Teiid Designer User Guide

1

11.1.2

1. Introduction	. 1
1.1. What is Teiid Designer?	. 1
1.2. Metadata Overview	. 2
1.2.1. What is Metadata	. 2
1.2.2. Business and Technical Metadata	. 4
1.2.3. Source and View Metadata	. 5
1.3. It's all in the Modeling	. 9
1.3.1. What Are Models?	. 9
1.3.2. How is a Model Defined?	11
1.3.3. Guiding through the process	11
1.3.4. Targeting Your Teiid Submoduler	11
1.3.5. Model Classes and Types	12
1.3.6. The Virtual Database	12
1.3.7. Model Validation	14
1.3.8. Testing Your Models	14
1.3.9. Model Object Extensions	15
2. Dive Right In!	19
2.1. Targeting the Teiid Server	19
2.1.1. Server Version Preference	19
2.1.2. Defining a Teiid Server	20
2.1.3. Server Version Status Panel	21
2.2. Guide Example	22
2.2.1. Model a Relational (via JDBC) Source	22
2.3. Cheat Sheet Example	37
2.3.1. Consume a SOAP Web Service	37
3. Server Management	51
3.1. Setting up a Server	51
3.1.1. Teiid Version Support	54
4. New Model Wizards	55
4.1. Creating New Relational Source Model	56
4.1.1. Generate File Translator Procedures	57
4.1.2. Generate Web Service Translator Procedures	58
4.1.3. Copy From Existing Model	60
4.2. Creating New Relational View Model	61
4.2.1. Copy From Existing Model	62
4.2.2. Transform From Existing Model	63
4.3. Creating XML Document View Model	63
4.3.1. Copy From Existing Model	63
4.3.2. Build XML Documents From XML Schema	64
4.4. Creating XML Schema Model	68
4.4.1. Copy From Existing Model	69
4.5. Creating Web Service View Model	69
4.5.1. Copy From Existing Model	70
4.5.2. Build From Existing WSDL File(s) or URL	71

4.5.3. Build From Relational Models	. 71
4.5.4. Build From XML Document View Models	. 72
5. Importers	. 75
5.1. Import DDL	. 75
5.2. Import From Relational Database	77
5.2.1. Relational Model Costing	. 84
5.3. Import From Teiid Data Source Connection	. 84
5.4. Import From Flat File Source	. 89
5.5. Import From XML Data File Source	104
5.6. Import From Salesforce	112
5.7. Import Metadata From Text File	117
5.7.1. Import Relational Model (XML Format)	118
5.7.2. Import Relational Tables (CSV Format)	122
5.7.3. Import Relational View Tables (CSV Format)	124
5.8. Import Data From REST Service	126
5.9. Import Data From SOAP Service	133
5.9.1. Circular References in WSDL Schemas	140
5.10. Import WSDL Into Web Service	141
5.10.1. Import WSDL From Workspace Location	142
5.10.2. Import WSDL From File System Location	148
5.10.3. Import WSDL From URL	154
5.11. Import from an XML Schema File	160
5.12. Import From an LDAP Server	164
6. Creating and Editing Model Objects	171
6.1. Creating New Model Objects	171
6.1.1. New Child Action	171
6.1.2. New Sibling Action	174
6.1.3. New Association Action	176
6.2. New Model Object Wizards	178
6.2.1. Create Relational Table Wizard	178
6.2.2. Create Relational Procedure Wizard	179
6.2.3. Create Relational Index Wizard	186
6.2.4. Create View Model Objects Wizards	187
6.3. Model Object Editors	190
6.3.1. Transformation Editor	192
6.3.2. Input Set Editor (XML)	210
6.3.3. Choice Editor (XML)	212
6.3.4. Recursion Editor (XML)	215
6.3.5. Operation Editor	219
6.4. Managing Model Object Extensions	221
6.4.1. Create New MED	222
6.4.2. Edit MED	223
6.4.3. Extending Models With MEDs	227
6.4.4. Setting Extended Property Values	228

7. Metadata-specific Modeling	231
7.1. Relational Source Modeling	231
7.1.1. Source Function	231
7.2. Relational View Modeling	231
7.2.1. Create Materialized Views	231
7.3. XML Document Modeling	235
7.3.1. Create XML View Documents from schema	235
7.4. Web Services Modeling	239
7.4.1. Create Web Service Action	239
7.4.2. Web Services War Generation	241
8. User Defined Functions	255
8.1. Modeling your functions	255
8.2. Utilizing your UDFs in transformations	256
8.3. Including functions in your VDB	257
8.4. Using legacy UDF models	257
9. Editing Models and Projects	261
9.1. Rename A Model	261
9.2. Move Model	263
9.3. Save Copy of Model	264
9.4. Clone Project	266
10. Managing VDBs	269
10.1. Creating a VDB	269
10.2. Editing a VDB	270
10.3. Test a VDB	270
10.4. Multi-source Binding Support	270
10.5. UDF support	271
10.6. Reusing VDBs	272
10.7. Security and Data Access	277
10.8. Working with Dynamic VDBs	281
10.8.1. Generate VDB Archive	281
10.8.2. Generate Dynamic VDB	283
11. Testing Your Models	287
11.1. Manage Connection Profiles	287
11.1.1. Set Connection Profile for Source Model	287
11.1.2. View Connection Profile for Source Model	287
11.1.3. Remove Connection Profile from Source Model	289
11.2. Previewing Data For a Model	289
11.2.1. Preview Relational Table or View	293
11.2.2. Preview Relational Table With Access Pattern	293
11.2.3. Preview Relational Procedure	294
11.2.4. Preview Web Service Operation	295
11.2.5. Sample SQL Results for Preview Data	296
11.2.6. Execution Plans	297
11.3. Testing With Your VDB	298

11.3.1. Creating Data Sources	298
11.3.2. Execute VDB from Model Explorer	300
11.3.3. Deploy VDB from Model Explorer	301
11.3.4. Executing a Deployed VDB	302
12. Materialization Support	309
12.1. Materialization Extension Properties	309
12.2. Create POJO	311
12.3. Materialize a View	312
12.4. JDG Connection Profile	316
13. Searching	319
13.1. Finding Model Objects	319
13.2. Search Transformation SQL	321
13.3. Search Models Via Metadata Properties	323
A. Supported Data Sources	325
B. Designer Metadata Usage Requirements In Teiid Runtime	327
C. User Preferences	339
C.1. Teiid Designer Preferences	339
C.1.1. Diagram Preferences	341
C.1.2. Editor Preferences	342
C.1.3. Validation Preferences	345
D. Teiid Designer Ui Reference	349
D.1. Teiid Designer Perspectives	349
D.1.1. Teiid Designer Perspective	349
D.1.2. Opening a Perspective	350
D.1.3. Further information	353
D.2. Teiid Designer Views	353
D.2.1. Model Explorer View	
D.2.2. Outline View	356
D.2.3. Server View	358
D.2.4. Connections View	365
D.2.5. Properties View	367
D.2.6. Description View	
D.2.7. Problems View	370
D.2.8. Search Results View	372
D.2.9. Datatype Hierarchy View	
D.2.10. Teiid Model Classes View	
D.2.11. System Catalog View	
D.2.12. SQL Reserved Words View	
D.2.13. Model Extension Definition Registry View (MED Registry View)	
D.2.14. Guides View	
D.2.15. Status View	
D.2.16. Cheat Sheets View	
D.3. Editors	
D.3.1. Model Editor	386

	D.3.2. VDB Editor	407
	D.3.3. Model Extension Definition Editor	412
D.4.	Teiid Designer Main Menu	415
	D.4.1. File Menu	416
	D.4.2. Edit Menu	419
	D.4.3. Refactor Menu	420
	D.4.4. Navigate Menu	421
	D.4.5. Search Menu	421
	D.4.6. Project Menu	423
	D.4.7. Metadata Menu	424
	D.4.8. Run Menu	424
	D.4.9. Window Menu	425
	D.4.10. Help Menu	426

Introduction

The Teiid Designer User's Guide provides detailed descriptions of Teiid Designer features and functionality.

1.1. What is Teiid Designer?

Teiid Designer is an Eclipse-based graphical modeling tool for modeling, analyzing, integrating and testing multiple data sources to produce Relational, XML and Web Service Views that expose your business data.

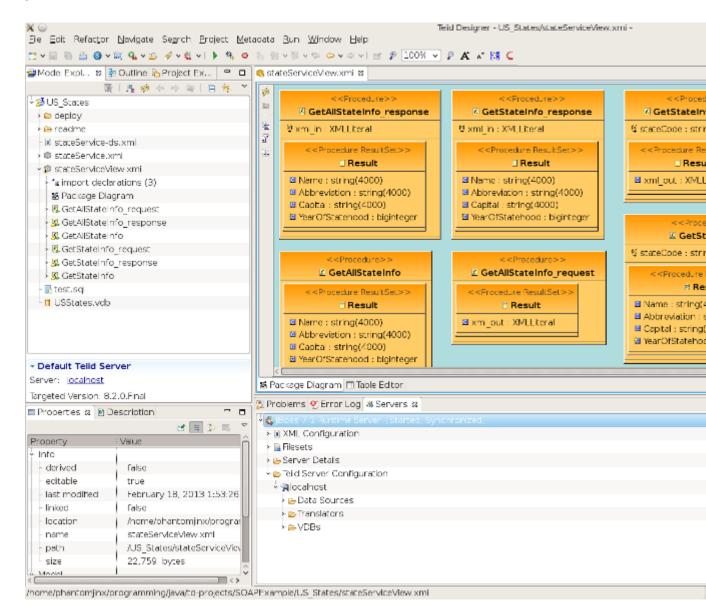


Figure 1.1. Teiid Designer

Why Use Teiid Designer?

Teiid Designer is a visual tool that enables rapid, model-driven definition, integration and testing of data services without programming. With Teiid Designer, not only do you map from data sources to target formats using a visual tool, but you can also:

- resolve semantic differences
- create virtual data structures at a physical or logical level
- use declarative interfaces to integrate, aggregate, and transform the data on its way from source to a target format which is compatible and optimized for consumption by your applications

This allows you to abstract the structure of the information you expose to and use in your applications from the underlying physical data structures. With Teiid Designer, data services are defined quickly, the resulting artifacts are easy to maintain and reuse, and all the valuable work and related metadata are saved for later reference.

You can use Teiid Designer to integrate multiple sources, and access them using the common data access standards:

- Web Services / SOAP / XML
- JDBC / SQL
- ODBC / SQL

1.2. Metadata Overview

1.2.1. What is Metadata

Metadata is data about data. A piece of metadata, called a meta object in the Teiid Designer, contains information about a specific information structure, irrespective of whatever individual data fields that may comprise that structure.

Let's use the example of a very basic database, an address book. Within your address book you certainly have a field or column for the ZIP code (or postal code number). Assuming that the address book services addresses within the United States, you can surmise the following about the column or field for the ZIP code:

- Named ZIPCode
- Numeric
- A string
- Nine characters long
- Located in the StreetAddress table

- Comprised of two parts: The first five digits represent the five ZIP code numbers, the final four represent the ZIP Plus Four digits if available, or 0000 if not
- Formatted only in integer numeric characters. Errors will result if formatted as 631410.00 or 6314q0000

This definition represents metadata about the ZIP code data in the address book database. It abstracts information from the database itself and becomes useful to describe the content of your enterprise information systems and to determine how a column in one enterprise information source relates to another, and how those two columns could be used together for a new purpose

You can think of this metadata in several contexts:

- What information does the metadata contain? (see Section 1.2.2, "Business and Technical Metadata")
- What data does the metadata represent? (see Section 1.2.3, "Source and View Metadata")
- How will my organization use and manage this metadata? (see ???)

Editing Metadata vs. Editing Data

The Teiid Designer helps you to create and describe an abstract graphic representation of your data structure of your data in the original data sources. It also describes whether those data sources are composed of Relational databases, text files, data streams, legacy database systems, or some other information type.

The Teiid Designer allows you to create, edit, and link these graphically-represented meta objects that are really a description of your data, and not the data itself.

So when this documentation describes the process of creating, deleting, or editing these meta objects, **remember** that you are not, in fact, modifying the underlying data.

Metadata Models

A **metadata model** represents a collection of metadata information that describes a complete structure of data.

In a previous example we described the field ZIPCode as a **metadata object** in an address book database. This **meta object** represents a single distinct bit of metadata information. We alluded to its parent table, StreetAddress. These **meta objects**, and others that would describe the other tables and columns within the database, would all combine to form a **Source Metadata** model for whichever enterprise information system hosts all the objects.

You can have **Source Models** within your collection of **metadata models** These model physical data storage locations. You can also have **View Models**, which model the business view of the data. Each contains one type of metadata or another. For more information about difference between Source and View metadata, (see *Section 1.2.3, "Source and View Metadata"*).



Note

For detailed information about creating models from your metadata, see Section 1.3, "It's all in the Modeling..."

1.2.2. Business and Technical Metadata

Metadata can include different types of information about a piece of data.

- **Technical metadata** describes the information required to access the data, such as where the data resides or the structure of the data in its native environment.
- **Business metadata** details other information about the data, such as keywords related to the meta object or notes about the meta object.



Note

The terms **technical and business metadata**, refer to the content of the metadata, namely what type of information is contained in the metadata. Don't confuse these with the terms "physical" and "view" metadata that indicate what the metadata represents. For more information, (see Section 1.2.3, "Source and View Metadata').

Technical Metadata

Technical metadata represents information that describes how to access the data in its original native data storage. Technical metadata includes things such as datatype, the name of the data in the enterprise information system, and other information that describes the way the native enterprise information system identifies the meta object

Using our example of an address book database, the following represent the technical metadata we know about the ZIP code column:

- Named ZIPCode
- Nine characters long
- A string
- Located in the StreetAddress table
- Uses SQL Query Language

These bits of information describe the data and information required to access and process the data in the enterprise information system.

Business Metadata

Business metadata represents additional information about a piece of data, not necessarily related to its physical storage in the enterprise information system or data access requirements. It can also represent descriptions, business rules, and other additional information about a piece of data.

Continuing with our example of the ZIP Code column in the address book database, the following represents business metadata we may know about the ZIP code:

- The first five characters represent the five ZIP code numbers, the final four represent the ZIP Plus Four digits if available, or 0000 if not
- The application used to populate this field in the database strictly enforces the integrity of the data format

Although the first might seem technical, it does not directly relate to the physical storage of the data. It represents a business rule applied to the contents of the column, not the contents themselves.

The second, of course, represents some business information about the way the column was populated. This information, although useful to associate with our definition of the column, does not reflect the physical storage of the data.

1.2.3. Source and View Metadata

In addition to the distinction between business and technical metadata, you should know the difference between **Source Metadata and View Metadata**.

Source and View metadata refer to what the metadata represents, not its content.

Source Metadata directly represents metadata for an enterprise information system and captures exactly where and how the data is maintained. Source Metadata sounds similar to technical metadata, but Source Metadata can contain both technical and business metadata. When you model Source Metadata, you are modeling the data that your enterprise information systems contain.

View Metadata, on the other hand, represent tailored views that **transform** the **Source Metadata** into the terminology and domain of different applications. **View Metadata**, too, can contain both technical and business metadata. When you model **View Metadata**, you're modeling the data as your applications (and your enterprise) ultimately use it.

Modeling Your Source Metadata

When you model the **Source Metadata** within your enterprise information systems, you capture some detailed information, including:

- Identification of datatype
- Storage formats
- Constraints
- Source-specific locations and names

The **Source Metadata** captures this detailed technical metadata to provide a map of the data, the location of the data, and how you access it.

This collection of **Source Metadata** comprises a direct mapping of the information sources within your enterprise. If you use the Teiid Designer Server for information integration, this technical metadata plays an integral part in query resolution.

For example, our ZIPCode column and its parent table StreetAddress map directly to fields within our hypothetical address book database.

To extend our example, we might have a second source of information, a comma-separated text file provided by a marketing research vendor. This text file can supply additional demographic information based upon address or ZIP code. This text file would represent another Enterprise Information System (EIS), and the meta objects in its Source Model would describe each comma-separated value.

Modeling Your View Metadata

When you create **View Metadata**, you are not describing the nature of your physical data storage. Instead, you describe the way your enterprise uses the information in its day-to-day operations.

View Metadata derives its classes and attributes from other metadata. You can derive View Metadata from Source Metadata that describes the ultimate sources for the metadata or even from other View Metadata. However, when you model View Metadata, you create special "views" on your existing enterprise information systems that you can tailor to your business use or application expectations. This View Metadata offers many benefits:

- You can expose only the information relevant to an application. The application uses this **View Metadata** to resolve its queries to the ultimate physical data storage.
- You can add content to existing applications that require different views of the data by adding the **View Metadata** to the existing **View Metadata** that application uses. You save time and effort since you do not have to create new models nor modify your existing applications.
- Your applications do not need to refer to specific physical enterprise information systems, offering flexibility and interchangeability. As you change sources for information, you do not have to change your end applications.
- The **View Metadata** models document the various ways your enterprise uses the information and the different terminology that refers to that information. They do so in a central location.

Our example enterprise information sources, the address book database, and the vendor-supplied comma-delimited text file, reside in two different native storage formats and therefore have two Source Metadata models. However, they can represent one business need: a pool of addresses for a mass mailing.

By creating a **View Metadata** model, we could accurately show that this single View Table, the AddressPool, contains information from the two enterprise information systems. The **View Metadata** model not only shows from where it gets the information, but also the SQL operations it performs to select its information from its source models.

This **View Metadata** can not only reflect and describe how your organization uses that information, but, if your enterprise uses the Teiid Designer Server, your applications can use the **View Metadata** to resolve queries.

To create this **View Metadata**, you create a view and define a **transformation** for that view, a special query that enables you to select information from the source (or even other view) metadata models. For more information, see "*Section 6.3.1, "Transformation Editor"*."

Metadata Transformations

By modeling View Metadata, you can illustrate the business view of your enterprise information sources. View Metadata models not only describe that business view, but also illustrate how the meta objects within the View Metadata models derive their information from other metadata models.

Let's return to the example of our address book database and the vendor's comma-separated list. We want to generate the View Metadata model, Address Pool, from these enterprise information systems.

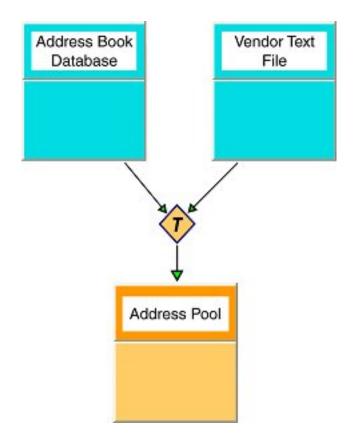


Figure 1.2. Data Flow for View Transformations

The transformation that joins these metadata models to create the virtual Address Pool metadata model contains a SQL query, called a union, that determines what information to draw from the source metadata and what to do with it.

The resulting Address Pool contains not only the address information from our Address Book database, but also that from our vendor-supplied text file.

SQL in Transformations

Transformations contain SQL queries that SELECT the appropriate attributes from the information sources.

For example, from the sources the transformation could select relevant address columns, including first name, last name, street address, city, state, and ZIP code. Although the metadata models could contain other columns and tables, such as phone number, fax number, e-mail address, and Web URL, the transformation acts as a filter and populates the Address Pool metadata model with only the data essential to building our Address Pool.

You can add other SQL logic to the transformation query to transform the data information. For example, the address book database uses a nine-character string that represents the ZIP Plus Four. The transformation could perform any SQL-supported logic upon the ZIPCode column to substring this information into the format we want for the Address Pool View metadata model.

Mapping XML Transformations

When you model View Metadata, you can also create a View XML Document model. This View Document lets you select information from within your other data sources, just like a regular View Metadata model, but you can also map the results to tags within an XML document.

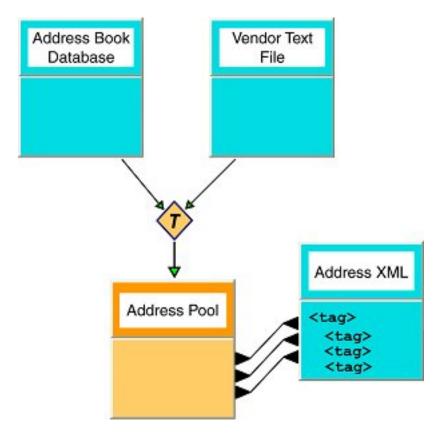


Figure 1.3. Data Flow for XML Transformations

In this example, the Address Pool View Metadata model still selects its information from the Address Book Database and the Vendor Text File, but it also maps the resulting columns into tags in the Address XML document.

1.3. It's all in the Modeling...

1.3.1. What Are Models?

A model is a representation of a set of information constructs. A familiar model is the relational model, which defines tables composed of columns and containing records of data. Another familiar model is the XML model, which defines hierarchical data sets.

In Teiid Designer, models are used to define the entities, and relationships between those entities, required to fully define the integration of information sets so that they may be accessed in a uniform manner, using a single API and access protocol. The file extension used for these models is .xmi (Example: NorthwindOracle.xmi) which adheres to the XMI syntax defined by the OMG.

Below is an example of the partial contents of a model file.

```
<?xml version="1.0" encoding="ASCII"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xm</pre>
  <mmcore:ModelAnnotation xmi:uuid="mmuuid:b0355f00-413b-1079-90</pre>
    <modelImports xmi:uuid="mmuuid:2d815780-4140-1079-9d18-8acf.</pre>
    <modelImports xmi:uuid="mmuuid:2e663940-4140-1079-9d18-8acf</pre>
  </mmcore:ModelAnnotation>
  <relational:BaseTable xmi:uuid="mmuuid:b20e64c0-413b-1079-9d1;</pre>
    <columns xmi:uuid="mmuuid:bb5ac3c0-413b-1079-9d18-8acf4a712"
      <type href="http://www.w3.org/2001/XMLSchema#long"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c0-413b-1079-9d18-8acf4a712"
      <type href="http://www.w3.org/2001/XMLSchema#string"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c1-413b-1079-9d18-8acf4a712"
      <type href="http://www.w3.org/2001/XMLSchema#string"/>
    </columns>
    <columns xmi:uuid="mmuuid:bc4ee7c2-413b-1079-9d18-8acf4a712"
      <type href="http://www.metamatrix.com/metamodels/SimpleDay
    </columns>
    <primaryKey xmi:uuid="mmuuid:d481f940-413b-1079-9d18-8acf4a"</pre>
  </relational:BaseTable>
```

Figure 1.4. Sample Model File



G

Note

Model files should never be modified "by hand". While it is possible to do so, there is the possibility that you may corrupt the file such that it cannot be used within Teiid Designer system.

The fundamental models in Teiid Designer define the structural and data characteristics of the information contained in data sources. These are referred to as source models (represented by

). Teild Designer uses the information in source models to federate the information in multiple sources, so that from a user's viewpoint these all appear to be in a single source.

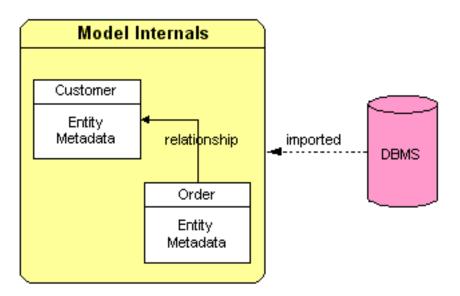


Figure 1.5. Model Internals

In addition to source models, Teiid Designer provides the ability define to а variety of view models(represented by TR

). These can be used to define a layer of abstraction above the physical (or source) layer, so that information can be presented to end users and consuming applications in business terms rather than as it is physically stored. Views are mapped to sources using transformations between models. These business views can be in a variety of forms:

- Relational Tables and Views
- XML
- Web services
- Functions

For full list of supported model types see Chapter 4, New Model Wizards

A third model type, logical, provides the ability to define models from a logical or structural perspective.

1.3.2. How is a Model Defined?

Models are defined using Teiid Designer in various ways:

- Created via importing source data characteristics. (see Chapter 5, Importers)
- Manual creation via Chapter 4, New Model Wizards
- Transforming or copying from one model into another (see Chapter 4, New Model Wizards options)
- · Various custom actions

1.3.3. Guiding through the process

To make the process of using Teiid Designer to build models more as easy as posssible, a guides view (*Section D.2.14, "Guides View*') has been introduced. It provides action sets which bring together the actions necessary to develop models for specific use-cases. Action sets are available for the following scenerios:

- Consume a SOAP Web Service
- Create a REST Web Service
- Model Flat File Source (a text file)
- Model JDBC Data Source
- Model Local XML File Source
- Model Remote XML File Source
- Modell Teiid Data Source (deployed on server)
- Teiid Server Actions

1.3.4. Targeting Your Teiid Submoduler

Like Teiid Designer, the Teiid runtime is under continuous development and as such multiple versions have been and are being released. Due to changes in both its code and the underlying JBoss server, the versions are not always backward compatible. Teiid Designer provides Teiid runtime validation for VDBs based on server versions. New models must be compatible with their targeted server version hence the correct server version must be selected prior to creating them.

To aid with selection of the correct server version, two changes have been made to Teiid Designer:

- A preference for the targeted server version that new models will be based on;
- The concept of the **default server** has been extended so that it will determine the targeted server version of new models (superceding the preference).

1.3.5. Model Classes and Types

Teiid Designer can be used to model a variety of classes of models. Each of these represent a conceptually different classification of models.

- **Relational** Model data that can be represented in table columns and records form. Relational models can represent structures found in relational databases, spreadsheets, text files, or simple Web services.
- XML Model that represents the basic structures of XML documents. These can be "backed" by XML Schemas. XML models represent nested structures, including recursive hierarchies.
- XML Schema W3C standard for formally defining the structure and constraints of XML documents, as well as the datatypes defining permissible values in XML documents.
- Web Services which define Web service interfaces, operations, and operation input and output parameters (in the form of XML Schemas).
- **Function** The Function metamodel supports the capability to provide user-defined functions, including binary source jars, to use in custom transformation SQL statements.

1.3.6. The Virtual Database

The critical artifact that **Teiid Designer** is intended to manage is the **VDB**, or **Virtual DataBase**. Through the Teiid server, VDB's behave like standard relational database schema which can be connected to, queried and updated based on how the VDB is configured. Since VDB's are just databases once they are deployed, they can be used as sources to other view model transformations. This allows creating and deploying re-usable or common VDB's in multiple layers depending on your business needs.

Starting in Teiid Designer 9.0, support was added for converting between archive (ZIP) VDBs and simple Dynamic VDB XML files. Dynamic VDBs provide users to define their source and view metadata via Teiid DDL statements. This capability allows some users to maintain a simpler version of their VDBs in a source control system. (See the *Section 10.8, "Working with Dynamic VDBs*" section for more details)

1.3.6.1. VDB Content and Structure

In Designer, the VDB file names use a "**.vdb**" file extension. VDBs are structurally just ZIP archive files containing 3 folders:

- META-INF
 - contains "vdb.xml" definition file
- runtime-inf
 - contains a binary INDEX file for each model included in your VDB. Note that these INDEX files
 represent the actual runtime metadata and is an optimized subset of data from your designtime metadata in your models.
- <project folder name>
 - contains of the models you will be adding in the VDB Editor (i.e. *.xmi and *.xsd files)

When deployed, the metadata is consumed by Teiid in order to create the necessary runtime metadata for your model definitions.

The vdb.xml file contains:

- VDB name, version, properties
- contained model information (name, translator name, connection info)
- · translator info
- · data role definitions for the referenced models
- import VDB references

The vdb.xml file example below highlights the basic model information.



Note

The VIRTUAL and PHYSICAL <model> elements containing property references to the INDEX files as well as the <source> element info for the PHYSICAL (aka source) model EU_CustomerAccounts.xmi.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vdb version="1" name="Financials">
        <model visible="true" type="VIRTUAL" name="US_CustomerAccounts" path="/
Financials/US_CustomerAccounts.xmi">
        <property value="4097408696" name="checksum"/>
        <property value="Relational" name="modelClass"/>
        <property value="false" name="builtIn"/>
        <property value="false" name="builtIn"/>
        <property value="1592679058.INDEX" name="indexName"/>
        <property value="/Financials/US_CustomerAccounts.xmi"
        name="imports"/>
```

Fortunately, Teiid Designer simplifies the management of your VDBs by providing a dedicated VDB Editor which maintains a consistent, valid vdb.xml file for you and assists in synchronizing your workspace models with any related models in your VDB. (See the Section D.3.2, "VDB Editor" section)

1.3.7. Model Validation

Models must be in a valid state in order to be used for data access. Validation of a single model means that it must be in a self-consistent and complete state, meaning that there are no "missing pieces" and no references to non-existent entities. Validation of multiple models checks that all inter-model dependencies are present and resolvable.

Modle and VDB validation is scoped to a model project.

Models must always be validated when they are deployed in a VDB for data access purposes.

Teiid Designer will automatically validate all models whenever they are saved.



Note

The "Project > Build Automatically" menu option must be checked. When editing models, the editor tabs will display a "*" to indicate that the model has unsaved changes.

1.3.8. Testing Your Models

Designing and working with data is often much easier when you can see the information you're working with. The Teiid Designer's **Preview Data** feature makes this possible and allows you to instantly preview the information described by any object, whether it's a physical table or a virtual view. In other words, you can test the views with actual data by simply selecting the table, view, procedure or XML document. The preview functionality insures that data access behavior in Teiid Designer will reliably match when the VDB is deployed to the Server. For more info on server management see *Chapter 3, Server Management*

Previewing information is a fast and easy way to sample the data. Of course, to run more complicated queries like what your application likely uses, simply execute the VDB in Teiid Designer and type in any query or SQL statement.

After creating your models, you can test them by using the **Preview Data** action

By selecting a desired table object and executing the action, the results of a simple query will be displayed in the Data Tools SQL Results view. This action is accessible throughout the Teiid Designer in various view toolbars and context menus.

Previewable objects include:

- Relational table or view, including tables involving access patterns.
- Relational procedure.
- Web Service operation.
- XML Document staging table.



Note

If attempting to preview a relational access pattern, a web service operation or a relational procedure with input parameters, a dialog will request values for required parameters.

1.3.9. Model Object Extensions

Teild Designer in conjunction with Teild provides an extensible framework to define custom properties for model objects over-and-above what is defined in the metamodel. These custom property values are added to your VDB and included in your runtime metadata. This additional metadata is available to use in your custom translators for both source query manipulation as well as adjusting your result set data being returned.

In the 7.6 release, Teiid Designer introduces a new **Model Extension Definition (MED)** framework that will replace the current EMF-based Model Extension metamodel in a later 8.0 release.

This new MED framework provides the following improvements:

- Eliminate need for separate EMF metamodel.
- Simpler approach including reduction of extendable metamodels and metamodel objects (Relational, Web Services, XML Document, User Defined Functions) and replacing EMF terminology with basic object types.

- · Allows metamodels to be extended by multiple MEDs
- · MEDs are stored in models so no added dependency needed in VDB

Also see: Section 6.4, "Managing Model Object Extensions" and Section D.3.3, "Model Extension Definition Editor".

1.3.9.1. Model Extension Definition (MED)

The purpose of a MED is to define one or more sets of extension properties. Each set of extension properties pertains to one model object type (or metaclass). Each MED consists of the following:

- Namespace Prefix a unique identifier. Typically only a small number of letters and can be used as an abbreviation for the namespace URI.
- Namespace URI a unique URI.
- Extended Metamodel URI (Model Class) the metamodel URI that is being extended. Each
 metamodel URI also has model class and that is typically what is shown in the Designer. The
 model classes supported for extension are: Relational, Web Service, XML Document, and
 Function.
- Version (currently not being used)
- **Description** an optional description or purpose.
- Extended Model Object Types (Metaclasses) a set of model object types, or metaclasses, that have extension properties defined.
- Properties the extension property definitions grouped by model object type.

A **MED** file is an XML file with an extension of "mxd." A MED schema file (see attached modelExtension.xsd file) is used to validate a MED file. Here is a sample MED file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<modelExtension xmlns:p="http://org.teiid.modelExtension/2011"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
metamodelUri="http://www.metamatrix.com/metamodels/Relational"
namespacePrefix="mymodelextension"
namespaceUri="org.my.extension.mymodelextension"
version="1"
xsi:schemaLocation="http://org.teiid.modelExtension/2011
modelExtension.xsd"
xmlns="http://org.teiid.modelExtension/2011">
<p:description>This is my model extension/2011">
<p:description>This is my model extension/2011">
<p:extendedMetaclass
name="com.metamatrix.metamodels.relational.impl.BaseTableImpl">
sp:property advanced="false" index="true" masked="false"
```

The *MED Registry* is where the MEDs used by Designer are stored. MED files can be edited by opening the .mxd file in the *Section D.3.3, "Model Extension Definition Editor*".

1.3.9.2. Model Extension Definition Registry (MED Registry)

A MED registry keeps track of all the MEDs that are registered in a workspace. Only registered MEDs can be used to extend a model. There are 2 different types of MEDs stored in the registry:

- Built-In MED these are registered during Designer installation. These MEDs cannot be updated or unregistered by the user.
- User-Defined MED these are created by the user. These MEDs can be updated, registered, and unregistered by the user.

The MED Registry state is persisted and is restored each time a new session is started.

Dive Right In!

We are going to dive right into a couple examples of common tasks in this section. These examples will give you a quick introduction to the capabilities that are built into Designer to assist you with common design tasks. Specifically, we will introduce the following concepts:

• Targeting the Teiid Server

The Teiid Server is the destination for Designer's modelling. It is essential to define the correct server version that models will be deployed to. This is achieved either by setting the server version preference or defining a teiid server in the Servers View.

Guides

The **Guides View** is a good starting point for many common modeling tasks. The view includes categorized Modeling Actions and also links to Cheat Sheets for common tasks. The categorized Modeling Actions simply group together all of the actions that you'll need to accomplish a task. You can launch the actions directly from the Guides view, rather than hunting through the various Designer menus.

Cheat Sheets

The **Cheat Sheets** go beyond even the categorized Action Sets, and walk you step-by-step through some common tasks. At each step, the data entered in the previous step is carried through the process when possible.

After seeing the Guides and Cheat Sheets in action, subsequent chapters will offer detailed explanations of the various concepts and actions.

2.1. Targeting the Teiid Server

In this section, the setting of the teild server version is demonstrated. This can be achieved by either setting a preference or by defining a teild server.

2.1.1. Server Version Preference

The default server version preference allows the target server version to be changed without actually having to define a teild server in Designer. The preference's list of possible values is determined by which teild runtime client plugins have been installed into the application.

Preferences	
type filter text	Teiid Designer 🔶 🔻 🖒 🔻
 General 	Specify values for general Designer preferences.
Ant	
 Data Management 	Preview Data/VDB Execution
Forge	
FreeMarker Editor	Enable auto-creating of a source model's data source on Teiid Instance (VDB Execution)
▶ Help	
HQL editor	Enable auto-toggling children of the checked model object
Install/Update	
▶ Java	👿 Enable Preview
Java EE	
 Java Persistence 	Generate Teiid Query Plan on execution
 JavaScript 	
 JBoss Tools 	Enable Preview Teiid Cleanup
JDBC Data Set	
JDBC Data Source	
Maven	120 Teiid Connection Importer Timeout (sec)
Plug-in Development	
Project Archives	Teiid Designer
Remote Systems	Always open editor without prompting
 Run/Debug 	
Server	Open Designer perspective when a model is opened
▶ Team	🔿 Always open 🕥 Never open 📵 Prompt
Teiid Designer	
Terminal	
TestNG	Check and update imports during save
Validation	8.5.x Default Teiid Instance Version (applicable if no teiid instance defined)
▶ Web	· · · · · · · · · · · · · · · · · · ·
 Web Services 	
▶ XML	Restore <u>D</u> efaults <u>A</u> pp
?	Cancel

Figure 2.1. Default Server Version Preference

2.1.2. Defining a Teiid Server

The defining of a Teiid Server is encouraged since it allows for models to be previewed and their deployment tested. There is no limit to the number of servers that can be defined. However, the default server will always be used for previewing and deployment, unless using the context menu actions in the *Section D.2.3, "Server View"*.

The Guides View provides the following Teiid Server actions.

😭 Guides 🕱	
Modeling Actions 🔻	
✓ Action Sets	
Execute selected action	
Teiid	÷
 Configure new JBoss Server Edit JBoss / Teiid Instance Properties Set the Default JBoss / Teiid Instance Refresh Teiid Instance 	

Figure 2.2. Teiid Server Category in the Guides View

The *New Teiid Server* action will display the wizard outlined in ??? and steps through the process of creating both the Teiid Server and its parent JBoss server in the Server View.

Should more than one Teiid Server be defined in the Server View then the Set the Default Server action allows for the default server to be changed appropriately. If a Teiid Server is currently selected in the Server View then this will be selected as the default server. However, should nothing be selected then a dialog will be displayed inviting the user to choose which server they wish to select.

Note

i

The version of the defined Teiid server always takes precedence over the Section 2.1.1, "Server Version Preference"

2.1.3. Server Version Status Panel

Whether the server version preference has been modified or a server defined, the server and server target version will be updated in the default server status panel. This will always reflect the current server version being targeted and the server being used to preview or deploy against.

▼ Default Server	Change
Name: 🗿 EAP6.1+Teiid8.7FINAL	
Teiid Version: 🕋 8.7.0	

Figure 2.3. Default Server Status Panel

2.2. Guide Example

In this section, the **Guides View** is demonstrated in detail by walking through a simple example. For this example, we will follow the **Model JDBC Source** Action Set. The actions appear in the following order:

- 1. Define Teiid Model Project
- 2. Create JDBC connection
- 3. Create source model for JDBC data source
- 4. Preview Data
- 5. Define VDB
- 6. Execute VDB

The action names are self explanatory. We will create a new "Model Project" in the workspace, then define our connection properties to a MySQL database. We will then connect to the database and import the 'metadata', creating a source model in Designer. Next we will 'preview' the database contents. Finally we will define a 'VDB' and then deploy it to a running Teiid Server to execute.

2.2.1. Model a Relational (via JDBC) Source

This section shows how to Model a Relatioanl Source, using the Guide View action set. We will connect to a MySQL database for this example, but you can use the same process to connect to any supported database.

1. Open Guides View

To open the **Teiid Designer's Guides** view, select the main menu's **Window > Show View > Other...** and select the **Teiid Designer > Guides** view in the dialog.

The Guides view is shown below, with the Model JDBC Source Action Set selected:

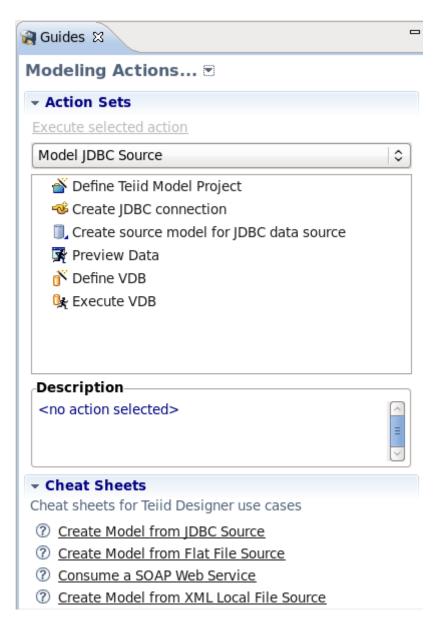


Figure 2.4. Guides View

2. Define Teiid Model Project

The **Define Teild Model Project** action launches the **New Model Project Wizard**. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). The wizard is launched as shown below:

New Model Project	
New Model Project	
Create a new model project.	
Project name: MyProject	
✓ Use default location	
Location: //home/mdrillin/Runtime-Workspaces/7_7_x/MyProject	Browse
Working sets	
Add project to working sets	
Working sets:	Select
(?) < Back Next > Cancel	Finish

Figure 2.5. New Project Wizard

Enter a project name, e.g. 'MyProject' for the name. Then click **Next**. The next page of the wizard is shown below:

E	New Model Project						
	Model Project Options Click Finish to create folders for your new project						
Create Fo	olders						
🗹 Name	sources						
🗹 Name	views						
🗆 Name	schemas						
🗆 Name	web_services						
🗆 Name	functions						
🗆 Name	extensions						
?	< Back Next > Cancel Fir	nish					

Figure 2.6. New Project Folders

Under 'Create Folders', de-select 'schemas' and 'web_services' - we won't need them for this example. Now, click **Finish** to exit the wizard. The project has now been created - your Model Explorer view should like like this:

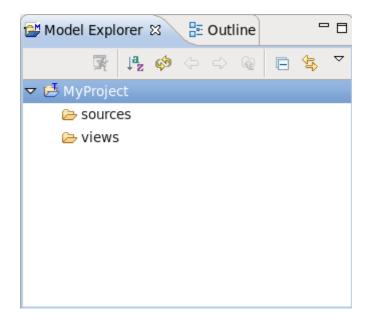


Figure 2.7. Model Explorer

3. Create JDBC connection

The **Create JDBC connection** action will create the 'Connection profile' for your database. The connection profile defines the properties and driver to be used when connecting to the database. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). The wizard is launched as shown below:

3	New Cor	nnection	Profi	le	
Connection Profile					\rightarrow
Create a MySQL connec	tion profile	2			
Connection Profile Type	5:				
type filter text					
Ingres					
MaxDB					
III ModeShape					=
😫 MySQL					_
Oracle					
Name:					
TestMySQL					
Description (optional):					
Sector 2 - Back	k	Next >		Cancel	Finish

Figure 2.8. Connection Profile Name and Type

Select the type of database that you are connecting to (e.g. MySQL), and enter a name for the connection profile, e.g. 'TestMySQL'. Click **Next**.

New Connection Profile	×
Specify a Driver and Connection Details	
Select a driver from the drop-down and provide login details for the connection.	
Drivers: MySQL JDBC Driver	6
Properties	
General Optional	
D <u>a</u> tabase: database	
URL: jdbc:mysql://localhost:3306/database	
User name: root	
Pass <u>w</u> ord:	
□ Sa <u>v</u> e password	
✓ <u>C</u> onnect when the wizard completes <u>T</u> est Connection	on
Connect every time the workbench is started	
? < Back Next > Cancel Finish	

Figure 2.9. Connection Profile properties

Now, select the driver and enter the login properties for your database. Click **Finish** to complete the profile creation.

4. Create source model for JDBC data source

The **Create source model for JDBC data source** action will now utilitze the **Connection profile** that you just created, to import the metadata from the database to create your Teiid Source Model. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). The wizard is launched as shown below:

JDBC Metadata Processor JDBC (default) Properties Driver: MySQL JDBC Driver URL: jdbc:mysql://localhost:3306/Accounts User Name: root Password: ******		Import Database via JDBC	
model. Connection Profile TestMySQL New E JDBC Metadata Processor JDBC (default) Properties Driver: MySQL JDBC Driver URL: jdbc:mysql://localhost:3306/Accounts User Name: root Password: ******	mport Datal	base via JDBC	
TestMySQL		C source configuration for the database to be imported into a relational	
JDBC Metadata Processor JDBC (default) Properties Driver: MySQL JDBC Driver URL: jdbc:mysql://localhost:3306/Accounts User Name: root Password: ******	Connection	Profile	
JDBC (default) Properties Driver: MySQL JDBC Driver URL: jdbc:mysql://localhost:3306/Accounts User Name: root Password: ******	TestMySQL	\$ New	Edit
Properties Driver: MySQL JDBC Driver URL: jdbc:mysql://localhost:3306/Accounts User Name: root Password: ******	JDBC Metad	ata Processor	
Driver: MySQL JDBC Driver URL: jdbc:mysql://localhost:3306/Accounts User Name: root Password: ******	JDBC (defau	lt)	\$
URL: jdbc:mysql://localhost:3306/Accounts User Name: root Password: ******	Properties		
User Name: root Password: ******	Driver:	MySQL JDBC Driver	
Password: *****	URL:	jdbc:mysql://localhost:3306/Accounts	
	User Name:	root	
	Password:	*****	
Cancel	?	< Back Next > Cancel Fini	ish

Figure 2.10. Select Connection Profile

On this page, select the 'TestMySQL' Connection profile that you created in the previous step. Click **Next**.

