reading><Timestamp>2013-04-18T16:00:13-09:00</Timestamp><Value>12
.87</Value></Reading>

<Reading><Timestamp>2013-04-18T17:00:13-09:00</Timestamp><Value>20
.49</Value></Reading>

<Reading><Timestamp>2013-04-18T18:00:13-09:00</Timestamp><Value>22
.45</Value></Reading>

<Reading><Timestamp>2013-04-18T19:00:13-09:00</Timestamp><Value>25
.30</Value></Reading>

<Reading><Timestamp>2013-04-18T20:00:13-09:00</Timestamp><Value>32
.05</Value></Reading>

<Reading><Timestamp>2013-04-18T21:00:13-09:00</Timestamp><Value>41
.24</Value></Reading>

<Reading><Timestamp>2013-04-18T22:00:13-09:00</Timestamp><Value>50
.50</Value></Reading>

<Reading><Timestamp>2013-04-18T23:00:13-09:00</Timestamp><Value>47
.95</Value></Reading>

```

```

        </Temp>
        <Pressure>

<Reading><Timestamp>2013-04-17T18:00:13-09:00</Timestamp><Value>31
.12</Value></Reading>

<Reading><Timestamp>2013-04-17T19:00:13-09:00</Timestamp><Value>32
.28</Value></Reading>

<Reading><Timestamp>2013-04-17T20:00:13-09:00</Timestamp><Value>31
.24</Value></Reading>

<Reading><Timestamp>2013-04-17T21:00:13-09:00</Timestamp><Value>40
.55</Value></Reading>

<Reading><Timestamp>2013-04-17T22:00:13-09:00</Timestamp><Value>37
.60</Value></Reading>

<Reading><Timestamp>2013-04-17T23:00:13-09:00</Timestamp><Value>44
.57</Value></Reading>

<Reading><Timestamp>2013-04-18T00:00:13-09:00</Timestamp><Value>41
.62</Value></Reading>

<Reading><Timestamp>2013-04-18T01:00:13-09:00</Timestamp><Value>32
.64</Value></Reading>

<Reading><Timestamp>2013-04-18T02:00:13-09:00</Timestamp><Value>26
.51</Value></Reading>

<Reading><Timestamp>2013-04-18T03:00:13-09:00</Timestamp><Value>28
.20</Value></Reading>

<Reading><Timestamp>2013-04-18T04:00:13-09:00</Timestamp><Value>21
.63</Value></Reading>

<Reading><Timestamp>2013-04-18T05:00:13-09:00</Timestamp><Value>15
.59</Value></Reading>

<Reading><Timestamp>2013-04-18T06:00:13-09:00</Timestamp><Value>24
.33</Value></Reading>

<Reading><Timestamp>2013-04-18T07:00:13-09:00</Timestamp><Value>22
.03</Value></Reading>

<Reading><Timestamp>2013-04-18T08:00:13-09:00</Timestamp><Value>28
.60</Value></Reading>

<Reading><Timestamp>2013-04-18T09:00:13-09:00</Timestamp><Value>34
.02</Value></Reading>

<Reading><Timestamp>2013-04-18T10:00:13-09:00</Timestamp><Value>24
.19</Value></Reading>

<Reading><Timestamp>2013-04-18T11:00:13-09:00</Timestamp><Value>17
.27</Value></Reading>

```

```

<Reading><Timestamp>2013-04-18T12:00:13-09:00</Timestamp><Value>19
.38</Value></Reading>

<Reading><Timestamp>2013-04-18T13:00:13-09:00</Timestamp><Value>16
.52</Value></Reading>

<Reading><Timestamp>2013-04-18T14:00:13-09:00</Timestamp><Value>14
.57</Value></Reading>

<Reading><Timestamp>2013-04-18T15:00:13-09:00</Timestamp><Value>11
.66</Value></Reading>

<Reading><Timestamp>2013-04-18T16:00:13-09:00</Timestamp><Value>13
.55</Value></Reading>

<Reading><Timestamp>2013-04-18T17:00:13-09:00</Timestamp><Value>22
.12</Value></Reading>

<Reading><Timestamp>2013-04-18T18:00:13-09:00</Timestamp><Value>28
.35</Value></Reading>

<Reading><Timestamp>2013-04-18T19:00:13-09:00</Timestamp><Value>27
.98</Value></Reading>

<Reading><Timestamp>2013-04-18T20:00:13-09:00</Timestamp><Value>25
.84</Value></Reading>

<Reading><Timestamp>2013-04-18T21:00:13-09:00</Timestamp><Value>25
.70</Value></Reading>

<Reading><Timestamp>2013-04-18T22:00:13-09:00</Timestamp><Value>25
.03</Value></Reading>

<Reading><Timestamp>2013-04-18T23:00:13-09:00</Timestamp><Value>24
.27</Value></Reading>
  </Pressure>
  <Concentration>

<Reading><Timestamp>2013-04-17T18:00:13-09:00</Timestamp><Value>4.
27</Value></Reading>

<Reading><Timestamp>2013-04-17T19:00:13-09:00</Timestamp><Value>-1
.36</Value></Reading>

<Reading><Timestamp>2013-04-17T20:00:13-09:00</Timestamp><Value>-5
.61</Value></Reading>

<Reading><Timestamp>2013-04-17T21:00:13-09:00</Timestamp><Value>-1
.19</Value></Reading>

<Reading><Timestamp>2013-04-17T22:00:13-09:00</Timestamp><Value>6.
67</Value></Reading>

<Reading><Timestamp>2013-04-17T23:00:13-09:00</Timestamp><Value>-0
.84</Value></Reading>

```

```

<Reading><Timestamp>2013-04-18T00:00:13-09:00</Timestamp><Value>-4
.21</Value></Reading>

<Reading><Timestamp>2013-04-18T01:00:13-09:00</Timestamp><Value>3.
41</Value></Reading>

<Reading><Timestamp>2013-04-18T02:00:13-09:00</Timestamp><Value>12
.19</Value></Reading>

<Reading><Timestamp>2013-04-18T03:00:13-09:00</Timestamp><Value>12
.82</Value></Reading>

<Reading><Timestamp>2013-04-18T04:00:13-09:00</Timestamp><Value>14
.44</Value></Reading>

<Reading><Timestamp>2013-04-18T05:00:13-09:00</Timestamp><Value>16
.39</Value></Reading>

<Reading><Timestamp>2013-04-18T06:00:13-09:00</Timestamp><Value>20
.99</Value></Reading>

<Reading><Timestamp>2013-04-18T07:00:13-09:00</Timestamp><Value>16
.94</Value></Reading>

<Reading><Timestamp>2013-04-18T08:00:13-09:00</Timestamp><Value>19
.92</Value></Reading>

<Reading><Timestamp>2013-04-18T09:00:13-09:00</Timestamp><Value>15
.00</Value></Reading>

<Reading><Timestamp>2013-04-18T10:00:13-09:00</Timestamp><Value>10
.25</Value></Reading>

<Reading><Timestamp>2013-04-18T11:00:13-09:00</Timestamp><Value>10
.41</Value></Reading>

<Reading><Timestamp>2013-04-18T12:00:13-09:00</Timestamp><Value>20
.20</Value></Reading>

<Reading><Timestamp>2013-04-18T13:00:13-09:00</Timestamp><Value>13
.62</Value></Reading>

<Reading><Timestamp>2013-04-18T14:00:13-09:00</Timestamp><Value>12
.87</Value></Reading>

<Reading><Timestamp>2013-04-18T15:00:13-09:00</Timestamp><Value>6.
37</Value></Reading>

<Reading><Timestamp>2013-04-18T16:00:13-09:00</Timestamp><Value>3.
68</Value></Reading>

<Reading><Timestamp>2013-04-18T17:00:13-09:00</Timestamp><Value>6.
22</Value></Reading>

<Reading><Timestamp>2013-04-18T18:00:13-09:00</Timestamp><Value>-2
.26</Value></Reading>

```

```

<Reading><Timestamp>2013-04-18T19:00:13-09:00</Timestamp><Value>2.
72</Value></Reading>

<Reading><Timestamp>2013-04-18T20:00:13-09:00</Timestamp><Value>7.
52</Value></Reading>

<Reading><Timestamp>2013-04-18T21:00:13-09:00</Timestamp><Value>4.
78</Value></Reading>

<Reading><Timestamp>2013-04-18T22:00:13-09:00</Timestamp><Value>6.
56</Value></Reading>