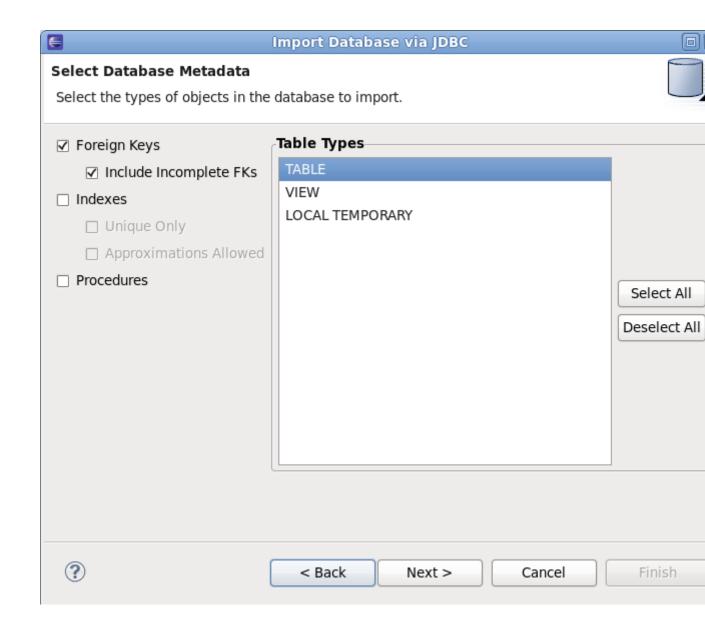


Figure 2.11. Select Database Metadata

On this page, select the database metadata that you want to import. When finished, click Next.

	Import Databa	ise via JDBC		
Select Database Objects				7
Press the "Next >" button to co	ntinue.			
📄 TestMySQL 🛛 🚸	Columns Indexes	Primary Key Imported F	oreign »3	
▷ information_schem △	TABLE_CAT TABL	E_SCHEM TABLE_NAME	COLUMN_NAME DATA	Α_'
∇ \overrightarrow{V} Accounts	Accounts	ACCOUNT	ACCOUNT_ID	
▼ ▼ TABLE	Accounts	ACCOUNT	SSN	
ACCOUNT	Accounts	ACCOUNT	STATUS	
CUSTOMER	Accounts	ACCOUNT	TYPE	
✓ HOLDINGS	Accounts	ACCOUNT	DATEOPENED	
PRODUCT	Accounts	ACCOUNT	DATECLOSED	
<	<			>
4 table object(s) selected to im	port.			
Deselect All				
Deselect All				
🗆 shaw o ha sala ta dishawa	_			
Show Only Selected Schema	S			
?	< Back	Next > Ca	ncel Finish	

Figure 2.12. Select Database Objects

On this page, select the specific objects from the database that you want to import. When finished, click **Next**.

E	Import Database via JDBC	OX
Specify Impor	t Options	
Press the "Next	: >" button to continue or the "Finish" button to finish.	
	odel Definition	
Model Name:	TestMySQL.xmi	
Into Folder:	MyProject/sources	
🗆 Make targe	t a view model	
🗆 Update (if e	existing model selected)	
Model Object ☑ Use Fully Q	log For Fully Qualified Names t Names (Tables, Procedures, Columns, etc) qualified Names (Example: partssupplier.dbo.PARTS) se For All Characters tions	
Make	All Upper Case (Example: Suppliers > SUPPLIERS)	
O Make	All Lower Case (Example: SUPPLIERS > suppliers)	
(?)	< Back Next > Cancel	Finish

Figure 2.13. Import Options

Finally, choose the name for the model to be created (defaults to 'profileName'.xmi). The 'Into Folder' field defines the target location for your new model. Select the 'MyProject/sources' folder. Now, click **Finish**. The source model has now been created - your Model Explorer view should like like this:

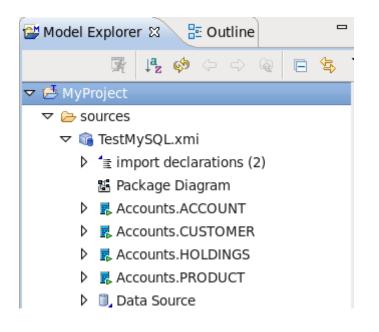


Figure 2.14. Model Explorer

5. Preview Data

All execution capabilities in Designer (Preview Data, VDB execution) require you to connect to a running Teiid Server. See *Chapter 3, Server Management* for instructions on establishing a Teiid Server connection. Once you are connected to a Teiid Server, you can proceed with the following steps.

The **Preview Data** action allows you to preview a sample of data rows from your source. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). In the dialog, select the source table you want to preview, as shown below:

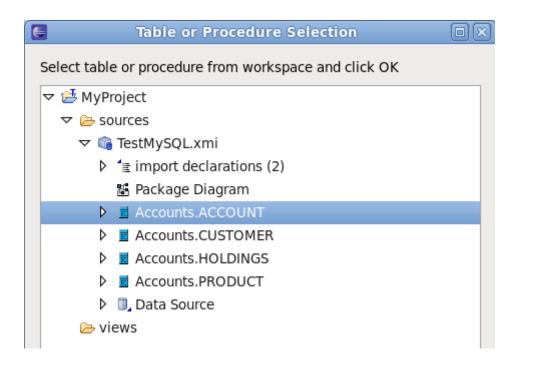


Figure 2.15. Select Preview Table

After selecting the table, click **OK**. Now, the preview results will be displayed:

Problems 🥺 Error Log 🔲 SQL Results 🕱 🛛 🛷 Teiid Execution Plan				
Type query expression here	Stat	tus Result1		
Status Operation Date Co		ACCOUNT_ID	SSN	STATUS
✓ Succe select * fro Jun 8, 2012 Trans	1	19980002	CST01002	Persona
	2	19980003	CST01003	Persona
	3	19980004	CST01004	Persona
	4	19980005	CST01005	Persona
	5	19980006	CST01006	Persona
	6	19990007	CST01007	Persona
	<u> </u>			
< III >	Tota	al 17 records shown		

Figure 2.16. Preview Results

6. Define VDB

The **Define VDB** action allows you to create a VDB (Virtual Database) artifact for deployment to a Teiid Server. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). The following dialog is displayed:

E	New VDB	
New VDB		
Press the "Fil	nish" button to finish.	
In Folder:	MyProject	
VDB Name:	myVDB	
Descriptio	n	
Selected M	lodels	
Add Remove	宿 TestMySQL.xmi	
?	Cancel	Finish

Figure 2.17. New VDB

In the dialog, select the target 'In Folder' location where the VDB will be placed. Enter a Name for the VDB, for example 'myVDB'. Finally, select the models that will be included in the VDB. When finished, click **Finish**. The VDB will be created in your Teild Model Project - as shown in the following figure.

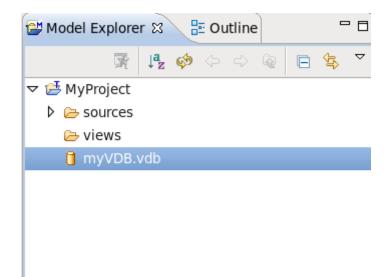


Figure 2.18. Model Explorer

7. Execute VDB

Finally, the **Execute VDB** action allows you to execute your VDB and run sample queries against it. In the Action Set list, double-click the action (or select it, then click 'Execute selected action'). In the dialog, select the VDB you want to execute, then click **OK**. The VDB will be deployed and executed, and the perpective will switch to the 'Database Development' perspective. You can now run queries against the VDB, as show in the following example:

📬 TestMySQL.xmi 🚺 myVDB.vdb		🖪 *SQL Scrapbook0 🛿		
Connection profile				
Type: Teiid_7.x		≎ <u>N</u> ame: m	nyVDB - localhost -	· Teiid Conne
SELECT * FROM Accounts.ACCOL	лт			
<				111
🔲 SQL Results 🕱 < 🖓 Teiid Executio	n Pla	an		
Type query expression here	Stat	us Result1		
Status Operation Date (ACCOUNT_ID	SSN	STATUS
✓ Succe select * fro Jun 8, 2012	1	19980002	CST01002	Personal
✓ Succe SELECT * FI Jun 8, 2012 I	2	19980003	CST01003	Personal
	2 3	19980003 19980004	CST01003 CST01004	Personal Personal

Figure 2.19. Execute VDB Example

2.3. Cheat Sheet Example

In this section, we introduce **Cheat Sheets** by walking through a simple example. For this example, we will follow the **Consume a SOAP Web Service** Cheat Sheet.

2.3.1. Consume a SOAP Web Service

This section shows how to consume a SOAP Web Service, using a Cheat Sheet. We will demonstrate connection to a publicly accessible web service. You can use this process as an example for modeling other web services

1. Open the Cheat Sheet

You can access the **Cheat Sheet** from the Designer Menu. From the Designer main menu, select **Window > Show View > Other...**, then select **Help > Cheat Sheets** in the dialog.

Alternately, you can access the **Cheat Sheet** from the **Guide View**. A sample Guide view is shown below, with the **Consume a SOAP Web Service** Action Set selected:

🍓 Guides ଅ	
Modeling Actions 🕾	
- Action Sets	
Execute selected action	
Consume SOAP Web Service	•
💣 Define Teiid Model Project	
🕸 Create Web Services (SOAP) connection	
🍓 Generate relational models from WSDL	
📴 Preview Data	
🔊 Define VDB	
🔍 Execute VDB	
Description	
Test a VDB by deploying to Teiid Server, connecting to it via JDBC and executing queries against it via Datatools' SQL Scrapbook	
- Cheat Sheets	
Cheat sheets for Teiid Designer use cases	
⑦ Create Model from JDBC Source	
⑦ Create Model from Flat File Source	
⑦ Consume a SOAP Web Service	
⑦ Create Model from XML Local File Source	

Figure 2.20. Guides View

To open the **Cheat Sheet** from the Guide View, expand the Cheat Sheet section in the lower portion of the Guide View, then select the **Consume a SOAP Web Service** link.

2. Begin the Cheat Sheet

The Consume a SOAP Web Service Cheat Sheet is shown below:

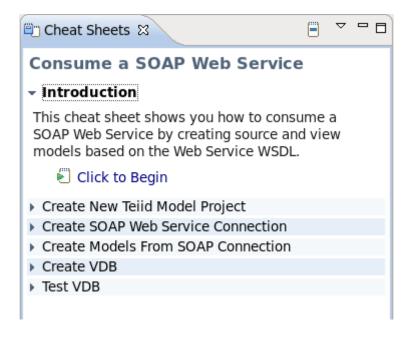


Figure 2.21. Consume SOAP Web Service Cheat Sheet

To start the Cheat Sheet process, expand the **Introduction** section, then select **Click to Begin**. The **Create New Teild Model Project** section opens, as shown.

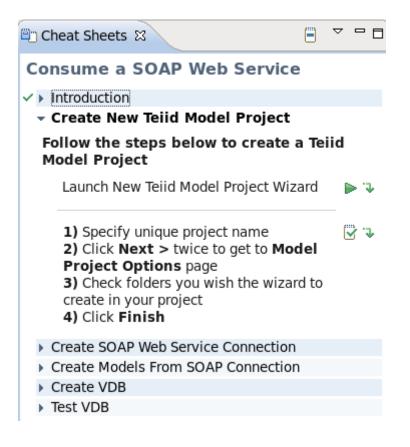


Figure 2.22. Create Model Project



Note

Each section of the sheet has basic instructions outlining what to do at each step.

Click

▶ 7

next to Launch New Teiid Model Project Wizard to launch the 'New Project' wizard.

Follow the wizard to create a new Model Project. For this example, we will use **SOAPProj** for our project name. On the second page of the wizard, select the 'sources' and 'views' folders. Click **Finish**. The new project is created.

In the Cheat Sheet, you can advance to the next step - once the wizard has completed. Click

ت 🖸

to advance to the next step.

3. Create SOAP Web Service Connection

This section of the Cheat Sheet provides instructions for creating a connection profile for the SOAP Web Service, as shown below:

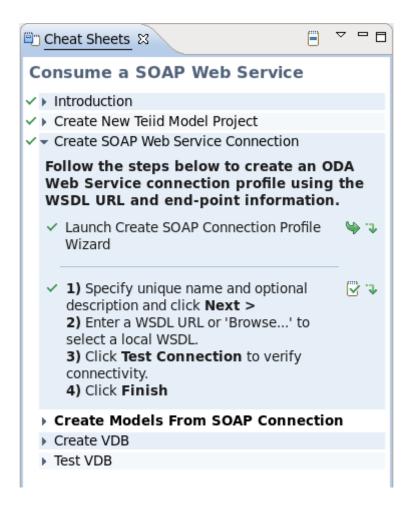


Figure 2.23. Create SOAP Connection Profile

Click

▶ 🤉

next to Launch Create SOAP Connection Profile Wizard to launch the wizard. The first page of the wizard is shown below:

New Connection Profile	
New Connection Profile	\diamond
Create a SOAP Web Service connection profile.	
Connection Profile Types:	
type filter text	
A Web Services Data Source (SOAP)	
Name:	
CountryInfoConn	
Description (optional):	
? < Back Next > Cancel Finish	

Figure 2.24. Create SOAP Connection Profile

The **Web Services Data Source (SOAP)** profile type will be selected. Enter **CountryInfoConn** for the profile name, then click **Next**. The next page of the wizard is shown below:

E	New connection profile
Web Service Connection Click Next or Finish	Properties
Properties	
Profile Name	CountryInfoConn
Profile Description	
Connection URL or File Path	http://www.oorsprong.org/websamples.countryinfo/CountryInfoService
	Workspace
Description	
	nection profile provides the hook to inject the 'Endpoint' property into Set Connection Profile' action. This 'Endpoint' value is required by the '
(see 'Import > Teiid Desig	ner > WSDL File or URL >> Source and View Model (SOAP))
	Ī
?	< Back Next > Cancel

Figure 2.25. SOAP Connection Properties

The connection profile properties are entered on this page. Click on the URL... button, then enter the following URL: http://www.oorsprong.org/websamples.countryinfo/ CountryInfoService.wso?WSDL

Select 'None' for SecurityType, then click **OK** to complete the wizard. In the Cheat Sheet, you can now continue - once the wizard has completed. Click

to advance to the next step.

4. Create Models from SOAP Connection

This section of the Cheat Sheet provides instructions for creating relational models using the previously-created connection profile for the SOAP Web Service, as shown below:

Consume a SOAP Web Service	
✓ ► Introduction	
✓ ► Create New Teiid Model Project	
Create SOAP Web Service Connection	
Create Models From SOAP Connection	
Follow the steps below to create models using the previously defined connection profile.	e
✓ Launch the Consume SOAP Web Service Wizard	₽ 🗣
 Select the connection profile, then press Validate WSDL. Select the Port and Service Mode as desired. Select one or more operations under Select the desired WSDL Operations. Click Next >. Select the location and name for the Source and View Models to be created. Click Next >. For each operation, define the request and response XML documents: Select the Operation Select the Request tab, then select desired element(s) under schema contents and Add to Element Info. Select the Response tab, then select desired element(s) under schema contents and Add to Column Info. Select the Wrapper Procedure tab to view the generated procedure SQL. When all operation(s) have been defined, click Finish. 	3.0
Create VDB	
Test VDB	

Figure 2.26. Create Models from SOAP Connection





next to **Launch the Consume SOAP Web Service Wizard** to launch the wizard. The first page of the wizard is shown below:

		Create Bala	tional Madal-6	rom Web Comice	
				rom Web Service	
		Model from Web Serv	/ice		
ress the "N	ext >"	button to continue.			
Connectio	n Pro	file			
CountryIn					0
WSDL UR	L or L	ocation:			
http://www	v.oorsp	prong.org/websamples.cou	ntryinfo/Country	InfoService.wso?WSDL	
Coloct Dort	Court	ta dafa Can da Cana			
Select Port	\geq	tryInfoServiceSoap		<u></u>	
Binding	SOAP	11	Service Mode	PAYLOAD	
Select the	e desii	red WSDL Operations-			
Select A	IL	Operation			
		🗹 🗅 CapitalCity			
Deselect	All	🗌 🕞 CountriesUsingCu	rrency		
		CountryCurrency			
		🗌 📄 CountrvFlag			
Selection [Details	:			
Operation:			Diadian		
bindir port:	ıg:	CountryInfoServiceSoap CountryInfoService			
servio	e:	CountryInfoService			
<			Ш		
					Connel
?			< Back	Next >	Cancel

Figure 2.27. Consume SOAP Wizard

For **Connection Profile**, select the previously-created **CountryInfoConn** profile. The available WSDL Operations will then be displayed under **Select the desired WSDL Operations**. Select only the first **CapitalCity** Operation for this example. Click **Next** to proceed to the next page, as shown below:

E	Create Relational Model from Web Service
Models D	efinition
All inputs	OK. Click 'Next>' to define custom procedures.
Source	Model Definition
Location	SOAPProj/sources
Name	CountryInfoService.xmi
Status	
	model CountryInfoService.xmi does not exist and will be created and contain the required procedure.
View Mo	odel Definition
Location	SOAPProj/views
Name	CountryInfoServiceView.xmi
Status	
View m procedu	odel CountryInfoServiceView.xmi does not exist and will be created and contain your gene ires.
Procedu	ire Generation Options
O User-	specified Procedures (recommended)
Define us and resp	ser-specified request and response procedures from your WSDL schema elements. This op onse.
🔿 Defau	Ilt Procedures
Generate	e default request and response procedures. A new procedure will be generated for each co
I	
?	< Back Next > Cancel

Figure 2.28. Consume SOAP Wizard

On the **Model Definition** page, the source and view model info section will be pre-filled. We will keep the names and location defaults for the source and view models. Click **Next** to proceed to the next page, as shown below:

E			Create Re	elatio	n <mark>al Model</mark> 1	from W	/eb Servic	e
	re Definitio umns are def		the response	e proce	edure result	set for t	he operatio	n: CapitalCity
Operati	ons							
Capital	City							
🗆 Over	write existin	g proced	ures for this	s opera	tion			
Request	Response	Wrapper	Procedure					
Genera	ted Procedur	e Name	CapitalCity	_reque	est			
BODY	HEADER							
Sche	ma Content	ts					Element	Info
	CapitalCity						Add	
	•••• sequence					_	Delete	e sCountryISOCo
		-	de				Up	
	🗟 strir	ng						
							Down	
Gener	ated SQL S	tateme	nt					
	E VIRTUAL P							
BEGIN								
SELI		NAME Ca	apitalCity, 🔉		MESPACES(D	DEFAULT	http://ww	w.oorsprong.org/
?					< Back		Next >	Cancel
							NEAL >	Cancer

Figure 2.29. Consume SOAP Wizard

On the **Procedure Definition** page, the **CapitalCity** Operation will be selected since it is the only one used for this example. On the **Request** tab, select the **sCountrylSOCode** element - then click the **Add** button. This will add the selected element to the request. Now select the **Response** tab, as shown below:

E Create Rela	tional Mod	el from Web Servic	e			
Procedure Definition Press the "Finish" button to finish.						
Operations						
CapitalCity						
Overwrite existing procedures for this ope	eration					
Request Response Wrapper Procedure						
Generated Procedure Name CapitalCity_res	ponse					
BODY HEADER						
Schema Contents	Column I	nfo				
	Root Path	/ns:CapitalCityRespor	ns:CapitalCityResponse			
	Add	Name	Ordinality	Data		
	Delete	CapitalCityResult		string		
a string			:	:		
	Down		1			
Generated SQL Statement						
CREATE VIRTUAL PROCEDURE BEGIN SELECT t.* FROM						
	< E	Back Next >	Can	cel		

Figure 2.30. Consume SOAP Wizard

On the **Response** tab, select the **Body** sub-tab. In the **Schema Contents**, select the **CapitalCityResult**, then click the **Add** button. This will add the selected element to the response.

Select the **Wrapper Procedure** tab to see the full Generated Procedure SQL, as shown below.

Create Relational Model from Web Service
Procedure Definition Press the "Finish" button to finish.
Operations
CapitalCity
Overwrite existing procedures for this operation
Request Response Wrapper Procedure
Generated Procedure Name
CapitalCity
Generated SQL Statement
CREATE VIRTUAL PROCEDURE BEGIN SELECT t.* FROM TABLE(EXEC CountryInfoServiceView.CapitalCity_request (CountryInfoServiceView.CapitalCity.sCountryISOCode) AS request, TABLE(EXEC CountryInfoService.invoke('SOAP11', null, REQUEST.xml_out, null)) AS response, TABLE(EXEC CountryInfoServiceView.CapitalCity_response(RESPONSE.result)) AS t; END
Output > Cancel

Figure 2.31. Consume SOAP Wizard

Click Finish to exit the wizard. In the Cheat Sheet, you can now continue. Click

to advance to the next step.

5. Create VDB

This section of the Cheat Sheet provides instructions for creating a VDB using the models that you created in the previous step. The Cheat Sheet section is shown below:



Figure 2.32. Create VDB

Click

```
▶ 🤉
```

next to Launch New VDB Wizard to launch the wizard. Follow the steps to create a VDB in your workspace. When complete, exit the wizard. In the Cheat Sheet, you can now continue. Click

🖸 🥆

to advance to the next step.

6. Test VDB

This	final	section	of	the	Cheat	Sheet	provides	inst	ructions
for	executing	the	VDB	create	d in	the	previous	step.	Click
<i>,</i>									

```
▶ 🤉
```

next to **Launch Execute VDB Dialog** to launch the wizard. Select the previously-created VDB to execute it.

Server Management

Teiid Designer is a design-time tool that allows setting up and testing deployable VDB artifacts. In order to deploy and test these VDB's a running *JBoss* [http://www.jboss.org] application server is required that contains an installed *Teiid* [https://www.jboss.org/teiid] submodule. This section describes how to set up, connect and maintain your servers and describes the various aspects of what features in Teiid Designer are enabled by this connection and how you can leverage these capabilities.

3.1. Setting up a Server

Teiid is installed as a component of JBoss hence connection to a Teiid Server requires the setting up and configuration of its parent JBoss Server. This is achieved using the Server View, see *Section D.2.3, "Server View"*, displayed as part of the Teiid Designer perspective. The detailed procedures for creating a JBoss Server configuration can be found in the documentation provided at *http://www.jboss.org/tools/docs/reference*. Thus, brief steps are only outlined here.

If no servers have been previously created then the Server View will display a new server hyperlink. To create a new JBoss Server configuration, click the hyperlink.

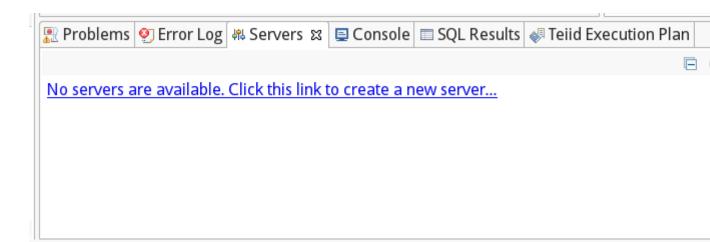


Figure 3.1. Server View with no created servers

Navigate through the wizard, configuring the details of the JBoss Server including its Runtime location, hostname and whether its externally managed. The final property determines whether the server is instantiated within the IDE or whether it is installed and started independently. Should the latter be the case then the Server View merely assumes connection to the independent server.

🖫 Problems 🥺 Error Log 🤼 Servers 🛱 📮 Console 🔲 SQL Results	Teiid Execution Plan
	E
🖶 localhost 7.1 server [Started, Synchronized]	
XML Configuration	
> 🗟 Filesets	
> 🗁 Server Details	
 Teiid Server Configuration 	
✓ ♀ mm://goshawk:9999	
> 🗁 Data Sources	
> 🗁 Translators	
> 🗁 VDBs	

Figure 3.2. Server View with a single server

As illustrated, the server has been installed with a Teiid Server and on clicking the green start button, Designer has successfully connected to the server, resulting in the display of the Teiid Server's configuration.

JBoss Tools provides an editor for the configuration of the JBoss Server. In addition, Designer provides an extra tab to this editor that displays the configuration of the Teiid Server. Only a few options can be modified since most of the configuration is determined by the parent JBoss Server. This editor can be displayed by double-clicking on any node in the JBoss Server tree.

EAP6.1+Teiid8.7FIN	IAL 23									Ē
Teiid Instance										
- Overview										
JBoss Server	EAP6.1+Teiid8.7FI	NAL								
Host	localhost									
Teiid Runtime Version	8.7.0									∇
 Administration Con 	inection									
Administration is perf	ormed via the jboss r	nanagement config	uration							
Port			9999							
Test Administration C	onnection									
➡ JDBC Connection										
Username			user							
Password			••••							
10530010										
Port			3100)						
JDBC connection uses	SSL									
Test JDBC Connection	L									
Overview Deployment	t Teiid Instance									
😽 Servers 🔀 🔐 Pi	roblems 📮 Consol	le 🔲 SQL Results	👰 Error Log	🖉 Teiid Execution Plar	n					Ē
				Ē	\$	蓉	0	Ď	<u>e</u> þ	
DV6_V4 [Stopped]]									
EAP6.1+Teiid8.6										
🔻 🖡 EAP6.1+Teiid8.7F		ronized]								
 X XML Configurat Filesets 	ION									
 Gerver Details 										
▼ 🗁 Teiid Instance C	Configuration									
▼ 👰 mm://localho										
Data Source	es									
Translators	5									
VDBs	_									
JBoss 5.1 Runtime	e Server [Stopped]									

Figure 3.3. Editor for Configuring the Teiid Server

3.1.1. Teiid Version Support

Teiid Designer is bundled with 4 versions of the Teiid Client Runtime, notably:

- Teiid Server 7.7 version installed on JBoss Server 5.1.
- Teiid Server 8.0, 8.1, 8.2 and 8.3 versions installed on JBoss Server 7.
- Teiid Server 8.4 thru 8.12 installed on JBoss Server 7.1.1, 7.2 and EAP 6.1 or greater.
- Teiid Server 9.0 installed on Wildfly 9+.

This allows multiple client runtime support from within a single installation of Teiid Designer. However, in order to facilitate this support it is necessary for a default server to be chosen by the user. The setting of the default server can be performed from the Teiid Server section of the Guides View or the Servers view, as described in *Section 2.1.2, "Defining a Teiid Server"*. Care should be taken to ensure that any new models are created against the correct version of server to ensure functionality and internal architecture is correct. Note that VDBs are validated against a server version and noted as such in the editor. Note that your VDBs will be validated against the current Teiid runtime client and that version will be persisted in your VDB.

New Model Wizards

Models are the primary resource used by Teiid Designer. Creating models can be accomplished by either directly importing existing metadata or by creating them using one of several **New Model** wizard options. This section describes these wizards in detail.

• The Teiid Designer currently supports the following types of models:

Section 4.1, "Creating New Relational Source Model"

G
 Section 4.2, "Creating New Relational View Model"

Section 4.3, "Creating XML Document View Model"

• 💽 Section 4.4, "Creating XML Schema Model"

• 📦

Section 4.5, "Creating Web Service View Model"

Use one of the following options to launch the New Model Wizard.

New Model Wizard

 Choose 	the	File	>	New	>	Metadata	Model	action
ð								

• Select a project or folder in the Section D.2.1, "Model Explorer View" and choose the same action in the right-click menu.

Select	the	New	button	on	the	main	toolbar
- 23							
and	select		the	Metadata		Model	action
ð							



Note

Model names are required to be unique within Designer. When specifying model names in new model wizards and dialogues error messages will be presented and you will prevented from entering an existing name.

	New Model Wizard	×
New Model	Wizard	
Specify model	name and options then create model file.	
Location:	NorthwindSales	Browse
Model Name:	Northwind	J
Model Class:	Relational	ļ
Model Type:	Source Model)
Colortanod		
	el builder (optional): File Translator Procedures	
-	Web Service Translator Procedures	
	n an existing model of the same model class	
?	< Back Next > Cancel	Finish

Figure 4.1. Import Wizard Selection Dialog

4.1. Creating New Relational Source Model

Create New Relational Source Model

- To create a new empty relational source model:
 - Step 1 Launch the New Model Wizard.

- Step 2 Specify a unique model name.
- Step 3 Select Relational option from Model Class drop-down menu.
- Step 4 Select Source Model from Model Type drop-down menu.
- Step 5 Click Finish.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty relational source model, the following builder options are available:
 - Copy from existing model of the same model class.

4.1.1. Generate File Translator Procedures

This builder option allows construction of a relational model containing one or more of the procedures required for accessing file-based data via a file translator.

- To create a new relational model containing file translator procedures, complete *Create New Relational Source Model* above and continue with these additional steps:
 - Step 5 Select the model builder labeled Generate File Translator Procedures and click Next >. The Generate File Translator Procedures dialog will be displayed.
 - Step 6 Check one ore more of the Available File Translator Procedures, then Click Finish

New Model Wizard	\otimes
Generate File Translator Procedures	
Generate default relational procedures compatible with Teiid File translator. Select desired procedures and 'Finish'	
Available File Translator Procedures	
getFiles('path/*.ext') return blob	
✓ getTextFiles('path/*.ext') return clob	
✓ saveFile('path', value) return void	
? < Back Next > Cancel	īnish

Figure 4.2. Generate File Translator Procedures Dialog

4.1.2. Generate Web Service Translator Procedures

This builder option allows construction of a relational model containing one or more of the procedures required for accessing web-service-based XML data via a web s translator.

- To create a new relational model containing web-service-based translator procedures, complete *Create New Relational Source Model* above and continue with these additional steps:
 - Step 5 Select the model builder labeled Generate Web Service Translator Procedures and click Next >. The Generate Web Service Translator Procedures dialog will be displayed.
 - Step 6 Check one ore more of the Available Web Services Translator Procedures, then Click Finish

⊖ ○ ⊙	New Model Wizard
Generate Web Service Translator F	rocedures
Generate default relational proce desired procedures and 'Finish'	edures compatible with Teiid Web Se
Available Web Service Translato	r Procedures
invoke(binding in String, action	n in STRING, request in OBJECT, endpoir
invokeHttp(action in STRING, r	equest in OBJECT, endpoint in STRING, i
?	< Back Next >

Figure 4.3. Generate Web Service Translator Procedures Dialog

4.1.3. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another relational source model, complete *Create New Relational Source Model* above and continue with these additional steps:
 - Step 5 Select the model builder labeled Copy from existing model of the same model class and click Next >. The Copy Existing Model dialog will be displayed.
 - Step 6 Select an existing relational model from the workspace using the browse button.



1

Note

An existing model will be pre-selected if a relational model in the workspace is selected in the *Section D.2.1, "Model Explorer View"* prior to starting the new model wizard.

• Step 7 - Check the Copy all descriptions option if desired. Click Finish

8	New Model	Wizard		×
Copy an Existing Mode Select an existing model to b				
Existing Model: USCustomer	s.xmi			
 ▷ ■ ACCOUNT ▷ ■ ACCOUNTHOLDINGS ▷ ■ CUSTOMER ▷ ■ Data Source 				
Copy all descriptions				
?	< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

Figure 4.4. Copy An Existing Model Dialog

4.2. Creating New Relational View Model

Create New Relational View Model

- To create a new empty relational view model:
 - Step 1 Launch the New Model Wizard.
 - Step 2 Specify a unique model name.
 - Step 3 Select Relational option from Model Class drop-down menu.
 - Step 4 Select View Model from Model Type drop-down menu.
 - Step 5 Click Finish.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty relational view model, the following builder options are available:
 - · Copy from existing model of the same model class.
 - Transform from existing model.

4.2.1. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another relational view model, complete *Create New Relational View Model* above and continue with these additional steps:
 - Step 5 Select the model builder labeled Copy from existing model of the same model class and click Next >. The Copy Existing Model dialog will be displayed.
 - Step 6 Select an existing relational model from the workspace using the browse button.



• Step 7 - Check the Copy all descriptions option if desired. Click Finish

8	New Model Wizard	×
Copy an Existin	ıg Model	
Select an existing		
Existing Model: US	SCustomers.xmi	
Model Contents		
D 🛛 ACCOUNT		
🕨 🛛 ACCOUNTH		
D E CUSTOMER		
Data Sour	ce	
🗹 Copy all descri	ptions	
(?)	< Back Next > Cancel	
•		

Figure 4.5. Copy An Existing Model Dialog

4.2.2. Transform From Existing Model

This option is only applicable for creating a relational view model from a relational source model with the added feature of creating default transformations (SELECT * FROM SourceModel.Table_X) for each source table. The steps are the same as for the Section 4.2.1, "Copy From Existing Model" described above.

There is an additional option in the second page of the wizard which can automatically set the relational table's supports update property to false. If this is unchecked the default value will be true.

4.3. Creating XML Document View Model

Create XML Document View Model

- To create a new empty XML document view model:
 - Step 1 Launch the New Model Wizard.
 - Step 2 Specify a unique model name.
 - Step 3 Select XML option from Model Class drop-down menu.
 - Step 4 Select View Model from Model Type drop-down menu.
 - Step 5 Click Finish.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty XML document view model, the following builder options are available:
 - Copy from existing model of the same model class.
 - Build XML documents from XML schema.

4.3.1. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another XML document view model, complete Create XML Document View Model above and continue with these additional steps:
 - Step 5 Select the model builder labeled Copy from existing model of the same model class and click Next >. The Copy Existing Model dialog will be displayed.
 - Step 6 Select an existing relational model from the workspace using the browse button.



• Step 7 - Check the Copy all descriptions option if desired. Click Finish

	New Model Wizard	×
Copy an Existing Mo Select an existing model		
Existing Model: USCuston Model Contents		
Copy all descriptions	< <u>B</u> ack Next > Cancel	Finish

Figure 4.6. Copy An Existing Model Dialog

4.3.2. Build XML Documents From XML Schema

This option creates an XML View document model based on a selected XML schema and its dependencies.

- To create a new XML document view model by from XML schema, complete *Create XML Document View Model* above and continue with these additional steps:
 - Step 5 Select the model builder labeled Build XML documents from XML schema and click Next >. The Select XML Schema dialog will be displayed.

• Step 6 - Select an existing schema model from the workspace using the browse button.



i

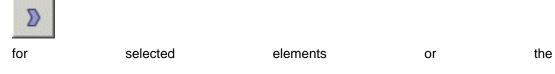
Note

An existing model will be pre-selected if an XSD model in the workspace is selected in the VDB explorer prior to starting the new model wizard. The schema must be found in the workspace so if you need to get one or more into the workspace use the XSD Schemas on file system importer.

9		New Mo	del Wizard		
Select XML Sc					
Select XML Schen	na file, then select Schem	a Root Elements. Oi	r press Finish to	create an em	pty XML Document.
	/BooksProject/Books.xsd)
Select one or mo Schema Root Ele	ore Schema Root Elements ements	to generate Virtua	l Documents	Virtua	Documents
e bookListing :	on : BooksNS:BookSetNest BooksNS:BookSetFlat d : BooksNS:BookSetMixed			> e bit >>> >>> >>>>	bliography : Bookst
Reset					
Document Option	1S				
Build full virt	ual documents from the s	chema			
 Build only fir 	st level of each document	(for large schema	where many por	tions will not b	e used)
Mapping Options					
🗹 Build Mappin	g Classes				
<					
			< <u>B</u> ack	<u>N</u> ext >	Cancel

Figure 4.7. Select XML Schema Dialog

• Step 7 - Move the available schema root elements you want to become virtual documents in the new model over to the Virtual Documents list by using the arrow button





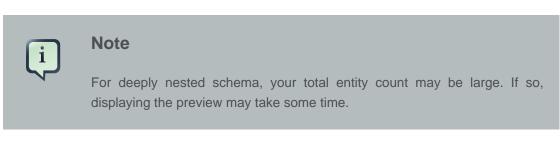
button to move all elements.

- Step 8 Select the appropriate document options and mapping options. Click Finish
- Step 9 Click Finish to create a model of all selected document entities or (optional) click Next > to view Selected Documents Statistics page which shows document entity statistics and gives you an idea the size of the model being created.

9	New Model Wizard 🛛 🗶
Selected Documents S	tatistics
	cuments to be generated. Select Next to ent, or Finish to create the document with default
Documents:	1
Elements:	7
Recursive Elements:	0
Complex Subtype Elements:	0
Attributes:	0
Total entity count:	13
?	< <u>B</u> ack <u>N</u> ext > Cancel <u>F</u> inish

Figure 4.8. Selected Documents Statistics Dialog

 Step 10 - (Optional) Click Finish to create a model of all selected document entities or click Next > to view Preview Generated Documents page that allows you to exclude document specific entities then click Finish.



🛞 New Model Wizard
Preview Generated Documents
Preview and edit the documents generated from the XSDs you selected. Elements marked with a boxed arrow contain recursive data. Elements marked with indicate that the builder stopped at the node.
▽ 🗹 🖹 bibliographyDocument
▽ 🗹 🕼 bibliography : BooksNS:Bibliography
▷ ▼ { } sequence
✓ 〈:〉 BooksNS = http://www.metamatrix.com/XMLSchema/DataSets/Books
✓ (:) xsd = http://www.w3.org/2001/XMLSchema
✓ 〈:〉 BookTypesNS = http://www.metamatrix.com/XMLSchema/DataSets/Books/BookDatatypes

Figure 4.9. Preview Generated Documents Dialog

4.4. Creating XML Schema Model

Create XML Schema Model

- To create a new empty XML schema (.xsd) model:
 - Step 1 Launch the New Model Wizard.
 - Step 2 Specify a unique model name.
 - Step 3 Select XML Schema (XSD) option from Model Class drop-down menu.
 - Step 4 Select Datatype Model from Model Type drop-down menu.
 - Step 5 Click Finish.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

• In addition to creating a new empty XML schema model, the following builder option is available:

• Copy from existing model of the same model class.

4.4.1. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another XML schema model, complete *Create XML Schema Model* above and continue with these additional steps:
 - Step 5 Select the model builder labeled Copy from existing model of the same model class and click Next >. The Copy Existing Model dialog will be displayed.
 - Step 6 Select an existing relational model from the workspace using the browse button.
 - ...
 - Step 7 Check the Copy all descriptions option if desired. Click Finish

Sew Model Wizard	
Copy an Existing Model Select an existing model to be copied.	
Existing Model: USCustomers.xmi	
 ▷ ■ ACCOUNT ▷ ■ ACCOUNTHOLDINGS ▷ ■ CUSTOMER ▷ □, Data Source 	
Copy all descriptions	
? < <u>B</u> ack <u>N</u> ext >	Cancel <u>F</u> inish

Figure 4.10. Copy An Existing Model Dialog

4.5. Creating Web Service View Model

Create Web Service View Model

- To create a new empty web service view model:
 - Step 1 Launch the New Model Wizard.
 - Step 2 Specify a unique model name.
 - Step 3 Select Web Service option from Model Class drop-down menu.
 - Step 4 Select View Model from Model Type drop-down menu.
 - Step 5 Click Finish.



Note

You can change the target location (i.e. project or folder) by selecting the **Browse...** button and selecting a project or folder within your workspace.

- In addition to creating a new empty web service view model, the following builder options are available:
 - Copy from existing model of the same model class.
 - Build from existing WSDL file(s) or URL.

4.5.1. Copy From Existing Model

This builder option performs a structural copy of the contents of an existing model to a newly defined model. You can choose a full copy or select individual model components for copy.

- To create a new relational model by copying contents from another web service view model, complete *Create Web Service View Model* above and continue with these additional steps:
 - Step 5 Select the model builder labeled Copy from existing model of the same model class and click Next >. The Copy Existing Model dialog will be displayed.
 - Step 6 Select an existing relational model from the workspace using the browse button.
 - Step 7 Check the Copy all descriptions option if desired. Click Finish

		New Mod	lel Wizard		×
Copy an Existin Select an existing	-	pied.			
Existing Model: U	SCustomers.xn	ni			
 ▷ I ACCOUNT ▷ I ACCOUNT ▷ I CUSTOME ▷ I Data Sour 	HOLDINGS R				
✓ Copy all descr	ptions				
?		< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

Figure 4.11. Copy An Existing Model Dialog

4.5.2. Build From Existing WSDL File(s) or URL

This builder option creates a Web service model based on a user-defined WSDL file and its referenced schemas. In addition, applicable XML schema files and XML View document models (optional) are created.

- To create a new relational model by copying contents from another web service view model, complete *Create Web Service View Model* above and continue with these additional steps:
 - Step 5 Select the model builder labeled Build from existing WSDL file(s) or URL and click Next >.
 - The remaining wizard steps are identical to those found using the Section 5.10, "Import WSDL Into Web Service" action option.

4.5.3. Build From Relational Models

See Section 7.4.1, "Create Web Service Action"

4.5.4. Build From XML Document View Models

Web Service models and their corresponding **Interfaces** and **Operations** can be generated in Teiid Designer from **XML View** model components. Namely, XML View Documents and XML View Document roots.

- To create a new Web service model from XML components::
 - Step 1 Select either a single XML Document or single XML Document root in Section D.2.1, "Model Explorer View".
 - Step 2 Right-click select Modeling > Create Web Service action
 - Step 3 Fill in missing properties in Web Service Generation Wizard shown below.

(@	Web S	ervice Generation V	Vizard	×
Generate a Web Se	rvice.			
Click 'Finish' to generate	the Web S	Service. Click 'Cancel'	to abort.	
Web Service Generation Wizard Generate a Web Service. Click 'Finish' to generate the Web Service. Click 'Cancel' to abort. Specify location for generated model(s) (i.e. Project or Folder) BooksProject Browse Bold Blue text indicates an existing workspace entity. Bold Red text indicates an error. WebService Model: bookCollection_WS Interface Name: bookCollection Operation Definition Input Message Element: e bookCollection : BooksNS:BookSetNested Input Message Rame: bookCollection : BooksNS:BookSetNested Output Message Name: bookCollection Output Output Message Name: bookCollection input Output Message Name: bookCollection Output				
Specify location for ger	nerated mo	odel(s) (i.e. Project or	Folder)	
BooksProject				Browse
Bold Blue text indicate	s an exist	ing workspace entity	Bold Red text indica	ates an error.
WebService Model:	bookColle	ection_WS		
Interface Name:	bookColle	ection		
Operation Name:	getbook	Collection		
Operation Definit	ion			
Input Message E	lement:	e bookCollection :	BooksNS:BookSetNe	ested
Input Message N	ame:	bookCollectionInput		
Output Message	Element:	e bookCollection : E	3ooksNS:BookSetNe	sted
Output Message	Name:	bookCollectionOuput		
?			Cancel	Finish

Figure 4.12. Generate A Web Service Dialog

• Step 4 - Click Finish to generate model. When model generation is complete, a confirmation dialog should appear. Click OK.



Figure 4.13. Generation Completed Dialog



Note

Users can change the **Web Service Model** and **Interface Name** values (via "..." buttons) to use existing **Web service** model components. This will create a new operation in an existing model.

Importers

The **Import Wizard** provides a means to create a model based on the structure of a data source, to convert existing metadata (i.e. **WSDL** or **XML Schema**) into a source model or to load existing metadata files into the current VDB.

To launch the **Import Wizard**, choose the **File > Import** action or select a project, folder or model in the tree and right-click choose "**Import...**"

🛞 🗇 Import	
Select Choose import source.	Ľ
Select an import source:	
type filter text	×
 Teiid Designer DDL File (General) >> Source or View Model DDL File (Teiid) >> Source or View Model Designer Text File >> Source or View Models File Source (Flat) >> Source and View Model File Source (XML) >> Source and View Model JDBC Database >> Source Model LDAP Service >> Source Model Salesforce >> Source Model Web Service Source >> Source and View Model (REST) Web Service Source >> Source and View Model (SOAP) WSDL File or URL >> Web Service Model XML Schemas 	
Cancel	Finish

Figure 5.1. Import Wizard

5.1. Import DDL

Source relational models can be created by importing DDL.