<Reading><Timestamp>2013-04-18T23:00:13-09:00</Timestamp><Value>5.
40</Value></Reading>
  </Concentration>
</Results>

```

3. Save the file as **readings.xml** in a directory location that you can easily navigate to from Studio.
4. Open Studio.
5. Select Shared on the Studio resource tree.
6. Right-click and select New Data Source.
7. Select File-XML.
8. Click Next.
9. Name the data source Readings.
10. Next to the Root Path field, click Browse and navigate to the location of the readings.xml file.
11. Click OK.
12. Click Create & Introspect.
13. Make sure readings.xml is selected.
14. Click Next.
15. Click Finish.
16. When introspection completes, click OK.

Starting the NameValue Transformation Procedure

The NameValue transformation is a procedure that will transform XML into a table so it is easier to analyze.

To create the NameValue transform

1. Open Studio.
2. In the Studio resource tree, select Shared.
3. Right-click and select New Transformation.
4. Select Any-Any Transformation and click Next.
5. Type NameValue in the Transformation Name field.
6. Click Finish.

The Transformation Editor opens with the Model tab displayed.

7. From the Studio resource tree, drag the readings.xml resource onto the NameValue transform editor workspace.
8. Add 3 loop operations and 6 cast operations to the transform.
9. Save the transform.

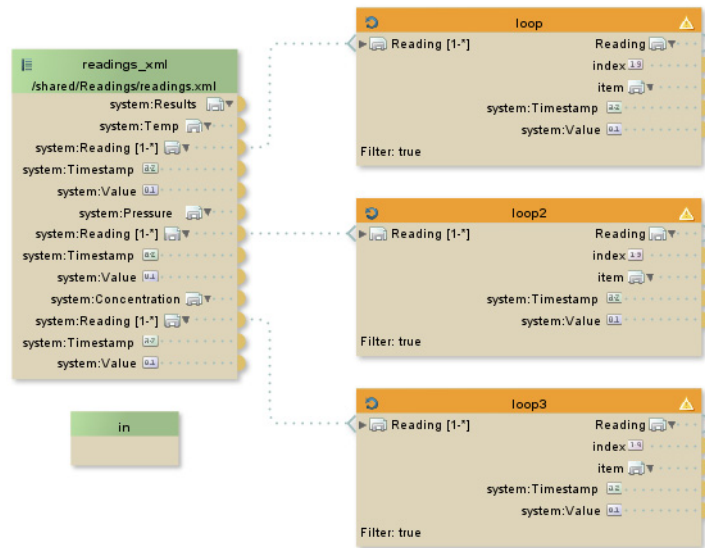
Creating the Resource to Loop Connections

This task creates the connections between the operations that you have added to the transform.

To create the resource to loop connections

1. Open Studio.
2. Open your NameValue transform.
3. Expand the hierarchy in the readings.xml operation.
4. Connect system:Reading[1-*], from under system:Result, to source in loop.
5. Connect system:Reading[1-*], from under system:Pressure, to source in loop2.

6. Connect `system:Reading[1-*`], from under `system:Concentration`, to source in `loop3`.

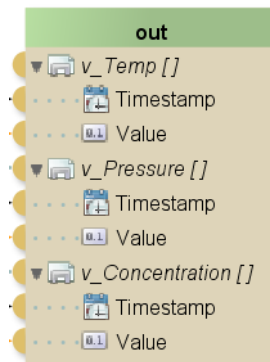


7. Save the transform.

Creating the Cursor Outputs

This procedure creates the outputs for your view.

Instead of creating each subparameter in the out container, you can copy the subparameters and use the paste into right-mouse menu option to create the multiple timestamp and value parameters.



To create the v_Temp cursor outputs

1. Select the out container in the NameValue transform.
2. Right-click and select Add Parameter.
3. Select Complex > Cursor.
4. Name the cursor v_Temp.
5. Right-click the operation handle next to v_Temp.
6. Select Add a parameter.
7. Select Add Parameter with the type Time > Timestamp and named Timestamp.
8. Right-click the operation handle next to v_Temp.
9. Select Add Parameter with the type Decimal > DOUBLE and named Value.

To create the v_Pressure cursor outputs

1. Right-click and select Add Parameter.
2. Select Complex > Cursor.
3. Name the cursor v_Pressure.
4. Right-click the operation handle next to v_Pressure.
5. Select Add a parameter.
6. Select Add Parameter with the type Time > Timestamp and named Timestamp.
7. Right-click the operation handle next to v_Pressure.
8. Select Add Parameter with the type Decimal > DOUBLE and named Value.

To create the v_Concentration cursor outputs

1. Right-click and select Add Parameter.
2. Select Complex > Cursor.
3. Name the cursor v_Concentration.
4. Right-click the operation handle next to v_Concentration.
5. Select Add a parameter.
6. Select Add Parameter with the type Time > Timestamp and named Timestamp.
7. Right-click the operation handle next to v_Concentration.

8. Select Add Parameter with the type Decimal > DOUBLE and named Value.
9. Save the transform.

Connecting the Loop Operations to out

To connect loop operations to out

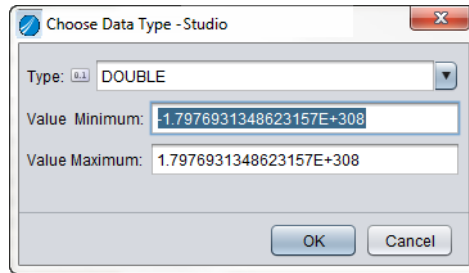
1. Open the NameValue transform.
2. Connect Reading from loop to v_Temp.
3. Connect Reading from loop2 to v_Pressure.
4. Connect Reading from loop3 to v_Concentration.
5. Save the transformation.

Connecting the Loop Operations to the Cast Operations

The XML formatted timestamp and values need to be transformed into data types that will work in a SQL view environment. The NMTOKEN values need to be converted to timestamp and the xs:decimal values need to be converted to DOUBLE.

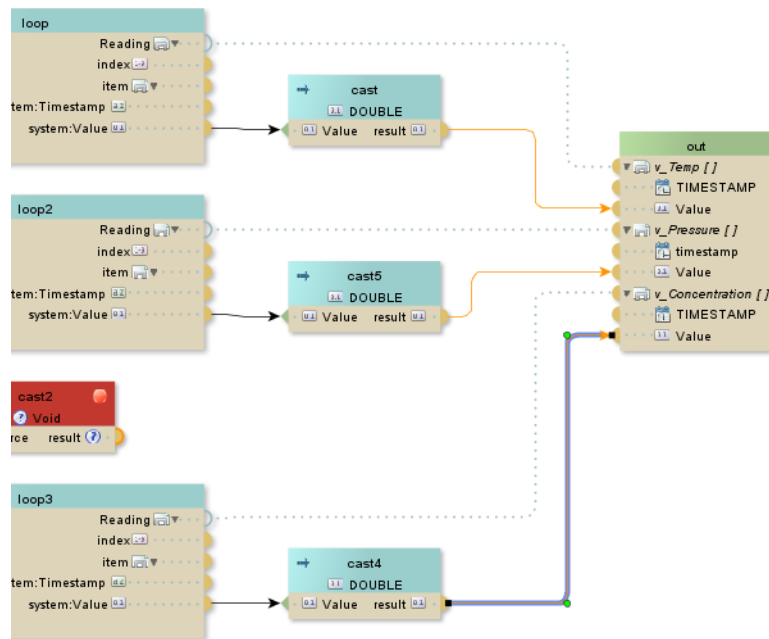
To connect loop operations to cast operations

1. Open the NameValue transform.
2. Connect loop system:Value to the source in cast.
3. Double-click cast to open the Choose Data Type editor.
4. From the drop down list select Decimal > DOUBLE.
5. For the minimum and maximum values, accept the defaults of:
-1.7976931348623157E+308 **and** 1.7976931348623157E+308



6. Connect cast result to the out v_Temp Value parameter.
7. Repeat these steps to create the cast operations for each of the loop system:Value parameters and connect them to the out v_Pressure and v_Concentration Value parameters.

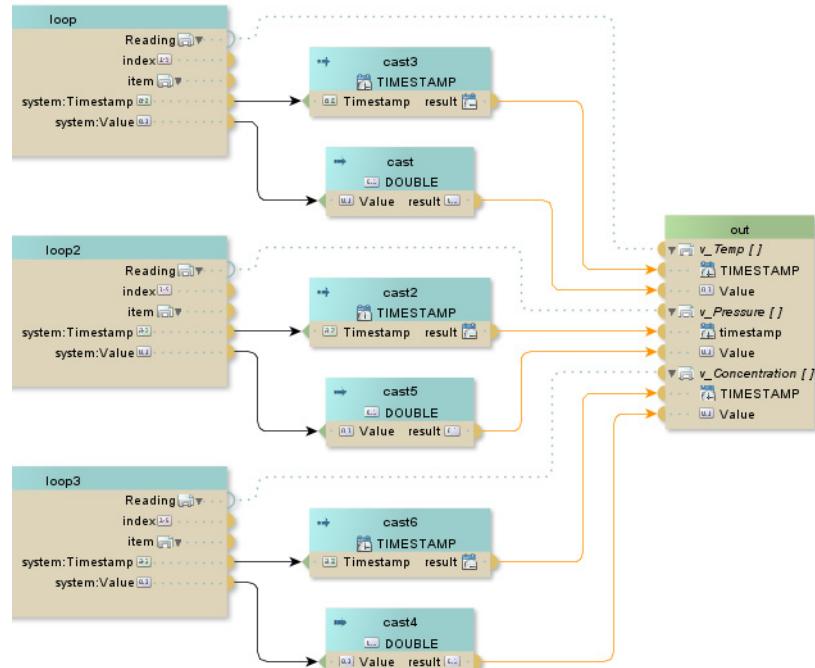
For example, your transform should have cast operations and connections similar to the following:



8. Connect loop system:Timestamp to the source in a cast operation.
9. Double-click cast to open the Choose Data Type editor.
10. From the drop down list select Time > TIMESTAMP.
11. Connect cast result to the out v_Temp Timestamp parameter.

12. Repeat these steps to create the cast operations for each of the loop system:Timestamp parameters and connect them to the out v_Pressure and v_Concentration Timestamp parameters.

For example, your transform should have cast operations and connections similar to the following:



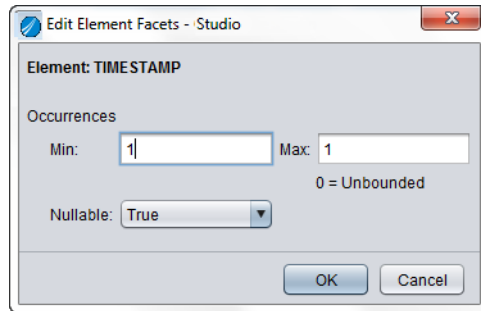
13. Save the transformation.

Allowing NULL Output Values

This same functionality is available from the Model tab.

To allow NULL values

1. Open the NameValue transform.
2. Right-click on any of the parameters and select Change Facets.



3. In the Nullable field, select True.
4. Click OK.
5. Repeat this for each output parameter.
6. Save your transform.

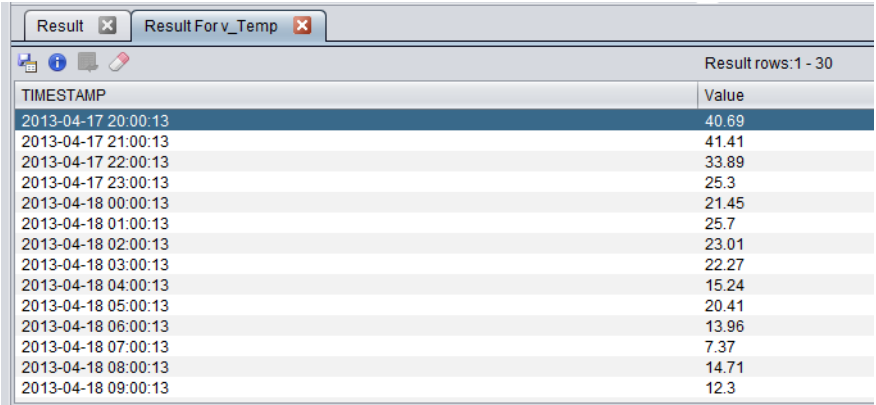
Executing and Testing Your Transform

After designing your sample transform it is valuable to execute it to see what data it retrieves.

To execute and test the results of your transform

1. Select Execute.
2. The right-side of Studio splits and the Result panel opens.
3. Click one of the Display buttons.

4. Validate that your results look similar to the following:



Result		Result For v_Temp	
		Result rows: 1 - 30	
TIMESTAMP		Value	
2013-04-17 20:00:13		40.69	
2013-04-17 21:00:13		41.41	
2013-04-17 22:00:13		33.89	
2013-04-17 23:00:13		25.3	
2013-04-18 00:00:13		21.45	
2013-04-18 01:00:13		25.7	
2013-04-18 02:00:13		23.01	
2013-04-18 03:00:13		22.27	
2013-04-18 04:00:13		15.24	
2013-04-18 05:00:13		20.41	
2013-04-18 06:00:13		13.96	
2013-04-18 07:00:13		7.37	
2013-04-18 08:00:13		14.71	
2013-04-18 09:00:13		12.3	

5. Save and close your transform.

Transform XML to Relational Tutorial

The purpose of this lab is to learn how to use the Any-to-Any Transformation Editor to transform a hierarchical XML document to a “flat” relational structure.

The steps to create and use this transform include:

- [Creating the School.xml Data File for the Tutorial, page 87](#)
- [Creating an XML Data Source for the Tutorial, page 92](#)
- [Creating the SchoolXML2Rel Transformation, page 93](#)
- [Creating the SchoolXML2Rel View, page 95](#)

Creating the School.xml Data File for the Tutorial

The XML file contain data for the execution of the transform is required for this tutorial. This procedure provides the code for you to copy and paste into the Studio editor to create the simple XML data file.

To create the School.xml data file

1. Open your favorite text editor.
2. Copy and paste the following XML code into the text editor.