• You can create relational source models from your DDL using the steps below.

• Step 1 - In Model Explorer choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teiid Designer > DDL File >> Source or View Model and click Next>
- Step 3 Select existing DDL from either Choose from file system... or Choose from workspace.... set the Model folder location, enter or select valid model name, set Model type (Source Model or View Model), set desired options and click NEXT> (or Finish if enabled)

🛞 🗆 Import DDL
Provide DDL source Create relational view model "test_import_ddl" using DDL file "/home/blafond/Testing/DDL/ view_example.ddl".
DDL File Definition File: /home/blafond/Testing/DDL/view_example.ddl
Dialect:
Model Definition Options DDL file contents
Model folder: /Test_View_DDL_Import Choose
Model name: test_import_ddl Choose
Model type: View Model 🛟 🗌 Generate valid default SQL (SELECT null AS column_name, etc)
(?) < Back Next > Cancel Finish

Figure 5.2. DDL Import Options

• **Step 4** - If *NEXT>* is pressed, a difference report is presented for viewing or de-selecting individual relational entities. Press *Finish* to complete.

🔚 Import DDL 🕅
Review Model Updates
Review the changes that will be applied to your model to bring it up to date with the source metadata. Uncheck any changes you do not wish to apply.
Only In Old Model (0LD): 0 Changes (♠): 0 Only In New Model (NEW): 0
< >>
Selection Details
(?) < Back

Figure 5.3. Review DDL Updates Dialog

5.2. Import From Relational Database

• You can create source models from your relational source schema data using the steps below.

	į			pendir	0	tail provided hrough 7 may				tion u	ırl informat	ion and
•	Step	1	-	In	Model	Explorer	choose	the	File	>	Import	action

in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teiid Designer > JDBC Database >> Source Model and click Next>
- Step 3 Select existing or previous connection profile from the drop-down selector or press New... button to launch the New Connection Profile dialog (See Eclipse Data Tools documentation) or Edit... to modify/change an existing connection profile prior to selection.

į	Note the Connection Profile selection list will be populated with only JDBC Database connections.
🙁 🗆 Impo	rt Database via JDBC
	source configuration C source configuration for the database to be imported into a relational model.
Connection JDBC Metad Properties Driver: URL: User Name: Password:	Iata Processor
?	< Back Next > Cancel Finish

Figure 5.4. Select JDBC Source Configuration Dialog

Because relational databases are different, special processing of your metadata to be required in order to convert datatypes or to interpret your metadata. The **JDBC Metadata**

Processor drop-down selector will be auto-selected based on your selected connection profile. Special processors are available for DB2, Modeshape, ODBC, Oracle, PostgeSQL, SQL Server and Sybase. For all other DB's ajdefault JDBC processor is available.

 Step 4 - After selecting a Connection Profile, input password (if not provided). Press Next> (or Finish if enabled)

× 🗆 Impor	rt Database via JDBC	
	source configuration t >" button to continue or the "Finish" button to finish.	
Connection	Profile	-
PartsOracle	11 * New Edit	
JDBC Metad	ata Processor	
Oracle	*) *	
Properties		
Driver:	Oracle Thin Driver	
URL:	jdbc:oracle:thin:@englxdbs11.mw.lab.eng.bos.redhat.com:1521:ORCL	_
User Name:	partssupplier	
Password:	**	
		Ľ
?	< Back Next > Cancel Finish	

Figure 5.5. Select JDBC Source Configuration Dialog

• Step 5 - On the Select Database Metadata page, select the types of objects in the database to import. Press Next> (or Finish if enabled).

😣 🗆 Import Database via JDBC		
Select Database Metadata Select the types of objects in the datab	ase to import.	
Foreign Keys	Table Types	
📝 Include Incomplete FKs	SYNONYM TABLE	
 Indexes Unique Only Approximations Allowed 	VIEW	Select All Deselect All
Procedures		
?	< Back Next > Cancel	Finish

Figure 5.6. Select Database Metadata Dialog

 Step 6 - On the Select Database Objects page, view the contents of the schema, or change selections. Select which database schema objects will be used to construct relational objects. Press Next> (or Finish if enabled)

😣 💷 Import Database via JDBC					
Select Database Objects					
Select the objects to import from data	abase "PartsOra	acle11".			
📄 PartsOracle11 🛛 🚳	Columns Inc	dexes Primary Key In	nported Foreign	»	
PARTSSUPPLIER	TABLE_CAT	TABLE_SCHEM	TABLE_NAME	COLUMN_NAME	DATA_TYP
TABLE		PARTSSUPPLIER	PARTS	PART_ID	
PARTS =		PARTSSUPPLIER	PARTS	PART_NAME	1
		PARTSSUPPLIER	PARTS	PART_COLOR	1
SHIP_VIA		PARTSSUPPLIER	PARTS	PART_WEIGHT	1
JOPPLIER V	(1)				<u>, ()</u>
5 table object(s) selected to import.					
Deselect All					
Show					
		Coloria de constitu	Charles		
🔵 All schema 🔘 Only selecte	ed schema	Select and connection	n filtered schen	na	
?		< Back	Next >	Cancel	Finish

Figure 5.7. Select Database Options Dialog

• Step 7 - On the Specify Import Options page, specify desired Model Name as well as any other options used to customize the naming of your relational objects. Press Finish to complete.

🛞 💷 Import Database via JDBC
Specify Import Options Press the "Next >" button to continue or the "Finish" button to finish.
Relational Model Definition Model Object Names (Tables, Procedures, Columns, etc)
Model Name: PartsOracle11.xmi
Into Folder: Test_TEIIDDES_2475
Make target a view model
Update (if existing model selected)
Include Cost Statistics (will increase time for large imports)
Include Catalog For Fully Qualified Names
Cancel Finish

Figure 5.8. Specify Import Options Dialog - Relational Model Definition

🛞 🗆 Import Database via JDBC
Specify Import Options Press the "Next >" button to continue or the "Finish" button to finish.
Relational Model Definition Model Object Names (Tables, Procedures, Columns, etc)
Use Fully Qualified Names (Example: partssupplier.dbo.PARTS)
Change Case For All Characters
Case Options Make All Upper Case (Example: Suppliers > SUPPLIERS)
Make All Lower Case (Example: SUPPLIERS > suppliers)
< Back Next > Cancel Finish

Figure 5.9. Specify Import Options Dialog - Object Naming Options

During the *Finish* processing, a monitor will be displayed providing feedback on the import progress.

	Progress Information	8
9	Importing from JDBC database: PartsOracle11	
Processi	ng database object (1 of 5) - creating BaseTable "PARTS"	
		Cancel

Figure 5.10. JDBC Import Progress Dialog

5.2.1. Relational Model Costing

On the final page of import, you are presented with a checkbox option to **Include Cost Statistics**. If this option is selected, the table cardinalities will be determined and the table cardinality properties will be set. Later, during query execution, if the table cardinalities are available they are used by the Teiid query engine to optimize the query plan.

After import, you can update the cardinalities at any time. If you would like to update the Cost Statistics for the entire model, select the model - then use the Modeling Context menu action **Update Source Data Statistics**. Likewise, you can update the cardinality for a specific table or tables, by selecting the table(s) in ModelExplorer then using the same **Update Source Data Statistics** action.

5.3. Import From Teild Data Source Connection

The **Teiid Connection >> Source Model** import option provides a means to create relational source models from relational and other deployed data sources that are not supported by other Teiid Designer importers.

NOTE: To launch this importer, you must have at minimum a Teiid 8.x server running in Designer. The Teiid importer deploys a dynamic VDB to Teiid containing the selected source type, then the schema (as determined by Teiid) is retrieved. We expect to move towards this type of import in future versions of Teiid Designer.

- You can create relational source models from your deployed data source connections using the steps below.
 - Step 1 In Model Explorer choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teiid Designer > Teiid Connection >> Source Model and click Next>
- Step 3 Select the datasource to use for the import. You can create a new source if it doesnt exist, as well as other source management functions. Click NEXT> to continue.

mporter De This importer	rutilizes a Teiid runtime o			ent Teiid DDL runtime metadata
Note that all o Data Source		ported	and may not return source metac	fata (tables, columns, etc)
	DataSource		JNDI Name	Driver or Module ID
New	java:/BooksOracle11_D)5	java:/BooksOracle11_DS	ojdbc14.jar
Delete	java:/EmployeeSource_		java:/EmployeeSource_DS	EmployeeSource_DS
Name * URL, End * WebServ * class-nar	java:/PartsOracle_DS java:/PartsOracle_DS_C issue:/DartsCC Properties	COPY Value http:/d None	java:/excel-file java:/myJNDI_Name java:/PartsDataFileSource_DS java:/PartsOracle_DS java:/PartsOracle_DS_COPY iava:/DarteSS	ojdbc14.jar ojdbc14.jar edidbe4.jse er/data/employees/file/EN
Authentic *-denotes	ation User Name ation User Password required property HTTP, Service Endpoint f	for SOA	P	0

Figure 5.11. Select Deployed Data Source

• Step 4 - On the next page select the appropriate translator for your data source type as well as defined the target relational model that you wish to create or update. You can also specify value for any built-in import properties in order to fine-tune the schema data you wish to have returned. Click **NEXT>** to continue.

🗙 🔲 Imp				
-	port using a Teiid Connection			
elect the tra	anslator for the import			
Press the "Ne	ext >" button to continue.			
erver: EAP6.	1+Teiid8.7FINAL			
Source Def	inition			
DataSources	ProductsMySQL			
Datasource.	rioducismysQu			
Driver:	mysql-connector-java-5.1.5.jar			
Translator:	mysql			
Indirstator.	nysqr			
Import Pro	perties			
Property		Value		
Auto Crea	te Unique Constraints	true		
catalog				
Column Na	ame Pattern			
Exclude Pr	rocedures			
Exclude Ta	bles			
Exclude Ta		true		
Exclude Ta Import Ap	bles	true		
Exclude Ta Import Ap Note: See	ables proximate Indexes	true		
Exclude Ta Import Ap Note: See Optional So	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See	ables proximate Indexes Teiid documentation for details on importer properties	true		
Exclude Ta Import Ap Note: See Optional So	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See Optional So	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See Optional So	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See Optional So Name	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See Optional So	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See Optional So Name	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See Optional So Name	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See Optional So Name	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See Optional So Name	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true		
Exclude Ta Import Ap Note: See Optional So Name	ables proximate Indexes Teiid documentation for details on importer properties purce Import Properties	true < Back	Next >	Cancel

Figure 5.12. Translator and Model Definition

• Step 5 - On the next page define your target relational model that you wish to create or update. Either enter a new unique name, or select an existing source model. At this point you can click on the *Advanced* tab to view or edit the temporary vdb.xml contents that will be deployed to Teiid in order to return the requested database schema/DDL. Click **NEXT>** to continue.

😣 🗆 🛛 Im	port using a Teiid Connection		
	arget model for the import Next >" button to continue.		
Default Se	erver: JDV 6.4 ER2		(*=-
	odel Definition Advanced		
Location:	Sample_Project		
Name:	PartsSuppllier		
	Model Status		
	Model name defined: PartsSuppllier		
Filte	ate existing model r unique constraints that mimic PKs. te and apply a Connection Profile.		
?	< Back	Next > Ca	ncel Finish

Figure 5.13. Target Model Definition

 Step 6 - When you move to next page of the wizard, a temporary dynamic vdb is actually deployed to you rserver and the schema your data source is retrieved in DDL form. This DDL is displayed (and can also be exported if desired). Click NEXT> to continue.

🛞 🗊 Import using a Teiid Connection
Get DDL for the Import Press the "Next >" button to continue.
Export to File System Export to Workspace Importer DDL CREATE FOREIGN TABLE PARTS (PART_ID string(50) NOT NULL OPTIONS (NAMEINSOURCE '"PART_ID"', N PART_NAME string(255) OPTIONS (NAMEINSOURCE '"PART_NAME"', NATIV PART_COLOR string(30) OPTIONS (NAMEINSOURCE '"PART_COLOR"', NATI PART_WEIGHT string(255) OPTIONS (NAMEINSOURCE '"PART_WEIGHT"', N) OPTIONS (NAMEINSOURCE '"PARTS"', UPDATABLE TRUE, CARDINALITY 16);
<pre>CREATE FOREIGN TABLE SHIP_VIA (SHIPPER_ID bigdecimal(2) NOT NULL OPTIONS (NAMEINSOURCE '"SHIPPE SHIPPER_NAME string(30) OPTIONS (NAMEINSOURCE '"SHIPPER_NAME"',) OPTIONS (NAMEINSOURCE '"SHIP_VIA"', UPDATABLE TRUE, CARDINALITY 3) CREATE FOREIGN TABLE STATUS (STATUS_ID bigdecimal(2) NOT NULL OPTIONS (NAMEINSOURCE '"STATUS_ STATUS_NAME string(30) OPTIONS (NAMEINSOURCE '"STATUS_NAME"', NA) OPTIONS (NAMEINSOURCE '"STATUS', UPDATABLE TRUE, CARDINALITY 3);</pre>
CREATE FOREIGN TABLE SUPPLIER (SUPPLIER_ID string(10) NOT NULL OPTIONS (NAMEINSOURCE '"SUPPLIER SUPPLIER_NAME string(30) OPTIONS (NAMEINSOURCE '"SUPPLIER_NAME"' SUPPLIER_STATUS bigdocimal(2) OPTIONS (NAMEINSOURCE '"SUPPLIER_OF
Set all tables read-only (UPDATABLE FALSE)
Cancel Finish

Figure 5.14. Review DDL Dialog

• Step 7 - On the final page of the wizard, a difference report is presented for viewing or deselecting individual relational entities. Press **Finish** to complete.

Select the Differences to Apply Select the items to apply, then click 'Finish'. Items for Import ✓ ✓ Objects to Create ✓ ✓ Ø Ø SHIP_VIA ✓ Ø SUPPLIER Ø Ø SUPPLIER Ø Ø SUPPLIER Ø Ø SuppLier_PARTS	
Items for Import ▼ Objects to Create ■ PARTS ■ PARTS ■ SHIP_VIA ■ STATUS ■ SUPPLIER ■ SUPPLIER_PARTS	
 Objects to Create PARTS SHIP_VIA STATUS SUPPLIER SUPPLIER_PARTS 	3
 PARTS SHIP_VIA STATUS SUPPLIER SUPPLIER_PARTS 	
 ✓ I SHIP_VIA ✓ I STATUS ✓ I SUPPLIER ✓ I SUPPLIER_PARTS 	
 ✓ I STATUS ✓ I SUPPLIER ✓ I SUPPLIER_PARTS 	
SUPPLIER SUPPLIER_PARTS	
SUPPLIER_PARTS	
🔽 🖬 table	
Import Messages	
(?) < Back Next > Cancel Finish	

Figure 5.15. Review Model Updates Dialog

5.4. Import From Flat File Source

 You can import metadata from your flat file data sources and create the metamodels required to query your data in minutes. Using the steps below you will define your flat file data source, configure your parsing parameters for the flat file, generate a source model containing the standard Teiid flat file procedure and create view tables containing the SQL defining the column data in your flat file.

Teiid supports Flat Files as data sources. Teiid Designer provides an Import wizard designed to assist in creating the metadata models required to access the data in your flat files. As with Designer's JDBC, Salesforce and WSDL importers, the Flat File importer is based on utilizing a specific Data Tools Connection Profile.

The results of the importer will include a source model containing the getTextFiles() procedures supported by Teiid.

The importer will also create a new view model containing a view table for your selected flat file source file. Within the view table will be generated SQL transformation containing the "getTextFiles()" procedure from your source model as well as the column definitions and parameters required for the Teiid TEXTTABLE() function used to query the data file. You can also choose to update an existing view model instead of creating a new view model.

The TEXTTABLE function, as defined in the Teiid documentation, processes character input to produce tabular ouptut. It supports both fixed and delimited file format parsing. The function itself defines what columns it projects. The TEXTTABLE function is implicitly a nested table and may be correlated to preceeding FROM clause entries.

```
TEXTTABLE(expression COLUMNS <COLUMN>, ...
[DELIMITER char] [(QUOTE|ESCAPE) char] [HEADER [integer]] [SKIP
integer]) AS name
```

Teiid Designer will construct the full SQL statement for each view table in the form:

```
SELECT A.Name, A.Sport,
A.Position, A.Team, A.City, A.StateCode, A.AnnualSalary FROM (EXEC
PlayerDataSource.getTextFiles('PlayerData.txt')) AS f,
TEXTTABLE(f.file COLUMNS Name string, Sport string, Position
string, Team string, City string, StateCode string, AnnualSalary
string HEADER 2 SKIP 3) AS A
```

To import from your flat file source follow the steps below.

• Step 1 - In Model Explorer choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

 Step 2 - Select the import option Teiid Designer > File Source (Flat) >> Source and View Model and click Next>

🛞 🗇 Import	
Select Choose import source.	Ľ
Select an import source:	
type filter text	×
 Teiid Designer DDL File (General) >> Source or View Model DDL File (Teiid) >> Source or View Model DDL File (Teiid) >> Source or View Models File Source (Flat) >> Source and View Model File Source (XML) >> Source and View Model JDBC Database >> Source Model JDBC Database >> Source Model LDAP Service >> Source Model Salesforce >> Source Model Web Service Source >> Source and View Model (REST) Web Service Source >> Source and View Model (SOAP) WSDL File or URL >> Web Service Model XML Schemas 	
Cancel	Finish

Figure 5.16. Import from Flat File Source

• Step 3 - On the first wizard page, select the flat file mode you wish to import, either Flat file on local file system or Flat file via remote URL

ata File S		t File Source
	ource S	election
Press the	"Next >"	button to continue.
Data File	e Source	
Employ	eesRemo	teFile \$ New Edit
Availabl	e Data Fi	les
Folder lo	cation:	boss.org/teiid/designer/data/employees/file/EMPLOYEEDATA.txt
File Filter	r:	*.*
Data File	e Name	
Selected	Data File:	teiid_designer_data_employees_file_EMPLOYEEDATA3637927986146424981.txt
	Model De	
Source	Model De	finition
Source I	Model De Employ empoye Model NEW M	ees_file_source

Figure 5.17. Flat File Source Model Options

Note that the local file system connection specifies a folder containing one or more comma separated text data file : /home/jdoe/employees/. The remote URL connection will specify a URL to a single data file : http://download.jboss.org/teiid/designer/data/employees/file/ EMPLOYEEDATA.txt.

• Step 4 - Select existing or previous connection profile from the drop-down selector or press New... button to launch the New Connection Profile dialog (See Eclipse Data Tools documentation) or Edit... to modify/change an existing connection profile prior to selection.

Note

i

1

The Flat File Source selection list will be populated with only Flat File connection profiles.

After selecting a **Connection Profile**, the file contents of the folder defined in the connection profile will be displayed in the **Available Data Files** panel. Check the the data file you wish to process. The data from this file, along with your custom import options, will be used to construct a view table containing the required SQL transformation for retrieving your data and returning a result set.

Lastly enter or unique source model name in the **Source Model Definition** section at the bottom of the page or select an existing source model using the browse button.

Note

The **Model Status** section which will indicate the validity of the model name, whether the model exists or not and whether the model already contains the getTextFiles() procedure. In this case, the source model nor the procedure will be generated.

When finished with this page, click Next>.

	Import From Flat File Source	(
Data Fil	le Source Selection	
Press th	he "Next >" button to continue.	
Data F	ile Source	
Employ	yeeData 🗘 New	Edit
Availab	ble Data Files	
Folder lo		
	le Name	
_	pDataFixedWidth.txt	
	ployeeData.txt yerData.txt	
V Play	yer bata.txt	
Selected	d Data File: PlayerData.txt	
6		
	Model Definition	
Location	TestXmlimport	
Name:	TextFileProcedures.xmi	
	Model Status	
	EXISTING MODEL: Source model TextFileProcedures.xmi already contains required	
	getTextFiles() procedure.	
?	< Back Next > Cancel	nish

Figure 5.18. Data File Source Selection Page

• Step 5 - The next page, titled Data Source Definition Options, provides the option for defining a unique JBoss JNDI name for the deployed data source that this imported model

will prepresent. This name will be stored as a property on your model and changeable via the **Modeling > Set JBoss JNDI Data Source Name** action. If there you have a default server defined and running, then **Auto-Create Data Source** check box will be enabled. On completion of this import, the wizard will check for an existing deployed data source with this JNDI name and create one if needed.

When finished with this page, click Next>.

🙁 🗆 Impor	rt From Flat File Source		
	efinition Options e JBoss JNDI name for this data source		3
JBoss Data S	Source Information		
JNDI Name	ÞartsSupplier_mysql		
Auto-cre	eate Data Source		
0		- Pack Nexts	Tinich
•		< Back Next > Ca	Finish

Figure 5.19. Data File Source Selection Page

• Step 6 - The next page, titled Flat File Column Format Definition, requires defining the format of your column data in the file. The options are Character delimited and Fixed width. This page contains a preview of the contents of your file to aid in determining the format. The wizard defaults to displaying the first 20 lines, but you can change that value if you wish.

When finished with this page, click Next>.

Import From Flat File Source
Flat File Column Format Definition
Press the "Next >" button to continue.
Selected Data File: EmployeeData.txt
File Preview Options
Number of lines in file : 52 Number of preview lines 20
Select Column Format
Character delimited Fixed width
File Contents Preview
LastName,FirstName,MiddleName,EmpId,Department,AnnualSalary,Title,HomePhone,M
Kisselmeyer,Abbiegale,Tikvica ,9000059,G,64000.00,MGMT WannaBe,670-270-7947,9
Glore,Diodie,Vojvoda ,9000060,G,71000,Associate,480-650-9750,9000073,127 State
Dawson,Pinckney,Ostoja ,9000061,G,71000,Associate,110-400-3600,9000073,135 St
Waldrip,Trixie,Curic ,9000062,G,57000,Newbie,820-210-7045,9000073,136 State S
Kitchen,Zilpha,Buic ,9000063,G,60000,MGMT WannaBe,660-390-3785,9000073,138 St
Wakeman,Gerard,Vlahovic ,9000064,G,78000,Newbie,700-190-5880,9000073,130 State
Rafferty,Dock,Korda ,9000065,G,70000,Newbie,400-190-6192,9000073,128 State St
Kersavage,Zelda,Mjesecevic ,9000066,G,56000,MGMT WannaBe,802-930-1482,9000073
Zummer,Gerda,Milan ,9000067,G,69000,Newbie,920-100-9701,9000073,131 State St.
Davies,Allwyne,Radic ,9000068,G,61000,Associate,470-820-6096,9000073,126 State
Deanford, Abe, Skrabalo , 9000069, G, 67000, Associate, 907-660-8233, 9000073, 124 Sta
Garcia,Orsal,Ucovic ,9000070,ML,79000,CF0,480-490-7710,9000075,150 State St.,
Zook,Orson,Bendevis ,9000071,G,71000,Newbie,350-260-8654,9000073,133 State St
Rainier,Adelaid,Marinovic ,9000072,G,67000,Newbie,316-550-3499,9000073,125 St
Nealon,General,,9000073,G,77000,C00,203-420-3113,9000075,129 State St.,Hartfo
Garahana, Jarrod, Cvjetkovic , 9000074, G, 66000, Associate, 620-430-1782, 9000073, 13
Neely,Petronella,Goravica ,9000075,G,80000,CE0,230-320-6330,9000076,134 State
(?) < <u>B</u> ack <u>Next</u> > Cancel

Figure 5.20. Data File Source Selection Page

Step 7a : Character Delimited Option - The primary purpose of this importer is to help you create a view table containing the transformation required to query the user-defined data file. This page presents a number of options you can use to customize the Generated SQL Statement , shown in the bottom panel, for the character delimited option. Specify header options (Column names in header, header line number and first data line number), Parse selected row, changed character delimiter and edit the TEXTTABLE() function options. See the Teiid User's Guide for details on the TEXTTABLE() function.

If columns names are not defined in a file header or if you wish to modify or create custom columns, you can use the **ADD**, **DELETE**, **UP**, **DOWN** to manage the column info in your SQL.

When finished with this page, click Next>.

Import From Flat File Source						
Flat File Delimited Columns Parser Settings Press the "Next >" button to continue.						
Selected Data File: EmployeeData.txt						
Format Options ✓ Column names in header Header line # 1 Data line # 2 Parse Selected Row Edit Delimiter Character Edit TEXTTABLE() function options	File Contents Preview LastName,FirstName,MiddleName,EmpId,De Kisselmeyer,Abbiegale,Tikvica,9000059 Glore,Diodie,Vojvoda,9000060,G,71000, Dawson,Pinckney,Ostoja,9000061,G,7100 Waldrip,Trixie,Curic,9000062,G,57000, Kitchen,Zilpha,Buic,9000063,G,60000,N Wakeman,Gerard,Vlahovic,9000064,G,780					
Column Information Column Name LastName FirstName MiddleName EmpId DELETE	Datatype string string string string					
A.HomePhone, A.Mgrld, A.Street, A.City FROM (EXEC AAAA.getTextFiles('EmployeeDa string, FirstName string, MiddleName st	ame, A.Empld, A.Department, A.AnnualSalary, A.Title, , A.State, A.ZipCode ata.txt')) AS f, TEXTTABLE(file COLUMNS LastName tring, Empld string, Department string, AnnualSalary Mgrld string, Street string, City string, State string,					
?	Back Next > Cancel Finish					

Figure 5.21. Flat File Delimited Columns Options Page

To aid in determining if your parser settings are correct you can select a data row in your **File Contents Preview** section and click the **Parse Selected Row** button. A dialog will be displayed showing the list of columns and the resulting column data. If your column data is not what you expected, you'll need to adjust your settings accordingly.

Parsed Column Data	×
The following list contains the parsed column data from t selected line	the
Parsed Column Data	
lastName : Glore	
FirstName : Diodie	
MiddleName : Vojvoda	
Empld : 9000060	
Department : G	
AnnualSalary : 71000	
Title : Associate	
HomePhone : 480-650-9750	
Mgrid : 9000073	
Street : 127 State St.	
City : Los Angeles	
State : CA	
ZipCode : 10005	
(2) Cancel	ОК
Cancel	

Figure 5.22. Parse Column Data Dialog

Step 7b : Fixed Column Width Option - The primary purpose of this importer is to help you create a view table containing the transformation required to query the user-defined data file. This page presents a number of options you can use to customize the Generated SQL Statement, shown in the bottom panel, for the fixed column width option. Specify header options (Column names in header, header line number and first data line number), Parse selected row, changed character delimiter and edit the TEXTTABLE() function options. See the Teiid User's Guide for details on the TEXTTABLE() function.

If columns names are not defined in a file header or if you wish to modify or create custom columns, you can use the **ADD**, **DELETE**, **UP**, **DOWN** to manage the column info in your SQL.

You can also utilize the cursor postion and text length values in the upper left panel to determine what your column widths are in your data file.

When finished with this page, click Next>.

	lm	port From Flat	File So	urce			
Flat File Fixed Colun	nns Width Par	ser Settings					
Press the "Next >" butt	on to continue.						
Selected Data File: Emp	DataFixedWidth txt	+					
		·					
Format Options	File Contents	Preview					
Data line # 2		tment string					
		integer WID	DTH 8,	Street	string	, WIDTH	414, Cit
Cursor Position 24	11						
Text Length 12	Kisselmeyer		Tikv:		900005		64000
	Glore Dawson	Diodie Pinckney	Vojv		900006		71000 71000
	Waldrip	Trixie	Osto Curi		900006		57000
	Kitchen	Zilpha	Buic		900006		60000
	Wakeman	Gerard	Vlah		900006		78000
	Rafferty	Dock	Kord		900006	35 G	70000 -
	<						>
Column Options	Column Infor	mation					
		nation					
ADD	Column Name			Datatype		Width	
DELETE	A firstName			string		12	
	A middleName			string		12	
UP]	A lastName			string		12	
DOWN							
]
Teiid TEXTTABLE() F	-unction Optio	ns					
🗌 Include HEADER 🗌	Include SKIP	🗌 In	clude Q	UOTE 🕛		Include	ESCAPE 1
Generated SQL Sta	tement						
SELECT	-	2					6
A.firstName, A.middl FROM	eName, A.column	_3					
(EXEC ssss.getTextFi	les('EmpDataFixed	Width.txt')) AS f.	TEXTTA	BLE(file C	OLUMNS f	irstName	e string width
12, middleName string	width 12, column_	3 string width 1	2) AS A				2
0		- Baali		Nexts		ancel	Finish
		< <u>B</u> ack		<u>N</u> ext >		ancel	<u>F</u> inish

Figure 5.23. Flat File Fixed Columns Width Options Page

li

• Step 8 - On the View Model Definition page, select the target folder location where your new view model will be created. You can also select an existing model for your new view tables.

Note

The **Model Status** section which will indicate the validity of the model name, whether the model exists or not. Lastly, enter a unique, valid view table name.

Press **Finish** to generate your models and finish the wizard.

	Import From Flat File Source	
View Model Definit	ion	E.
Press the "Finish" butt	ton to finish.	
Selected Data File: Play	yerData.txt	
View Model Defini	ition	
Location	TestXmlImport	
Name:	MyEmployeeViews.xmi	
	Model Status	
	EXISTING MODEL: New view tables will be created in your existing view model	
	MyEmployeeViews.xmi on FINISH.	
New view table name:	PlayerDataTable	
~		
?	< <u>B</u> ack <u>N</u> ext > Cancel	<u>F</u> ini

Figure 5.24. View Model Definition Page

When your import is finished your source model will be opened in an editor and show a diagram containing the your getTextFiles() procedure.

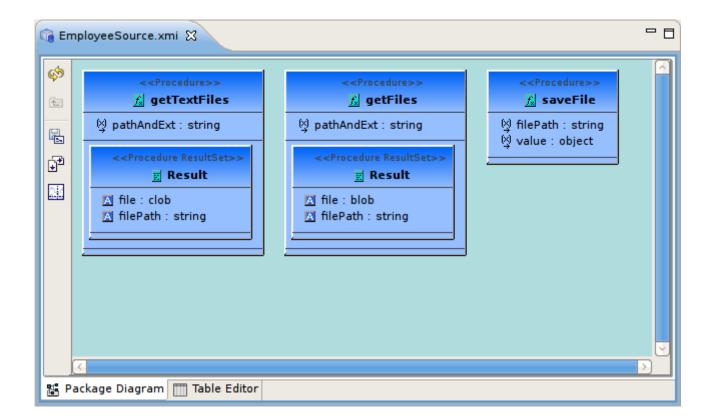


Figure 5.25. Generated Flat File Procedures

In addition, the view model will be opened in an editor and will show the generated view tables containing the completed SQL required to access the data in your flat file using the "getTextFiles" procedure above and the Teiid TEXTTABLE() function. The following figure is an example of a generated view table.

👘 Er	mploy	eeViews.xmi	x												
69		1	VIEW			SOU	RCI	ES							
		A LastNam A FirstNam A FirstNam A MiddleNa A Empld : A Departm A AnnualS A Title : st	<base table=""/> > ployeeDataView ne : string(10) ne : string(10) string(10) nent : string(10) alary : string(10) string(10) string(10) string(10) ring(10)			< <p ▲ g Emp ♥ pathAndt</p 	rocedu etTex loyeeS Ext : s ure Re Resu ob	ure>> ttFile: Source string esultSe ult	1						
Tr SE A.M Firs	Dr LECT A.L Mgrid, FROM (E) stNan	astName, A A.Street, A EC Employe ne string, Mi	ram Table Editor Cursor at (3, 6) FirstName, A.Midd City, A.State, A.Zip SeSource.getTextFile iddleName string, E	oCod es('Ei implo	me, , e mplo d stri	yeeData.txt') ng, Departme) AS ent s	f, TE) tring,	KTTAB Annu	ILE(f. l JalSa	file Co lary s	ary, A DLUM string	INS L , Title	.astN e strir	om Iam

Figure 5.26. Generated Flat File View Table

5.5. Import From XML Data File Source

 Teiid supports XML Files as data sources. You can import from these data sources and create the metamodels required to query your data in minutes. Using the steps below you will define your flat file data source, configure your parsing paramaters for the xml data file, generate a source model containing the required Teiid procedure and create a view table containing the SQL defining the column data in your xml data file.

As with Designer's JDBC, Salesforce and WSDL importers, the XML File importer is based on utilizing a specific Data Tools Connection Profile.

The results of the importer will include a source model containing the getTextFiles() procedure or invokeHTTP() procedure which are both supported by Teiid.

The importer will also create a new view model containing a view table for your selected flat file source file. Within the view table will be generated SQL transformation containing the "getTextFiles()" procedure from your source model as well as the column definitions and parameters required for the Teiid XMLTABLE() function used to query the data file. You can also choose to update an existing view model instead of creating a new view model.

The XMLTABLE function uses XQuery to produce tabular ouptut. The XMLTABLE function is implicitly a nested table and may be correlated to preceeding FROM clause entries. XMLTABLE is part of the SQL/XML 2006 specification.

```
XMLTABLE([<NSP>,] xquery-expression
[<PASSING>] [COLUMNS <COLUMN>, ... )] AS name
```

```
COLUMN := name (FOR ORDINALITY | (datatype [DEFAULT
expression] [PATH string]))
```

Teiid Designer will construct the full SQL statement for each view table in the form:

```
SELECT A.entryDate AS entryDate,
A.internalAudit AS internalAudit FROM (EXEC
CCC.getTextFiles('sample.xml')) AS f,
XMLTABLE(XMLNAMESPACES('http://www.kaptest.com/schema/1.0/party'
AS pty), '/pty:students/student' PASSING
XMLPARSE(DOCUMENT f.file) COLUMNS entryDate FOR ORDINALITY,
internalAudit string PATH '/internalAudit') AS
A
```

To import from your XML data file source follow the steps below.

• Step 1 - In Model Explorer choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

 Step 2 - Select the import option Teiid Designer > File Source (XML) >> Source and View Model and click Next>

🛞 🗇 Import	
Select Choose import source.	Ľ
Select an import source:	
type filter text	×
 Teiid Designer DDL File (General) >> Source or View Model DDL File (Teiid) >> Source or View Model DDL File (Teiid) >> Source or View Models File Source (Flat) >> Source and View Model File Source (XML) >> Source and View Model JDBC Database >> Source Model JDBC Database >> Source Model Salesforce >> Source Model Salesforce >> Source Model Web Service Source >> Source and View Model (REST) Web Service Source >> Source and View Model (SOAP) WSDL File or URL >> Web Service Model XML Schemas 	
Sack Next > Cancel	Finish

Figure 5.27. Import from XML File Source

 Step 3 - The next page of the wizard allows selection of the XML Import mode that specifies whether the XML file is local or remote. The description at the top describes what operations this wizard will perform. Select either the XML file on local file system or XML file via remote URL and click Next>

🎯 🕙 🛛 Import P	From XML File Source
XML Import File Options	
Select the desired XML Import Mode	
•	
Description	
This wizard automates the construction of the source from your XML file source. The view table transformatio utilize the XMLTABLE() function.	procedure and view table transformation necessary to que n will contain a getTextFiles() (or invokeHTTP()) procedure
The XML structure of your file will be presented to aid in	n selecting data elements to define your table column data
Select XML Import Mode	
 XML file on local file system 	
O XML file via remote URL	
?	< Back Next > Enish Ca

Figure 5.28. XML Import File Options Page

 Step 4 - Select existing or previous connection profile from the drop-down selector or press New... button to launch the New Connection Profile dialog (See Eclipse Data Tools documentation) or Edit... to modify/change an existing connection profile prior to selection.

After selecting a **Connection Profile**, the XML data file from the connection profile will be displayed in the **Available Data Files** panel. Check the the data file you wish to process. The data from this file, along with your custom import options, will be used to construct a view table containing the required SQL transformation for retrieving your data and returning a result set.

Lastly enter or unique source model name in the **Source Model Definition** section at the bottom of the page or select an existing source model using the browse button.



Note

The **Model Status** section which will indicate the validity of the model name, whether the model exists or not and whether the model already contains the getTextFiles() procedure. In this case, the source model nor the procedure will be generated.

When finished with this page, click Next>.

ی 🕲	Import From XML File Source		
XML Data	File Source Selection		
Press th	e "Next >" button to continue.		
Data File	Source		
Parts XM			New
			140.00
XML Dat			
Folder lo	cation: /usr/share/teiid		
Data File	Name		
🗹 parts	xml		
Selected	Data File: parts.xml		
Source	Model Definition		
Location:	parts		
Name:	PartXMLSource		
	Model Status		
	NEW MODEL: Source model [PartXMLSource] does not exist.		
	Model with required getTextFiles() procedure will be created on FINISH.		
0			
(?)	< <u>B</u> ack <u>N</u> ext > Enis	h	Ca

Figure 5.29. XML Data File Source Selection Page

• Step 5 - The next page, titled Data Source Definition Options, provides the option for defining a unique JBoss JNDI name for the deployed data source that this imported model will prepresent. This name will be stored as a property on your model and changeable via

the **Modeling > Set JBoss JNDI Data Source Name** action. If there you have a default server defined and running, then **Auto-Create Data Source** check box will be enabled. On completion of this import, the wizard will check for an existing deployed data source with this JNDI name and create one if needed.

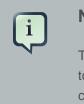
When finished with this page, click Next>.

	rt From Flat File Source		
	efinition Options e JBoss JNDI name for this data source		
JBoss Data S	Source Information		
JNDI Name	PartsSupplier_mysql		
V Auto-cre	eate Data Source		
L			
(?)		< Back Next >	Cancel Finish

Figure 5.30. Data File Source Selection Page

• Step 6 - The primary purpose of this importer is to help you create a view table containing the transformation required to query the user-defined data file. This page presents a number of options you can use to customize the **Generated SQL Statement**, shown in the bottom panel. The to panel contains an XML tree view of your file contents and actions/buttons you can use to create column entries displayed in the middle, **Column Information** panel.

To create columns, select a root XML element and right-click select **Set as root path** action. This populates the root path value. Next, select columns in the tree that you wish to include on your query and select **Add selection as new column** button. You can also modify or create custom columns, by using the **ADD**, **DELETE**, **UP**, **DOWN** to manage the column info in your SQL.



Note

The **Path** property value for a column is the selected element's path relative to the defined root path. If no root path is defined all paths are absolute. Each column entry requires a datatype and an optional default value. See the Teiid User's Guide for details on the XMLTABLE() function.

When finished with this page, click Next>.

🛞 💷 Import From XML File Source			
XML Data File Import Options Press the "Next >" button to continue.			
XML File EMPLOYEEDATA.xml XML File Contents V EmployeeData V EmployeeData E EmployeeData E LastName E FirstName E MiddleName E Empld E Department E AnnualSalary E Title SELECT A.LastName AS LastName, A.FirstName AS FROM (EXEC SourceModel.getTextFiles('EMPLOY EmployeeData/EmployeeData' PASSING XML 'FirstName/text()) AS A	'EEDATA.xml ')) A	/EmployeeData/Em Name LastName FirstName	ployeeData Column Info ordinal, type = string, default = , PATH = LastNa ordinal, type = string, default = , PATH = FirstNa MESPACES('http://org.jboss.teiid' ASEMPLOYEES_NS) stName string PATH 'LastName/text()', FirstName string
?			< Back Next > Cancel

Figure 5.31. XML File Delimited Columns Options Page

• Step 7 - On the View Model Definition page, select the target folder location where your new view model will be created. You can also select an existing model for your new view tables.

Note

i

The **Model Status** section which will indicate the validity of the model name, whether the model exists or not. Lastly, enter a unique, valid view table name.

Press **Finish** to generate your models and finish the wizard.

9 ن	Import From XML File Source
View Model Definitio	
Press the "Finish" but	on to finish.
Selected Data File: par	ts.xml
View Model Definiti	on
Location:	parts
Name:	PartView
	Model Status
	NEW MODEL: New view tables will be created in a new view model [PartView] on FINISH.
New view table name:	PartsViewTable
?	< <u>B</u> ack <u>Next</u> <u>Finish</u> Ca

Figure 5.32. View Model Definition Page

5.6. Import From Salesforce

• You can create relational source models from your Salesforce connection using the steps below.

i		pendir		tail provided hrough 7 ma				tion u	ırl informat	ion and
• Step 1		In selec			choose				Import	action
in the toolb	ar or	selec	t a project	t, folder or m	odel in the	tree ar	nd choo	se In	nport	

- Step 2 Select the import option Teiid Designer > Salesforce >> Source Model and click Next>
- Step 3 Select existing or previous connection profile from the drop-down selector or press New... button to launch the New Connection Profile dialog (See Eclipse Data Tools documentation) or Edit... to modify/change an existing connection profile prior to selection.



Note

The Connection Profile selection list will be populated with only Salesforce connection profiles.

E	Create Relational Model from SalesForce Data Model	×
SalesForce	Credentials	(X)
Validate the Sa	alesForce Connection Profile	
Connection	Profile	
Billing Sales		
Properties		5
URL:	<default></default>	
User Name:	joeBlow@bogus.ortg	
Password:	******	
Ping		
?	< <u>B</u> ack <u>N</u> ext > Cancel <u>Finish</u>	

Figure 5.33. Select Salesforce Credentials Dialog

• **Step 4** - After selecting a *Connection Profile*, input password (if not provided). Press **Next>** to display the Salesforce Objects selection page.

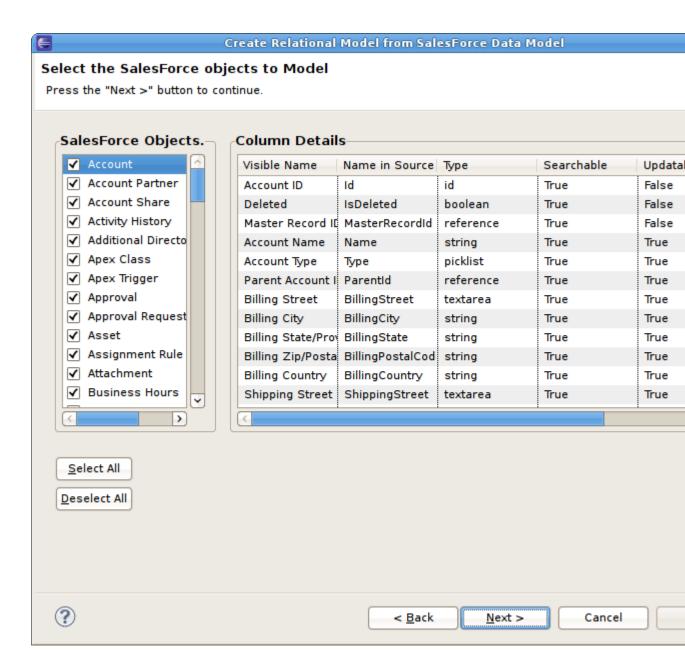


Figure 5.34. Select Salesforce Objects Dialog

 Step 5 - On the Target Model Selection page, specify the target folder location for your generated model, a unique model name and select desired import options. Press Next> (or Finish if enabled).

	Create Relatio	nal Model fro	n SalesForce D	ata Model	
arget Model	Selection				
Press the "Next	>" button to continue	or the "Finish" b	outton to finish.		
Select Target R	elational Model				
Model Name:	BillingSourceModel				Browse
Location:	SalesforceImportTest				Browse
Select Import C	ptions				
Model aud	it fields.				<u>^</u>
Selecting this	option will cause the i	importer to mod	lel the 'Audit Field	s' for each SalesFo	orce object.
(Created By,	Created Date, Last Mo	dified By, Last N	lodified Date, Sys	tem Modification T	imestamp)
🗌 Do not gat	her Cardinalities				
Selecting this	option will stop the in	porter from cal	culating and setti	ng the cardinalities	metadata
For large sale	sforce applications thi	is can become a	a long running op	eration.	
🗌 Gather Co	lumn Distinct Value Co	unt			
Selecting this	option will cause the i	importer to calc	ulate and set the	distinct value coun	t metadata
This will scan	the data in each field	individually and	can be a very lor	ng running operatio	in.
🗌 Set name	to SalesForce label.				
Selecting this	option will cause the i	importer to set	the name metada	atain the genera	ted model 1
By default the	importer uses the int	ernal data nam	e because SalesF	orce labels are oft	en invalid f
🗌 Create a p	rocedure for the GetU	pdated operatio	in.		
Selecting this	option will cause the i	importer to crea	te a procedure fo	or the getUpdated o	operation ir
🗌 Create a p	rocedure for the GetD	eleted operation	n.		
Selecting this	option will cause the i	importer to crea	te a procedure fo	or the getDeleted o	peration in
					>
?		< <u>B</u> ack	<u>N</u> ext >	Cancel	<u>F</u> inish

Figure 5.35. Target Model Selection Dialog

• Step 5a - If you are updating an existing relational model, the next page will be **Review Model** Updates page. Any differences. Press **Finish** to create your models and tables.