```
<?xml version="1.0" encoding="UTF-8"?>
<Classes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://compositesw.com/transform/example/school">
  <Class>
    <CourseName>BIO-101</CourseName>
    <CourseDescription>Intro to biology</CourseDescription>
    <Department>Science</Department>
    <CreditNumber>1101</CreditNumber>
    <Teacher>
      <TeacherName>Bob Darwin</TeacherName>
      <TeacherEmail>bdarwin@school.com</TeacherEmail>
    </Teacher>
    <Student><StudentName>Evelina
Ruffino</StudentName><StudentEmail>ERuffino@school.com</StudentEmail>
</Student>
    <Student><StudentName>Krystal
Deeter</StudentName><StudentEmail>KDeeter@school.com</StudentEmail>
</Student>
    <Student><StudentName>Temika
Kanter</StudentName><StudentEmail>TKanter@school.com</StudentEmail>
</Student>
```

```

        <Student><StudentName>Zina
Borey</StudentName><StudentEmail>ZBorey@school.com</StudentEmail><
/Student>
        <Student><StudentName>Eugenio
Bossard</StudentName><StudentEmail>EBossard@school.com</StudentEma
il></Student>
        <Student><StudentName>Bess
Zucker</StudentName><StudentEmail>BZucker@school.com</StudentEmail
></Student>
        <Student><StudentName>Gisele
Landsman</StudentName><StudentEmail>GLandsman@school.com</StudentE
mail></Student>
        <Student><StudentName>Lorretta
Strawn</StudentName><StudentEmail>LStrawn@school.com</StudentEmail
></Student>
        <Student><StudentName>Juanita
Hamby</StudentName><StudentEmail>JHamby@school.com</StudentEmail><
/Student>
        <Student><StudentName>Fredda
Nakagawa</StudentName><StudentEmail>FNakagawa@school.com</StudentE
mail></Student>
        <Student><StudentName>Fletcher
Sigg</StudentName><StudentEmail>FSigg@school.com</StudentEmail></S
tudent>
        <Student><StudentName>Latrina
Rumore</StudentName><StudentEmail>LRumore@school.com</StudentEmail
></Student>
        <Student><StudentName>Neg
Wisener</StudentName><StudentEmail>NWisener@school.com</StudentEma
il></Student>
        <Student><StudentName>Hwa
Speegle</StudentName><StudentEmail>HSpeegle@school.com</StudentEma
il></Student>
        <Student><StudentName>Mariella
Stonerock</StudentName><StudentEmail>MStonerock@school.com</Studen
tEmail></Student>
        <Student><StudentName>Evan
Pagaduan</StudentName><StudentEmail>EPagaduan@school.com</StudentE
mail></Student>
        <Student><StudentName>Gertrud
Millman</StudentName><StudentEmail>GMillman@school.com</StudentEma
il></Student>
        <Student><StudentName>Bill
Baynard</StudentName><StudentEmail>BBaynard@school.com</StudentEma
il></Student>
        <Student><StudentName>Neal
Sheilds</StudentName><StudentEmail>NSheilds@school.com</StudentEma
il></Student>
        <Student><StudentName>Emery
Galles</StudentName><StudentEmail>EGalles@school.com</StudentEmail
></Student>
    </Class>
    <Class>
        <CourseName>BIO-201</CourseName>
        <CourseDescription>Evolution</CourseDescription>
        <Department>Science</Department>
        <CreditNumber>1201</CreditNumber>

```



```

    <Teacher>
      <TeacherName>Bob Darwin</TeacherName>
      <TeacherEmail>bdarwin@school.com</TeacherEmail>
    </Teacher>
    <Student><StudentName>Leeann
Trimpe</StudentName><StudentEmail>LTrimpe@school.com</StudentEmail
></Student>
    <Student><StudentName>Georgeann
Sones</StudentName><StudentEmail>GSones@school.com</StudentEmail><
/Student>
    <Student><StudentName>Chrissy
Loiselle</StudentName><StudentEmail>CLoiselle@school.com</StudentE
mail></Student>
    <Student><StudentName>Natalya
Fife</StudentName><StudentEmail>NFife@school.com</StudentEmail></S
tudent>
    <Student><StudentName>Kandi
Markwell</StudentName><StudentEmail>KMarkwell@school.com</StudentE
mail></Student>
    <Student><StudentName>Tommie
Weller</StudentName><StudentEmail>TWeller@school.com</StudentEmail
></Student>
    <Student><StudentName>Elin
Schimpf</StudentName><StudentEmail>ESchimpf@school.com</StudentEma
il></Student>
    <Student><StudentName>Spring
Luke</StudentName><StudentEmail>SLuke@school.com</StudentEmail></S
tudent>
    <Student><StudentName>Devona
Bertelsen</StudentName><StudentEmail>DBertelsen@school.com</Studen
tEmail></Student>
    <Student><StudentName>Celinda
Chaplin</StudentName><StudentEmail>CChaplin@school.com</StudentEma
il></Student>
    <Student><StudentName>Annamarie
Eustice</StudentName><StudentEmail>AEustice@school.com</StudentEma
il></Student>
    <Student><StudentName>Ambrose
Keating</StudentName><StudentEmail>AKeating@school.com</StudentEma
il></Student>
    <Student><StudentName>Kiley
Cranor</StudentName><StudentEmail>KCranor@school.com</StudentEmail
></Student>
    <Student><StudentName>Jani
Gibeault</StudentName><StudentEmail>JGibeault@school.com</StudentE
mail></Student>
    <Student><StudentName>Willian
Gochenour</StudentName><StudentEmail>WGochenour@school.com</Studen
tEmail></Student>
    <Student><StudentName>Jeanna
Marmolejo</StudentName><StudentEmail>JMarmolejo@school.com</Studen
tEmail></Student>
    <Student><StudentName>Norah
Revelle</StudentName><StudentEmail>NRevelle@school.com</StudentEma
il></Student>

```

```

        <Student><StudentName>Claud
Lares</StudentName><StudentEmail>CLares@school.com</StudentEmail><
/Student>
        <Student><StudentName>Lacy
Bossert</StudentName><StudentEmail>LBossert@school.com</StudentEma
il></Student>
        <Student><StudentName>Ernesto
Dangelo</StudentName><StudentEmail>EDangelo@school.com</StudentEma
il></Student>
    </Class>
    <Class>
        <CourseName>CSC-450</CourseName>
        <CourseDescription>Compilers I</CourseDescription>
        <Department>Computer Science</Department>
        <CreditNumber>2450</CreditNumber>
        <Teacher>
            <TeacherName>Chuck Babbage</TeacherName>
            <TeacherEmail>cbabbage@school.com</TeacherEmail>
        </Teacher>
        <Student><StudentName>Fredda
Nakagawa</StudentName><StudentEmail>FNakagawa@school.com</StudentE
mail></Student>
        <Student><StudentName>Fletcher
Sigg</StudentName><StudentEmail>FSigg@school.com</StudentEmail></S
tudent>
        <Student><StudentName>Latrina
Rumore</StudentName><StudentEmail>LRumore@school.com</StudentEmail
></Student>
        <Student><StudentName>Nan
Wisener</StudentName><StudentEmail>NWisener@school.com</StudentEma
il></Student>
        <Student><StudentName>Hwa
Speegle</StudentName><StudentEmail>HSpeegle@school.com</StudentEma
il></Student>
        <Student><StudentName>Mariella
Stonerock</StudentName><StudentEmail>MStonerock@school.com</Studen
tEmail></Student>
        <Student><StudentName>Evan
Pagaduan</StudentName><StudentEmail>EPagaduan@school.com</StudentE
mail></Student>
        <Student><StudentName>Gertrud
Millman</StudentName><StudentEmail>GMillman@school.com</StudentEma
il></Student>
        <Student><StudentName>Bill
Baynard</StudentName><StudentEmail>BBaynard@school.com</StudentEma
il></Student>
        <Student><StudentName>Neal
Sheilds</StudentName><StudentEmail>NSheilds@school.com</StudentEma
il></Student>
        <Student><StudentName>Emery
Galles</StudentName><StudentEmail>EGalles@school.com</StudentEmail
></Student>
        <Student><StudentName>Ambrose
Keating</StudentName><StudentEmail>AKeating@school.com</StudentEma
il></Student>

```