Create Relational Model from SalesForce Data Model	X
Review Model Updates	×.
Review the changes that will be applied to your model to bring it up to date with the source metadata. Uncheck any changes you do not wish to apply.	
Only in Old Model (OLD): ① Changes (스): ① Only in New Model (NEW): ① 무 압	
√ file:/TestSF/1314365845645_temp.xmi	
✓ NEW salesforce	
Selection Details	1
salesforce - this selection is an Addition	
? < Back	

Figure 5.36. Review Model Updates Dialog

When finished, the new or changed relational model's package diagram will be displayed showing your new tables.

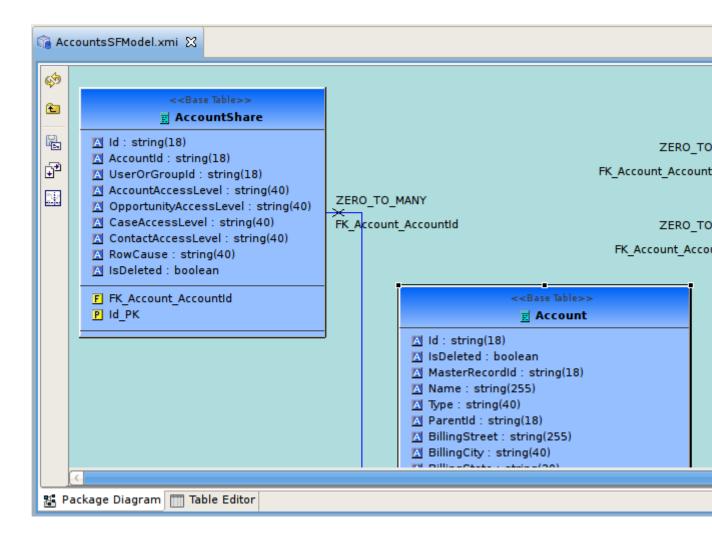


Figure 5.37. New Saleforce Tables Diagram

5.7. Import Metadata From Text File

- The Teiid Designer provides various import options for parsing comma delimited text file metadata into models. This is accomplished via the Import > Teiid Designer > Designer Text File >> Source or View Models option.
 - Step 1 In Designer choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teild Designer > Designer Text File >> Source or View Models and click Next>
- Step 3 Select an import type via the drop-down menu shown below.

elect Impo	rt Type	Alb.
Select the type	of text file import desired.	
Import Type: (Relational Model (XML Format)	~
Purpose: Imp	Relational Model (XML Format)	
relational sou	Relational Tables (CSV Format)	
	Relational Virtual Tables (CSV Format)	
Sample File Fo	Datatypes (CSV Format)	

Figure 5.38. Import Wizard

- These steps are required for each type are defined below:
 - Section 5.7.1, "Import Relational Model (XML Format)"
 - Section 5.7.2, "Import Relational Tables (CSV Format)"
 - Section 5.7.3, "Import Relational View Tables (CSV Format)"
 - ???

5.7.1. Import Relational Model (XML Format)

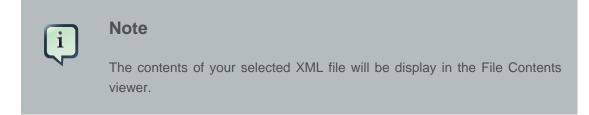
- To create relational tables from imported xml text file metadata:
 - Perform Steps 1 through 3 (above) and select the Relational Model (XML Format) import type, then click Next >

🗧 🛛 Import Metadata From Text File 🛛 🕅
Select Import Type
Select the type of text file import desired.
Import Type: Relational Model (XML Format)
Import Type: Relational Model (XML Format)
Purpose: Imports relational tables, views, procedures and indexes from an XML File into a relational source model.
Sample File Format:
<pre><relational-model> </relational-model></pre>

Figure 5.39. Select Import Type - Relational Model (XML Format)

• Perform **Steps 4** - On the next page, select the XML file on your local file system via the **Browse...** button. Select a target model to which the imported relational objects will be added

via the second **Browse...** button. The dialog allows selecting an existing relational model or creating a new model.



Click Finish to create your new model.

Select Source Text File and Target Relational Model. Select Metadata File to Import and Destination Model or model folder Select Source Text File r/Release_7.6/planning/SampleRelationalXML.xml relations Target Location ProductsFromXMLText.xmi Browse Target Location ProductsFromXMLText.xmi Browse File Contents <pre> </pre> <pre> </pre> <pre> </pre> <pre> State Contents </pre> <pre> </pre>	📕 Import Metadata From Text File	×
Select Source Text File pr/Release_7.6/planning/SampleRelationalXML.xml Browse Target Location ProductsFromXMLText.xmi Browse File Contents <pre> <pre> </pre> </pre> <pre> Select Source Text File productsFromXMLText.xmi Browse File Contents </pre> <pre> </pre> <pre> </pre> <pre> Select Source Text File productsFromXMLText.xmi Browse File Contents </pre> <pre> </pre> <pre> </pre> <pre> </pre> <pre> </pre> <	Select Source Text File and Target Relational Model.	19h
Target Location ProductsFromXMLText.xmi Browse File Contents <pre> </pre> </td <td></td> <td></td>		
Target Location ProductsFromXMLText.xmi Browse File Contents <pre> </pre> </td <td>· .</td> <td>\checkmark</td>	· .	\checkmark
Target Location ProductsFromXMLText.xmi Browse File Contents <pre> </pre> </td <td></td> <td></td>		
File Contents <pre></pre> <pre></pre> <pre></pre> <pre></pre> <pre>Contents </pre> <pre></pre> <pre></pre> <pre>Contents </pre> <pre></pre> <pre>Contents </pre> <pre></pre> <pre></pre> <pre>Contents </pre> <pre> </pre> <pre>Contents </pre> <pre></pre> <pre>Contents </pre> <pre> </pre> <pre>Contents </pre> <pre> </pre> <pre> </pre> <pre>Contents </pre> <pre> </pre> </pre> </pre> </pre> </pre> </pre> <pre> </pre> </pre> <pre> <pre> <pre> </pre> </pre> </pre> <pre> <pre> <pre> <pre> <pre> <pre> </pre> <td>Select Source Text File r/Release_7.6/planning/SampleRelationalXML.xml</td><td>B<u>r</u>owse</td></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre></pre>	Select Source Text File r/Release_7.6/planning/SampleRelationalXML.xml	B <u>r</u> owse
File Contents <pre></pre> <pre></pre> <pre></pre> <pre></pre> <pre>Crelational-model name="myRelationalModel" xmlns="http://www.teiid.org" xsi:schemaLocation="http://www.teiid.org file:///home/blafond/Documents/Teiid_Designer/Rele <pre></pre> <pre></pre> <pre></pre> <pre></pre> <pre>// table name="ProductData" nameInSource="dbo.products.ProductData" </pre> <pre>description="Basic stocks or bond data table second line &#x cardinality="100" materialized="true" supportsUpdate="true" system="true"> </pre> <pre></pre> <pre></pre> <pre>// table name="ProductID" description="Unique ID for this product" nameInSource="dbo.products.ProductData.ProductID" datatype="varchar" length="11" nullable="NO_NULLS"> </pre> </pre> <pre></pre> <pre></pre> <pre>// column name="ProductType" nameInSource="dbo.products.ProductData.Pro</pre>		
<pre><?xml version="1.0" encoding="UTF-8"?> <relational-model 100"="" description="Basic stocks or bond data table second line &#x cardinality=" materialized="true" name="myRelationalModel" nameinsource="dbo.products.ProductData" productdata"="" supportsupdate="true" system="true" xmlns="http://www.teiid.org" xsi:schemalocation="http://www.teiid.org file:///home/blafond/Documents/Teiid_Designer/Rele <table name="> <column datatype="varchar" description="Unique ID for this product" length="11" name="ProductID" nameinsource="dbo.products.ProductData.ProductID" nullable="NO_NULLS"> </column> <column datatype="varchar" description="Unique ID for this product" length="11" name="ProductID" nameinsource="dbo.products.ProductData.ProductID" nullable="NO_NULLS"> </column> <</relational-model></pre>	File Contents	
<pre>xsi:schemaLocation="http://www.teiid.org file:///home/blafond/Documents/Teiid_Designer/Rele description="Basic stocks or bond data table second line &#x
cardinality="100"
materialized="true" supportsUpdate="true" system="true">
<column name="ProductID" description="Unique ID for this product"
nameInSource="dbo.products.ProductData.ProductID" datatype="varchar"
length="11" nullable="NO_NULLS">
</column>
<column name="ProductName" nameInSource="dbo.products.ProductData.ProductData.ProductData.ProductData.ProductData.ProductData.ProductDatatype="varchar"
datatype="varchar" length="60" nullable="NULLABLE">
</column>
<column name="ProductType" nameInSource="dbo.products.ProductData.Product
datatype="varchar" length="15" nullable="NULLABLE">
</column>
<column name="ProductType" nameInSource="dbo.products.ProductData.Product
datatype="varchar" length="15" nullable="NULLABLE">
</column>
<column name="ProductType" nameInSource="dbo.products.ProductData.Product
datatype="varchar" length="15" nullable="NULLABLE">
</column>
<column name="ISSUFR" nameInSource="dbo.products.ProductData.ISSUFR"</pre></td><td><?xml version="1.0" encoding="UTF-8"?></td><td>6</td></tr><tr><td>description="Basic stocks or bond data table second line &#x
cardinality="100"
materialized="true" supportsUpdate="true" system="true">
<column name="ProductID" description="Unique ID for this product"
nameInSource="dbo.products.ProductData.ProductD" datatype="varchar"
length="11" nullable="NO_NULLS">
</column>
<column name="ProductName" nameInSource="dbo.products.ProductData.Produc
datatype="varchar" length="60" nullable="NULLABLE">
</column>
<column name="ProductType" nameInSource="dbo.products.ProductData.Product
datatype="varchar" length="15" nullable="NULLABLE">
</column>
<column name="ProductType" nameInSource="dbo.products.ProductData.Product
datatype="varchar" length="15" nullable="NULLABLE">
</column>
<column name="ProductType" nameInSource="dbo.products.ProductData.Product
datatype="varchar" length="15" nullable="NULLABLE">
</column></td><td><relational-model name="myRelationalModel" xmlns="http://www.teiid.org"</td><td></td></tr><tr><td><pre>description="Basic stocks or bond data table &#xa; second line &#xa;&#x9;&#x cardinality="100" materialized="true" supportsUpdate="true" system="true"> <column name="ProductID" description="Unique ID for this product"</td><td>xsi:schemaLocation="http://www.teiid.org file:///home/blafond/Documents/Teiid_Des</td><td>signer/Rele</td></tr><tr><td><pre>cardinality="100" materialized="true" supportsUpdate="true" system="true"> <column name="ProductID" description="Unique ID for this product" nameInSource="dbo.products.ProductData.ProductID" datatype="varchar" length="11" nullable="NO_NULLS"> </column> </column name="ProductName" nameInSource="dbo.products.ProductData.ProductData.ProductDatatype="varchar" length="ength="60" nullable="NULLABLE"> </column name="ProductType" nameInSource="dbo.products.ProductData.ProductDatatype="varchar" length="160" nullable="NULLABLE"> </column name="ProductType" nameInSource="dbo.products.ProductData.ProductData.ProductDatatype="varchar" length="160" nullable="NULLABLE"> </column name="ProductType" nameInSource="dbo.products.ProductData.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE"> </column name="ProductType" nameInSource="dbo.products.ProductData.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE"> </column name="ProductType" nameInSource="dbo.products.ProductData.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE"> </column name="ProductType" nameInSource="dbo.products.ProductData.ProductData.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE"> </column> </column> </column></td><td><table name="ProductData" nameInSource="dbo.products.ProductData"</td><td></td></tr><tr><td><pre>materialized="true" supportsUpdate="true" system="true"></td><td>description="Basic stocks or bond data table second line </td><td> &#x</td></tr><tr><td><pre><column name="ProductID" description="Unique ID for this product" nameInSource="dbo.products.ProductData.ProductID" datatype="varchar" length="11" nullable="NO_NULLS"> </column> </column name="ProductName" nameInSource="dbo.products.ProductData.ProductData.ProductDatatype="varchar" length="60" nullable="NULLABLE"> </column> </column> </column> </column name="ProductType" nameInSource="dbo.products.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE"> </column> </column> </column="ProductType" nameInSource="dbo.products.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE"> </column> </column> </column="ProductType" nameInSource="dbo.products.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE"> </column> </column="ProductType" nameInSource="dbo.products.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE"> </column> </column="ProductType" nameInSource="dbo.products.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE"> </column> </column></td><td>cardinality="100"</td><td></td></tr><tr><td>nameInSource="dbo.products.ProductData.ProductID" datatype="varchar"
length="11" nullable="NO_NULLS">
</column>
<column name="ProductName" nameInSource="dbo.products.ProductData.ProductData.ProductDatatype="varchar" length="60" nullable="NULLABLE">
</column>
<column name="ProductType" nameInSource="dbo.products.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE">
</column>
<column name="ProductType" nameInSource="dbo.products.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE">
</column>
</column name="ProductType" nameInSource="dbo.products.ProductData.ProductDatatype="varchar" length="15" nullable="NULLABLE">
</column>
</column></td><td></td><td></td></tr><tr><td><pre>length="11" nullable="NO_NULLS"></td><td></td><td></td></tr><tr><td></column> </column> </column name="ProductName" nameInSource="dbo.products.ProductData.Pro</td><td></td><td>varchar"</td></tr><tr><td><pre><column name="ProductName" nameInSource="dbo.products.ProductData.Product</td><td></td><td></td></tr><tr><td>datatype="varchar" length="60" nullable="NULLABLE"> </column> </column name="ProductType" nameInSource="dbo.products.ProductData.Product</td><td></td><td></td></tr><tr><td></column>
<column name="ProductType" nameInSource="dbo.products.ProductData.Product
datatype="varchar" length="15" nullable="NULLABLE">
</column>
<column name="ISSUER" nameInSource="dbo_products_ProductData_ISSUER"</td><td></td><td>ata.Produ</td></tr><tr><td><column name="ProductType" nameInSource="dbo.products.ProductData.Product
datatype="varchar" length="15" nullable="NULLABLE">
</column>
<column_name="ISSUER" nameInSource="dbo_products_ProductData_ISSUER"</td><td></td><td></td></tr><tr><td>datatype="varchar" length="15" nullable="NULLABLE"> </column> </column_name="ISSUER" nameInSource="dbo_products_ProductData_ISSUER"</td><td></td><td>ta Product</td></tr><tr><td></column>
<column_name="ISSUER"_nameInSource="dbo_products_ProductData_ISSUER"</td><td></td><td>a. Froduct</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td><column_name="ISSUER"_nameInSource="dbo_products_ProductData_IS</td><td>SUFR"</td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td>< <u>Back</u> <u>Next</u> > Cancel</td><td>Finish</td></tr></tbody></table></pre>		

Figure 5.40. Select Source Text File and Target Relational Model Page

If the target model contains named children (tables, views, procedures) that conflict with the objects being imported, a dialog will be displayed giving you options on how to proceed including: replacing specific existing objects, creating new same-named objects or cancel import entirely.

🗧 🛛 Duplicate Objects Exist 🛛 🕷
Options
 Replace Existing Objects (See list below)
 Create New Objects Anyway (results in validation errors)
 Cancel Import
Duplicate Objects
getProductinfo
✓ ProductData
ProductiDindex
✓ ProductSymbols
ProductView
Select All Deselect All
Cancel OK

Figure 5.41. Duplicate Objects Dialog

5.7.2. Import Relational Tables (CSV Format)

- To create relational tables from imported text file metadata:
 - Perform Steps 1 through 3 (above) and select the Relational Tables (CSV Format) import type, then click Next >

8	Import Metadata From Text File
	I mport Type ne type of text file import desired.
Import	Type: Relational Tables
A samı	 Imports relational schema, catalogs, tables, columns, and indexes from a CSV text file into a relational schema, Catalog, and Table data is expected to be of the form: > TYPE (i.e. SCHEMA, CATALOG, or TABLE), Name, "Description" (Optional), Location (Optional) > Locations are of the form: ProjectName/FolderName/ModelName/SchemaName. > If the project, folder, model or schema/catalog containers do not exist, they will be created. 2) The Column data is expected to be in the form: > COLUMN, ColumnName, JDBCType, Length, "Description" 3) Column data rows for each table must appear immediately following the table data row. 4) The Index data is expected to be of the form: > INDEX, IndexName, Type, Uniqueness, Tablespace, Column 5) Index data rows must appear immediately following the table column data rows.
· · ·	G, Catalog_1, "Catalog_1 Description", Project_1/MyModel_1
	A, Schema_1, "Schema_1 Description", Project_1/MyModel_1/Catalog_1
SCHEM	, schema_1, schema_1 Description, Project_1/MyModel_1/catalog_1

Figure 5.42. Select Import Type - Relational Tables (CSV Format)

• **Step 4** - In the next page, you'll need to provide a source text file containing the metadata formatted to the specifications on the previous page.

Import Metadata From Text File	
Select Source Text File and Target Relational Model. Select CSV File to Import and Destination Model or model folder	
Select Source Text File	B <u>r</u> owse
Target Location CustomerInfo.xmi	Browse
File Contents	
Options	
Use default datatype: string(255)	
(?) < <u>Back</u> <u>Next</u> > Cancel	Finish

Figure 5.43. Select Source Text File and Target Relational Model

- **Step 5** Select an existing relational model as the target location for your new relational components using the **Browse...** button to open the Relational Model Selector Dialog. Select a relational model from your workspace or specify a unique name to create a new model.
- Step 6 Select any additional options and choose Finish.

5.7.3. Import Relational View Tables (CSV Format)

- To create relational virtual tables from imported text file metadata:
 - Perform Steps 1 through 3 (above) and select the Relational Virtual Tables (CSV Format) import type, then click Next >

(9)	Import Metadata From Text File
Select Import Type	
Select the type of text file import desired.	
Import Type: Relational Virtual Tables	
Purpose: Imports virtual relational tables and proce Table data is expected to be of the form: >> TableName, "SQL Statement", De A sample of typical input data is shown below: Sample File Format:	edures from a CSV text file into a relational virtual model.
VTable1, "SELECT * FROM PMod1.PTab1, PMod1.PTab	le2", "Descrip, no embedded quotes"
VTable2, "SELECT * FROM PMod1.PTab2", "Descrip 2	with quotes ""xxx"""
VTable3, "SELECT * FROM PMod1.PTab3", "Descrip 3	u

Figure 5.44. Select Import Type - Relational Virtual Tables (CSV Format)

• **Step 4** - In the next page, you'll need to provide a source text file containing the metadata formatted to the specifications on the previous page.

9	Import Metadata From Text File 🛛 🛞
	Text File and Target Virtual Relational Model
Select Source Te	ext File Browse
Target Location	Browse
File Contents	
?	<u>Back</u> <u>Next</u> > Cancel <u>Finish</u>

Figure 5.45. Select Source Text File and Target Virtual Relational Model

- Step 5 Select an existing relational virtual model as the target location for your new model components using the **Browse...** button to open the Virtual Model Selector Dialog. Select a virtual relational model from your workspace or specify a unique name to create a new model.
- Step 6 Select Finish.

5.8. Import Data From REST Service

 In addition to SOAP web services, Teiid supports REST web services as data sources. Using the steps below you will define your REST web service data source, select the elements to include in your generated view table and generate a source model containing the required Teiid procedure.

To import from your target REST web service source:

• Step 1 - In Model Explorer choose the File > Import action in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teiid Designer > WSDL File or URL >> Source and View Model (SOAP) and click Next>
- **Step 3** On the next page select an existing Web Service Connection Profile from the list, or click the **New** Button to create a new profile.

😕 💷 Create Relational Model from Web Service	
Source and WSDL Operations Definition Press the "Next >" button to continue.	S
Connection Profile	
GlobalWeather_WSDL_SOAP	▼ New Edit
WSDL URL or Location:	
http://www.webservicex.net/globalweather.asmx?WSDL	Validate WSDL
End Point End Point Name GlobalWeatherSoap End Point	URI http://www.webservicex.net/globalweather.asmx
Binding SOAP11 S	ervice Mode PAYLOAD 💌
Select All Operation Deselect All Image: Construction Image: Construction <	
Selection Details:	
service: GlobalWeather id: {http://www.webserviceX.NET}GlobalWea input message: GetWeatherSoapIn output message: GetWeatherSoapOut fault names: none	atherSoap.GetWeather
?	< Back Next > Cancel Finish

Figure 5.46. WSDL Source Selection

• Step 4 - Select individual Web Service Operations to model. The default behavior of this page selects all available operations in the tree. Operations can be de-selected if they are not being modeled. The Selection Details panel displays static information about the operation such as the names of the input and output messages, and faults thrown by the operation.

Click Next >

• Step 5 - The next page entitled Model Definiton requires both a model location (i.e. folder or project) and a valid model name for both source and view models. Use the **Browse...** button to select existing folders or models. Click **Next>** when all the information is defined.

9 🗆 C	reate Relational Model from Web Service
odels Def Il inputs C	finition DK. Click 'Next>' to define custom procedures.
Target P	roject
Select Op	en Model Project >> WeatherInfo
Source	Model Definition
Location	WeatherInfo
Name	GlobalWeather.xmi
Status	
	del Definition
Location	WeatherInfo
Name	GlobalWeatherView.xmi
Status	
View mo generate	del GlobalWeatherView.xmi does not exist and will be created and contain your ed procedures.
	< Back Next > Cancel Finish

Figure 5.47. WSDL Source Selection

Step 6 - The next page, titled Data Source Definition Options, provides the option for defining a unique JBoss JNDI name for the deployed data source that this imported model will prepresent. This name will be stored as a property on your model and changeable via the Modeling > Set JBoss JNDI Data Source Name action. If there you have a default server defined and running, then Auto-Create Data Source check box will be enabled. On completion of this import, the wizard will check for an existing deployed data source with this JNDI name and create one if needed.

When finished with this page, click Next>.

Data Source Definition Options
Specify a unique JBoss JNDI name for this data source
JBoss Data Source Information
JNDI Name GlobalWeather
Auto-create Data Source
Server Unavailable
A data source cannot be created unless connected to a server
Please define and start a server then re-launch this importer
Cancel Finish

Figure 5.48. Data File Source Selection Page

Step 7 - The next page entitled Operations Type Selection gives you the option to define custom procedures (Recommended)shown in Step 8 below. or generate default procedures you can edit later. button to select existing folders or models. If customer procedure is selected, click Next> to proceed or Finish if default option is selected

S Create Relational Model from Web Service
Operations Type Selection All inputs OK. Click 'Next>' to define custom procedures.
Procedure Generation Options Image: Construction of the second
(?) < Back Next > Cancel Finish

Figure 5.49. WSDL Source Selection

• **Step 8** - This wizard generates both request and response procedures that are used in the queryable wrapped procedure. The next page, Procedure Definition, provides the means to define the details of your request and response structures.

In the Request tab, select and double-click the schema elements you wish to be input parameters for your request. These will be added to the Element Info panel and the resulting generated SQL statement will be updated to reflect the new element.

Note

i

The BODY and HEADER tabs which exist on both the Request and Response tabs. If the selected service mode for this procedure is set to MESSAGE, the HEADER tab will be enabled and allow you to define the SOAP header variables utilizing the same schema tree.

Select the Response tab and create the response procedures result set columns in the same way.

Repeat this process for all operations by changing the selection target operation via the **Operations** selector at the top.

Create Relational Model from Web Service	
rocedure Definition Press the "Finish" button to finish.	Ð
Operations	
GetWeather	▼
Overwrite existing procedures for this operation	
Request Response Wrapper Procedure	
Generated Procedure Name GetWeather_request	
BODY HEADER	Element Info
 ♥ end of the end of	Add Delete Up Down
Generated SQL Statement BEGIN SELECT XMLELEMENT(NAME "tns:GetWeather", XMLNAMESPA XMLELEMENT(NAME "tns:CityName", GlobalWeather XMLELEMENT(NAME "tns:CountryName", GlobalWea END	ACES('http://www.webserviceX.NET' AS tns), rView.GetWeather_request.CityName), atherView.GetWeather_request.CountryName)) AS xml_out;
?	< Back Next > Cancel Finish

Figure 5.50. Procedure Definition Page

• **Step 7** - Click **Finish**. After generation the new models can be found in the specified location in your workspace.

5.9. Import Data From SOAP Service

In addition to REST web services, Teiid supports SOAP web services as data sources. This
importer is accessed by launching Eclipse's "Import..." action and selecting the "Teiid Designer
> WSDL File or URL >> Source and View Model (SOAP)" option. Web Services Connection
Profile defined by a WSDL file in your workspace or defined by a URL. Designer will interpret
the WSDL, locate any associated or dependent XML schema files, generate a physical model
to invoke the service, and generate virtual models containg procedures to build and parse the
XML declared as the service messages.

Using the steps below you will define your SOAP web service data source, select the elements to include in your generated view table and generate a source model containing the required Teiid procedure.

To import from your target SOAP web service source:

• Step 1 - In Model Explorer choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

 Step 2 - Select the import option Teild Designer > Web Service Source >> Source and View Model (SOAP) and click Next>

🛞 🗇 Import	
Select Choose import source.	Ľ
Select an import source:	
type filter text	×
 Teiid Designer DDL File (General) >> Source or View Model DDL File (Teiid) >> Source or View Model Designer Text File >> Source or View Models File Source (Flat) >> Source and View Model File Source (XML) >> Source and View Model JDBC Database >> Source Model JDBC Database >> Source Model Salesforce >> Source Model Salesforce >> Source Model Web Service Source >> Source and View Model (REST) Web Service Source >> Source and View Model (SOAP) WSDL File or URL >> Web Service Model XML Schemas 	
Cancel	Finish

Figure 5.51. Import from SOAP Web Service Source

• Step 3 - On the first page of the wizard, select an existing or previous connection profile from the drop-down selector or press New... button to launch the New Connection Profile dialog or Edit... to modify/change an existing connection profile prior to selection.

Create Re	elational Model from Web Service	
rce and WSDL (Operations Definition	
ess the "Next >" bu	outton to continue.	
Connection Pro	ofile	
CountryInfo_SO	DAP	‡ New
VSDL URL or L	Location:	
http://www.oor	rsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL	Validate
nd Point		
ind Point Name	CountryInfoServiceSoap End Point URI http://www.oorsprong.org/websamples.countryin	to/CountryInfoServio
nding SOAP1	11 Service Mode PAYLOAD	
alact the decir	ired WSDL Operations	
Select the desil		
Select All	Operation	
Deselect All	CapitalCity	
	CountryFlag	
	CountryISOCode	
election Details	le:	
Operation: Capita		
binding:	CountryInfoServiceSoapBinding	
service:	CountryInfoServiceSoap CountryInfoService	
id: input messag	{http://www.oorsprong.org/websamples.countryinfo}CountryInfoServiceSoapBinding.CapitalCil ge: CapitalCitySoapReguest	ty
(III III	
	< Back Next >	Cancel

Figure 5.52. Source and WSDL Operations Definition

Name your new SOAP Connection Profile and click the Next >

	Properties
rofile Name	CountryInfo_SOAP
rofile Description	
onnection URL or File Path	http://www.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL
he URL defined for this con	Workspace Eile System URL Description nection profile provides the hook to inject the 'Endpoint' property into a source model via the
Modeling > Set Connection	Description
Modeling > Set Connection	Description nnection profile provides the hook to inject the 'Endpoint' property into a source model via the Profile' action. This 'Endpoint' value is required by the 'ws' translator.
Modeling > Set Connection	Description nnection profile provides the hook to inject the 'Endpoint' property into a source model via the Profile' action. This 'Endpoint' value is required by the 'ws' translator.
Modeling > Set Connection	Description nnection profile provides the hook to inject the 'Endpoint' property into a source model via the Profile' action. This 'Endpoint' value is required by the 'ws' translator.

Figure 5.53. New SOAP Connection Profile

Set your Connection URL from either your workspace, your local file system or define a remote URL. Click **Next >** and select an endpoint and binding type, click **Finish**.

😣 🗆 New connec	tion profile			
Web Service End Po	int Property			12
Click Next or Finish				
		Properties		
Profile Name	CountryInfo_SOAP			
Profile Description				
End Point	CountryInfoServiceSoap			ţ)
Binding Type	SOAP11			
Sonnect when the	e wizard completes			
Connect every tin	ne the workbench is <u>s</u> tarted			
?		< Back Next >	Cancel	Finish

Figure 5.54. New SOAP Connection Profile Endpoint Definition

Once the connection profile is defined, you can view the WSDL URL and validate it via the **Validate WSDL** button. Select Service Model and desired WSDL Operations and click **Next >**

 Step 4 - On the Models Definition page, define your source and view model names and locations and click Next >

😸 🗆 🛛 Cr	reate Relational Model from Web Service
Models Def	inition
All inputs O	IK. Click 'Next>' to define custom procedures.
Source	Model Definition
Location	Customer_Accounts
Create Relational Model from Web Service Models Definition All inputs OK. Click 'Nexts' to define custom procedures. Source Model Definition Location Location Customer_Accounts	
Source m	
View Mo	del Definition
Location	Customer_Accounts
Name	CountryInfoServiceView.xmi
Procedu	re Generation Options
Define us generates O Defa	er-specified request and response procedures from your WSDL schema elements. This option also s a wrapper procedure allowing you to execute your request and response. ault Procedures
Generate wrapper p	default request and response procedures. A new procedure will be generated for each complex type. The procedure must be manually created with this option.
?	< Back Next > Cancel Finish

Figure 5.55. Models Definition

• Step 5 - On the Procedure Definition page, defined your request and response elements for each operation selected previously. When all operations are sufficiently defined click **Finish**

Careta Dalational Madel Gran Web Consist	
Create Relational Model from Web Service	
rocedure Definition	
Press the "Finish" button to finish.	
Operations	
CapitalCity	* *
Overwrite existing procedures for this operation	
Request Response Wrapper Procedure	
Generated Procedure Name CapitalCity_request	
BODY HEADER	
Schema Contents	Element Info
▼ e CapitalCity	Add
▼ •••• sequence	Delete CountryISOCode
▼ e sCountryISOCode	
a string	Up
	Down
Generated SQL Statement	
CREATE VIRTUAL PROCEDURE	
BEGIN SELECT	
XMLELEMENT(NAME "tns:CapitalCity", XMLNAMESPACES('http websamples.countryinfo' AS tns),	p://www.oorsprong.org/
XMLELEMENT(NAME "tns:sCountryISOCode", CountryInfoServiceView.CapitalCity_request.sCountryISOCode))	ASymption the
END	ASXIIIL_OUL,
	J
	Next > Cancel Finish
< Back	Next > Cancel Finish

Figure 5.56. Procedure Definition

In the **Model Explorer** you can see the importer created the following a single **physical model** containing a single procedure called invoke. This model and procedure correspond to the single port declared in the WSDL.

A single **view model** was also created containing your new procedures named after the operations declared in the WSDL. For each operation a wrapper procedure was created which can be previewed in Designer. Below is an example dependency diagram showing the sources for the wrapper procedure as request, response procedures and the invoke() source procedure.

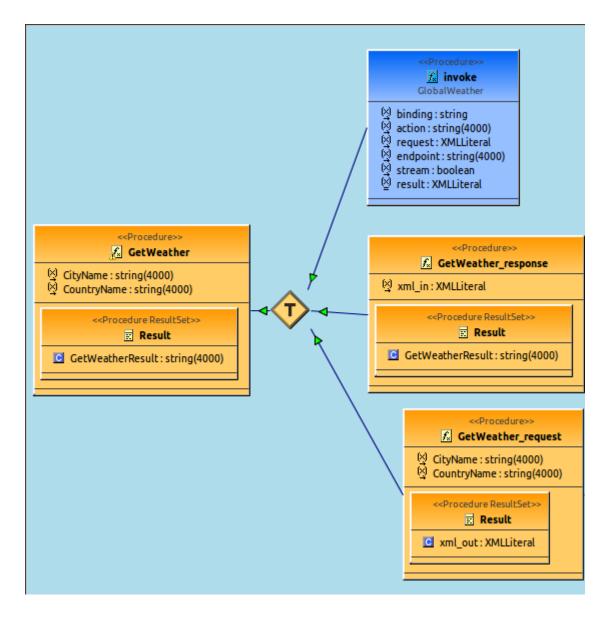


Figure 5.57. Example Web Services Wrapper Procedure

5.9.1. Circular References in WSDL Schemas

It is possible for a WSDL schema to either contain a very deep set of XML type references or indeed for such references to be circular. This is legal in the WSDL schema but can make

processing the schema in Designer difficult. If left unchecked such circular references can result in a JVM **StackOverFlow** exception and exiting of the application.

To mitigate this possibility a depth limit of 750 references has been introduced. Should the depth exceed this limit then a warning is displayed and further processing of that fragment of the schema will end. It may be the case that the reference in question is not circular but just very deep so in such a case it is possible to increase the depth limit by setting the JVM property **WsdlSchemaHandlerRecursiveDepth** to a larger value, eg. -D WsdlSchemaHandlerRecursiveDepth=800. This should only be used with caution as on some systems it is possible the JVM throws a **StackOverFlow** exception before the new depth limit is reached.

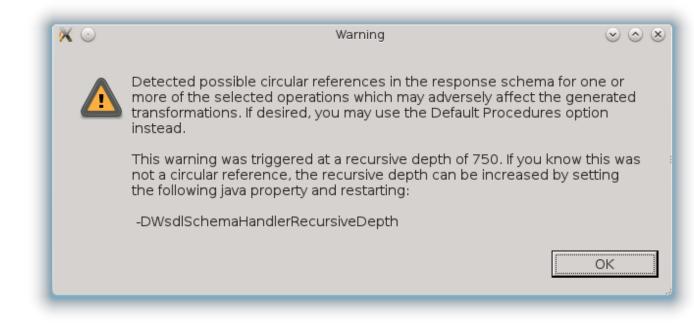


Figure 5.58. Warning message displayed if depth limit has been reached

5.10. Import WSDL Into Web Service

You can create a **Web Service** model by selecting a **WSDL** file in your workspace, importing **WSDL** files from the file system or by defining a URL. The Teiid Designer will interpret the **WSDL**, locate any associated or dependent **XML Schema** files, generate an **XML View** of the schema components and create a **Web Service** model representing the interfaces and operations defined in the **WSDL**.

- There are three options for selecting the WSDL for your Web Service generation
 - Workspace Location
 - File System Location

• URL

Detailed steps for each of these options is described below, as well as a description of how the wizard handles **WSDL** errors.

5.10.1. Import WSDL From Workspace Location

- You can create a Web Service model by selecting a WSDL file from your workspace.
 - Step 1 Choose the File Import choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teiid Designer > WSDL File or URL >> Web Service Model option shown below and click Next>
- Step 3 Input a valid name for your Web Service model and select the Workspace... button. Locate your workspace WSDL file in the selection dialog and click OK>. Click Next> to continue.

Screate Web Service from WSDL File	×
WSDL File Selection Press the "Next >" button to continue.	
Press the Next > bullon to continue.	
Web Service Model Name OnHandWS	
Workspace [File System] URL	
WSDL Files	6
Ider/Test In JBDS/JBDS_H45/workspace/WSDLImportProject/OnHand	l.wsdl
<	\rightarrow
Cancel Enci	sh

Figure 5.59. WSDL File Selection Dialog

8	WSDL File Selection	×
Choose a WSDL	file to add:	
🗢 📑 WSDLIm	portProject	
🖉 OnHa	ind.wsdl	
?	Cancel OK	

Figure 5.60. WSDL File Workspace Selection Dialog

Note

i

- If no WSDL is selected or specified then the importer will only create an empty Web Service model. No XML Schema or XML View models will be generated.
- Any referenced files (**WSDL**s or schemas) must either be embedded in the **WSDL** file or exist on your file system.
- Step 4 The next page is titled Namespace Resolution. This page identifies successful and errant WSDL namespace resolution. The main WSDL document will essentially always be resolved, since the workspace file chooser is used to obtain the path. Problems will occur when the main WSDL file imports other WSDL files that cannot be resolved. If no errors, select Next to proceed, or Finish (if enabled) to complete with default options.

		tion olved. Select "Next" to modify the created schemas workspace	3
lam	espaces		4
4	Namespace	Path	
	http://dladb07/	/home/blafond/Test Designer Folder/example files/wsdls/OnHand.ws	dl

Figure 5.61. Namespace Resolution Dialog

• Step 5 - The next page WSDL Operations Selection allows customizing the resulting content of your Web Service model by selecting/deselecting various operations and interfaces in the following dialog.

	Create Rela	tional Model f	rom Web Servic	e	
SDL Operations	Selection				
ress the "Finish" butt	on to finish.				3
Select the desir	ed WSDL Operati	ons			
🗢 🗹 📄 Share	M_Real_Time_Data_De _BSM_Real_Time_Data_	Design_Server_	RealTimeData_po	rtPort0	Select All
	are_BSM_Real_Time_Da	ata_Design_Serv	er_RealTimeData	_portBinding	Deselect All
✓ 585	OnHand				
Selection Details:					
OnHand [Operation] id: input message: output message: fault names:	{http://dladb07, _OnHandInput OnHandOutput none	/}Share_BSM_Re	eal_Time_Data_De	sign_Server_Re	alTimeData_p
<.					>
0					

Figure 5.62. Namespace Resolution Dialog

 Step 6 - The next page is titled Schema Workspace Location Selection. This page lists all schemas imported by the WSDL (along with any dependent schemas referenced within schemas) as well as schemas embedded in the WSDL and indicates whether or not they are resolvable. All resolved schemas will be created in a separate file and added to the workspace. The editor panel allows you to change the default file name of the new schema file(s).

If no errors, select Next to proceed, or Finish to complete with default option

		Create Web Ser	vice from WSDI	L File	
Sche	ma Workspace Location	Selection			
All w	orkspace schema locations are v	alid. Click "Next" to cont	tinue.		
Sch	ema Location Information				
4	Namespace				Source File
	http://localhost/Share_BSM_Re	al_Time_Data/Design_Se	erver/RealTimeDa	ta_port/OnHand	/home/blafond/To
K	Editor - http://localhost/Share_BS	M_Real_Time_Data/Desi	gn_Server/RealTi	meData_port/On	Hand
	Folder:	/WSDLImportProject			
	Name (without file extension):	OnHand			
0)		< <u>B</u> ack	Next >	Cancel
•				<u>n</u> ext >	

Figure 5.63. Namespace Resolution Dialog

• Step 7 - The last page titled XML Model Generation allows you to change the name of the XML View model if the Generate virtual XML document model is checked. Input desired name or use the default name provide. Select Finish to complete.

@	Create Web Service from WSDL File	×
	I Generation nerate the Web service model	8
🗹 Generate	e virtual XML document model	
Location:	/WSDLImportProject	≎
XML Model:	OnHandWSResponses	
?	< Back Next > Cancel	<u>Finish</u>

Figure 5.64. Namespace Resolution Dialog

In order to successfully generate Web Services from WSDL, the WSDL must be error free. WSDL validation is performed during *Step 3* above. If errors do exist, a error summary dialog will be displayed (shown below) and you will not be able to *Finish* the wizard until the WSDL problems are fixed or you re-import and select a valid WSDL file.

Screate Web Service from V	WSDL File
Validation Problems	
🔇 One or more selected WSDL files have errors. Please correct or remov	e the file(s)to proceed.
WSDL Validation Panel	
▲ message	wsdl file
🔞 cvc-elt.1: Cannot find the declaration of element 'wsdl:definitions'.	http://terraservice.net/terraservice2.asm
(?) < <u>B</u> ack	<u>N</u> ext > Cancel

Figure 5.65. WSDL Validation Problems Dialog

5.10.2. Import WSDL From File System Location

- You can create a Web Service model by selecting a WSDL file from your local file system.
 - Step 1 Choose the File Import choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teiid Designer > WSDL File or URL >> Web Service Model and click Next>
- Step 3 Input a valid name for your Web Service model and select the File System... button. Locate your file system WSDL file in the selection dialog and click OK>.

Screate Web Service from WSDL File			×
WSDL File Selection			
Press the "Next >" button to continue.			
Web Service Model Name OnHandWS Workspace File System			
WSDL Files	D	×	
/home/blafond/Test Designer Folder/example files/wsdls/OnHand.	wsdl		
(?) < <u>Back</u> <u>Next</u> > Cancel		inish	

Figure 5.66. WSDL File Selection Dialog

Note

i

- If no WSDL is selected or specified then the importer will only create an empty Web Service model. No XML Schema or XML View models will be generated.
- Any referenced files (**WSDL**s or schemas) must either be embedded in the **WSDL** file or exist on your file system.
- Step 4 The next page is titled Namespace Resolution. This page identifies successful and errant WSDL namespace resolution. The main WSDL document will essentially always be resolved, since the workspace file chooser is used to obtain the path. Problems will occur when the main WSDL file imports other WSDL files that cannot be resolved. If no errors, select Next to proceed, or Finish (if enabled) to complete with default options.

		tion olved. Select "Next" to modify the created schemas workspace	3
lam	espaces		4
4	Namespace	Path	
	http://dladb07/	/home/blafond/Test Designer Folder/example files/wsdls/OnHand.ws	dl

Figure 5.67. Namespace Resolution Dialog

• Step 5 - The next page WSDL Operations Selection allows customizing the resulting content of your Web Service model by selecting/deselecting various operations and interfaces in the following dialog.

		Create Relati	onal Model fr	om Web Service	2	
ed WSDL Operations M_Real_Time_Data_Design_Server_RealTimeData_portService BSM_Real_Time_Data_Design_Server_RealTimeData_portBinding OnHand {http://dladb07/}Share_BSM_Real_Time_Data_Design_Server_RealTimeData_port OnHandOutput OnHandOutput none	SDL Operations	Selection				
SM_Real_Time_Data_Design_Server_RealTimeData_portService _BSM_Real_Time_Data_Design_Server_RealTimeData_portPort0 are_BSM_Real_Time_Data_Design_Server_RealTimeData_portBinding OnHand	ress the "Finish" but	on to finish.				8
Select All Deselect All OnHand	Select the desir	ed WSDL Operation	1s			
OnHand (http://dladb07/}Share_BSM_Real_Time_Data_Design_Server_RealTimeData_p OnHandInput OnHandOutput none	▽ 🖌 📄 Share	BSM_Real_Time_Data_D	esign_Server_F	RealTimeData_por	rtPort0	
<pre></pre>			_Design_Serve	er_RealTimeData_	portBinding	Deselect All
{http://dladb07/}Share_BSM_Real_Time_Data_Design_Server_RealTimeData_p _OnHandInput OnHandOutput none	✓ 585	OnHand				
{http://dladb07/}Share_BSM_Real_Time_Data_Design_Server_RealTimeData_p _OnHandInput OnHandOutput none						
_OnHandInput OnHandOutput none	<					
_OnHandInput OnHandOutput none	Selection Details:					
	OnHand [Operation] id: input message: output message: fault names:	_OnHandInput OnHandOutput	Share_BSM_Re	al_Time_Data_De	sign_Server_Re	
	<.					

Figure 5.68. Namespace Resolution Dialog

 Step 6 - The next page is titled Schema Workspace Location Selection. This page lists all schemas imported by the WSDL (along with any dependent schemas referenced within schemas) as well as schemas embedded in the WSDL and indicates whether or not they are resolvable. All resolved schemas will be created in a separate file and added to the workspace. The editor panel allows you to change the default file name of the new schema file(s).

If no errors, select Next to proceed, or Finish to complete with default option

		Create Web Ser	vice from WSDI	L File	
Sche	ma Workspace Location	Selection			
All w	orkspace schema locations are v	alid. Click "Next" to cont	tinue.		
Sch	ema Location Information				
4	Namespace				Source File
	http://localhost/Share_BSM_Re	al_Time_Data/Design_Se	erver/RealTimeDa	ta_port/OnHand	/home/blafond/To
K	Editor - http://localhost/Share_BS	M_Real_Time_Data/Desi	gn_Server/RealTi	meData_port/On	Hand
	Folder:	/WSDLImportProject			
	Name (without file extension):	OnHand			
0)		< <u>B</u> ack	Next >	Cancel
•				<u>n</u> ext >	

Figure 5.69. Namespace Resolution Dialog

• Step 7 - The last page titled XML Model Generation allows you to change the name of the XML View model if the Generate virtual XML document model is checked. Input desired name or use the default name provide. Select Finish to complete.

@	Create Web Service from WSDL File	\otimes
	l Generation	
Ready to gen	nerate the Web service model	
🗹 Generate	e virtual XML document model	
Location:	/WSDLImportProject 2	
XML Model:	OnHandWSResponses	
?	< <u>Back N</u> ext > Cancel <u>Finish</u>	

Figure 5.70. Namespace Resolution Dialog

In order to successfully generate Web Services from WSDL, the WSDL must be error free. WSDL validation is performed during *Step 3* above. If errors do exist, a error summary dialog will be displayed (shown below) and you will not be able to *Finish* the wizard until the WSDL problems are fixed or you re-import and select a valid WSDL file.

Create Web Service from V	VSDL File
Validation Problems	
😵 One or more selected WSDL files have errors. Please correct or remove	e the file(s)to proceed.
WSDL Validation Panel	
▲ message	wsdl file
🔞 cvc-elt.1: Cannot find the declaration of element 'wsdl:definitions'.	http://terraservice.net/terraservice2.asm
? < <u>B</u> ack	<u>N</u> ext > Cancel

Figure 5.71. WSDL Validation Problems Dialog

5.10.3. Import WSDL From URL

- You can create a Web Service model by selecting a WSDL file based on a URL.
 - Step 1 Choose the File Import choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teiid Designer > WSDL File or URL >> Web Service Model and click Next>
- Step 3 Input a valid name for your Web Service model and select the URL... button.
 - Enter a valid WSDL URL. If the URL cannot be validated then an error will be displayed and the **OK>** button disabled.
 - If the WSDL is protected by basic HTTP authentication then this option should be selected and the appropriate username and password entered.
 - Click **OK>** to continue.

Click Next> to continue.

9	WSDL URL
Enter WSDL Url:	http://terraservice.net/terraservice2.asmx?wsdl
	Security Type HTTPBasic 🔻
	User Name user
	Password
	Cancel OK

Figure 5.72. WSDL URL Dialog

Ĺ	Note • If no WSDL is selected or specified then the importer will only create an empty Web Service model. No XML Schema or XML View models will be generated.
	 Any referenced files (WSDLs or schemas) must either be embedded in the WSDL file or exist on your file system.
• Step 4 - Th	e next page is titled Namespace Resolution . This page identifies successful and

 Step 4 - The next page is titled Namespace Resolution. This page identifies successful and errant WSDL namespace resolution. The main WSDL document will essentially always be resolved, since the workspace file chooser is used to obtain the path. Problems will occur when the main WSDL file imports other WSDL files that cannot be resolved. If no errors, select Next to proceed, or Finish (if enabled) to complete with default options.

Screate Web Service from WSDL File	×
WSDL File Selection	
Press the "Next >" button to continue.	
	1000 CONTRACTOR CONTRA
Web Service Model Name TerraServiceWS	
Workspace File System URL	
WSDL Files	
http://terraservice.net/terraservice2.asmx?wsdl	
(?) < Back Next > Cancel	Freich
	Finish

Figure 5.73. Namespace Resolution Dialog

• Step 5 - The next page WSDL Operations Selection allows customizing the resulting content of your Web Service model by selecting/deselecting various operations and interfaces in the following dialog.

	Create W	eb Service from WSDL File	δ
ame	espace Resolution		
	mespaces are resolved. Sele pace locations.	ct "Next" to modify the created schemas	-
Name	espaces		
4	Namespace	Path	
I		http://terraservice.net/terraservice2.asm	

Figure 5.74. Namespace Resolution Dialog

 Step 6 - The next page is titled Schema Workspace Location Selection. This page lists all schemas imported by the WSDL (along with any dependent schemas referenced within schemas) as well as schemas embedded in the WSDL and indicates whether or not they are resolvable. All resolved schemas will be created in a separate file and added to the workspace. The editor panel allows you to change the default file name of the new schema file(s).

If no errors, select Next to proceed, or Finish to complete with default option

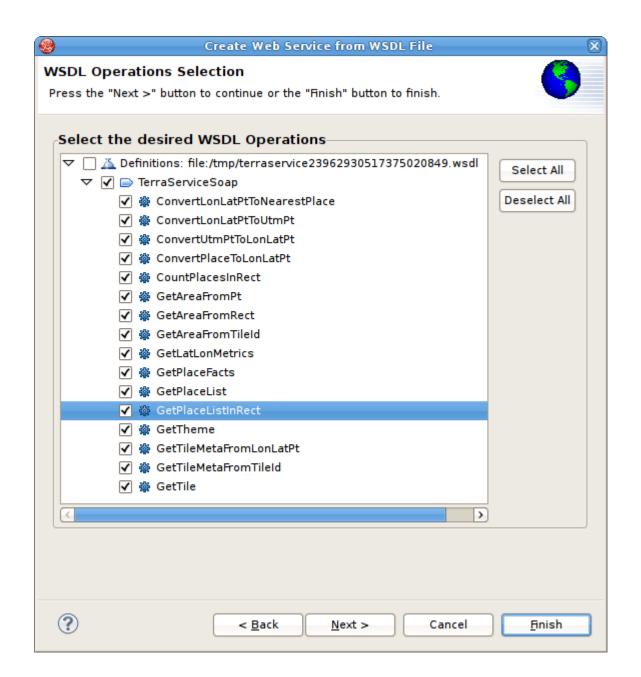


Figure 5.75. Namespace Resolution Dialog

 Step 7 - The last page titled XML Model Generation allows you to change the name of the XML View model if the Generate virtual XML document model is checked. Input desired name or use the default name provide. Select Finish to complete.

		Create W	eb Service from	WSDL File		
	ma Workspace Local rkspace schema locations		Next" to continue.			
che	ma Location Information					
4	Namespace	Source File	Target Path			
	http://terraservice-usa.co	om/ /tmp/terraser	rvice2 WSDLImport	Project/terraserv	vice2396293051	737502
f E	ditor - http://terraservice-u	usa.com/				
f E	ditor - http://terraservice-u Folder:	usa.com/ /WSDLImpo	ortProject			
E		/WSDLImpo	ortProject :e23962930517375	020849		
∦ E	Folder:	/WSDLImpo		020849		
f E	Folder:	/WSDLImpo		020849		
j E	Folder:	/WSDLImpo		020849		
∦ E	Folder:	/WSDLImpo		020849		
₹ E	Folder:	/WSDLImpo		020849		

Figure 5.76. Namespace Resolution Dialog

In order to successfully generate Web Services from WSDL, the WSDL must be error free. WSDL validation is performed during *Step 3* above. If errors do exist, a error summary dialog will be displayed (shown below) and you will not be able to *Finish* the wizard until the WSDL problems are fixed or you re-import and select a valid WSDL file.

VSDL File
e the file(s)to proceed.
wsdl file
http://terraservice.net/terraservice2.asm
Next > Cancel

Figure 5.77. WSDL Validation Problems Dialog

5.11. Import from an XML Schema File

- You can import an XML Schema file (XSD) files using the steps below.
 - Step 1 In Model Explorer choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teiid Designer > XML Schemas and click Next>
- Step 3 Select either Import XML Schemas from file system or Import XML Schemas via URL and click Next >
- Step 4a If importing from file system, the Import XML Files dialog is displayed. Click on the Browse button to find the directory that contains the XML file(s) you wish to import.
 - To select all of the XML files in the directory, click the checkbox next to the folder in the left panel.
 - To select individual XML files, click the checkboxes next to the files you want in the right panel

🧐 Import XML Schema Files 🛛 🕅
File system Import resources from the local file system.
From directory. np/testdata/johndoe/test-data/schemas/book\$ V Browse
🗖 🗁 books 🗌 🕅 🕅 BookDatatypes.xsd
Books.xsd
🗌 🖹 BooksInput.xsd
Filter Types Select All
Into folder: BooksProject Browse
Options
Overwrite existing resources without warning
O <u>C</u> reate complete folder structure
Create selected folders only
Advanced >>
Add Dependent Schema Files
< Back

Figure 5.78. Select XML From File System

• **Step 4b** - If importing from URL, select the *Import XML Schemas via URL* option and click *OK* to display the final *Add XML Schema URLs* wizard page.

9	lmp	ort XML Sche	ma Files	\square
Add XML	Schemas UF	₹Ls		SÌ
XML schem	a URLs			C1 ×
	BooksProject			Browse
Options Overwr	ite existing res	ources without	warning	
	pendent Schen		warning	
?	< <u>B</u> ack	<u>N</u> ext >	Cancel	Enish

Figure 5.79. Add XML Schema URLs Dialog

Step 5 - Click the Add XML Schema URL button

Enter a valid schema URL. Click OK. Schema will be validated and resulting entry added to the list of XML Schema URLs.

8	XML Schema Url	X	
Enter XML schema URL: http://ns.hr-xml.org/2_4/HR-XML-2_4/StandAlone/Resume.xsd			
Optional User Name Password Verify Hostname (HT	TPS)		
	Cancel OK		

Figure 5.80. Add XML Schema URLs

The schema URL is now displayed in the XML Schema URLs list.

Add XML Schemas URLS	9	Imp	ort XML Scher	na Files	×
http://ns.hr-xml.org/2_4/HR-XML-2_4/StandAlone/Resume.xsd Into folder: BooksProject Options Options Options Overwrite existing resources without warning	Add XML S	Schemas UR	≀Ls		SP
Into folder: BooksProject Browse Options Options Overwrite existing resources without warning	XML schema	a URLs			Eî 🗙
Options Options Overwrite existing resources without warning	http://ns.h	nr-xml.org/2_4/	HR-XML-2_4/Star	ndAlone/Resume	e.xsd
✓ Add Dependent Schema Files	Options <u>O</u> verwri	ite existing res		warning	Browse
(?) < Back Next > Cancel Finish	Add Dep			Cancel	Finish

Figure 5.81. Add XML Schema URLs

• Step 6 - Click Finish.

Note

i

XML files may have dependent files. This importer will determine these dependencies and import these as well if Add Dependent Schema Files is checked

5.12. Import From an LDAP Server

The Lightweight Directory Access Protocol, or LDAP, is a network protocol for accessing directory services over TCP/IP. A directory contains a collection of related data, organized hierarchically in a tree format. Each node in the tree is a directory entry, and each entry consists of a set of attribute-value pairs. Each directory entry has a unique identifier, known as its Distinguished Name (DN). The DN consists of a Relative Distinguished Name (RDN), constructed from an attribute from the entry itself, followed by the parent entry's DN.

In Teiid Designer, an LDAP Server can be modelled by doing the following:

- LDAP subtrees are represented as if they were tables in a relational database.
- Each node in the subtree is represented as a row in the table.
- Each attribute of the given node can be represented as a column in the table.
- The RDN (or DN) can be used to represent a primary key.

The LDAP metadata is modeled using the relational metamodel. Each table in the relational model represents a directory entry while each row in the table represents a child entry of the directory entry. Each column of the table represents an attribute of the child entry that may exist. In general, each table and column defines the LDAP-specific information in the property "Name In Source". This allows the connector to identify the attribute or Base DN name within the data source, ie. within LDAP. The actual name of the table and column can differ from the name in source, allowing for more descriptive labelling in models and queries.

- The relational source model can be created from LDAP data using the steps below.
 - Step 1 In Model Explorer choose the File > Import action

in the toolbar or select a project, folder or model in the tree and choose Import...

- Step 2 Select the import option Teild Designer > LDAP Service >> Source Model and click Next>
- Step 3 Select an existing or previous LDAP connection profile from the drop-down selector or press New... button to launch the New Connection Profile dialog (See Eclipse Data Tools documentation) or Edit... to modify/change an existing connection profile prior to selection.

Note

Information required for a new connection:

- Connection Username / password an administrator account to browse the ldap tree, eg. cn=Manager,dc=birds-of-prey,dc=org
- Connection URL, eg. Idap://falcon:389
- Principal Distinguished Name (DN) Suffix the root DN of the Idap tree
- An LDAP Connection Factory implementation class, eg. com.sun.jndi.ldap.LdapCtxFactory

Selection of the connection profile populates the LDAP Service URL and DN Suffix fields. The remaining requirements for the wizard page is the choosing of a suitable model file as the destination of the imported tables. If the selection is an existing model then the wizard will merge the new tables with the model's current content.

× 💿	Create Relational Model from LDAP Service		
Source and LDAP Definition			
③ Source model LdapFalcon does not exist and will be created and contain the required invoke() web served			
	Connection Profile		
LDAP-Falcon			
	LDAP Service URL:		
LDAP Service URL:	ldap://falcon:389		
LDAP Principal DN Suffix:	dc=birds-of-prey,dc=org		
	Source Model Definition		
Location LDAPModelling	/models		
Name LdapFalcon			
Ŷ	< <u>B</u> ack <u>N</u> ext >		

Figure 5.82. LDAP Definition Wizard Page

- Step 4 After selecting a Connection Profile, click Next>
- Step 5 On the Select LDAP Entries to be modelled as tables page, select the LDAP entries from the tree to be created as tables in the source model. Entries are selected by the ticking of their respective checkboxes in the tree. The highlighting of an entry displays the following attributes:
 - Table Name this is the table's label and can be modified to a more readable value
 - Table Source Name the fully qualified entry name. This is not editable in the wizard and should remain unchanged in the subsequently created source model
 - Table Source Name Suffix an additional suffix can be added that further limits the scope of the table's search criteria. The suffix is in the format of *?search_scope? objectClass_name* where search_scope is one of OBJECT_SCOPE (first and only one entry returned), ONELEVEL_SCOPE (only entries directly below the selected entry are returned) or SUBTREE_SCOPE (recursively return all entries below the selected entry) and objectClass_name is the name of a specific type of objectClass in the LDAP tree, eg.

return only the 'inetOrgPerson' entries. Both criteria are optional (but the '?'s are not) so it is possible to have a suffix such as ? ? *inetOrgPerson*. Click **Next>**.

💥 🍥	Create Relational Model from LDAP Service		
Select the LDAP Entries to be modelled as tabl	es		
Press the "Next >" button to continue.			
9 8		Source Model Table Attributes	
Idap://falcon:389/dc=birds-of-prey, Idap://falcon:389/dc=birds-of-prey,	Table Name	Hosts	
> 🗹 ou=Group	Table Source Name	ou=Hosts,dc=birds-of-prey,dc=org	
> ✓ ou=Hosts	Table Source Name Suffix		
<			
Deselect All			
(?)		< <u>B</u> ack <u>N</u> ext >	

Figure 5.83. Select LDAP entries to be modelled as tables page

- Step 6 The Select the LDAP Attributes to be modelled as columns page displays the
 previously selected LDAP entries and the attributes of their *child* entries. The purpose of a
 selected attribute is to be created as a column in the relevant source model table. Attributes
 are selected by the ticking of their respective checkboxes in the tree. The highlighting of an
 attribute displays the following properties:
 - Column Name this is the column's label and can be modified to a more readable value
 - Column Source Name the real LDAP attribute name. This is not editable in the wizard and should remain unchanged in the subsequently created source model
 - Column Distinct Value Count The number of distinct values assigned to the specific attribute in the LDAP service. This value is useful in optimising queries using the source model. This is not editable in the wizard and should remain unchanged in the subsequently created source model.

- Column Null Value Count The number of entries where the specific attribute has no value assigned in the LDAP service. This value is useful in optimising queries using the source model. This is not editable in the wizard and should remain unchanged in the subsequently created source model.
- Column Length The maximum length of existing values assigned to the attribute in the LDAP service. This value is assigned as the maximum length of the column. This is not editable in the wizard but can be edited in the source model later should this be required.
 Click Finish> .

× 📼	Create Relational Model from LDAP Service		
Select the LDAP Attributes to be modelled as t	able columns		
Press the "Finish" button to finish.			
9		Source Model Column Attributes	
r ≤ Cn	Column Name	IpAddress	
- 🗆 userPassword	Column Name	Thyadiesa	
– 🗹 gidNumber	Column Source Name	ipHostNumber	
 objectClass 	Column Distinct Value Count	15	
🗆 🗆 memberUid			
✓ Hosts	Column Null Value Count	-1	
🗹 ipHostNumber	Column Length	14	
 objectClass 			
- 🗹 cn 🔷			
Deselect All			
?		< <u>B</u> ack <u>N</u> ext >	
_			

Figure 5.84. Select the LDAP Attributes to be modelled as columns page

Once the wizard has completed, the new source model will be created.

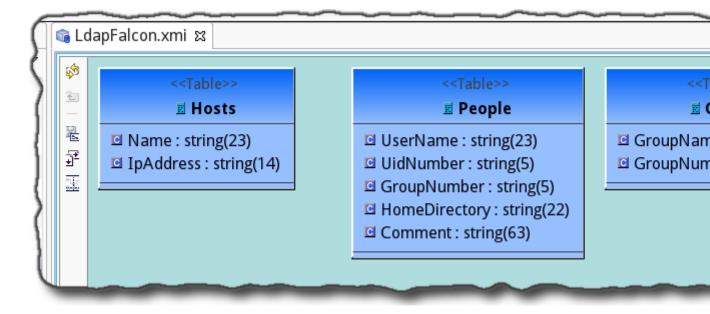


Figure 5.85. Example of an LDAP source model

The LDAP connector also provides an update capability. However, additional modeling requirements are imposed beyond those required for read-only access. The additional requirements are described below.

- Supports Update table property to enable updates, each source model table must have this
 property set to 'true';
- Updateable column property to enable updates, each column in the source model table must have this property set to 'true';
- Additional required columns:
 - DN for all update types (INSERT, UPDATE, and DELETE), the distinguished name must be modeled as a column, setting the *name in source* to *dn*. For UPDATE and DELETE capability, the DN must be specified in the criteria clause while for INSERT, the DN must be one of the column values to be set.
 - *objectClass* for INSERT, the objectclass must be modelled as a column, setting the *name in source* to *objectClass*. It must also be one of the column values to be set.
 - *additional* each entry defined in the LDAP directory's schema may also have one or more additional required columns. This is dependent on the LDAP server implementatation so consult the LDAP documentation accordingly.

A selection of example queries:

- SELECT * FROM LdapModel.People
- INSERT INTO LdapModel.People (dn, sn,

.

٠

objectclass, Name) VALUES ('cn=JoeYoung,ou=people,dc=example,dc=org','Young','person', 'Joe Young')

UPDATE LdapModel.People SET PhoneNumber='(314) 299-2999' WHERE DN='cn=JoeYoung,ou=people,dc=example,dc=org'

DELETE FROM LdapModel.People WHERE DN='cn=JoeYoung,ou=people,dc=example,dc=org'

Creating and Editing Model Objects

This section summarizes Teiid Designer features for creating and editing existing model objects contained in your models.

6.1. Creating New Model Objects

As discussed in the introduction, Section 1.1, "What is Teild Designer?", Teild Designer provides a framework to model various types of metadata. Each metamodel type has a set of parentchild relationships that establish constraints on what can be created and where. You cannot, for example, create a column attribute in a stored procedure, nor can you create a mapping class column in a Web service operation's output message.

The Teiid Designer provides a common set of actions to create new children of these models as well as children of children.

- You can create new model objects directly in the Section D.2.1, "Model Explorer View" view, Section D.3.1.2, "Diagram Editor" or Section D.3.1.3, "Table Editor" using the following actions:
 - New Child Action
 - New Sibling Action
 - New Association Action

6.1.1. New Child Action

- To create new child model objects in the Section D.2.1, "Model Explorer View":
 - **Step 1** Select the parent object to which you want to add a child. For example, you can add a package to a package or an attribute to a class.
 - Step 2 Right-click on a container object. From the pop-up menu, select New Child. You can now select the child object you would like to add.

▼ 🚰 TwoSourceParts ▼ 📬 PartsDB2.xmi ▷ ⁴≣ import dec ಔ Package Di			
 ▶ ■ PARTS ▶ ■ SHIP_VIA ▶ ■ STATUS ▶ ■ SUPPLIER ▶ ■ SUPPLIER_ 	New Child New Sibling New Association Modeling	> > > >	A Access Pattern Column Foreign Key Primary Key
 Data Sour math PartsSQL.xm 	<₽ Undo		Unique Constraint

Figure 6.1. New Child Action In Model Explorer

• Step 3 - The new model object displays on the Section D.2.1, "Model Explorer View" and is highlighted for renaming.

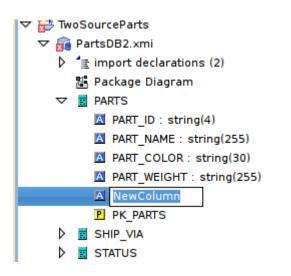


Figure 6.2. New Model Object In Explorer

- To create new child model objects in the Section D.3.1.2, "Diagram Editor":
 - **Step 1** Select the parent object to which you want to add a child. For example, you can add a package to a package or an attribute to a class.
 - Step 2 Right-click on a container object. From the pop-up menu, select New Child. You can now select the child object you would like to add.

< <base table=""/> >	i		
SHIP_VIA	New Child	\rightarrow	A Access Pattern
	New Sibling	>	🔺 Column
A SHIPPER_ID : bigdecimal	New Association	>	F Foreign Key
SHIPPER_NAME : string(Modeling	>	P Primary Key
P PK_SHIP_VIA	💛 Undo Create NewColumn		Unique Constraint
·	🍄 Redo		

Figure 6.3. New Child Action In Diagram

• Step 3 - The new model object displays on the diagram and is highlighted for renaming.

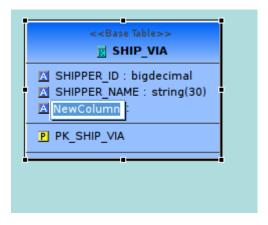


Figure 6.4. New Model Object In Diagram

- To create new child model objects in the Section D.3.1.3, "Table Editor":
 - **Step 1** Select the row for the parent object to which you want to add a child. For example to add a column, click the **Base Table** tab and select base table row.
 - Step 2 Right-click on a table row. From the pop-up menu, select New Child. You can now select the child object you would like to add.

artsDB2.xr	mi 🖾 🧃] PartsViews.xmi	
🗾 Base Table	s 🖪 Colun	nns F Foreign Keys 🗜 Prima	iry Keys
Location	Name	Name In Source	System Cardinality Supports I
	PARTS SHIP_VIA STATUS SUPPLIER SUPPLIER_	New Child New Sibling New Association Modeling Vindo Delete multiple object	A Access Pattern A Column F Foreign Key P Primary Key Unique Constraint

Figure 6.5. New Child Action In Table Editor

• Step 3 - The selected tab in the Table Editor changes to the tab for the child object type, the new model object row is displayed and the row's name table cell is highlighted for renaming.

6.1.2. New Sibling Action

- To create new sibling model objects in the Section D.2.1, "Model Explorer View":
 - **Step 1** Select the object to which you want to add a sibling. For example, you can add a column sibling to a column.
 - Step 2 Right-click on that object. From the pop-up menu, select New Sibling. You can now select the sibling object you would like to add.

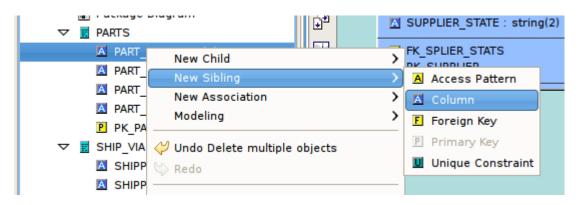


Figure 6.6. New Sibling Action In Model Explorer

- Step 3 The new model object displays on the Section D.2.1, "Model Explorer View" and is highlighted for renaming.
- To create new sibling model objects in the Section D.3.1.2, "Diagram Editor":

- Step 1 Select the object to which you want to add a sibling. For example, you can add a column sibling to a column.
- Step 2 Right-click on that object. From the pop-up menu, select New Sibling. You can now select the sibling object you would like to add.

< <base table=""/> > SHIP_VIA			
SHIPPER_ID : bigdecin	New Child	>	
	New Sibling	\rightarrow	Access Pattern
P PK_SHIP_VIA	New Association	>	A Column
	Modeling	>	F Foreign Key
	💛 Undo Delete multiple objects		P Primary Key
	> Redo		Unique Constraint

Figure 6.7. New Sibling Action In Diagram

- Step 3 The new model object displays on the diagram and is highlighted for renaming.
- To create new sibling model objects in the Section D.3.1.3, "Table Editor":
 - **Step 1** Select the row for the object to which you want to add a sibling. For example, you can add a column sibling to a column.
 - Step 2 Right-click on a row. From the pop-up menu, select New Sibling. You can now select the sibling object you would like to add.

📬 PartsDB2.	xmi 🛛 🕞 Parts	Views.xmi				
🗾 Base Tab	oles 🖪 Columns 🗜	Foreign Key	vs P Primary Keys			
Location	Name	Name In So	ource System	Cardinality	Supp	oorts Up Materialized
	PARTS	PARTS	New Child	: :	>	false
	SHIP_VIA	SHIP_VIA	New Sibling		\rightarrow	🖬 Base Table
	STATUS	STATUS	New Association		~	👝 Catalog
	SUPPLIER	SUPPLIER	Modeling		>	I Index
	SUPPLIER_PARTS		✓ Undo Delete multip	ole objects		Logical Relationsh
			of Cut	Ctrl	I+X	View
			Сору	Ctrl	+C	🖺 Custom Diagram
1			💼 Table Paste			

Figure 6.8. New Sibling Action In Table Editor

• Step 3 - The selected tab in the Table Editor changes to the tab for the child object type, the new model object row is displayed and the row's name table cell is highlighted for renaming.

6.1.3. New Association Action

- To create new associations between model objects in the Section D.2.1, "Model Explorer View":
 - Step 1 Select two objects you wish to associate. For example, select columns in different base tables.
 - Step 2 Right-click. From the pop-up menu, select New Association > Foreign Key Relationship..

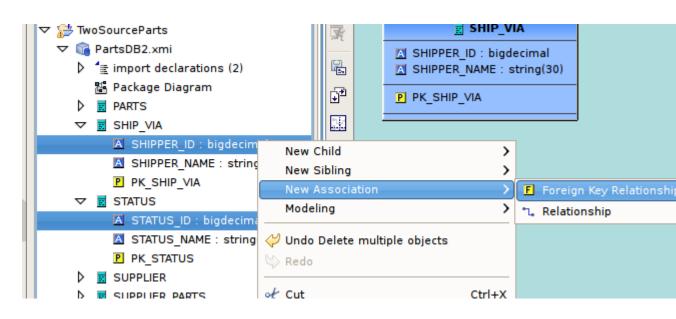


Figure 6.9. New Association Action In Model Explorer

• Step 3 - The new relationship link is displayed in the diagram.

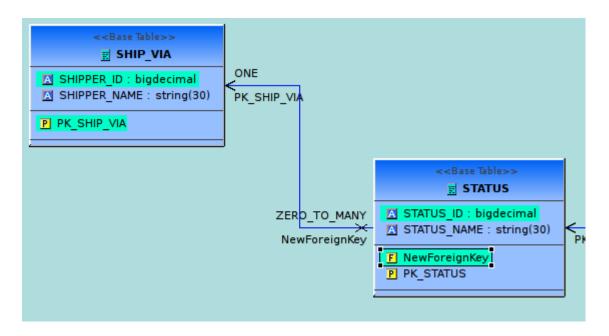


Figure 6.10. New Association In Diagram

- To create new associations between model objects in the Section D.3.1.2, "Diagram Editor":
 - Step 1 Select two objects you wish to associate. For example, select columns in different base tables.
 - Step 2 Right-click. From the pop-up menu, select New Association > Foreign Key Relationship..

• **Step 3** - The new relationship link is displayed in the diagram. The Column, Foreign Key, Primary Key reference properties are properly set on the selected columns, new primary key and new foreign key.

OR

- Step 1 Select a column in table.
- **Step 2** Drag the column to another table and drag over a column and drop onto this column. The target column should highlight in Yellow.
- **Step 3** The new relationship link is displayed in the diagram. The Column, Foreign Key, Primary Key reference properties are properly set on the selected columns, new primary key and new foreign key.
- To create new associations between model objects in the Section D.3.1.3, "Table Editor":
 - Step 1 Select two objects you wish to associate. For example, select columns in different base tables.
 - Step 2 Right-click. From the pop-up menu, select New Association > Foreign Key Relationship.
 - **Step 3** New Foreign Key and Primary Key objects will be added to the contents of their respective tabs in the Table Editor. The Column, Foreign Key, Primary Key reference properties are properly set on the selected columns, new primary key and new foreign key.

6.2. New Model Object Wizards

In addition to the simple new object actions, Teiid Designer provides New Child and New Sibling wizards for children of view and source relational models. Namely tables, views, procedures and indexes. The menu now looks like:

New	+ <u> </u>
New Child	🕨 📩 Base Table
Modeling	Procedure
Open	👻 View
Open With	► î Index
Сору	隆 Custom Diagram
📑 Paste	Ctrl+V

Figure 6.11. Relational Model New Child Menu

6.2.1. Create Relational Table Wizard

The **Create Relational Table Wizard**, shown below, allows creating a complete relational table including columns, unique keys, foreign keys definition and indexes.

🛛 🙁 Create R	elational Table
Create Relational T	able (
Click OK to add new	object to model
Model File	ProductsOracle.xmi
Name	productinfo
Name In Source	"productinfo"
Properties	Columns P Primary Key Unique Constraint F Foreign Keys I Indexes S Native Query
Cardinality 1	
Supports	Update 🗌 Is System Table
🗌 Materializ	zed
Materialized Ta	able
Descriptio	n
(?)	Cancel OK

Figure 6.12. Create Relational Table Dialog

Note that the relational view wizard is identical to the relational table wizard.

6.2.2. Create Relational Procedure Wizard

The **Create Relational Procedure Wizard**, shown below, allows creating a complete relational Procedure including columns, unique keys, foreign keys definition and indexes. The relational procedure object can represent different types of procedures, including a standard procedure, source function (pushdown function) and native query procedures. When the **New Child > Procedure...** action is launched the first dialog gives you the option of selecting the procedure type. Note that user defined functions can be modeled in a view model.

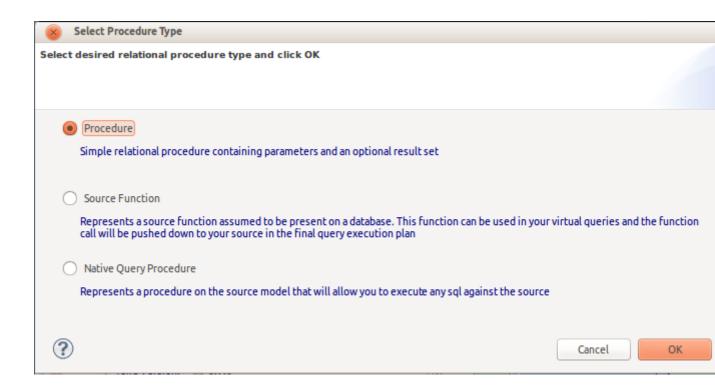


Figure 6.13. Select Procedure Type Dialog

The second dialog customizes the **Create Relational Procedure** dialog based on your selected type. The following dialog provides standard procedure data input.

00	Create Relational Procedure
Create Relational Provide unique nar	Procedure me, properties and desired parameters and result set
This wizard allow	s creating a complete relational procedure including parameters and optional result set.
Model File	Customer_Accounts.xmi
Name	
Name In Source	
Properties	(X) Parameters Result Set Description Native Query
Update Count	AUTO 😂
Non-Prepar	red
?	Cancel OK

Figure 6.14. Create Relational Procedure Dialog

Source functions are procedures that are already deployed and accessable on your database. By defining source functions in your relational model, you can call these functions from within your transformation SQL and the functions will be pushed down to your database for execution.

ccounts_Source.xmi		
ame		
ame In Source		
Properties (×) Parameters	B Description	
Function Properties		
Deterministic Ret	urns null on null 🗌 Variable Arguments	
Aggregate Properties		
Aggregate Allow	s Distinct 🗌 Allows Order-by	
	nposable Uses Distinct Rows	
	iposable 🔄 oses Distilict Rows	

Figure 6.15. Create Relational Source Function Dialog

Native query procedures are procedures that are already deployed and accessable on your database. By defining source functions in your relational model, you can call these functions from within your transformation SQL and the functions will be pushed down to your database for execution.

The resulting Create Relational Native Query Procedure dialog defaults to the following procedure structure including 0 or more input parameters (defined by the user) and a result set with a single output column with object data type (user can edit/change the column name).

🛛 😑 🛛 Create Re	elational Native Quer	ry Procedu	re					
Create Relational Native Query Procedure Click OK to add new object to model								
Model File								
Moderrite	Customer.xmi							
Name	getFullName							
Name In Source								
Hame in Source								
Properties	🛛 Parameters 📃	Result Set		S Native Query				
			-					
Add Delete	Move Up Move	e Down						
Parameter Na	me Data Type	Length	Direction					
(×) first	string	4000	IN					
(×) middle	string	4000	IN					
(×) last	string	4000	IN					
?				Canc	el OK			
-								

Figure 6.16. Create Relational Native Query Procedure Dialog

6.2.2.1. Create Function from SQL Transformation

Sometimes users will be creating SQL transformations that represent procedures or functions that have yet to be defined. Teiid Designer allows creating Source Functions or User Defined Functions based on the function defined in your SQL.

Below is an example of a query using a getBookInfo() function that does not exist yet. The result is a SQL validation error.

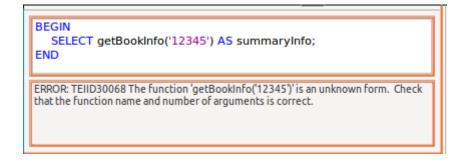


Figure 6.17. Undefined Function in Transformation

By selecting the function name and right-clicking, you can now select the Create Function action.

BEGIN					
	Bookinfo('12345') AS	summarvl	nfo;		
END	of Cu <u>t</u>				
	С ору				
ERROR: TEIID3006 that the function r			s an unknown form. Check ect.		
	Select <u>A</u> ll				
	< ₽ U ndo				
🗖 Data Sources 📓	🍄 Redo	Ctrl+Y	Console	SQL Results	
0 errors, 3 warnings,	🛷 Find			E SQUICSORS	
Description	Apply Template				
Warnings (3 it	Create Function				
	Input Methods	Þ			

Figure 6.18. Create Function Action

You can then choose to create a Source Function (i.e. pushdown) or a User Defined Function option.

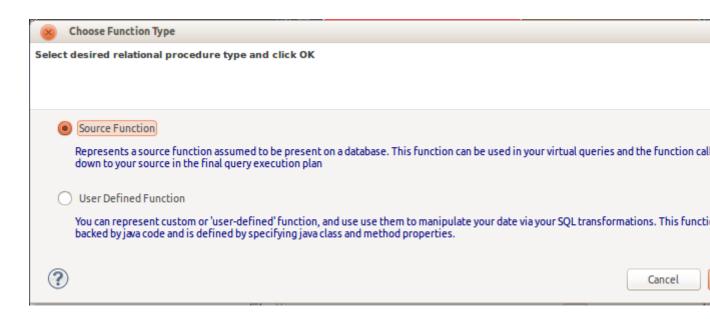


Figure 6.19. Select Function Type

For either option, you'll be presented the appropriate new function/procedure wizard which will seed the parameter definitions based on the procedure definition in the SQL.

🛞 🗊 Create Relation	al Source Functi	ion		
Create Relational Source	Function			
Click OK to add new object	to model			
Model File				
BooksSource.xmi				
Name				
getBookInfo				
Name In Source				
Dressertion M Pa	rameters 🎲 D	Vacariatia		
Properties (X) Pa)escriptio		
Add Delete M	ove Up Move	Down		
Parameter Name	Data Type		Direction	
(×) arg1	string	0	IN	
?			Cancel	ОК

Figure 6.20. Create Source Function

	Create User Defi	ined Function			
Create	User Defined Fund	tion			
This w specif	vizard allows creating ic properties like the	a complete user function's java cl	defined lass and n	function including parameters and function- nethod.	
Мо	odel File				
B	ooksInfo.xmi				
Nar	me				
ge	etBookInfo				
Nar	me In Source				
	Properties (X) Par	ameters 🍄 D	escriptio	n	
	Add Delete Mo	we Up Move [Down		
	Parameter Name	Data Type	Length	Direction	
	(×) arg1	string	0	IN	
	(×) return_param	string	0	RETURN	
?	1			Cancel	OK

Figure 6.21. Create User Defined Function

In either case, you'll be prompted to select a target Source of View model as the location for your new function.

6.2.3. Create Relational Index Wizard

Indexes can be created at the same time as your relational table object (see Indexes tab in **Create Relational Table** image above) or via **New Child > Index...** action and dialog as shown below.

x 🗆 Create	Relational Index					
Create Relational	Create Relational Index					
Click OK to add ne	w object to model					
This wizard allows	s creating a complete relational index including referenced columns.					
Model File	ProductsMySQL.xmi					
Name	ProductDataIndex					
Name In Source						
C Referenced (Columns I Properties Description					
Muto Updat	.e					
Nullable						
Unique						
Filter Condition						
0						
(?)	Cancel OK					

Figure 6.22. Create Relational Index Dialog

6.2.4. Create View Model Objects Wizards

For view models, only tables, procedures (standard procedures and user defined functions) and indexes can be created. For view tables and procedures, the primary difference in the wizards is that they include a SQL Transformation tab.

Create Relational View Table

🔇 Table name car	nnot be null or empty					
This wizard allow	s creating a complete i	elational table	including column	s, unique keys and foreig	n keys definition.	
Model File	PartsViews.xmi					
Name						
Name In Source						
Properties	Transformation SQL	Columns	Primary Key	Unique Constraint	F Foreign Keys	I Indexes
Select SQL Tem	nplate					
SQL Definitio		L2], [TABLEB.O	OL1] FROM [TABL	EA], [TABLEB] WHERE [TA	BLEA.COL1] = [TAB	LEB.COL1]
						-

Figure 6.23. Create Relational View Table Dialog

6.2.4.1. Create User Defined Functions Wizard

User-defined functions can be defined as view model procedures that of type "function". When the **New Child > Procedure...** action is launched for a view model, the first dialog gives you the option of selecting the procedure type which includes either a standard view procedure or a user-defined function.

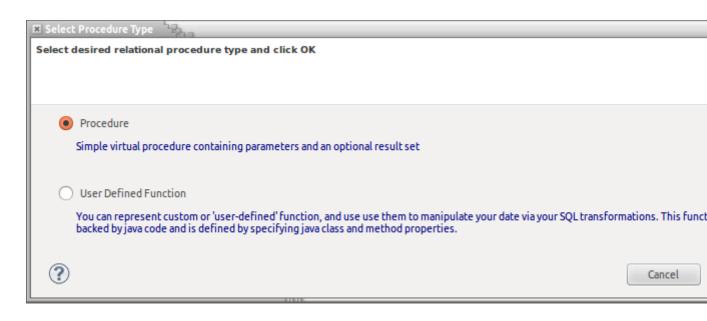


Figure 6.24. Select Procedure Type Dialog

User Defined Functions require additional properties such as **Java Class** and **Method** as well as a path to the jar file containing the code as shown below.

Create User Define	ed Function Land
reate User Defined	Function
Click OK to add new ob	ject to model
Model File	
MyPortfolio.xmi	
Name	
getLatestQuote	
Name In Source	
	500 p 1 1 1
<u> R</u> Properties (X	Parameters Parameters
Function Prope	erties
Function Categor	stock function
Java Class	StockUtilities
Java Method	getLatestQuote
UDF Jar Path	/Stock/lib/stockUtilities.jar
🗹 Determinist	tic Returns null On null Variable Arguments
Aggregate Pro	perties
🖌 Aggregate	Allows Distinct Allows Order-by
Analytic	Decomposable Uses Distinct Rows
?	Cancel OK

Figure 6.25. Create Relational User Defined Fuction Dialog

Note you can also create a User Defined Function from within a SQL transformation. See *Section 6.2.2.1, "Create Function from SQL Transformation"* for details.

6.3. Model Object Editors

The primary actions for editing model objects are:

• • t

Cut - Deletes the selected object(s) and copies it to the clipboard.

Copy - Copies the selected object(s) to the clipboard.

• 💼

Paste - Pastes the contents of the clipboard to the selected context.

• **Clone** - Duplicates the selected object in the same location with the same name; user is able to rename the new object right in the tree.

• 🗙

Delete - Deletes the selected object(s).

• Rename - Allows a user to rename an object.

These actions are presented in Teiid Designer's main **Edit** menu and also in the right-click context menus for model objects selected in the Section D.2.1, "Model Explorer View", Section D.3.1.2, "Diagram Editor" and Section D.3.1.3, "Table Editor".

Modeling Sub-Menu. In addition to the New Child/Sibling/Association menus available for object creation Designer provides a *Modeling* > sub-menu which presents various object-specific actions which can be performed.

If you select a source table, for instance, the modeling menu below would be presented:

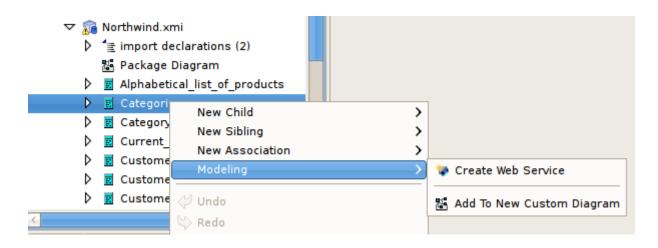


Figure 6.26. Modeling Sub-Menu for Source Table

If a view table is selected, the menu would reflect the actions related to virtual operations:

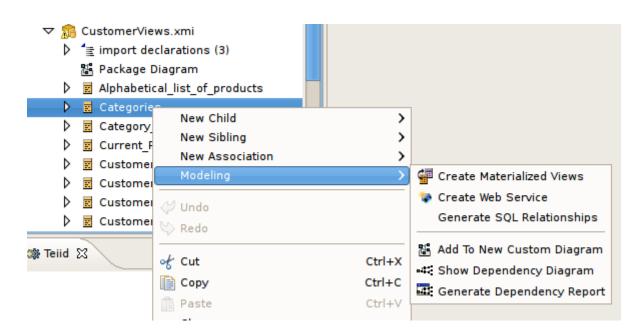


Figure 6.27. Modeling Sub-Menu for Source Table

Teiid Designer also provides specialized object editors to handle complex model objects and their unique properties. These editors include:

- Section 6.3.1, "Transformation Editor"
- Section 6.3.2, "Input Set Editor (XML)"
- Section 6.3.3, "Choice Editor (XML)"
- Section 6.3.4, "Recursion Editor (XML)"
- Section 6.3.5, "Operation Editor"

This section describes these editors in detail.