```

        <Student><StudentName>Kiley
Cranor</StudentName><StudentEmail>KCranor@school.com</StudentEmail
></Student>
        <Student><StudentName>Jani
Gibeault</StudentName><StudentEmail>JGibeault@school.com</StudentE
mail></Student>
        <Student><StudentName>Willian
Gochenour</StudentName><StudentEmail>WGochenour@school.com</Studen
tEmail></Student>
        <Student><StudentName>Jeanna
Marmolejo</StudentName><StudentEmail>JMarmolejo@school.com</Studen
tEmail></Student>
        <Student><StudentName>Norah
Revelle</StudentName><StudentEmail>NRevelle@school.com</StudentEma
il></Student>
        <Student><StudentName>Claud
Lares</StudentName><StudentEmail>CLares@school.com</StudentEmail><
/Student>
        <Student><StudentName>Lacy
Bossert</StudentName><StudentEmail>LBossert@school.com</StudentEma
il></Student>
        <Student><StudentName>Ernesto
Dangelo</StudentName><StudentEmail>EDangelo@school.com</StudentEma
il></Student>
    </Class>
    <Class>
        <CourseName>ECON-323</CourseName>
        <CourseDescription>Macro Supply and
Demand</CourseDescription>
        <Department>Economics</Department>
        <CreditNumber>3323</CreditNumber>
        <Teacher>
            <TeacherName>Scrooge Duck</TeacherName>
            <TeacherEmail>sduck@school.com</TeacherEmail>
        </Teacher>
        <Student><StudentName>Evelina
Ruffino</StudentName><StudentEmail>ERuffino@school.com</StudentEma
il></Student>
        <Student><StudentName>Krystal
Deeter</StudentName><StudentEmail>KDeeter@school.com</StudentEmail
></Student>
        <Student><StudentName>Temika
Kanter</StudentName><StudentEmail>TKanter@school.com</StudentEmail
></Student>
        <Student><StudentName>Zina
Borey</StudentName><StudentEmail>ZBorey@school.com</StudentEmail><
/Student>
        <Student><StudentName>Eugenio
Bossard</StudentName><StudentEmail>EBossard@school.com</StudentEma
il></Student>
        <Student><StudentName>Bess
Zucker</StudentName><StudentEmail>BZucker@school.com</StudentEmail
></Student>
        <Student><StudentName>Gisele
Landsman</StudentName><StudentEmail>GLandsman@school.com</StudentE
mail></Student>

```