6.3.1. Transformation Editor

The **Teiid Designer's Transformation Editor** enables you to create the query transformations that describe how to derive your virtual metadata information from physical metadata sources or other virtual metadata and how to update the sources.

The **Transformation Editor** provides a robust set of tools you can use to create these SQL queries. You can use these tools, or you can simply type a SQL query into the Transformation Editor.

To edit a transformation you can:

- Double-click Edit
 - A relational view table or procedure in the Model Explorer or Diagram Editor

- A transformation node in a transformation diagram or mapping transformation diagram
- A mapping class in a mapping diagram or mapping transformation diagram
- Right-click Edit action on selected object in the Model Explorer, Diagram Editor or Table Editor
 - A relational view table or procedure
 - A transformation node in a transformation diagram or mapping transformation diagram
 - A mapping class in a mapping diagram or mapping transformation diagram

If a Model Editor is not currently open for the selected object's model, a Model Editor will be opened.

After the corresponding transformation diagram is opened in the Diagram Editor, the Transformation Editor is displayed in the lower section of the Diagram Editor.

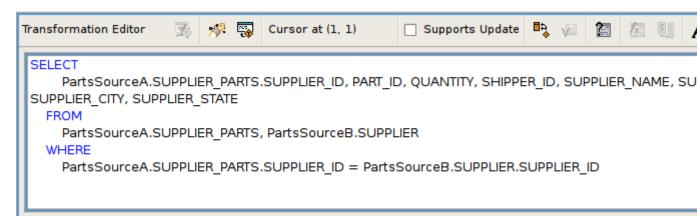


Figure 6.28. Editing String Property

If this virtual class supports updates, the tabs on the bottom of the **Transformation Editor** allow you to enter SQL for each type of query this virtual class supports. If this virtual class does not support updates, only the **SELECT** tab is available.

You can enter separate SQL queries on each available tab to accommodate that type of query.

Within the Transformation Editor, you can:

- Disable specific update transformation types on this virtual class.
- Start your transformation with a provided SQL Template.
- Build or edit a criteria clause to use in your transformation.
- Build or edit an expression to use in your transformation.
- Find and replace a string within your transformation.
- Validate the transformation to ensure its content contains no errors.

• Reconcile target attributes to ensure the symbols in your transformation match the attributes in your virtual metadata class.

You can also set preferences that impact the display of your **Transformation Editor**. For more information, see Section C.1.2.3, "Transformation Editor Preferences"

• The Transformation Editor toolbar actions are summarized below.

• 录

Prevew Virtual Data - executes a simple preview query for the target table or procedure of the transformation being edited.

• 🚀

Search Transformations - provides a simple way select and edit another transformation based SQL text search criteria.

• 🔄

Edit Transformation - provides a simple way to change which transformation to edit without searching in a diagram or the Model Explorer. Simply click the action and select from a list of views, tables, procedures or operations from the currently edited model.

Cursor at (1, 1)

Cursor Position (line, column) - shows the current line and column position of the insertion cursor. For example, Cursor Position(1,4) indicates that the cursor is presently located at column 4 of line 1.

Supports Update

Supports Update - checkbox allows you to enable or disable updates for the current transformation target. If 'Supports Update' is checked, the editor shows four tabs at the bottom for the Select, Update, Insert and Delete transformations. If 'Supports Update' is unchecked, all updates are disabled and only the Select transformation is displayed.

•∎⊅

Reconcile - allows you to resolve any discrepancies between the transformation symbols and the target attributes. Pressing this button will display the "Reconcile Virtual Target Attributes" dialog box in which you can resolve discrepancies. See *Section 6.3.1.1, "Using the Reconciler"* for more information about the Reconciler Dialog.

• v_e

Save/Validate - saves edits to the current transformation and validates the transformation SQL. Any Warning or Error messages will be displayed at the bottom of the editor in the messages area. If the SQL validates without error, the message area is not displayed.

• 🚊

Criteria Builder - allows you to build a criteria clause in your transformation. The button will enable if the cursor position is within a query that allows a criteria. Pressing the button will launch the Criteria Builder dialog. If the Criteria Builder is launched inside an existing criteria, that criteria will be displayed for edit, otherwise the Criteria Builder will be initially empty. See *Section 6.3.1.3, "Using the Criteria Builder"* for further information.

• 檀

Expression Builder - allows you to build an expression within your transformation. The button will enable if the cursor position is at a location that allows an expression. Pressing the button will launch the Expression Builder dialog. If the Expression Builder is launched inside an existing expression, that expression will be displayed for edit, otherwise the Expression Builder will be initially empty. See *Section 6.3.1.4, "Using the Expression Builder"* for further information.

• 💵

Expand Select * - allows you to expand a "SELECT *" clause into a SELECT clause which contains all of the SELECT symbols. The button will enable only if the cursor is within a query that contains a SELECT * clause that can be expanded.

۰ Ą

Increase Font Size - increases the font size of all editor text by 1.

• •

Decrease Font Size - decreases the font size of all editor text by 1.

Show/Hide Messages - toggles the display of the message area at the bottom of the transformation editor.

• a.b.c i∔ c

Optimize SQL - when toggled 'ON', will use the short names of all SQL symbols that can be optimized. Some symbol names may remain fully qualified in the event of a duplicate name or if the optimizer is unable to optimize it. When the action is toggled 'OFF', all symbol names will be fully-qualified.

• 💪

Import SQL Text - allows you to import a sql statement from a text file on your file system. Pressing this button will display an import dialog in which you can navigate to the file.

• 🚰

Export SQL Text - allows you to export the currently displayed SQL statement into a text file on your file system. Pressing this button will display an export dialog in which you can choose the location for export.

- **Close** "X" closes the transformation editor.
- The **Transformation Editor** context menu can be displayed by Rt-Clicking within the editor's text area. The context menu is show below:

BEGIN SELECT gett	Rool	(1050 ⁽¹¹ 23451) AS	summarvli	nfo;	
END ERROR: TEIID3006 that the function r	D	Cu <u>t</u> Copy Paste Select <u>A</u> ll		s an unkr ect.	nown form. Check
	Ý	<u>U</u> ndo			
Data Sources	\heartsuit	Redo	Ctrl+Y	Console	SQL Results
0 errors, 3 warnings,	P	Find			
Description		Apply Template			
Warnings (3 it		Create Function			
		Input Methods	Þ		

Figure 6.29. Transformation Editor context menu

Following is a summary of the context menu actions:

- Cut Copy Paste Typical text editor actions to cut, copy or paste text within the editor.
- Undo Redo Allows you to Undo or Redo the previous action.
- **Find** Displays a Find-Replace Dialog which allows you to search and replace text within the transformation.
- Apply Template... Displays the 'Choose a SQL Template' Dialog, which allows you to choose a starting SQL Template from a list of common SQL patterns. See ??? for a description of this dialog.
- **Create Function...** Allows creating a source function or user-defined fuction based on the selected function call defined in your SQL. See *Section 6.2.2.1, "Create Function from SQL Transformation"* for a description of this action.

6.3.1.1. Using the Reconciler

The Transformation Editor's **Reconciler** offers you a quick, graphical means to reconcile the Target View attributes and the Transformation SQL. As you make changes, the overall status will appear at the top of the dialog to assist you in successfully completing your edits.

To launch the Reconciler, click on the **Reconcile Transformation** button

in the Transformation Editor. The Reconciler Dialog is shown below:

(Reconcile Virtual Target Attri	butes	
Reconciler Status			
M The SOL contains unmatched symbol	ols - Remove them or create new attribute	s.	
SCORE: Recentilize Over			
SCOPE: Reconciling Query			Unmatched SQL Syr
Note: The data types shown are the r	untime data types.		S PART_ID
Target Lasked	Attribute COL Rindings		S QUANTITY
Target Locked	Attribute - SQL Bindings		SHIPPER_ID
Virtual Target Attributes	SQL Symbol		
SUPPLIER ID : string	SUPPLIER ID : string	< Bind	
SUPPLIER_NAME : string	SUPPLIER_NAME : string	Unbind >	
SUPPLIER_STATUS : bigdecimal	SUPPLIER_STATUS : bigdecimal		
SUPPLIER_CITY : string	SUPPLIER_CITY : string	< New	
SUPPLIER_STATE : string	SUPPLIER_STATE : string	< Null	
PART_ID : string	S PART_ID : string		
QUANTITY : bigdecimal	S QUANTITY : bigdecimal	2010	
SHIPPER_ID : bigdecimal	SHIPPER_ID : bigdecimal		
Top Up Swap Down Bottom	Delete Type Resolver		
✓ Show SQL Display			Add Remove
SELECT PartsDB2.SUPPLIER.SUPPLIER ID.	SUPPLIER NAME, SUPPLIER STATUS, SUPP	PLIER CITY, SUP	PLIER STATE, PART ID
SHIPPER_ID, PartsSQL.SUPPLIER_PARTS	.PART_ID, PartsSQL.SUPPLIER_PARTS.QUAI	NTITY, PartsSQL	SUPPLIER_PARTS. SHI
FROM PartsDB2.SUPPLIER, PartsSQL.SU	IPPLIER PARTS		
\odot			Cancel

Figure 6.30. Reconciler Dialog

To summarize the different sections of the dialog:

• Target Attributes - SQL Symbol Table: This table shows the target attributes in the left column and the SQL Symbols in the right column. The SQL Symbols are the symbols that are 'projected' from the SQL transformation. A symbol is referred to as being 'bound' to a target attribute when it is displayed next to the attribute.

If a target attribute is 'unbound', its row is highlighted in red. The transformation is not valid until all attributes have a corresponding SQL symbol binding.

Here are a few things you can do in the table section:

- Lock Target Attributes: To 'lock' the target attribute ordering, check the 'Lock Target Attributes' checkbox. This will lock the attributes in place.
- **Re-Order Attributes**: To change the ordering of the target attributes, use the 'Top', 'Up', 'Swap', 'Down', and 'Bottom' controls beneath the table. Select or multi-select the table rows, then click the desired action button.
- **Delete Attributes**: To delete one or more of the target attributes, select the table row(s) you want to delete then click the 'Delete' button.
- **Resolve Types**: If an Attribute-SQL Symbol binding has a datatype conflict, a message will be displayed. To assist in resolving the datatype conflict, a 'Datatype Resolver Dialog' is provided. Click on the table row, then click the 'Type Resolver...' button to display the dialog. See *Section 6.3.1.2, "Using the Datatype Resolver"* for further information.
- Unmatched SQL Symbols list: This list is to the right of the attribute-symbol binding table, and shows the SQL symbols from the transformation SQL that are not 'bound' to a target table attribute.

Here are a few things you can do in the list section:

- Add SQL Symbols: To 'Add' SQL Symbols to the list, click the 'Add' button. You will be presented with a dialog showing all available symbols from your transformation source tables. Click on the symbols you want to add, then click 'OK'.
- **Remove or Clear Symbols**: To remove one or more of the SQL symbols, select the list items then click the 'Remove' button. To clear the entire SQL symbols list, click the 'Clear' button.
- **Sort Symbols**: By default, the symbols are shown in the order that they appear in the SQL query. To show them alphabetically in the list, click the 'Sort' button.
- **Binding Controls**: The 'Binding Controls' are located between the Attribute-Symbol table and the Unmatched SQL Symbols list. Use these buttons to define the Attribute-Symbol bindings.

Here are a few things you can do with the binding controls:

• **Bind**: This button will 'Bind' a SQL Symbol to a target attribute. Select an Unmatched SQL symbol and select a target attribute, then click 'Bind' to establish the binding.

- **Unbind**: This button will 'Unbind' an Attribute-Symbol binding. Select an already-bound attribute in the table, then click 'Unbind'. The SQL Symbol will be released to the Unmatched Symbols list.
- New: This button will create a new target attribute, using an Unmatched SQL Symbol. Select an Unmatched Symbol from the list, then click 'New'. A new target attribute will be added to the bottom of the Attribute-Symbol table, bound to the selected SQL symbol.
- Null: This button allows you to bind 'null' to a target attribute instead of binding a SQL Symbol to it. Select a row in the Attribute-Symbol table, then click 'Null'. The target attribute will be bound to 'null'. If it was originally bound to a SQL Symbol, the symbol will be released to the Unmatched Symbol list.
- Function: This button allows you to define an expression instead of just a SQL Symbol for the binding. To define the expression, select a row in the Attribute-Symbol table, then click the 'Function' button. The Expression Builder Dialog will display, allowing you to define any type of expression. See Section 6.3.1.4, "Using the Expression Builder" for further information about the Expression Builder.
- SQL Display: The current transformation SQL is shown at the bottom of the reconciler dialog. As you add / remove SQL symbols and make other changes, you can see the SQL display change to reflect those changes. When you 'OK' the dialog, this SQL will be your new transformation SQL. If desired, the SQL Display can be hidden by un-checking the 'Show SQL Display' checkbox.

Once you are finished defining the bindings and resolving datatypes, click 'OK' to accept the changes. The transformation SQL will change to reflect your edits.

6.3.1.2. Using the Datatype Resolver

This dialog is accessible from the **Reconciler** dialog (See Section 6.3.1.1, "Using the Reconciler") and offers you a quick way to resolve datatype conflicts between a target attribute and its SQL Symbol. The Datatype Resolver Dialog is shown below:

🛞 Resolve Data	type Conflicts 🛞				
One or more of the Bindings has a type conflict.					
Attribute - SQL Bindings					
Virtual Target Attributes	SQL Symbol				
SHIPPER_ID : biginteger	SHIPPER_ID : bigdecimal				
EDIT Virtual Target Attribute SHIPPER_ID Runtime Type:	jinteger				
	eger : xs:decimal				
SQL Symbol					
Runtime Type: bigdecimal Converted Symbol: convert(SHIPPER_ * Warning * this type conversion may Apply Convert All	result in loss of precision				
?	Cancel OK				

Figure 6.31. Datatype Resolver Dialog

To summarize the different sections of the dialog:

- Target Attribute SQL Symbol Table: This table shows all target attribute SQL Symbol bindings from the Reconciler Dialog which have a type conflict. Select on a table row to populate the lower Edit Panel
- **Edit Panel**: The lower panel shows the Target Attribute and SQL Symbol datatype information for the selected binding. You can resolve the conflict in one of the following ways:

- Virtual Target Attribute: Resolve the type conflict by changing the target attribute type to be compatible with the SQL Symbol type. The attribute's current runtime type is shown, along with a potential new datatype and some button controls:
 - **Apply Button**: If the suggested datatype is acceptable, click 'Apply' to allow the attribute type to be changed.
 - **Convert All Button**: If you wish to change all of the attribute types in the table to be compatible with its corresponding SQL Symbol datatype, click the 'Convert All' button.
 - **Change Button**: If the suggested datatype is not acceptable, click 'Change' to choose your own datatype from a datatype dialog.
- **SQL Symbol**: Resolve the type conflict by applying a CONVERT function to the SQL Symbol, so that its type is compatible with the target attribute type. The SQL Symbol's current type is shown, along with a suggested CONVERT function and two button controls:
 - **Apply Button**: If the suggested CONVERT function is acceptable, click 'Apply' to apply the CONVERT function to the SQL Symbol.
 - **Convert All Button**: If you wish to apply a CONVERT function to all of the SQL Symbols in the table so that their datatype is compatible with the corresponding attribute datateyp, click the 'Convert All' button.

Once you are finished resolving datatypes, click 'OK' to accept the changes. You are directed back to the Reconciler Dialog, which will be updated to reflect your edits.

6.3.1.3. Using the Criteria Builder

The Transformation Editor's **Criteria Builder** offers you a quick, graphical means to build criteria clauses in your transformations based on meta objects in your diagram. If you launch the **Criteria Builder** with your cursor within an existing criteria in your transformation SQL, the builder will open in Edit mode. If your cursor is not in an existing criteria location, the builder will open in create mode and allow you to create it from scratch.

This procedure provides an example of building a criteria clause using the **Criteria Builder**. When building your own criteria, you can mix and match the values and constants with whatever logic you need to build powerful and complex criteria.

- To use the Criteria Builder:
 - Step 1 In the Transformation Editor, click the Launch Criteria Builder button.
 - Step 2 The Criteria Builder displays.

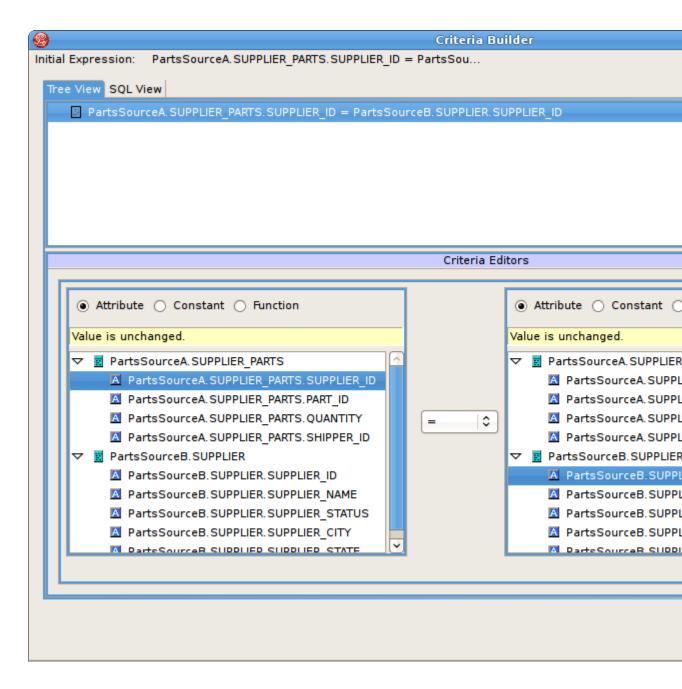


Figure 6.32. Editing String Property

The two tabs at the top, **Tree View** and **SQL View**, show the current contents of the criteria you have built.

The **Criteria Editor** at the bottom allows you to build a criteria clause. To build a criteria clause, you must add information to the left side of the predicate, select a comparison operator, and add a value to the right side.

• Step 3 - The radio buttons on either side of the **Predicate Editor** let you choose what type of content to place in that side of your predicate. Click the radio button of the type of content you want to place in your criteria. You can click:

• Attribute to add an attribute to the predicate. If you click the Attribute radio button, the Predicate Editor looks like this:

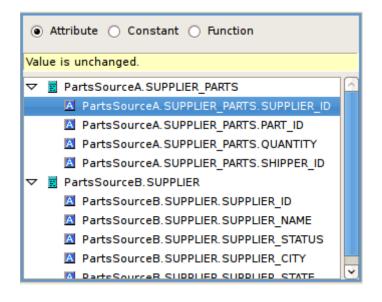


Figure 6.33. Attribute Panel

From the tree, select the attribute you want to add to the expression. You can select an attribute from any of the source classes in the transformation.

• **Constant** to add a hard-wired constant value to the predicate. If you click this radio button, the **Predicate Editor** looks like this:

Attribute Constant Function				
'Apply' to replace selected Attribute with Constant below.				
Type: string				
Value				
CompanyID				

Figure 6.34. Constants Panel

Select the datatype for this constant from the Type drop-down list and enter the value in the Value edit box.

• Function to add a function. If you click the Function radio button, the Predicate Editor looks like this:

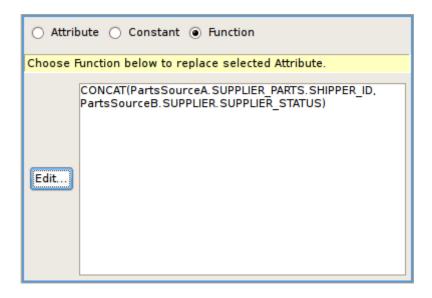


Figure 6.35. Functions

Click the Edit button to use the Expression Builder to construct a function to use in the predicate of your SQL Criterion. For more information about the Expression Builder, see *Section 6.3.1.4, "Using the Expression Builder"*

- Step 4 Set a value left side of the predicate and, when necessary, the right side of the predicate. If the right side of the predicate does not require a value of some sort, the Criteria Builder will not let you enter one.
- Step 5 Click Apply.
- **Step 6** When you have created both a Left Expression and a Right Expression in the Predicate Editor, click Apply to add the criterion to the tree view at the top of the dialog box.

The criteria clause displays in the Criteria tree.

You can create complex criteria by joining other criteria with this one. To join criteria with this one, select the criteria in the Criteria tree and click:

- Delete to remove the selected criterion.
- AND to create a new criterion that must also be true.
- OR to create a new criterion that can be true instead of the selected criterion.

• NOT to establish negative criterion.

If you join a criterion to the one you just completed, you build the expression the same way, using the Expression Editors panel and the Predicate Editor panel. You can create complex, nested criteria by judicious use of the AND and OR buttons.

Once you have created the complete criteria you want, click OK to add it to your transformation.

6.3.1.4. Using the Expression Builder

The **Transformation Editor's Expression Builder** offers you a quick, graphical means to build expressions in your transformations. This **Expression Builder** lets you create:

- Attributes by selecting an attribute.
- Constants by selecting the datatype and value.
- Functions from both the standard Teiid Designer SQL functions and your enterprise's custom user-defined functions. If you select a function before you launch the Expression Builder, you can use the Expression Builder to edit the selected function; otherwise, you can create a new function from scratch.
- To use the Expression Builder:
 - Step 1 In the Transformation Editor, click the location where you want to insert the function.
 - Step 2 Click the Expression Builder

button.

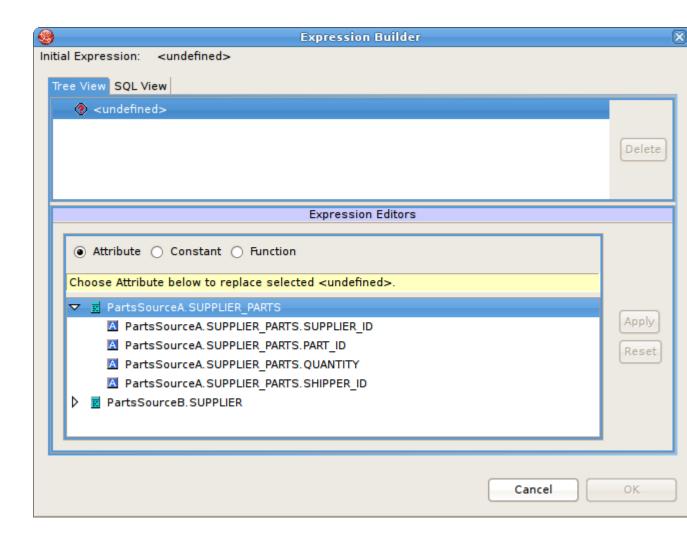


Figure 6.36. Expression Builder

The two tabs at the top, Tree View and SQL View, show the current contents of the expression you have built. To build an expression, you must specify the type of expression you want to build and populate it. In most cases, you will use the Expression Builder to construct a complex expression.

• Step 3 -Click the Function radio button to add a function.

i	Note
$\langle \gamma \rangle$	You can simply add constants and attributes as expressions by themselves using the Attribute or Constant radio buttons; however, the Expression Editor is most useful for functions.

• Step 4 - The Expression Editor displays the Function editor.

9		Expression Builder		×
Initial Expression	n: <unde< th=""><th>fined></th><th></th><th></th></unde<>	fined>		
Tree View SQL	View			
📀 <undef< th=""><th>ined></th><th></th><th></th><th></th></undef<>	ined>			
				Delete
		Expression Editors		
		Expression Editors		
 Attribut 	e 🔿 Con	stant 🖲 Function		
'Apply' to re	nlace sele	ected <undefined> with Function below.</undefined>		
Apply to re	prace ser			
Category:	STRING		•	Apply
Function:	CONCAT(STRING1, STRING2)	0	Reset
Argument	Name	Value		Reset
STRING		<undefined></undefined>		
STRING	G2	<undefined></undefined>		
		_		
			Cancel	OK

Figure 6.37. Function Panel Selected

From the Category drop-down list, choose the type of function you want to add. By default, the Teiid Designer System offers the following categories:

- Conversion for functions that convert one datatype into another.
- Datetime for functions that handle date or time information.
- Miscellaneous for other functions.
- Numeric for mathematic and other numeric functions.
- String for string manipulation functions.



Note

Any additional categories represent those containing user-defined functions your site has created.

- **Step 5** From the **Function** drop-down list, select the function you want. The table beneath the drop-down lists displays the number of arguments required for this function.
- Step 6 Click Apply.
- Step 7 Your function displays in the tree at the top. Sub nodes display for each argument you need to set for this function.

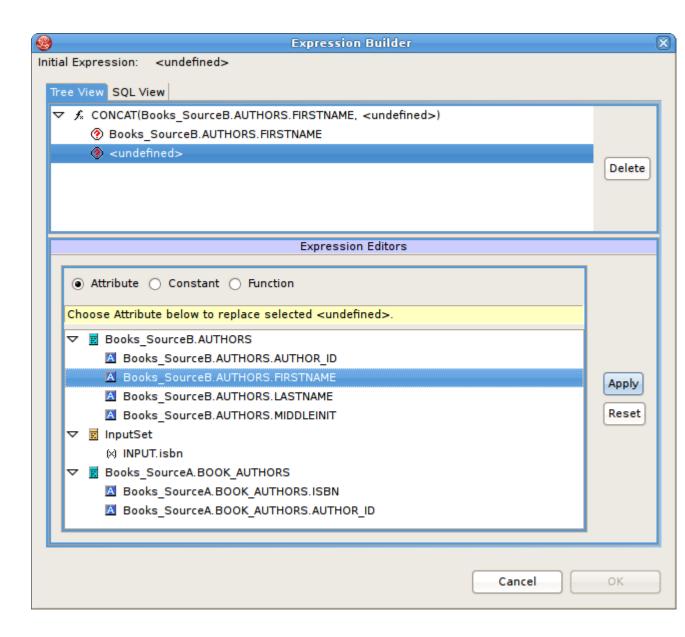


Figure 6.38. New Blank Function Created

You need to set an attribute or constant value for each sub node in the tree to specify the arguments this function needs. You can also nest another function in the tree using the **Function** editor.

		Expression Builder	
al Expression	n: <undefined:< th=""><th>></th><th></th></undefined:<>	>	
ree View SQL	View		
		B.AUTHORS.FIRSTNAME, Books SourceB.AUTHORS.LASTNAME)	
	_	HORS.FIRSTNAME, BOOKS_SOURCED.AOTHORS.LASTNAME)	
	-	THORS.LASTNAME	
0 000	C3_SOULCED.AOT	HORO, EASTMANE	Delete
		Expression Editors	
Value is und	hanged		
Category.	STRING	G1, STRING2)	Apply
Category:	STRING	G1, STRING2)	Apply Reset
Category: Function: Argument	STRING CONCAT(STRING Name	G1, STRING2) Value	
Category: Function: Argument STRING	STRING CONCAT(STRING Name G1	G1, STRING2) Value Books_SourceB.AUTHORS.FIRSTNAME	
Category: Function: Argument	STRING CONCAT(STRING Name G1	G1, STRING2) Value	
Category: Function: Argument STRING	STRING CONCAT(STRING Name G1	G1, STRING2) Value Books_SourceB.AUTHORS.FIRSTNAME	
Category: Function: Argument STRING	STRING CONCAT(STRING Name G1	G1, STRING2) Value Books_SourceB.AUTHORS.FIRSTNAME	
Category: Function: Argument STRING	STRING CONCAT(STRING Name G1	G1, STRING2) Value Books_SourceB.AUTHORS.FIRSTNAME	

Figure 6.39. Nested Function Example

- **Step 8** Click each sub node in the tree and use the editors at the bottom of the dialog box to apply an attribute, constant, or function value to it.
- Step 9 When you have added values to all nodes, as shown below, click OK. to add this expression to your query or Cancel to close the dialog box without inserting the expression.

If the **OK** button does not enable, you have not added a value to all nodes in the tree. You can also nest functions within your expressions by selecting an argument and selecting a function for that argument. The nested function displays in the tree beneath your root function and its arguments display as well. Using the Expression Builder and nested functions, you can create complex logic within your query transformations.

6.3.2. Input Set Editor (XML)

The **Input Set** represents a special class that contains attributes from a parent mapping class. When you create mapping classes for an **XML Document** model, Teiid Designer automatically adds an **Input Set** to all XML transformation diagrams for mapping classes beneath the highest node in the Document meta object.

The **Input Set** proves especially useful for information integration using the **Teiid Server**. Through the **Input Set**, you can access a row of data generated by any XML transformation in a mapping class higher in the XML document's hierarchy. You can use **Input Set** attributes, which are individual columns from the rows of data, within the criteria of an XML transformation query of the child mapping class.

You cannot use the **Input Set** attributes within the SELECT portion of the XML transformation query.

To use an Input Set, you must use the Input Set Editor to bind attributes from parent classes.

Once you have created an **Input Set**, you can use the attributes within it as source material for the XML transformation diagram's query.

The **Input Set** only serves to enable data flow between nested mapping classes. If you use the **Teiid Server** for data access, your applications cannot directly query an **Input Set**. **Input Sets** only display in the XML transformation diagram to which they belong. **Input Sets** do not display on the *Section D.2.1, "Model Explorer View*" view and you cannot use them as you would a normal class, such as for source classes in other transformations.

To open the **Input Set Editor**, either double-click the input set in the **Mapping Transformation Diagram** or click the edit button on the **Input Set** in the diagram. (see below)



Figure 6.40. Edit Input Set Button

Input Parameters	Mapping Class Bindings	▼ 🖬 book
(x) isbn : string	book.isbn : string	New A isbn : string A type : string A title : string A title : string A publisher : string A publishDate : string A edition : string

Figure 6.41. Input Set Editor Panel

The **Input Parameters** table contains a list of mapping attributes within the input set and the mapping attributes bound to input set mapping attributes. The tree on the right displays the parent mapping classes and the attributes available from each.

Using the Input Set Editor, you can:

- Add a mapping attribute from a parent mapping class to the **Input Set**. In the tree on the right, select the symbol for which you want to create an attribute and click **New**. The item displays in the **Input Parameters** and **Mapping Class Bindings** table.
- Delete a mapping attribute from the Input Set. Click the row in the Input Parameters and Mapping Class Bindings table that you want to delete and click Delete. The Teiid Designer removes this row from the table and this mapping attribute from your Input Set.
- Bind and Unbind Input Parameters.

Once you have created the mapping attributes within the **Input Set** that you need, you can use the **Input Set Parameters** within a mapping class transformation to produce mapping attributes you can map to your XML document.

6.3.3. Choice Editor (XML)

Within an XML Document model, a choice compositor defines all possible document data structures (sometimes called fragments) that can appear at that location in an XML instance document. When the Teiid Server populates an XML instance document at runtime based upon your virtual XML document, it will choose the first fragment that matches the criteria you specify within the **Choice Editor**.

To view the choice editor, right-click on the choice node in the mapping diagram's XML Document tree view and select **Edit** from the right-click pop up menu.

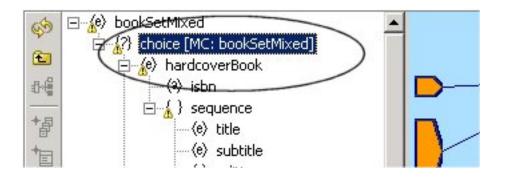


Figure 6.42. Opening The Choice Editor

	Option Name	Include	Criteria
1	😟 hardcoverBook		AudioBooks.bookSetMixedDocument.MappingClass
2	(e) softcoverBook	~	UCASE(AudioBooks.bookSetMixedDocument.Mapp
3	(e) audioBook		AudioBooks.bookSetMixedDocument.MappingClass

Figure 6.43. The Choice Editor

The table on this panel displays fragment options for the choice, each represented by the top node of the document fragment.

The Summary tab, shown below, displays a SQL-like version of the current choice criteria.

Choice Editor: bookSetMixed / choice	Def
 (1) hardcoverBook: AudioBooks.bookSetMixedDocument.MappingClasses.bookSetMixed.TYPE = 'Har (2) softcoverBook: UCASE(AudioBooks.bookSetMixedDocument.MappingClasses.bookSetMixed.TYP (3) audioBook: AudioBooks.bookSetMixedDocument.MappingClasses.bookSetMixed.TYPE IN ('Bo Default: (3) audioBook 	E) LIKE 'SOF%'
Edit Summary	

Figure 6.44. Choice Criteria Summary

6.3.3.1. Using the Choice Editor

You should address each choice option by performing one of the following:

- **Specify a criteria** statement for the Teiid Server to apply in order to determine which elements or elements to insert into the result document.
- Exclude or include the option's fragment from the document.

- Set the elements' criteria test order.
- Set a default action that occurs if none of the criteria you set is met.

6.3.3.2. Excluding Fragments

The XML Schema upon which you based the XML Document model determines the nature of the options available to the choice. A schema you share with other, external sources (such as business partners) might include information that you do not want to include within XML files.

For example, Sample Financial Services shares an XML schema with its partners Example Mutual Insurance, Illustrative Brokerage, and FinancialPartners.com. The partners created the schema broadly, to cover all possibilities for information they might need to interchange. As such, the customer information XML document might include a choice compositor based on a list of all products all companies offer.

However, Sample Financial does not offer a credit card; so it could exclude those elements from the XML documents its Teiid Designer Server creates since it will never have credit card information for an XML document.

The table on the **Choice Editor** contains the **Include** column. By default, all elements specified by the schema are included. You can click to remove the checkmark beside any element you do not want to include within your XML documents generated by this virtual XML document metadata model. By removing the checkmark, you are not removing the element from the XML Document model; you are merely telling the Teiid Server that it will never use this element as part of the choice.

You cannot edit criteria for excluded elements. However, if you exclude an option for which you have established a criteria, Teiid Designer will retain the criteria if you want to include the option in the future.

6.3.3.3. Editing Choice Criteria

- To edit the criteria for a choice element:
 - Step 1 In the table on the Choice Editor panel, select the element you want to edit..
 - Step 2 Click Edit Criteria button to launch the Criteria Builder dialog.
 - Step 3 Use the Criteria Builder to create the conditions for which the Teiid Server will test to determine whether to choose this option in the XML instance document.
 - Step 4 Click OK. The criteria you set displays both in the table and in the summary tab.

You must set a criterion for each option in your document unless you have selected to exclude that option or specify that option will be the default option.

6.3.3.4. Setting Choice Element Order

To edit the criteria for a choice element:

The Teiid Server evaluates the choice criteria in the order in which they appear, and when one choice criteria is met, the Teiid Server populates the XML instance document with that option. The Teiid Server might not test all criteria for all options, so their order matters a great deal.

Therefore, the order in which your options appear within the choice criteria often determines what information appears ultimately in your XML instance documents. You can reorder the option list within the choice to set the order in which the Teiid Server tests the criteria.

To set this order, select an element in the table and use the or or



button to move it into a new position in the table. The new order displays both in the table and in the Choice Criteria box and reorders the XML document as well.

6.3.3.5. Setting a Default Choice Action

The default action represents the course the Teiid Server should take if none of the criteria you set evaluates to true.

You can set this default using the combo box available in the Choice Editor's toolbar to:

- Any of the options within the table except those you have excluded from the document.
- **THROW** to throw a Teiid Server exception.
- **RECORD** to record the Teiid Server exception.
- DISCARD to place no element within the XML instance document.



Note

A default action for the choice criteria must be set.

6.3.4. Recursion Editor (XML)

Some **XML schemas** define data structures that contain self-referencing elements or datatypes. When generating XML documents, such data structures can produce an endless repetition of nested tags. This self-nesting pattern is known as **recursion**.

When generating virtual documents from **XML Schema**, Teiid Designer detects recursive data structures in the XML Schema model and halts the recursive nesting pattern after two cycles. These two cycles serve different purposes when mapping the document:

- The **first cycle** can be thought of as an "**entry condition**" for the **recursion**. The mapping class located at this node defines a normal mapping transformation like that of any other in the document model.
- The **second cycle** defines a mapping transformation that will be performed repeatedly until conditions are met that will halt the document instance being generated by the Teiid Server. This fragment of the document model is called the **recursive fragment**. The mapping transformation for this fragment is no different from the first, except that you can access the first cycle's mapping class attributes, plus you have the opportunity to specify the conditions that will halt the recursion.

You can recognize a mapping class located at the second, recursive document fragment by the looping arrow button in the top-left-hand corner of the diagram object as shown below.

When you model a virtual document based on an **XML Schema** model containing recursion, you can choose whether to treat the nested fragments as recursive. You should only use recursion when the data access pattern from your data source(s) is also recursive; in other words, when the same query transformation should be executed over and over to generate and map the nested document's data content.

By default, Teiid Designer does not mark the recursive fragments in document models to execute recursively in the Teiid Server. To take advantage of this behavior, you must open the **Recursion Editor** in the recursive mapping class *Section D.3.1.2.5, "Mapping Transformation Diagram*", mark the transformation query as recursive, and specify the recursion limit properties.

6.3.4.1. The Recursion Editor

The Recursion Editor lets you enable and limit recursion. The Recursion Editor button only displays on mapping classes, which have recursive patterns. For example, if you have an element named Employee which contains a element named Supervisor which itself contains an Employee element nested within it, you might need to limit the number of times the elements are nested within the document.

You can set the following conditions to limit the recursion:

- A fixed number of results to the query.
- A SQL-based criteria limit condition.
- A combination of both.

То	open	the	Recursion	Editor,	click	on	the	Recursion	Editor	button
----	------	-----	-----------	---------	-------	----	-----	-----------	--------	--------

Q

on the displayed mapping class.

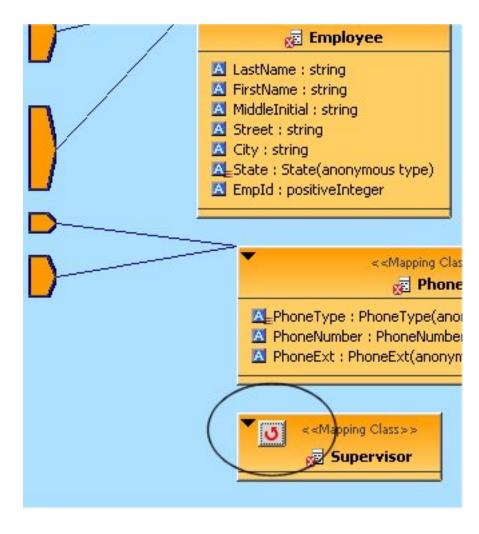


Figure 6.45. Open Recursion Editor Button

Recursion Edito	or: Supervisor		
Count Limit:	5	Action When Count Limit Excee	eded: THROW 💌
Limit Cond	ition Edit		
Employee	Docs.Employee	eDocument.MappingClasses.Sup	ervisor.name = 'Joe Smith'

Figure 6.46. Recursion Editor

- To edit recursion properties:
 - Step 1 Click the Enable Recursion check box if you want the Teiid Server to perform the query you specify to generate the nested tags within the XML document.
 - Step 2 Click the arrows beside the Count Limit box to limit the number of times to recursively perform the query. If you do not set a Limit Condition in the text area, the recursion finishes when the query reaches this limit. You can only set this limit to a maximum supported by your Teild Server. For more information about this limit, contact your system administrator
 - Step 3 Click the Action When Count Limit Exceeded drop down menu to instruct the Teiid Server what to do if it encounters more results for the query than the count limit before it reaches the limit condition.
 - Step 4 Click the Edit button to launch the SQL Section 6.3.1.3, "Using the Criteria Builder" to build a limiting condition for this recursion.



Note

The Teiid Server will evaluate this condition each time it recursively performs this query. If this criteria clause evaluates false, the Teiid Server performs the query recursively again unless it has reached the **Count Limit**. If the criteria evaluates true, the Teiid Server performs the mapping for the current level and ends its recursive loop.

When you have created the criteria, it displays in the Limit Condition box.

When the Teiid Server dynamically populates your XML documents at runtime, it will use the recursion specifications you entered here.

6.3.5. Operation Editor

Editing of **Web Service** Operation transformations is simplified via the **Operation Editor**. When editing a Web Service model, an additional editor tab labeled "**Operation Editor**" is available. This editor, shown below is comprised of:

- **Operations** section showing a tree view of Interfaces and Operations contained within the Web Service model.
- Input Variables section providing editing of desired Input Variable declarations.
- **Procedure** section providing SQL editing of the procedure.

Operation Editor	The second se	S	Cursor at (1, 1)	VE.	A	A
Operations ▼ ① bookCollection ③ getbookCollection	Input Variables The variables representing values fro operation's procedure. Edit variables IN_author : string IN_edition : string IN_isbn : string IN_publishDate : string IN_publishDate : string Procedure The body of the selected operation's BEGIN/END clauses). CREATE VIRTUAL PROCEDURE BEGIN DECLARE string VARIABLES.IN (BOOKCOLLECTION_WS.BOOKCO t, '/*:bookCollection/*:book/@*:is DECLARE string VARIABLES.IN (BOOKCOLLECTION_WS.BOOKCO	proce L_isbn DLLEC sbn'); L_title DLLEC	dure (i.e., it is not neco = xpathvalue TION.GETBOOKCOLLI = xpathvalue	at are a essary ECTION	to incl	ude
🎦 Package Diagram 🥅 Table	t, '/*:bookCollection/*:book/@*:is DECLARE string VARIABLES.IN (BOOKCOLLECTION_WS.BOOKCO t, '/*:bookCollection/*:book/*:title DECLARE string VARIABLES.IN	sbn'); I_title)LLEC e');	= xpathvalue TION.GETBOOKCOLLI			

Figure 6.47. Operation Editor

The **Operations** section contains all interfaces and operations currently defined in the model.

Selecting an operation will display the variables related to the input parameter's content in the **Input Variables** section and the body of its procedure (minus the CREATE VIRTUAL PROCEDURE BEGIN - END keywords and the input variable declarations and assignments) in the **Procedure section**.

When pasting in SQL, do not include the **CREATE VIRTUAL PROCEDURE BEGIN - END** keywords. Input variables will be automatically generated when the Content via Element property is set on an operation's input parameter. Input variables may be edited using the Edit link in the **Input Variables** section, and may only represent XPath values to single attributes and elements within the input contents; other variable declarations and assignments must be typed directly into the **Procedure** section. Clicking the *Edit* link will display the following dialog:

Edit Input Variables	
Request Document Check a request document component to define an input variable for the Web Service procedure. Uncheck a request document component to delete an existing input variable. ▼ ● bookCollection : BooksNS:BookSetNested ▼ ● book : BooksNS:BookNested ▼ ● book : BooksNS:BookNested ③ isbn : BookTypesNS:ISBN ✓ ● title : xsd:string ✓ ● edition : BookTypesNS:BookEdition ✓ ● edition : BooksNS:Authors ● ● publishingInformation : BooksNS:PublishingInformation	Input Variables Select an input variable to the XPath that will be used select its value from the request document. IN_author : string IN_edition : string IN_edition : string IN_publishDate : string IN_publisher : string IN_subtitle : string IN_title : string
XPath for IN_title /*:bookCollection/*:book/*:title	ОК

Figure 6.48. Edit Input Variables Dialog

You can create new operations in the *Operation Editor* by right-clicking an interface or operation and selecting the *Create Operation* action. This will display a dialog for entering the definition for your new operation.

Operation Editor		
Operations		
Products_productd		_
🔅 getproductdata	😳 Create Operation	
Products_products	ymbols	
🔅 getproductsymb	ols	
🔅 new_operation		
484		

Figure 6.49. Create Operation Action

😣 🗆 Create Operation		
Create Operation		
Enter name and properties to crea	ate a new operation	
Model Products_WS	xmi	
Operation Name getProductDa	ta	
Properties Description		
Input Message		
Name	[input_msg	
Schema Content Via Element	productdata_Input	
Output Message		
Name	output_msg	
Schema Content Via Element	productdata_Output	
Output XML Document	productdata_OutputView	
?		Cancel OK

Figure 6.50. Create Operation Dialog

6.4. Managing Model Object Extensions

Extending a model adds extra properties to its model objects. One good use of these extension properties is for passing data to a customized Teiid translator. The Designer model extension framework consists of:

• Model Extension Definitions (MEDs) (See Section 1.3.9, "Model Object Extensions")

- **MED Registry** -keeps track of all the MEDs that are registered in a workspace. Only registered MEDs can be used to extend a model. (See Section D.2.13, "Model Extension Definition Registry View (MED Registry View)")
- MED Editor (See Section D.3.3, "Model Extension Definition Editor")

6.4.1. Create New MED

To create a new MED select the **File > New > Other...** action to display the New wizard dialog. Select the **Teiid Designer > Teiid Model Extension Defn** option which displays the **New Model Extension Definition** dialog. Browse and select existing project or project folder location for MED file and specify unique file name and press **Finish**.



Note

If a project is already selected when wizard is launched, the location field will be pre-populated).

New Moo	del Extension Definition 🛛 🗙
New Model Extension Defin Press the "Finish" button to finish.	ition
Folder Location: Model Extension Definition Name:	ExtendedParts ExtraParts
✓ Open in Extension Editor	
? < <u>B</u> ack	<u>N</u> ext > Cancel <u>Finish</u>

Figure 6.51. MED Editor Overview Tab

6.4.2. Edit MED

To edit an MED file select an existing '.mxd' file in your workspace and right-click select the **Open** action. The MED Editor will be opened to allow editing (See Section D.3.3, "Model Extension Definition Editor").

On the Overview tab, you can specify or change the **Namespace Prefix**, **Namespace URI**, the **Model Class** you wish to extend (Relational, Web Service, XML Document, and Function) and a description. Note that version number is available, but is not currently being used.

After entering the basic MED info, you can now switch to the **Properties** tab and begin creating your extended property definitions for specific model objects supported by selected model class.

Proper	ties 🔻							
• Extend	ed Model Ob	ojects 📍	; /; ×;					
Manage tl	he extended r	model ob	ojects that	have ext	ension	propert	ies	
BaseTabl	e							
• Extensi	on Propertie	25 🖶	7					
	ion Propertie			tended n	adel of	aiact		
Manage e	xtension prop	perties fo	or each ex			-		
Manage e ID	xtension prop Runtime Type	perties fo	or each ex Modifiable			Indexed	Default	Value
Manage e	xtension prop Runtime Type	perties fo	or each ex			-	Default	Value
Manage e ID	xtension prop Runtime Type	perties fo	or each ex Modifiable			Indexed	Default	Value
Manage e ID	xtension prop Runtime Type	perties fo	or each ex Modifiable			Indexed	Default	Value
Manage e ID	xtension prop Runtime Type	perties fo	or each ex Modifiable			Indexed	Default	Value
Manage e ID	xtension prop Runtime Type	perties fo	or each ex Modifiable			Indexed	Default	Value

Figure 6.52. MED Editor Properties Tab

Start by selecting the **Add Extended Model Object** toolbar button to display the **Model Object Name** selection dialog. Select an object and press **OK**.

		Add Model Object Name	×
¢	Model Object Name	Choose the Model object that will have extension properties.	
$\overline{}$	BaseTable		
	Column		
	PrimaryKey		
	ForeignKey		
	AccessPattern		
	UniqueConstraint		
∇	View		
	Column		
	AccessPattern		
\neg	Procedure		
	Column		
	ProcedureParameter		
	Index		
(?	Cancel OK	

Figure 6.53. Select Model Object Name Dialog

Next, select the model object in the **Extended Model Objects** section and use the actions and propertes table in the lower **Extension Properties** section to add/remove or edit your actual extended properties. Selecting the add or edit extension properties actions displays a dialog containing sections to edit general properties, value definition (required, masked, allowed values) as well as display name and description values which can be internationalized.

000	Edit Property Definition				
💣 Property Defin	ition Enter the property's general information, value info				
▼ General					
Enter general prop	erty characteristics				
Model Object:	BaseTable				
Namespace Prefix:	mymodelextension				
ID:	copyable				
Runtime Type:	boolean				
Should only be	modified by advanced users				
Will be used by	Teiid server				
 Value Definition 	n				
Enter the character	ristics of the property value				
📄 A value is requi	ired				
Mask the prope	erty value when shown to user				
The value must	t be one of the following:				
true					
false					
🗌 Use this initial	value:				
The value can only be this value					
Display Name					
Description					
Figure 6 54 Edit Prope	erty Definition Dialog				
Figure 6.54. Edit Property Definition Dialog					

6.4.3. Extending Models With MEDs

MEDs must be applied to a model in order for their extension properties to be available to that model's model objects. To manage the applied MEDs for a specific model select the model and right-click select the **Modeling > Manage Model Extension Definitions** action. This will display a dialog listing the current applied MEDS and actions and buttons to add or remove MEDs from a model, extract a MED from a model and save a copy of it locally as a '.mxd' file and lastly, update the version of MED in a model if it differs from a version in your MED registry.

	00	Manage Model Extension Definitions
--	----	------------------------------------

Manage Model Extension Definitions

Add or Remove Model Extension Definitions to be associated with this model.

Location:	Test/					
Model:	View					
Current	Model Ex	tensions for the	selecte	d model:		
Registere	d Different	Namespace Prefix	Version	Description		
						ſ
						6
0					Cancel	
(I)					cancer	

Figure 6.55. Manage Model Extension Definitions Dialog

Selecting the Add button displays a list of applicable MEDS based on model class.

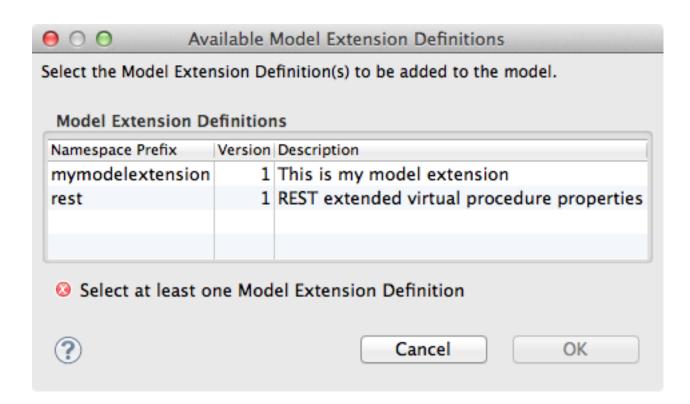
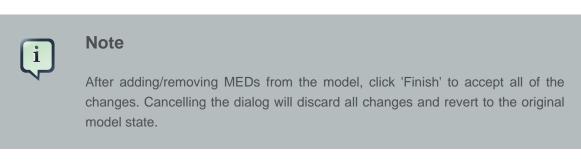


Figure 6.56. Add Model Extension Definitions Dialog



6.4.4. Setting Extended Property Values

Extension properties are user-defined properties available to any extended model object via the Properties View. As shown below, all extension properties are available up under the "Extension" category and are prefixed with a MED's namespace prefix. If there is an initial value for an extension property it will be set to the default value using the property definition found in the MED.

Properties 🛿 📄 Description	ᄩ 券 💀 🛃 🌄 🗖
Property	Value
▼Extension	
mymodelextension:Copyable	true
▼Info	
Object URI	
▼Misc	
Cardinality	L1.1 O
Logical Relationships	
Materialized	™ false
Materialized Table	LE.
Name	NewBaseTable

Figure 6.57. Properties View For Extended Model Object

Metadata-specific Modeling

This chapter discusses various features targeted at defining and managing metamodel-specific objects.

7.1. Relational Source Modeling

7.1.1. Source Function

The ability to utilize functions within your transformations can be a powerful tool for manipulating and managing the structure and content of your virtual data. Teiid Designer exposes a wealth of system functions through the Expression Builder wizard to select and define function calls within your SQL.

Teiid Designer also provides features for defining source functions to represent functions for your target relational data base, as well as user-defined functions coded in java.

If you define a source function in a source model or user-defined function in a view model, that new function will be available to use in your transformations.

See Section 6.2.2, "Create Relational Procedure Wizard" for details.

7.2. Relational View Modeling

This section contains descriptions of various features related to creating and managing relational view model objects.

7.2.1. Create Materialized Views

For any relational view table you can enable it's materialized view option by setting the **Materialized** property to TRUE and setting the **Materialized Table** reference, as shown in the figure below.



Note

You are required to have already created your relational tables.

nfo	
lisc	
Cardinality	Let 0
Logical Relationships	
Materialized	If true, the table can be materialized (cached). Only applie
Materialized Table	PARTS (Path=/PartsTest/MaterializedParts.xmi/MaterializedParts)
Name	A PARTS
Name In Source	A PARTS
Supports Update	😽 false
System	🙀 false
	isc Cardinality Logical Relationships Materialized Materialized Table Name Name In Source Supports Update

Figure 7.1. Materialized Table Properties

Designer includes a feature to assist in quickly creating materialized relational tables based on your existing view tables.

To create materialized views:

• Step 1 - Right-click on one or more view tables in the Section D.2.1, "Model Explorer View" and select the Modeling > Create Materialized Views action.

Te Pa	ckage Diagram	
🕨 🔂 PA	राऽ	
▶		>
> 🛃 ST > 🛃 SU	New Sibling	>
P ≣ 30	New Association	>
▷ 😭 Produ	Modelina	Create Materialized Views
👕 Salesfor		🔯 Create Web Service
👕 SearchPr	- WREND	뭡 Add To New Custom Diagram
Testir	-	Add to Metadata Favorites
TestRels	of Cut	Ctrl+X
<	📄 Сору	Ctrl+C
and the state of the	📋 Paste	Ctrl+V

• Step 2 - In the Create Materialized View Model dialog specify or select a target relational model for your generated relational tables.

Create Materialized View Model 🛛 🛞					
Create Materialized View Model					
All inputs OK. Press Finish to create materialized view tables.					
Materialization Model	MaterializedParts				
Target Location					
	MaterializedParts				
Selected Virtual	lables				
PARTS					
SHIP_VIA					
(?)	Ca	ncel <u>F</u> inish			
U					

• Step 2a - Selecting the browse '...' button displays the **Relational Model Selector dialog** where you select an existing relational model or specify a unique name for a new model.

	Relational Mo	del Selector Dialog		×
Select the Relational	Model or schema locat	ion within a model in w	which the entities	will be created.
🔻 🛃 PartsTest				
Northwind.x	mi			
PartsSupplie	er.xmi			
PartsTestVD				
PartsViewMo				
ProductsOra	icle.xmi			
New Model Name:	MaterializedParts			
Click OK to create the	relational model			
?		Properties	Cancel	ОК

• Step 3 - Click OK to create relational tables corresponding to your selected view tables and automatically set the **Materialized** property to TRUE and the **Materialized Table** reference value to your newly generated table.

When finished your view tables will be configured with their new materialized properties and the corresponding relational tables will be shown in their package diagram.

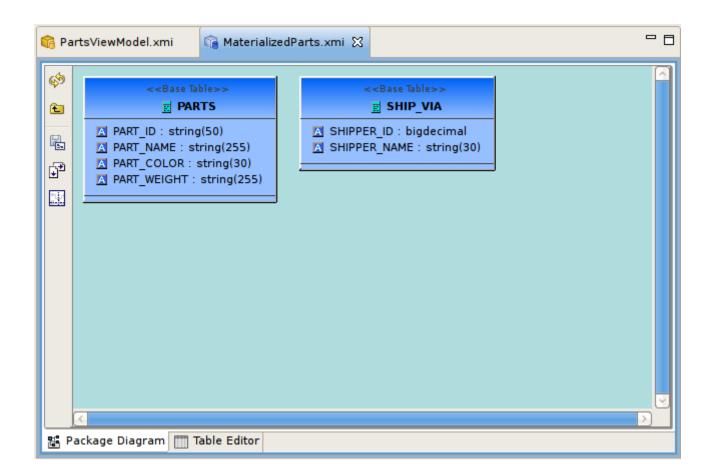


Figure 7.2. Materialized Table Properties

7.3. XML Document Modeling

7.3.1. Create XML View Documents from schema

You can create XML View Documents by selecting an element in the Model Explorer and selecting the **Modeling > Create XML View Documents** action.

V us bookDatatypes.xsd			
▼ 🕞 Books.xsd			
S Books.xsd			
4— BookDatatyp	es.xsd		
🗓 MetaMatrix D	ata Set Library Version 1.1. Boo		
e bibliography	: BooksNS:Bibliography		1
e bookListing :	Now Child	>	
e bookCollection	New Sibling	>	
e bookSetMixe	New Association	>	
Bibliography	Modeling	\rightarrow	Create Scalar To XML Tr
Bilbiography			Create XML To Scalar Tr
BookSetFlat	♀ ◇ Undo Set org.ecli) to FK_Account_AccountId		Create XML View Docum
BookFlat	🖏 Redo		
			☆ Add to Metadata Favorit
BookSetNest	of Cut	Ctrl+X	
BookNested	Copy	Ctrl+C	

Figure 7.3. Create XML View Documents Action

The action will query you for a target XML Document model. You can either select an existing XML Document model from your workspace, or enter a unique model name and the wizard will create a new model for you.

Select or create XML View model	X
▼ 🛃 BooksXMLTest	<u></u>
BookDatatypes.xsd	
Books.xsd	
Booksinput.xsd	
👂 😪 BooksXML.xmi	
▽ ট PartsTest	
Northwind.xmi	
PartsSupplier.xmi	
PartsTestVDB.vdb	
ProductsOracle.xmi	~
New Model Name:	
😢 Enter name and click OK to create a new XML View Model in folder "BooksXMLTest"	
? Properties Cancel	ок

Figure 7.4. Select or Create XML View Model Dialog

After selecting or creating your new XML Document model, the XML Document builder page will be displayed. This page is explained in greater detail in *Section 4.3.2, "Build XML Documents From XML Schema*" section.

Build XML Documents From XML Schema					
Select XML Sc Select XML Schen		lements. Or press Finish to create an empty XML Document.			
XML Schema File:	BooksXMLTest/Books.xsd				
Select one or more Schema Root Elements to generate Virtual Documents					
Schema Root Ele		Virtual Documents			
e bookListing :	on : BooksNS:BookSetNested BooksNS:BookSetFlat d : BooksNS:BookSetMixed	bibliography : BooksNS:Bibliography			
Reset		Keep sorted alphabetically			
Document Option	Document Options				
Build full virt	ual documents from the schema				
O Build only fir	st level of each document (for larg	e schema where many portions will not be used)			
Mapping Options					
✓ Build Mappin					
Mapping Class	s Locations				
Locate at	XML Schema iteration nodes (max	occurs > 1) - for schema-compliant data access			
🔿 Locate at	XML Schema complex types - for s	semantic use only			
Mapping Class	s Column Datatypes				
Use XML f	types from the document				
O Use "String" mapping type (disables automatic output conversion to closest built-in XSD type)					
		< <u>Back</u> <u>N</u> ext > Cancel <u>Finis</u>			

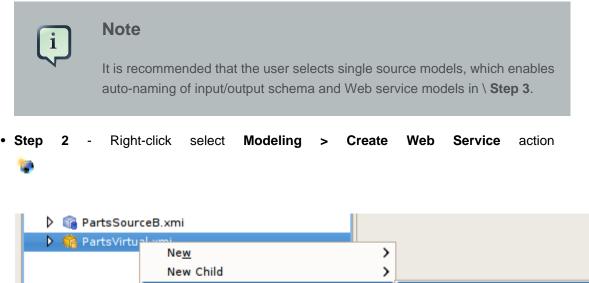
Figure 7.5. Build XML Documents From XML Schema Dialog

7.4. Web Services Modeling

7.4.1. Create Web Service Action

This method is recommended for experienced users for consistent and rapid deployment of **Web** services designed to query relational sources. It provides detailed control of all **Web** service interfaces, operations and required transformations from **XML Views**

- To create a Web service model from relational models or objects:
 - Step 1 Select any combination of relational models, tables and/or procedures in the Section D.2.1, "Model Explorer View" tree.



	New child	1	
	Modeling	>	😻 Create Web Service
🐗 Teiid 🕴	Open Open Wit <u>h</u>	_	Generate SQL Relationships
		、	Transformations - Export SQL
		_	Transformations - Import SQL
	Eopy		

Figure 7.6. Create Web Service Action

• Step 3 - In the Create Web Service dialog, specify file names for the generated Input Schema file, Output Schema file and Web service model. Change options as desired. Click Finish when done.

😸 🗆 Create	Web Service			
Create Web Ser				
Click Finish to ge	enerate.			
Location				
Specify location	for generated m	odel(s) (i.e. Project or Folder)		
Customer_Acc	ounts			Browse
Model Definiti	on			
🧭 Generate	Input Schema	CustomerAccounts_Input		Generate flat XSD
	Output Schema	CustomerAccounts_Output		Generate flat XSD
	Web Service	CustomerAccounts_WS		Generate default SQL for XML mappings
			_	
?				Cancel Finish

Figure 7.7. Create Web Service Dialog

• Step 4 - When model generation is complete, a confirmation dialog should appear. Click OK.

9	Generation completed successfully.	8
?	Generation completed successfully	
New Mo	odel PartsVirtual_Output.xsd created	
New Mo	odel PartsVirtual_Input.xsd created	
New Mo	del PartsVirtual_Output_View.xmi created	
New Mo	odel PartsVirtual_WS.xmi created	
	ОК]

Figure 7.8. Generation Completed Dialog

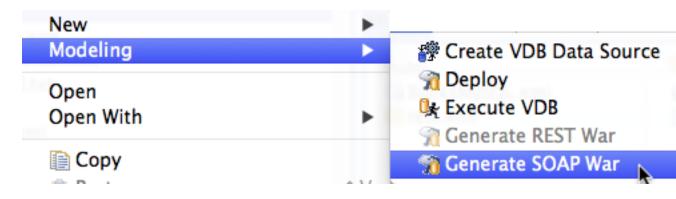
7.4.2. Web Services War Generation

Teiid Designer allows you to expose your VDBs via a SOAP or REST interface. JBossWS-CXF or RESTEasy wars can be generated based on models within your VDBs. This section describes these wizards in detail.

7.4.2.1. Generating a SOAP War

The Teiid Designer provides SOAP web service generation capabilities in the form of a JBossWS-CXF war. Once you have added your **Web Service Models** as described in *Section 4.5, "Creating Web Service View Model"* to your **VDB**, deployed the **VDB** to a running Teiid instance and created your VDB's data source, you are ready to expose the web service using the generated war.

- To generate a new SOAP war using the VDB:
 - Step 1 Right-click on the VDB containing your web service model(s) and select the Modeling > Generate JBossWS-CXF War action.



• Step 2 - Fill in missing properties in Web Service War Generation Wizard shown below.

-			
× Create Web Service	WAR File		
Create a WAR file to deploy a Enter the required information	as a Web Service In, then click OK to create the WAR file.		
WAR Creation Information			
Context Name:	CustomersVDB		
Web Server Host: lo	calhost		
Web Server Port: 80	080		
VDB JNDI Name: {R	{REPLACE_WITH_VDB_JNDI_NAME}		
General HTTPBasic Optio	ns WS-Security Options		
When using HTTPBasic sec	curity, a local Teiid connection is required using the PassthroughAuthentication property.		
None			
HTTPBasic			
O WS-Security (Username	e Token)		
General Options			
Enable MTOM			
Target namespace: ht	ttp://teiid.org		
WAR File Save Location: //home/blafond/Runtime-Workspaces/WARFiles			
Change			
Restore Default			
?	Cancel OK		

Figure 7.9. Generate a SOAP War Web Service Dialog

Table 7.1. Field Descriptions

Field Name	Description
Name	The name of the generated war file.
Host	The server host name (or IP).
Port	The server port.

Field Name		Description
VDB JNDI Name		The JNDI connection name to the deployed Teiid source VDB.
Security options		• None - no username/password required to connect to the VDB through the generated web service.
		 HTTP Basic - the specified security realm and role will be used. The default realm value is the realm that comes out-of-the-box with Teiid (teiid-security). The role needs to be defined in the appropriate security mechanism. In the case of Teiid, use the teiid-security-roles.properties file. When using HTTPBasic, a local Teiid connection using the PassthroughAuthentication property is required. See the Teiid user's manual for details on PassthroughAuthentication. WS-Security - a password callback class will be generated for you which will validate that the username/password values you specified in the war generator dialog are passed in. This is meant to be a testing mechanism for your WS-Security mechanism for your WS-Security mechanism should be implemented in this class. All source code is included in the generated war along with the compiled class files.
Target namespace		This is the target namespace that will be used in the generated WSDL and subsequent generated web service classes.
MTOM (Message T Optimization Mechanism)	ransmission	If selected, MTOM will be enabled for the web service endpoint(s). You will also need to update your output schema accordingly by adding the xmlns:xmime="http://www.w3.org/2005/05/xmlmim" schema and adding type="xs:base64Binary" xmime:expectedContentTypes="application/octet-stream" to the output element you wish to optimize.

Field Name	Description
War File Save Location	The folder where the generated WAR file
	should be saved.

• Step 3 - Click OK to generate the web service war. When war generation is complete, a confirmation dialog should appear. Click OK.

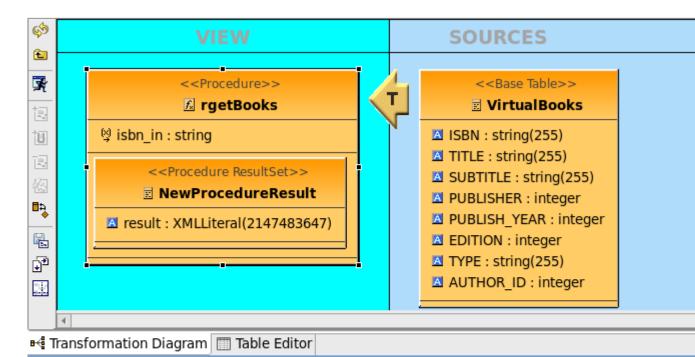


Figure 7.10. Generation Completed Dialog

7.4.2.2. Generating a REST War

In Teiid Designer, it is also possible to expose your VDBs over REST using a generated RESTEasy war. Also, if your target virtual model has update, insert and delete SQL defined, you can easily provide CRUD capabilities via REST. Accepted inputs into the generated REST operations are URI path parameters and/or XML/JSON. JSON is exposed over a URI that includes "json". For example, "http://{host}:{port}/{war_context}/{model_name}/resource" will accept URI path parameters and/or XML while "http://{host}:{port}/{war_context}/{model_name}/json/resource" will accept URI path parameters and/or JSON.

• Step 1 - In a virtual model, add a procedure(s) that returns an XMLLiteral object. The target of your procedure can be any models in your VDB. Here is an example procedure that selects from a virtual table (VirtualBooks) and returns the results as an XMLLiteral:



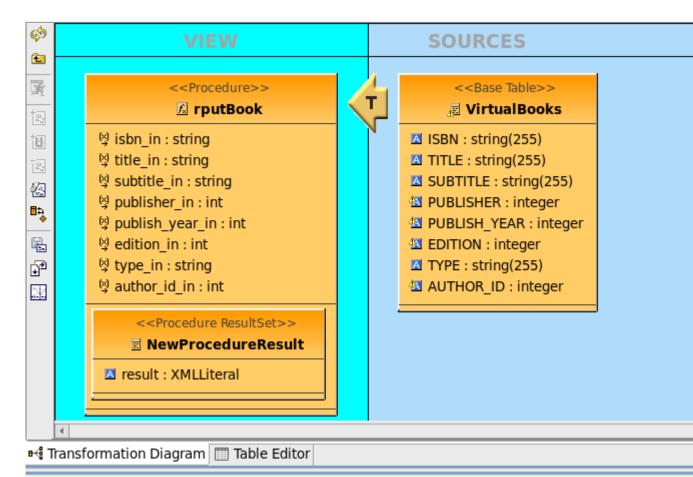
Transformation Editor

CREATE VIRTUAL PROCEDURE BEGIN SELECT XMLELEMENT(NAME books, XMLAGG(XMLELEMENT(NAME book, XMLFOREST(RestfulBooks RestfulBooks.VirtualBooks.SUBTITLE, RestfulBooks.VirtualBooks.PUBLISHER, RestfulBooks.VirtualBooks RestfulBooks.VirtualBooks.TYPE, RestfulBooks.VirtualBooks.AUTHOR_ID)))) AS result FROM RestfulBooks RestfulBooks.VirtualBooks.ISBN; END

Notice the syntax used to convert the relation table result of the select from VirtualBooks, to an XMLLiteral. All XML functions are documented in the **Scalar Functions** chapter of the **Teiid Reference Guide**.

Here is an example of an update procedure that will insert a row and return an XMLLiteral object:

7



```
Transformation Editor
```

CREATE VIRTUAL PROCEDURE
BEGIN
DECLARE integer VARIABLES.insert_count = 0;
INSERT INTO RestfulBooks.VirtualBooks (RestfulBooks.VirtualBooks.ISBN, RestfulBooks.VirtualBook
RestfulBooks.VirtualBooks.PUBLISH_YEAR, RestfulBooks.VirtualBooks.EDITION, RestfulBooks.VirtualB
RestfulBooks.rputBook.title_in, RestfulBooks.rputBook.subtitle_in, RestfulBooks.rputBook.publisher_i
RestfulBooks.rputBook.type_in, RestfulBooks.rputBook.author_id_in);
VARIABLES.insert_count = VARIABLES.ROWCOUNT;
IF(VARIABLES.insert_count = 1)
BEGIN
SELECT XMLCOMMENT('Insert was successful!') AS result;
END
ELSE
BEGIN
SELECT XMLCOMMENT('Insert failed!') AS result;
END
END

The input format for the REST procedure could be URI parameters, an XML/JSON document, or some combination of both. When using an XML document your root node should be **<input>** and the XML nodes should correspond to order of the procedure's input parameters. For example, here is the input for the above insert procedure:

<input> <ISBN>0-13-014714-1-999999</ISBN> <TITLE>The XML Handbook</TITLE> <SUBTITLE>Updated Edition</SUBTITLE> <PUBLISHER>16</PUBLISHER> <PUBLISH_YEAR>2000</PUBLISH_YEAR> <EDITION>2</EDITION> <TYPE>Hardback</TYPE> <AUTHOR_ID>49</AUTHOR_ID> </input>

Figure 7.11. Sample XML Input

When using a JSON document, your values should match the order of your procedure input parameters as well. Here is the input for the above insert procedure:

```
{
"ISBN": "1-55615-484-4",
"TITLE": "Code Complete",
"SUBTITLE": "A Practical Handbook of Software Construction",
"PUBLISHER": 5,
"PUBLISH_YEAR": 1993,
"EDITION": 1,
"TYPE": "Hardback",
"AUTHOR_ID": 31
}
```

Figure 7.12. Sample JSON Input

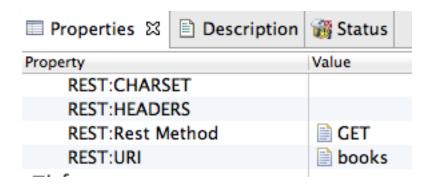
• Step 2 - Now we need to identify our procedure as REST eligible. To do this, select the procedure and note the REST properties in the property panel. The REST Method and URI are the only required properties to make the procedure REST eligible. All of the properties are defined in the table below:

Table 7.2. Properties for RESTful Procedures

Property Name	Description	Required
Rest Method	The HTTP method that will determine the REST mapping of this procedure. Supported	Yes

Property Name	Description	Required
	methods are: GET, PUT, POST and DELETE	
URI	The resource path to the procedure. For example, if you use "books/{isbn}" as your URI value for a procedure, http://{host}:{port}/ {war_context}/ {model_name}/books/123 would execute this procedure and pass 123 in as a parameter.	Yes
Description	Optional property that describes the procedure. This value is used by Swagger when documenting the REST API.	No
CHARSET	Optional property for procedures that return a Blob with content type that is text- based. This character set to used to convert the data.	No
HEADERS	Semi-colon delimited list of HTTP Header parameters to pass into the REST service operation. Ex. header1;header2;etc. These value will be passed into the procedure first.	No

Here's what the above example would look like in the Property tab:



i

Note

The generated URI will have the model name included as part of the path, so full URL would look like this: http://{host}:{port}/{war_context}/{model_name}/ books/123. If you wanted a REST service to return all books, you would write your procedure just as it is above, but remove the input parameter. The URI property would then just be 'books' (or whatever you want) and the URL would be http://{host}:{port}/{war_context}/{model_name}/books.

Once you have added all of your procedures along with the required extended properties, be sure and add the model to your VDB or synchronize if it's already included in the VDB. You will then need to re-deploy the VDB.

Important

If you redeploy your VDB during development, you may receive an "Invalid Session Exception" due to a stale connection obtained for the pool. This can be corrected by flushing the data source or, alternatively, you could add a test query to your VDB connection's -ds.xml file. This will insure you get a valid connection after redeploying your VDB. The syntax for the test query is as follows: <checkvalid-connection-sql>some arbitrary sql</check-valid-connection-sql>"

- Step 3 3. If you have not already done so, you will need to create a data source for your VDB. This can be done in the Teiid View of Designer. Right-click on your deployed VDB and select Create Data Source. The Generate REST WAR dialog will ask you for the JNDI name for your created source so that it can connect to your VDB.
- Step 4 Right-click on the VDB containing your virtual model(s) with REST eligible procedures and select the **Modeling > Generate RESTEasy War** action. If there are no procedures that are REST eligible, the "Generate RESTEasy War" option will not be enabled.

📋 Borl	opectude.	/BOOKSKESTW
	Ne <u>w</u>	bksToo.xmi /BooksRESTW
🗴 M	Modeling	🕨 😚 Deploy
📑 m	Open	🚱 Execute VDB
• 🔊 M	Open With	🕥 Generate JBossWS-CXF War
• 😘 Re 🚃		👘 Generate RESTEasy War

• Step 5 - Fill in missing properties in the REST War Generation Wizard shown below.

New Modeling	Create VDB Data Source
Open Open With	Contraction Contra
	😙 Generate REST War 📡
Copy	Generate SOAP War

Figure 7.13. Generate a REST WAR War File Dialog

Table 7.3. Field Descriptions

Field Name	Description
Name	The name of the generated war file.
Connection JNDI Name	The JNDI connection name to the deployed Teild source VDB.
War File Save Location	The folder where the generated WAR file should be saved.
Include RESTEasy Jars in lib Folder of WAR	If selected, the RESTEasy jars and there dependent jars will be included in the lib foled of the generated WAR. If not selected, the jars will not be included. This should be de- selected in environments where RESTEasy is installed in the classpath of the server installation to avoid conflicts.

• Step 6 - Click OK to generate the REST war. When war generation is complete, a confirmation dialog should appear. Click OK.

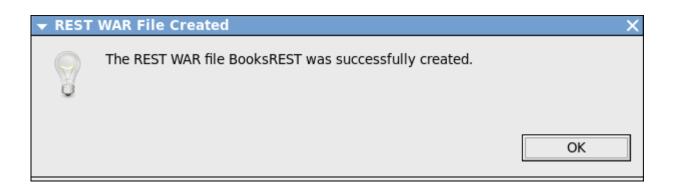


Figure 7.14. Generation Completed Dialog

7.4.2.3. Deploying Your Generated WAR File

Once you have generated your war file, you will need to deploy it to your JBoss AS instance. There are a few ways to accomplish this.

• From JBDS or JBoss Tools

1. Insure target JBossAS is configured and running.

2. Select your WAR file in the Model Explorer view. If you didn't generate your war to that location, you can copy and paste it there.

3. Right-click on the WAR file and select 'Mark as Deployable'. This will cause you WAR file to be automatically deployed the JBoss AS instance you have defined.

🔻 불 SoapyBooksProje	ect	
🔋 BooksWS.vdb		🚳 Dynamic Web Project
📄 BooksWS.war	New	🖘 Java EE Web Dreiest
👂 院 MySqlBooks_C		· · ·
🕨 🕞 MySqlBooks_C	<u>O</u> pen Open Wit <u>h</u>	•
👂 🕵 MySqlBooks_V	<u>С</u> ору	
👂 📬 MySqlBooks.xi	Paste	Ctrl+V
👂 🚰 test	<u>D</u> elete	
IS_States	Refactor	•
	🚵 Import	
	🛃 Exp <u>o</u> rt	
	Refresh	
	👈 Mark as Deploya	able
	<u>V</u> alidate	

• Using the JBoss AS Administration Console

Using the administration console that comes with JBoss AS, you can deploy WAR files. The administration console is available at http://{host:port}/admin-console. Once logged on, simply use the "Add a New Resource" button of the "Web Application (WAR)" resource folder.

Manual Deployment to JBossAS

It is possible to deploy the generated WAR by manually copying the file to the "deploy" folder of the target JBoss AS. If the server is running, the WAR will deploy automatically via "hot" deploy. Otherwise, the WAR will deploy at the next start of the server.

7.4.2.4. Testing Your Generated WAR Files

Once you have deployed your war file, you are ready to test it out. There are a few ways to accomplish this.

SOAP WAR Testing

• Determining Your WSDL URL

You can get your WSDL URL at http://{server:port}/jbossws/services. This is where all the deployed web services for the target JBossAS server will be listed. Find your service and click the Endpoint Address link. This will retrieve your web service's WSDL and the WSDL URL address will appear in the browser's address bar.

Endpoint Name Endpoint Address	jboss.ws:context=books, http://127.0.0.1:8080/boo	endpoint=MySqlBooks_BOOKS bks/MySqlBooks_BOOKS	
StartTime		StopTime	
Mon Jun 11 15:16	:02 CDT 2012		
RequestCount		ResponseCount	FaultCount
0		0	0
MinProcessingTim	ne	MaxProcessingTime	AvgProcessingTime
0		0	0

Now that you have your WSDL URL, you can use any SOAP testing tool such as the Web Service Tester that comes with JBDS and JBoss Tools or an external tool like soapUI.

• Using the JBoss AS Administration Console

Using the administration console that comes with JBoss AS, you can deploy WAR files. The administration console is available at http://{host:port}/admin-console. Once logged on, s simply use the "Add a New Resource" button of the "Web Application (WAR)" resource folder.

REST WAR Testing

• What is my URI?

When you modeled your REST procedures, you assigned a URI for each HTTP Operation you defined along with the corresponding operation (GET, PUT, POST or DELETE). The full path of each URI is defined as /{war_context}/{model_name}/{resource} for XML input/output and / {war_context}/{model_name}/fresource} for JSON input/output.

Swagger is a framework that exposes REST APIs and it is embedded in the generated REST WAR. This allows you to execute your methods directly from the deployed WAR. It is available at the root of the WAR (http://{host:port}/{context}).



Using your REST URL, you can also use any testing tool with REST support such as the Web Service Tester included with JBDS and JBoss Tools or an external tool like soapUI or cURL.

7.4.2.5. Auto-generate a REST WAR

Instead of generating the REST WAR in Designer, you can also set a property on your VDB containing a REST eligible procedure to auto-generate a REST war at VDB deployment time. In the VDB Editor, the properties tab has a checkbox to tell the Teiid engine to generate a REST WAR upon deployment. In addition to checking the **Generate REST WAR** checkbox, a role of **rest** is required for the user accessing the REST URL. Please see the Teiid documentation for more information about the auto-generated REST WAR.

Location: /Customer_A Version 1 Sys	Accounts	Last validated :	Wed Jul 08 09:22:25 CDT 2015	Teiid runtime version :
Data Roles Properties General Query Timeout (sec) Security Domain GSS Pattern Password Pattern Authentication Type	my_domain *.xyz #\$abc		lowed Languages	
Content Advanced				

Figure 7.16. REST WAR Auto-generation Option

User Defined Functions

Teiid Designer enables users to extend Teiid's scalar or aggregate function library via modeling and definition of custom or User Defined Functions(UDFs).

The following are used to define a UDF.

- Function Name When you create the function name, keep these requirements in mind:
 - You cannot overload existing Teiid System functions.
 - The function name must be unique among user-defined functions in its model for the number of arguments. You can use the same function name for different numbers of types of arguments. Hence, you can overload your user-defined functions.
 - The function name cannot contain the '.' character.
 - The function name cannot exceed 255 characters.
- Input Parameters- defines a type specific signature list. All arguments are considered required.
- Return Type- the expected type of the returned scalar value.
- Pushdown can be one of REQUIRED, NEVER, ALLOWED. Indicates the expected pushdown behavior. If NEVER or ALLOWED are specified then a Java implementation of the function should be supplied. If REQUIRED is used, then user must extend the Translator for the source and add this function to its pushdown function library.
- **invocationClass/invocationMethod** optional properties indicating the method to invoke when the UDF is not pushed down.
- Deterministic if the method will always return the same result for the same input parameters. Defaults to false. It is important to mark the function as deterministic if it returns the same value for the same inputs as this will lead to better performance. See also the Relational extension boolean metadata property "deterministic" and the DDL OPTION property "determinism". Defaults to false. It is important to mark the function as deterministic if it returns the same value for the same inputs as this will lead to better performance. See also the Relational extension boolean metadata property "deterministic" and the DDL OPTION property "determinism".

Even Pushdown required functions need to be added as a UDF to allow Teiid to properly parse and resolve the function. Pushdown scalar functions differ from normal user-defined functions in that no code is provided for evaluation in the engine. An exception will be raised if a pushdown required function cannot be evaluated by the appropriate source.

8.1. Modeling your functions

Teiid Designer provides a new object wizard to assist in modeling your UDF. Simply right-click select a model and right-click select **New > Procedure** action. In the wizard there will be an option

to specify the functions's jar location on your file system, as well as the java class and java method. (See Section 6.2.2, "Create Relational Procedure Wizard" for further information)

If creating a function by other means, you'll need to select the procedure and edit the different properties for the procedure in the Properties view.

🔲 Properties 🔀 👯 Serve	rs 🖹 De	scription	🎆 Status			
				달 😫	EZ.	~
Property		Value				
▼ Extension						
relational:Aggregate		false				
relational:Determinist	ic	false				
relational:Function Category		📄 name	Utils			
relational: Java Class		📄 org.te	iid.designe	r.examples.NameE	Builder	
relational: Java Method		📄 buildN	Name			
relational:Native Quer	/					
relational:Non-Prepared		false				
relational:Returns null on null		false				
relational:UDF Jar Path		📄 lib/na	me_builder.	jar		
relational:Variable Arg	uments	false				

Figure 8.1. VDB UDF Jar Files

A User Defined Function represents a defined method in a java class. In order for the Teiid runtime to register the function and allow it's use in your transformations, you'll need to specify the following as properties in the figure above:

- **Function Category** unique name for a grouping of user-defined functions. This category will be selectable in the *Expression Builder* wizard so you can locate and select your function.
- Java Class fully qualified name of the java class containing your function method.
- Java Method the name of the function method in your java class.
- **UDF Jar Path** the relative path in your project for the jar containing your UDF which will be in a *lib* folder in your project. If you are defining your jar for the first time and there is no jars in your lib folder, the property editor will allow you to select a jar from your file system. If one or more jars already exist in a *lib* folder, the editor will allow you to choose between selecting from your workspace or your file system.

8.2. Utilizing your UDFs in transformations

Once you've modeled your functions you can use your function calls in your transformation SQL. These functions will be accessible through the Transformation Editor's expression builder wizard,

just like any other built-in function. (See Section 6.3.1.4, "Using the Expression Builder" for further information)

8.3. Including functions in your VDB

In order for Teiid to become aware of your functions, the actual code must be deployed on your server and available to your Teiid submodule. Your Teiid Designer workspace is aware of any models containing functions and their referenced jars and class info. When a source model containing UDFs is added to a VDB, the jar containing the defined function will be added to your VDB and visible in the VDB Editor's UDF Jars tab.

1odel.xmi 🔀 📋 Al	lModelsVDB.vdb ⊠			
: /AllModelsProject		Last validated :	Wed Jun 11 09:56:00 CDT 2014	Teiid runtime version :
n 1 Synchron	hize All Show Impor	rt VDBs		
UDF Jars Other File	es Description			
	Location in VDB	Description		
ame_builder.jar	/lib			
	•			
Ē.				
Advanced				
	n: /AllModelsProject	a: /AllModelsProject a 1 Synchronize All Show Impore a UDF Jars Other Files Description builder.jar /lib b	a: /AllModelsProject Last validated : n 1 Synchronize All Show Import VDBs i UDF Jars Other Files Description i Location in VDB Description ame_builder.jar /lib	A: /AllModelsProject I Synchronize All Show Import VDBs UDF Jars Other Files Description Location in VDB Description ame_builder.jar /lib

Figure 8.2. VDB UDF Jar Files

8.4. Using legacy UDF models

Function models have been deprecated and tooling support to create or edit them will be removed in Teiid Designer 11.0. In 10.0, we've added a feature to convert the functions in your legacy function models into User Defined Functions (virtual procedures) in a target view model.

To convert your function model:

• Step 1 - In the Model Explorer select your function model and select the Modeling > Convert Legacy Functions to Virtual Procedures action.

🔊 oldUDF.vdb		
v 👘 udfTokenFunctions.		
撞 import declarati	New	•
🚰 Package Diagram	New Child	
🕨 🔣 getToken	Modeling	 Convert Legacy Functions to Virtual Procedures
▶ 🛃 getToken_2 🗊 Project_1	Open	In Show Model Statistics
	Open With	Update Imports

Figure 8.3. Convert Legacy Functions to Virtual Procedures Action

• Step 2 - In the dialog, check the functions you wish to convert and select the target virtual model and click OK>.

🛞 💷 Convert Legacy Functions	
Convert Legacy Functions Click OK to generate your UDF virtual procedures	
Select functions to convert	
🛃 getToken	
getToken_2	
Target Model	
Name /BQT/MyUDFs.xmi	
?	Cancel OK

Figure 8.4. Convert Legacy Functions Dialog

The selected scalar functions are now defined as virtual model User Defined Functions and available to use in your transformations

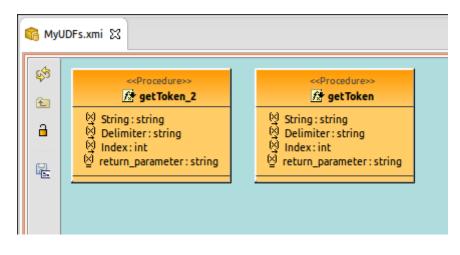


Figure 8.5. Converted Functions Diagram

Editing Models and Projects

Teiid Designer offers three basic model edit actions: **Rename**, **Move** and **Save As...** and one project-related action, **Clone Project**. These actions are described below.

9.1. Rename A Model

- To rename a model in your workspace:
 - Step 1 Select a model in the Section D.2.1, "Model Explorer View".
 - Step 2 Right-click select the Refactor > Rename action.

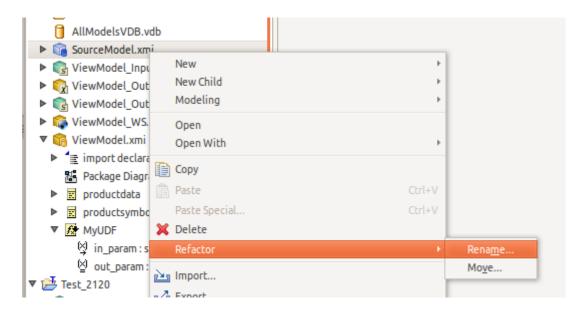


Figure 9.1. Refactor Rename Action In Model Explorer

• Step 3 - Specify unique model name in the Rename Model File dialog. Click OK.

Rename Resource SourceModel.xmi			
New name:	New5ourceModel.xmi		
	Preview > Cancel OK		

Figure 9.2. Rename Model File Dialog

• Step 4 (Optional) - The next Rename Resource File dialog will preview the changes that will be applied as part of renaming your model. Click OK to execute the changes or Cancel.