```

        <Student><StudentName>Lorretta
Strawn</StudentName><StudentEmail>LStrawn@school.com</StudentEmail
></Student>
        <Student><StudentName>Juanita
Hamby</StudentName><StudentEmail>JHamby@school.com</StudentEmail><
/Student>
        <Student><StudentName>Fredda
Nakagawa</StudentName><StudentEmail>FNakagawa@school.com</StudentE
mail></Student>
        <Student><StudentName>Fletcher
Sigg</StudentName><StudentEmail>FSigg@school.com</StudentEmail></S
tudent>
        <Student><StudentName>Kiley
Cranor</StudentName><StudentEmail>KCranor@school.com</StudentEmail
></Student>
        <Student><StudentName>Jani
Gibeault</StudentName><StudentEmail>JGibeault@school.com</StudentE
mail></Student>
        <Student><StudentName>Willian
Gochenour</StudentName><StudentEmail>WGochenour@school.com</Studen
tEmail></Student>
        <Student><StudentName>Jeanna
Marmolejo</StudentName><StudentEmail>JMarmolejo@school.com</Studen
tEmail></Student>
        <Student><StudentName>Norah
Revelle</StudentName><StudentEmail>NRevelle@school.com</StudentEma
il></Student>
        <Student><StudentName>Claud
Lares</StudentName><StudentEmail>CLares@school.com</StudentEmail><
/Student>
        <Student><StudentName>Lacy
Bossert</StudentName><StudentEmail>LBossert@school.com</StudentEma
il></Student>
        <Student><StudentName>Ernesto
Dangelo</StudentName><StudentEmail>EDangelo@school.com</StudentEma
il></Student>
    </Class>
</Classes>

```

3. Save the file with the name `school.xml`.

Creating an XML Data Source for the Tutorial

There are some Studio resources that you must create before starting this tutorial.

To create the XML data source

1. Open Studio.
2. Select Shared on the Studio resource tree.

3. Right-click and select New Data Source.
4. Select File-XML.
5. Click Next.
6. Type school.
7. Browse to the directory location where you saved the school.xml data file.
8. Click OK.
9. Click Create & Introspect.
10. Select the school.xml file.
11. Click Next.
12. Click Finish.
13. When introspection is complete, click OK.

The XML definition set is created for you.

Creating the SchoolXML2Rel Transformation

To transform XML to relational data

1. Create a folder called Transformations.
2. Right-click the Transformations folder, and select New Transformation.
3. Select Any-Any Transformation, and click Next.
4. Type SchoolXML2Rel, and select Finish.

The transformation editor opens showing an empty transformation.

5. Expand the data source File_SchoolXML, and drag school.xml to the transformation design area.
6. Completely expand the data nodes (the little triangle to the right of school:Class[1-*]) in the school_xml object to see the entire structure.
7. Drag a loop object from the Transformation Editor palette onto the design area.

Use a Loop Link to connect the input sequence school:Class[1-*] to the loop source.

8. Select Loop Link Mode on the toolbar.

9. Click and drag to connect the output of the school:Class [1-∗] element of the school_xml file to the source input tab of the loop object.
10. Drag another Loop Link from the input sequence school:Student[1-∗] to the loop. When the words “insert source” appear, release the mouse button and a new loop source is created.
11. Create an output cursor. Right-click the out block and click Add Parameter.
12. Select Complex > CURSOR.

A cursor is added to the out block.

13. Rename the cursor to ClassList.
14. Use a Loop Link to connect the loop object’s Student output handle to ClassList input handle of the out object.

Doing so ensures that the output rows are based on the bottom level of the XML input hierarchy. To connect to the ClassList handle, make sure that when you release the mouse, ClassList[] is highlighted and “insert param” does not appear. If “insert param” appears, you will connect to a newly created parameter instead of ClassList.

15. Expand the school:Teacher node on the loop, so that you see all the columns.
16. Select Assign Link mode by clicking the button on the toolbar.
17. Drag the Assign Link lines from each of the loop’s individual data items to the out box. Release the mouse button when the words “insert param” appear, so that a new parameter is created.

The connection lines are red, the next few steps will change that.

18. Make sure that ClassList[] is at the top of the list. Select ClassList[] and use the toolbar arrow to move it to the top.
19. Select each field under ClassList[], and use the toolbar Move In button to indent.

The red connection lines vanish, because the newly indented columns are now a different category of data, belonging to the ClassList[] cursor. Their names and types are now defined. All that remains is to reconnect them.

20. Connect all the bottom-level loop fields to the corresponding out fields, and the design is complete.
21. Save the transformation and execute it.

Creating the SchoolXML2Rel View

To create the SchoolXML2Rel view

1. Create a view called SchoolXML2Rel.
2. Drag the SchoolXML2Rel transform into the view.
3. Save the view.
4. Right-click the SchoolXML2Rel view and select Publish.
5. Publish the view as a data service with the name SchoolXML2Rel.
6. Open a command prompt and navigate to the <TDV_install_dir>\apps\jdbc directory.
7. Test the published view with the following command and verify that you see the output and no errors:

```
JdbcSample.bat TDVTraining localhost 9401 admin admin composite  
"select * from SchoolXML2Rel"
```


SAP Hana View Tutorial

TDV supports multidimensional, calculation, and graphical views for SAP Hana. This tutorial focuses on how to create views that make use of those SAP Hana features.

- [Creating a TDV View using an SAP Hana Data Source, page 97](#)
- [Using GROUP BY and COUNT with an SAP Hana Data Source, page 97](#)

Creating a TDV View using an SAP Hana Data Source

SAP Hana has several unique data structures that can be introspected and brought into TDV. Because those structures are unique to SAP Hana you need to work with them in a specific way to get the results that are expected.

To create a view using an SAP Hana data source

1. Create a view that selects all the columns from the graphical view or calculation.
2. Create the aggregate function.
3. Refer to that aggregate function from within a view that selects all the columns from the graphical view or calculation.

For example:

```
select count from (select *from graphical_view_name)
where condition
select count from (
select* from
"_SYS_BIC"."eps-staging.salesDiscountGuidance.Cubes/DEALS_F")
where site_id=5310563
```

Using GROUP BY and COUNT with an SAP Hana Data Source

Create the base view with GROUP BY to force TDV to select all the columns.

For example

1. Create a View similar to the following which is named NEW_TEST_COUNT:

```
SELECT * from
/users/composite/admin/SAP/New_SAP_Hana/"PUBLIC"/"eps-staging.sale
sDiscountGuidance.Cubes::DEALS_F"
group by site_id,
QUOTE_KEY,DEAL_TYPE,
GLOBAL_ULTIMATE_ID,
BUSINESS_ENTITY,
DEALS_LIST_PRICE,
DEALS_NET_PRICE
```

2. Create a view that uses the count function on NEW_TEST_COUNT, for example:

```
SELECT
count(*) as TT from
/users/composite/admin/SAP/NEW_TEST_COUNT
where site_id=5310763
```