ViewModel.xmi-AllModelsProject		
🛛 🞯 🏡 Rename 'AllModelsProject/SourceModel.xmi' to 'NewSourc	eModel.xmi'	
🥑 赨 Synchronize resource AllModelsVDB.vdb		
· · · · · ·		
🗊 ViewModel.xmi	📣 😥 🖓	
Driginal Source	Refactored Source	
<nested td="" transformation:transfor<="" xsi:type="transformation:Transformat</td><td><nested xsi:type="></nested>		
<inputs href="SourceModel.xmi#mmuuid/2253€</td><td><pre><inputs href=" newsourcemodel.xmi#mmuuid<="" pre=""></inputs>		
<nested pre="" transformation:transfor<="" xsi:type="transformation:Transformat</td><td><pre><nested xsi:type="></nested>		
<inputs href="SourceModel.xmi#mmuuid/41c61</td><td><pre><inputs href=" newsourcemodel.xmi#mmuuid<="" pre=""></inputs>		
<nested pre="" transformation:transfor<="" xsi:type="transformation:Transformat</td><td><pre><nested xsi:type="></nested>		
<inputs href="SourceModel.xmi#mmuuid/0ec2c</td><td><pre></td></tr><tr><td></nested></td><td></nested></td></tr><tr><td><nested xsi:type=" td="" transformation:transformat<=""><td><pre><nested a79d{<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transfor</pre></td></tr><tr><td><inputs href="><td><pre></pre></td></nested></pre></td></inputs>	<pre><nested a79d{<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transfor</pre></td></tr><tr><td><inputs href="><td><pre></pre></td></nested></pre>	<pre></pre>
<nested pre="" transformation:transfor<="" xsi:type="transformation:Transformat</td><td><pre><nested xsi:type="></nested>		
<inputs href="SourceModel.xmi#mmuuid/allbo</td><td><pre></td></tr><tr><td></nested></td><td></nested></td></tr><tr><td><nested xsi:type=" td="" transformation:transformat<=""><td><pre><nested 4dbl1<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transfor</pre></td></tr><tr><td><inputs href="><td><pre></pre></td></nested></pre></td></inputs>	<pre><nested 4dbl1<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transfor</pre></td></tr><tr><td><inputs href="><td><pre></pre></td></nested></pre>	<pre></pre>
<pre>cnested xsi:type="transformation:Transformat</pre>	<pre>nested xsi:type="transformation:Transfor</pre>	

Figure 9.3. Preview Changes For Renamed Model



Note

Renaming a model that is a dependency to another model will automatically change the model imports for those models. If source model CustomerSource is renamed to OldCustomerSource, for instance, the import statement for the view model CustomerAccounts which imports CustomerSource will be changed to reflect the new name.

9.2. Move Model

- To move a model in your workspace:
 - Step 1 Select a model in the Section D.2.1, "Model Explorer View".
 - Step 2 Right-click select the Refactor > Move action.

AllModelsVDB.vdb			
SourceModel.xmi			
🕨 🕞 ViewModel_Ιηρι	New	•	
ViewModel_Out	New Child	•	
ViewModel_Out	Modeling	F.	
ViewModel_WS.	Open		
🔻 🕞 ViewModel.xmi	Open With	Þ.	
🕨 📲 import declara	E Coore		
Fockage Diagn	Сору		
productdata	Paste	Ctrl+V	
productsymbc	Paste Special	Ctrl+V	
MyUDF	💢 Delete		
lin_param∶s	Refactor	•	Rena <u>m</u> e
🖉 out_param :	🔄 Import		Mo <u>v</u> e
🛡 📂 Test_2120	A Export		

Figure 9.4. Refactor Move Action In Model Explorer

• Step 3 - Select a new location (i.e. Project or Folder) and click OK.

8 D Move Resource		
Choose the destination for the resource:		
SourceModel.xmi		
ViewModel.xmi		
ViewModel_Input.xsd		
ViewModel_Output.xsd		
ViewModel_Output_View.xmi		
ViewModel_WS.xmi		
🕨 🗁 lib		
🖕 sources		
Preview > Cancel OK		

Figure 9.5. Move Model Dialog

• Step 4 (Optional) - The next Move Resource File dialog will preview the changes that will be applied as part of mvving your model. Click OK to execute the changes or Cancel.

C A Structure de La Calinde de la Decient				
🗸 🚰 ViewModel.xmi - AllModelsProject				
🥃 🔬 Move resource 'AllModelsProject/SourceModel.xmi' to	o 'sources'			
Synchronize resource AllModelsVDB.vdb				
ViewModel.xmi	📣 🖄 🖓			
Driginal Source	Refactored Source			
<nested td="" transformation:transformat:<="" xsi:type="transformation:Transfo</td><td><nested xsi:type="></nested>				
<inputs href="SourceModel.xmi#mmuuid/a</td><td><pre></td></tr><tr><td></nested></td><td></nested></td></tr><tr><td><nested xsi:type=" td="" transformation:transfo<=""><td><nested 4<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transformat:</td></tr><tr><td><inputs href="><td><pre><inputs href="sources/SourceModel.xmi#mmuu;</pre></td></tr><tr><td></nested></td><td></nested></td></tr><tr><td><nested xsi:type=" td="" transformation:transfo<=""><td><nested b<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transformat:</td></tr><tr><td><inputs href="><td><pre></pre></td></nested></td></inputs></pre></td></nested></td></inputs>	<nested 4<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transformat:</td></tr><tr><td><inputs href="><td><pre><inputs href="sources/SourceModel.xmi#mmuu;</pre></td></tr><tr><td></nested></td><td></nested></td></tr><tr><td><nested xsi:type=" td="" transformation:transfo<=""><td><nested b<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transformat:</td></tr><tr><td><inputs href="><td><pre></pre></td></nested></td></inputs></pre></td></nested>	<pre><inputs href="sources/SourceModel.xmi#mmuu;</pre></td></tr><tr><td></nested></td><td></nested></td></tr><tr><td><nested xsi:type=" td="" transformation:transfo<=""><td><nested b<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transformat:</td></tr><tr><td><inputs href="><td><pre></pre></td></nested></td></inputs></pre>	<nested b<="" sourcemodel.xmi#mmuuid="" td="" xsi:type="transformation:Transformat:</td></tr><tr><td><inputs href="><td><pre></pre></td></nested>	<pre></pre>
<nested td="" transformation:transformat:<="" xsi:type="transformation:Transfo</td><td><nested xsi:type="></nested>				
<inputs href="SourceModel.xmi#mmuuid/4</td><td><pre><inputs href=" pre="" sourcemodel.xmi#mmuu;<="" sources=""></inputs>				
<inputs <="" href="SourceModel.xmi#mmuuid/8b4</td><td><pre><inputs href=" pre="" sourcemodel.xmi#mmuuid="" sources=""></inputs>				
<transformationmappings td="" transformation"<="" xsi:type="transfor</td><td><transformationMappings xsi:type="></transformationmappings>				
chelper xsi:type="transformation:SolTran	transformation:SolTransformation			

Figure 9.6. Preview Changes For Renamed Model

9.3. Save Copy of Model

The **Save As...** action performs a similar function as the **Refactor > Rename** action except the renamed model is a structural copy of the original model.

Note

Each model object maintains it's own unique ID, so copying a model will result in a exact structural copy of your original model but with re-generated unique object IDs. Be aware that locating and copying your models via your local file system may

i

result in runtime errors within Designer. Each model is expected to be unique and duplicate models are not permitted.

- To create a duplicate model using Save As...:
 - Step 1 Open the model you wish to copy in a **Model Editor** by double-clicking the model in Section D.2.1, "Model Explorer View" or right-click select **Open** action.
 - Step 2 Select the editor tab for the model you opened.

	î <u>No</u>	orthwind.xmi 🛛
	ø	< <base table=""/> >
-	£	🔣 Territories

Figure 9.7. Select Editor Tab

• Step 3 - Select File > Save As... action to open the Save Model As dialog.

۲	Save Model As	×	
s	elect a destination for the copy		
	🔻 🟓 AllCustomerAccounts		
	🕨 📬 Northwind.xmi		
	🔁 Sources		
1	👂 🛃 BooksProject		
1	👂 🚰 PartsProject		
1	WSDLImportProject		
Ent	er the new model name		
No	orthwind_West	٦	
Re	eady to copy		
(Cancel OK	1	
	č	,	

Figure 9.8. Save Model As Dialog

• Step 4 - Enter a unique model name in the new model name text field and click OK.

• Step 5 - If dependent models are detected, the Save Model As - Import References dialog is presented to give you the opportunity to change any of the dependent models imports to reference the new model or not.

🥮 Save Model As - Import References 🛛 🕅		
The following model files use or reference the model to be copied.		
Select any that should be modified to use the new copy.		
✓ n AllCustomerAccounts/CustomerViews.xmi		
Select All Deselect All		
Cancel OK		

Figure 9.9. Save Model As Dialog

9.4. Clone Project

Because each instance of of a model contains a unique ID and each object in each model contains a unique ID, copying a project is a delicate task. For this reason, the Clone Project action was created to manage the creation of exact structural copies of all models in the source project.

- The following lists specific rules and limitations for this action.
 - This action clones a complete model project containing any number of model (XMI or XSD) files organized in a user-defined directory structure.
 - All object references (UUIDs) within the original project will be replaced with new unique references.
 - Any model dependencies or internal object references are refactored to reflect the dependencies within the cloned project.
 - Any model references to models in projects external to the original project will NOT be replaced.
 - Only XMI and XSD files are cloned. All other file types in your project will NOT be processed nor copied into your newly cloned project including VDBs

- If one or more editors that require "save" are open, the user will be asked to save them before continuing with the cloning process.
- To clone a model project ::
 - Step 1 Select an existing model project in the Section D.2.1, "Model Explorer View".
 - Step 2 Right-click, then select Model Project > Clone in the context menu. Otherwise you can select the Project > Clone Project action, located in Teiid Designer's main menu bar.

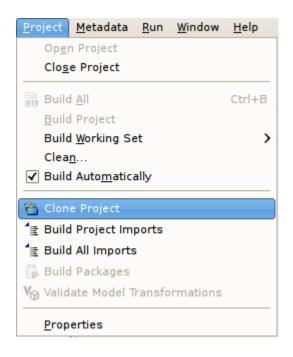


Figure 9.10. Clone Project In Project Menu

• Step 3 - On the Clone Project wizard page, provide a name for your new project.

9	Clone Model Project		
Clone Model Project Specify cloned model project name.			
<u>P</u> roject name:	ValuedCustomerAccounts		
✔ Use <u>d</u> efault	location		
Location: /hon	ne/blafond/Test Designer Folder/T	Browse	
Choo	se file system: default 🗘		
?	Cancel	<u>Finish</u>	

Figure 9.11. Clone Project In Project Menu

- **Step 4** (Optional) If you wish to create your cloned project in a location other than your default workspace location, uncheck the **Use default location** check-box and specify (type in or browse to) a new directory location on your local file system.
- Step 5 Click Finish to generate your new project.

Managing VDBs

As stated in the introduction, the critical artifact for Teiid Designer the VDB, or Virtual DataBase. This section describes the details of how to create, edit, deploy and test your VDBs.

10.1. Creating a VDB

To create an empty VDB launch Eclipse's **New** wizard, open the **Teiid Designer** category folder and select **Teiid VDB**. You can also select one or more models in a model project, right-click and select **New > Teiid VDB** action.>

Launching this wizard will open the New VDB dialog. If you launched with one or more models selected the dialog will contain the pre-selected models for inclusion in the new VDB.

	New VDB	× × × × × × × × × × × × × × × × × × ×
New VDB		
Press the "Fi	nish" button to finish.	
In Folder:		
in Folder.	Books Project	
VDB Name:	Books	
Descriptio	on	
Join of tw	o books sources	
Selected	Models	
Add	🕞 BooksView.xmi	
	📬 Books_SourceA.xmi	
Remove	宿 Books_SourceB.xmi	
L		
~		
?	Can	cel <u>F</u> inish

Figure 10.1. New VDB Dialog



Note

A VDB is scoped to be aware of models and files within the same model project as the VDB. You will not be allowed to add models to a VDB that exist in a different project.

10.2. Editing a VDB

To Edit an existing VDB, select the VDB in the explorer and right-click select Open action or simply double-click the VDB. The VDB will be opened in a VDB Editor. (See the Section D.3.2, "VDB Editor" section)

10.3. Test a VDB

For details on how to test your VDB, see Section 11.3, "Testing With Your VDB" section

10.4. Multi-source Binding Support

Teiid Designer now supports the Teiid feature of defining relational source models and binding them to multiple data sources.

Multi-source models can be used to quickly access data in multiple sources with homogeneous metadata. When you have multiple instances using identical schema, Teiid can help you gather data across all the instances, using "multi-source" models. In this scenario, instead of creating/ importing a model for every data source, one source model is defined to represents the schema and is configured with multiple data "sources" underneath it. During runtime when a query issued against this model, the query engine analyzes the information and gathers the required data from all sources configured and gathers the results and provides in a single result. Since all sources utilize the same physical metadata, this feature is most appropriate for accessing the same source type with multiple instances

The VDB editor's **Models** tab now contains a simplified model table on the left and a new tabbed panel on the right containing **Model Details** and **Source Binding Definition** tabs. Click the **Multi-source** check box if you wish to add additional source bindings. Note that each binding must be defined with a unique **Source Name** as well as unique **JNDI Name** representing a deployed data source you your server.

📋 TestUDF.vdb 🔀	
Location: /TestUDF Version 1 Synchronize All Show Import VDBs	Last validated : Tue Oct 15 12:50:15 CDT 2013 Teiid runtime version :
Models UDF Jars Other Files Description	
Model Image: Comparison of the second se	
	Add Source Name Translator Name JNDI Delete EMPLOYEE_DATA_SOURCE ws EMPLOY
Content Advanced	

Figure 10.2. VDB Editor's Source Binding Definition

10.5. UDF support

In Teiid Designer you can create, manage and use User Defined Functions (UDFs). These functions allow you to perform simple or complex java operations on your data during runtime. This is accomplished by deploying your custom UDF jars on your server and creating a scalar function representation of your function method to use in your view transformation. In the VDB Editor, you have the option of including your UDF jars as part of the VDB artifact. If included in the VDB, the jars will automatically be deployed to the server for you when the VDB is deployed.

The figure below illustrates a sample project setup which includes a UDF jar in a lib folder under a project. When a model defining a UDF is added to a VDB, each function is interrogated and it's referenced UDF jar (if available) is added to the VDB as well as shown in the UDF Jars tab in the editor.

🔛 Model Explorer 🔀 📴 Outline 🗖 🗖	TestUDF.vdb 🔀
🚰 🔻 🖳 🞼 🤹 🖓 🖓 👘	Location: /TestUDF
 ▼ → TestUDF ▶ → datasource ▼ → lib 	Version 1 Synchronize All Show Import VDBs
ame_builder.jar	Models UDF Jars Other Files Description
 ▼ ⊱ src ♪ NameBuilder.java ☆ .project ▶ 👔 EMPLOYEE_DATA_SOURCE.xmi ▼ 😘 EMPLOYEE_DATA_VIEWS.xmi ▶ * import declarations (2) ♀ Package Diagram ▶ EMPLOYEE_DATA 	File Location in VDB Description Image: mame_builder.jar /lib
Default Server Change Name: A SADE 1: Toild8 75(1)(4)	
Name: AP6.1+Teiid8.7FINAL Teiid Version: A 8.7.0	Content Advanced

Figure 10.3. UDF Jar In Project and VDB

10.6. Reusing VDBs

Teiid 8.1 introduced the ability to treat your deployed VDB as just another database where the database category is your VDB name and each visible model in your VDB is treated as a schema. This is accomplished via a new <import-vdb> element in the vdb.xml definition. (see *Teiid VDB Reuse section* [https://docs.jboss.org/author/display/TEIID/VDB+Reuse]). By allowing VDB's to referenced other VDBs, users can create reusable database components and reduce the amount of modeling required to create complex transformations.

The sample vdb.xml file below highlights the <import-vdb> element and the corresponding import-vdb-reference within the view model's <model> element.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<vdb version="1" name="PartssupplierViewsVDB">
        <property value="false" name="preview"/>
        <import-vdb import-data-policies="false" version="1"
        name="PartssupplierSourcesVDB"/>
        <model visible="true" type="VIRTUAL" name="PartsViewModel" path="/
PartssupplierProject/PartsViewModel.xmi">
        <property value="type="VIRTUAL" name="PartsViewModel" path="/
PartssupplierProject/PartsViewModel.xmi">
        <property value="1623826484" name="checksum"/>
        <property value="Relational" name="modelClass"/>
        <property value="false" name="builtIn"/>
        <property value="655076658.INDEX" name="indexName"/>
```

```
<property value="PartssupplierSourcesVDB" name="import-vdb-
reference"/>
</model>
</vdb>
```

Teiid Designer exposes this capability by allowing users to import metadata from deployed VDBs via the JDBC Import option. Through this import, relational VDB source models are created which structurally represent the Catalog (VDB), Schema (Model) and Tables in Virtual DataBase.

When dealing with the these VDB source models there are some limitations or rules, namely:

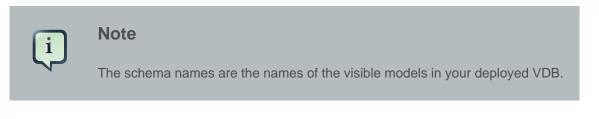
- VDB source models are read-only
- VDB source model name is determined by the deployed model name (schema) from the VDB it was imported from
- Model names have to be unique within a model project
 - VDB source models have to be imported/created in a project different than the project used to create and deploy the Reuse VDB
- The JDBC Import Wizard will restrict your options to comply with these rules

To create a VDB source model:

- Step 1 Deploy your VDB
- Step 2 Launch the JDBC Import Wizard via the "Import > Teiid Designer > JDBC Database >> Source Model" action
- Step 3 On the first page of the wizard create/select a valid connection profile for your deployed VDB.
 - The wizard will detect that the connection profile is a Teiid VDB connection and a section will be displayed on the wizard page titled **Teiid VDB Source Options**
 - If **Import as VDB source model** is NOT checked, then the wizard will continue importing as a normal JDBC import

🔳 Import Data	ibase via JDBC
	ource configuration t >" button to continue or the "Finish" button to finish.
Connection P	rofile
Barry_Test_V	
JDBC Metadat	a Processor
JDBC (defau	lt)
Properties	
Driver:	Teiid 8.x
URL:	jdbc:teiid:BarryParts_VDB@mm://localhost:31000
User Name:	user
Password:	****
Teiid VDB Sou	
	SVDB source model
A JDBC conne	ection to a deployed VDB allows creating a read-only VDB source model that you can use in your view transforma
VDBs contain source model	ing these view models will reference an import to the original deployed VDB and will not contain the metadata f I.
If checked, or	nly one schema in your VDB can be selected for import at a time.
?	< Back Next >

• Step 4 - On the 3rd page, titled Select Database Objects, select a single schema to use to create as VDB source model.



×	Import Database via JDBC	_			
s	Select Database Objects				
🛞 Can only select a single schema for VDB source model import.					
L					
	Rarry Test VDB				
	Barry_Test_VDB				
	AllParts				
	PartsSupplierA				
	PartsSupplierB				

- **Step 5** The final page shows the name of the resulting VDB source model and the name is NOT editable.
 - All other options are disabled
 - The target **Into Folder** must NOT contain a model with the same name or the **Finish** button will be disabled

関 Import Databa	ase via JDBC	
Specify Import O	ptions	
Press the "Next :	>" button to continue or the "Finish" button to finish.	
Relational Mode	el Definition	
Model Name:	AllParts	
Into Folder:	TeiidProject_A	
Make targe	t a view model	
🗌 Update (if	existing model selected)	
Include Cata	log For Fully Qualified Names	
Model Object N	ames (Tables, Procedures, Columns, etc)	
🗌 Use Fully Q	ualified Names (Example: partssupplier.dbo.PARTS)	
Change Ca	se For All Characters	
Case Opti	ions	
 Make 	All Upper Case (Example: Suppliers > SUPPLIERS)	
O Make	All Lower Case (Example: SUPPLIERS > suppliers)	
0		
		<pre>< Back Next > ()</pre>

You can use your VDB source model like any other source model in your project. VDB source model tables can be used in your transformation queries and the view models will contain model imports to your VDB source models. However, when your view model is added to a VDB, any referenced VDB source models do NOT get added to your VDB. Instead, an <import-vdb> element (described above) reference is added in it's place.

If VDB imports exist for a VDB, the **Show Import VDBs** button will be enabled and allow viewing the names of the imported VDBs as shown below. Note that the dialog now supports editing of the version number for each referenced VDB.

Location: /TestVdbImport	Last validated : N	1on Mar 28 10:47:40 CDT	2016 Teiid runtime
Version 1 Synchronize All Show Imp	port VDBs		
	📉 🙁 🗆 Import VDBs		
Models Schemas UDF Jars Other Files Descrip	VDB Names		
Model 02	Name	Version	Import Data Policies
🙀 Employee_Top_Views.xmi	EmployeesVDB	2	false
			Cancel
			Cancel O

10.7. Security and Data Access

You have some options on defining your data access security for your VDB via the VDB Editor (See the Section D.3.2, "VDB Editor" section).

The first level is provided by the model visibility check-box in the Models section (Spyglass column). If unchecked, that model and it's contents will not be returned by the Teiid runtime with the standard JDBC metadata.

The next level of security is provided defining permissions for your data roles which can be managed via the lower panel in the VDB Editor. For a unique data role, each model and most objects within that model can have specific values of data access including:

- Security (Row-based condition and column masking)
- Create
- Read

- Update
- Delete
- Execute
- Alter

The image below is an example of the **Permissions** defined for a data role.

😣 💷 Edit VDB Data Role						
Edit VDB Data Role						
Select Finish to save data role						
Name Internal Only						
Permissions Options Mapped Role Names Description						
Set object level permissions (i.e. tables, procedures, columns, etc.)					
Model Row Filter Column Masking Allowed Languages						
Model	Security	Create	Read	Update	Delete	Execu
▼ 🙀 CustomerAccounts.xmi			✓			
	0		1			
ACCTID: bigdecimal	0		✓			
CUSTID: string(12)	Ō		✓			
TYPE : string(10)	Õ		✓			
STATUS: string(10)	Ō		✓			
DATEOPENED : dateTime	Õ		✓			
DATECLOSED : dateTime	0		✓			
ACCOUNTHOLDINGS	0		✓			
CUSTOMER	0		✓			
	•					
?						Cancel
0						

Figure 10.4. Permissions Section

Teiid provides 2 security concepts for object level permissions: (see *Permissions* [https:// docs.jboss.org/author/display/teiid88final/Permissions])

 Row-Based Security is a permission against a fully qualified table/view/procedure may also specify a condition.

- Unlike the allow CRUD actions defined above, a condition is always applied not just at the user query level. The condition can be any valid SQL referencing the columns of the table/ view/procedure. Procedure result set columns may be referenced as proc.col. The condition will act as a row-based filter and as a checked constraint for insert/update operations.
- An example of a condition might be: column1=user()
- Column Masking is "a permission against a fully qualified table/view/procedure column may also specify a mask and optionally a condition."
 - When the query is submitted the roles are consulted and the relevant mask/condition information are combined to form a searched case expression to mask the values that would have been returned by the access. Unlike the CRUD allow actions defined above, the resulting masking effect is always applied - not just at the user query level. The condition and expression can be any valid SQL referencing the columns of the table/view/procedure. Procedure result set columns may be referenced as proc.col.
 - An example of a mask might be: CASE WHEN column1=user() THEN column1 END

You can define Row and Column Based security in Designer's Data Row Wizard. By doubleclicking a target table, view, procedure or column in the Models table, the appropriate editor dialog will be displayed. Note those objects that already have existing security defined are highlighted in blue.

If a table, view or procedure is double-clicked, the Row Filter Definition dialog is displayed. Enter a valid SQL condition and specify whether or not this filter should be treated as a constraint or not.

8 🗆 Row Filter Definition	
Row Filter Definition	
Click OK To Finish	
Target ProductsOracle.PRODUCTS.PRODUCTDATA	
Constraint 🞯	
Condition	
INSTR_ID>'123'	
?	Cancel OK

Figure 10.5. Row Filter Definition Dialog

If a column is double-clicked, the Row Filter Definition dialog is displayed. Enter valid column masking SQL expression, an optional order (see Teiid documentation) and an option condition expression.

🙁 🗆 Colu	mn Masking Definition
Column Maskir	ng Definition
Click OK To Fini	sh
Target Column	ProductsOracle.PRODUCTS.PRODUCTSYMBOLS.INSTR_ID
Condition	
column1() = '	xx'
Order	4
Column Mask	ing
хуг	
?	Cancel OK

Figure 10.6. Column Masking Definition Dialog

You can also edit these these values via the **Add**, **Edit and Remove** buttons on the respective Row Filter and Column Masking tabs

Permissions Options Mapped Role Na	mes Descr	iption		
Set object level permissions (i.e. tables	, procedure	s, columns, etc)		
Model Row Filter Column Masking Allowed Languages				
Filter out sensitive rows of results fi	rom table/vi	ew based condition provided for this data-role		
Name	Constraint	Condition		
CustomerAccounts. ACCOUNT	true	column1=user()		
Add Edit Remove				

Figure 10.7. Row Filter Tab

Permissions Options Mapped Role Names	Descr	iption
Set object level permissions (i.e. tables, pro	cedure	s, columns, etc)
Model Row Filter Column Masking Allo	wed Lar	iguages
Name	Order	Mask
CustomerAccounts. ACCOUNT. ACCTID	1	CASE WHEN ACCTID=user() THEN ACCTID END
·		
Add Edit Remove		

Figure 10.8. Column Masking Tab

10.8. Working with Dynamic VDBs

This section desribes the Designer features which allow generating a VDB archive with XMI models from existing dynamic VDBs (*-vdb.xml) files and generating dynamic VDBs from existing VDB archives.

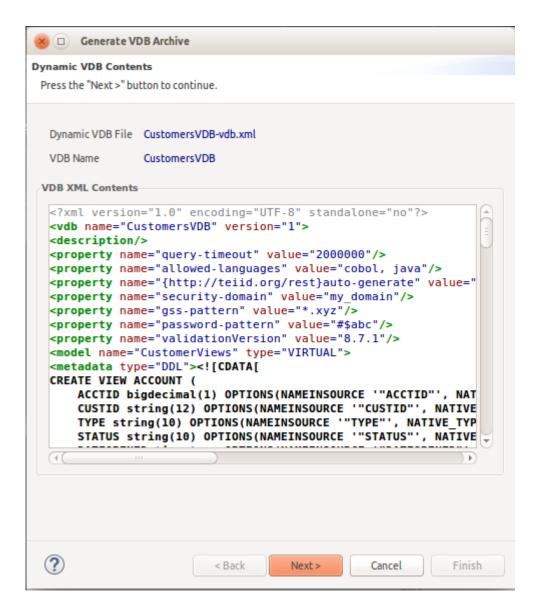
10.8.1. Generate VDB Archive

To generate a VDB archive and models from a dynamic vdb (*-vdb.xml) file in your workspace:

- Step 1 Import (or drag and drop) a dynamic vdb file into a model project or folder
- Step 2 Select the dynamic vdb file in the Model Explorer, then right-click and select the Modeling > Generate Archive VDB and Models action.

•
P Peploy
Generate VDB Archive and Models
•

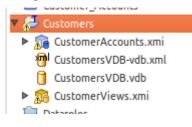
 Step 3 - The first page of thw wizard will show the Dynamic VDB Contents Verify the contents and click Next >



• Step 4 - On the second page, specify details and options for yor VDB archive including actual VDB name, VDB archive file name and target location (project or folder). Click Generate to create the VDB archive and associated models in your target location. Then click Finish

😕 🗊 Generate VDB A	rchive	
Specify VDB archive out	put	
Click Finish to save VDB ar		
VDB Details		
Original VDB Name	CustomersVDB	
Location	/Customers	Change
Version	1	
Archive VDB Name	CustomersVDB	
VDB Archive File Name	CustomersVDB.vdb	
	Set parsed DDL a	s the generated tables' descriptions
Source Models		View Models
CustomerAccounts		CustomerViews
🧭 Overwrite	existing files	Generate
?	< Back	Next > Cancel Finish

The generated VDB and XMI models will now be visible in the **Model Explorer** view.



10.8.2. Generate Dynamic VDB

To generate a dynamic VDB from a VDB archive (*.vdb) in your workspace:

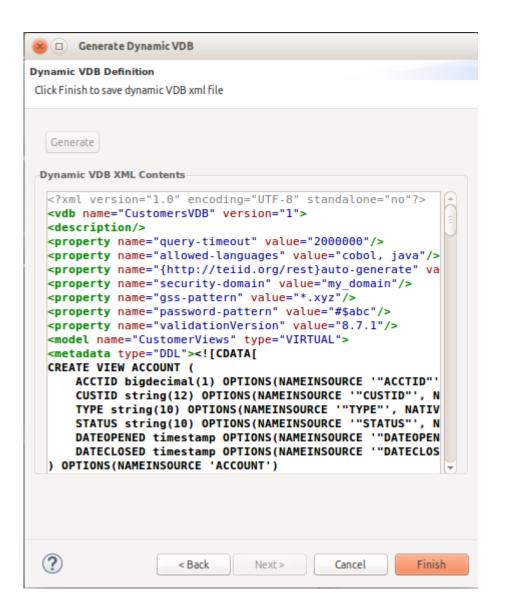
• Step 1 - Select a VDB archive file in the Model Explorer, then right-click and select the Modeling > Generate Dynamic VDB action.

🔻 🛃 Custo	mers			
🗐 💭 Cus	tomersVDB.vdb			
📋 D	New	Þ		
D D	Modeling	Þ	1	Create VDB Data Source
▶ 🚰 D ▶ 🚰 D	Open			Deploy
▶ 🛃 E	Open With	Þ	0,	Execute VDB
🗂 F 📄	Сору		xmi	Generate Dynamic VDB
		a. 1. 1.	3	Generate REST War

Step 2 - The first page of thw wizard will show the selected VDB archive file information. You can also set the VDB name adn version as well as specify the target location and actual file name (*-vdb.xml). Click Next >

Senerate Dynamic VDB
Archive VDB Information Press the "Next >" button to continue.
Selected VDB Archive VDB /Customers/CustomersVDB.vdb Name CustomersVDB Version 1 Dynamic VDB Definition Dynamic VDB Name CustomersVDB Version 1
Dynamic VDB Destination
Location /Customers Change
Dynamic VDB File Name CustomersVDB-vdb.xml
Options Exclude source DDL metadata Suppress attributes with default values Overwrite existing files
? < Back Next > Cancel Finish

 Step 3 - On the second page, click the Generate button. The XML contents will be displayed in the viewer which represents the same basic metadata from the origional VDB. Click Generate to create the VDB archive and associated models in your target location. Then click Finish to save the XML string to the file specified in Step 2.



Testing Your Models

As described briefly in Section 1.3.8. "Testing Your Models", can you test your models in Teiid Designer by using the Preview Data action S.

or test your models via your deployable VDB. These two options will be described in detail in this chapter as well as managing your required connection profiles.

11.1. Manage Connection Profiles

Teiid Designer utilizes the Eclipse Data Tools Platform (DTP) Connection Profile framework for connection management. Connection Profiles provide a mechanism to connect to relational and non-relational sources to access metadata for constructing metadata source models. Teiid Designer also provides a custom Teiid connection profile template designed as a JDBC source to a deployed VDB.

By selecting various Teiid Designer Import options, any applicable Connection Profiles you have defined in your Database Development perspective will be available to use as your import source. From these import wizards you can also create new connection profiles or edit existing connection profiles without leaving the wizard.

The Teiid server contents in the Section 3.1, "Setting up a Server" provides access to running Teiid instances and shows data source and VDB artifacts deployed there. The "Create Data Source" action available on this view utilizes the available and applicable connection profiles.

11.1.1. Set Connection Profile for Source Model

Teiid Designer integrates Data Tools Connection Profiles by persisting pertinent connection information in each source model. This can occur through Importing process or through the **Modeling > Set Connection Profile** action.

11.1.2. View Connection Profile for Source Model

In addition to setting the connection profile on a source model you can also view a source model's connection profile information via the **Modeling > View Connection Info** action which displays the detailed properties of the connection.

roperties	
name	value
connectionProfileDescription	
connectionProfileCategory	org.eclipse.datatools.connectivity.db.category
connectionProfileProviderId	org.eclipse.datatools.enablement.oracle.connectionProfile
connection-url	jdbc:oracle:thin:@englxdbs11.mm.atl2.redhat.com:1521:ORCL
connectionProfileName	PartsOracle11
connectionProfileInstanceID	org.eclipse.datatools.enablement.oracle.connectionProfile
driver-class	oracle.jdbc.OracleDriver
translator name	oracle
user-name	partssupplier

Figure 11.1. Connection Profile Information Dialog

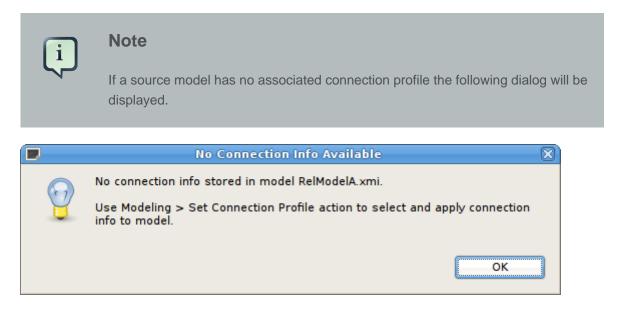


Figure 11.2. No Connection Info Dialog

11.1.3. Remove Connection Profile from Source Model

As a user, you may not want this connection information (i.e. URL, username, etc...) shared through your VDB. Designer provides a means to remove this connection information via a **Modeling > Remove Connection Info** action. When adding a source model without connection information will require the user to supply or select the correct translator type.

11.2. Previewing Data For a Model

Designing and working with data is often much easier when you can see the information you're working with. The Designer's Preview Data feature makes this possible and allows you to instantly preview the information described by any object, whether it's a physical table or a virtual view. In other words, you can test the views with actual data by simply selecting the table, view, procedure or XML document. Previewing information is a fast and easy way to sample the data. Of course, to run more complicated queries like what your application likely uses, simply execute the VDB Via DTP and type in any query or SQL statement.

After creating your models, you can test them by using the **Preview Data** action

By selecting a desired table object and executing the action, the results of a simple query will be displayed in the Section 11.2.5, "Sample SQL Results for Preview Data" view. This action is accessible throughout the Teiid Designer in various view toolbars and context menus.

There are two requirements for previewing your data:

1. The selected object must be one of several previewable model object types

2. All source models within the model dependency tree must reference a connection profile. Model objects that can be previewed include: relational tables and views (including tables involving access patterns), relational procedures, Web service operations and XML document staging tables.



Note

Any virtual table, view or procedure is previewable as long as all "physical" source models reference sufficient connection info. (See ??? view)

After selecting the Preview Data action, Designer will insure that all source models have corresponding deployed data sources .

If the model selected for preview is a source model and there is insufficient connection info for that model, the following dialog will be displayed and the action terminated.



Figure 11.3. Preview Not Available

If any of the source models in the corresponding project require a password that can't be retrieved from an existing connection profile, the user will be queried for each missing password

E	Missing Password Required 🛛 🕺
	a requires password for model >> Products.xmi connection profile >> AllProducts
Password:	****
	Cancel OK

Figure 11.4. Missing Password

In order to preview, the Teiid Runtime requires deployed metadata representing your source and view metadata. If all required data sources are deployed the Preview Data dialog will be displayed containing 2 tabs to view the simple SQL Query to be executed as well as the VDB Xml Content that will be temporarily deployed in order to execute the query. Clicking OK will execute the SQL query.

😣 🗆 Custo	om Preview Data	
Preview Data		
Customize you	r SQL query and view your preview VDB XML content	
Preview SQL	Preview VDB XML Content	
Options		
Cener	rate Execution Plan	
SQL Query	у	
?		C

Figure 11.5. Preview Data: SQL Query Tab

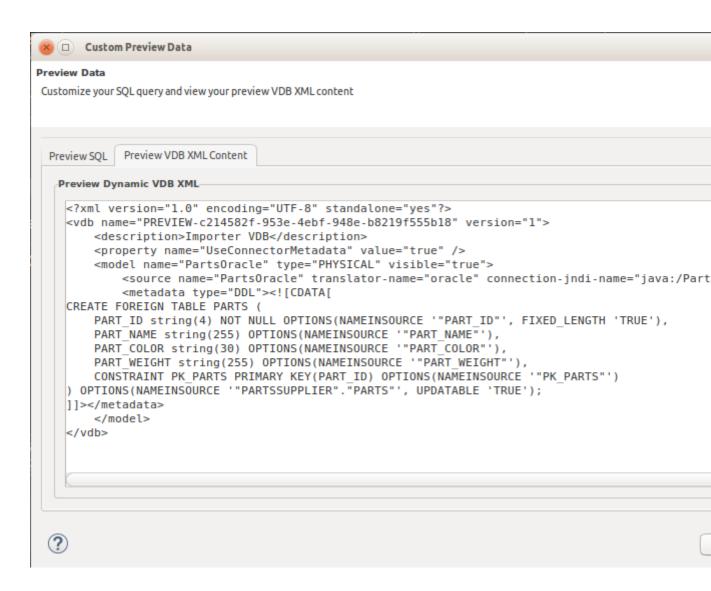


Figure 11.6. Preview Data: VDB XML Content Tabb

Testing Your Transformations

When editing transformation SQL in the Transformation Editor, а special SQL Results in the editor tool-bar data action is provided 5

You can change your transformation SQL, re-validate and preview your the data for your modified SQL.

The following sections provide steps for previewing your data.



Note

All steps assume that all source models referenced by your models, either directly or through dependencies, are bound to connector bindings.

11.2.1. Preview Relational Table or View

- To preview a relational table, relational view or staging table:
 - Step 1 Select a relational table or view in the Section D.2.1, "Model Explorer View" or diagram. The table or view can be in a view model as well as a source model. Staging tables are not visible in the Section D.2.1, "Model Explorer View", so you need to open the mapping diagram and select it there.
 - Step 2 Right-click select the Preview Data action

You can also select the same action in the tool-bar of either the Section D.2.1, "Model Explorer View" or diagram.

• Step 3 - Your query results will be displayed in the Section 11.2.5, "Sample SQL Results for Preview Data" view. The view will automatically open or get focus if not visible in your perspective.

11.2.2. Preview Relational Table With Access Pattern

- To preview a relational table or view with access pattern:
 - Step 1 Select a relational table or view in the Section D.2.1, "Model Explorer View" or diagram that contains an access pattern. The table or view can be in a view model as well as a source model.
 - Step 2 Right-click select the Preview Data action

You can also select the same action in the tool-bar of either the Section D.2.1, "Model Explorer View" or diagram.

• Step 3 - A column input dialog is presented. Select each access pattern and enter a value for each required column.

I



Note

If data entered does not match the column datatype (String, integer, etc...), an error message will be displayed in the dialog header. When all required values are entered, click the **OK** button to execute the query.

	1500	Preview	Data
--	------	---------	------

To preview data from this table, select an access pattern and provide a va

All access pattern column values are set. Select OK to continue.

Access Patterns	Required Columns	
A NameAndID	PUBLISHER_ID : long	20022
	NAME : string(255)	Mifflin
J		

Figure 11.7. Access Pattern Column Input Dialog

• Step 4 - Your query results will be displayed in the Section 11.2.5, "Sample SQL Results for Preview Data" view. The view will automatically open or get focus if not visible in your perspective.

11.2.3. Preview Relational Procedure

- To preview a relational procedure:
 - **Step 1** Select a relational procedure in the Section D.2.1, "Model Explorer View" or diagram. The procedure can be in a view model as well as a source model.
 - Step 2 Right-click select the Preview Data action

You can also select the same action in the tool-bar of either the Section D.2.1, "Model Explorer View" or diagram.

• Step 3 - An input parameter input dialog is presented. Enter a valid value for each parameter.

	0	lick the OK button	0		requirea	
🐢 Preview Dat	ia					
To execute th All parameter val	장애 이 가 가 가 가 많이 했다.			equired.		
- Input Param colorIn : string	eters blue					
					ОК	Cancel

If data entered does not match the parameter datatype (String, integer, etc...),

Figure 11.8. Procedure Parameter Input Dialog

• Step 4 - Your query results will be displayed in the Section 11.2.5, "Sample SQL Results for Preview Data" view. The view will automatically open or get focus if not visible in your perspective.

11.2.4. Preview Web Service Operation

• To preview a Web service operation:

Note

- Step 1 : Select a Web service operation in the Section D.2.1, "Model Explorer View" or diagram. The operation can be in a view model as well as a source model.
- Step 2 : Right-click select the Preview Data action

You can also select the same action in the tool-bar of either the Section D.2.1, "Model Explorer View" or diagram.

• Step 3 : An input parameter input dialog is presented. Enter a valid value for each parameter.

i	Note
	If data entered does not match the parameter datatype (String, integer, etc), an error message will be displayed in the dialog header. When all required values are entered, click the OK button to execute the query.
	riew Data
	cute this procedure, input values are required. Immeter values are set. Select OK to continue.
<i></i>	
Input	: Parameters
AUTHO	OR ID ; xs;long 10022
AUTHO	OR_ID : xs:long 10022
AUTHO	OR_ID : xs:long 10022

Figure 11.9. Procedure Parameter Input Dialog

• Step 4 - Your query results will be displayed in the Section 11.2.5, "Sample SQL Results for Preview Data" view. The view will automatically open or get focus if not visible in your perspective.

11.2.5. Sample SQL Results for Preview Data

Preview Data results are displayed in the Eclipse Datatools SQL Results view as shown below.



Note

There are a number of display preference and filter options for this view via toolbar buttons and the dropdown menu.

🖹 Problems 🔇 Message Log 🔲 SQL Results 🕱				\$
Type query expression here		INSTR_ID	NAME	T
Status A Operation Date Connection Profile	1	PRD01088	Novell Incorporated	St
✓ Succee select * from Aug 28, 2010 Transient.org.teiid.d	2	PRD01089	Amazon.com, Incorporated	St
✓ Succee select * from Aug 28, 2010 Transient.org.teiid.d	3	PRD01090	Juniper Networks, Incorporated	St
	4	PRD01091	Red Hat, Incorporated	St
	5	PRD01092	Boston Scientific Corporation	St
	6	PRD01093	Inex Pharmaceuticals, Incorpora	St
	7	PRD01094	Pfizer, Inc.	St
	8	PRD01095	Cytovax Biotechnologies Incorpo	St
	9	PRD01096	Commonwealth Biotechnologies	St
	10	PRD01097	British Biotechnology plc	St
	11	PRD01220	Unisys Corporation	St
	12	PRD01099	Honeywell International	St
	13	PRD01100	Hilton Hotels Corporation	St
	14	PRD01101	Hilton Hotels Corporation	С
	15	PRD01102	Mercury Interactive Corporation	S
	16	PRD01103	Fidelity Freedom Income Fund	м
		00001104	Eddler Freedom 2000 Fried	
	<			

Figure 11.10. SQL Results View

11.2.6. Execution Plans

When Preview Data is executed, the **Teiid Execution Plan** is also displayed as shown below. The Execution Plan may also be obtained by rt-clicking on a previewable object, then selecting **Modeling > Show Execution Plan** in the context menu.

SQL Results 🕼 Teiid Execution Plan 🕱
Preview Data for : myTbl
SQL
select * from "vModel"."myTbl"
Select from Wilder myrsi
Execution Plan
Text Plan Tree
<2ml vertion="1.0" encoding="UTE_0" standalene="vec"2>
xml version="1.0" encoding="UTF-8" standalone="yes"? <node name="AccessNode"></node>
<pre><pre>chode hame= AccessNode > </pre> <pre><pre>conde hame= AccessNode ></pre></pre></pre>
<value>INTKEY (integer)</value>
<value>STRINGKEY (string)</value>
<value>INTNUM (integer)</value>
<value>STRINGNUM (string)</value>
<value>FLOATNUM (double)</value>
<value>LONGNUM (long)</value>
<value>DOUBLENUM (double)</value>

Figure 11.11. Teiid Execution Plan View

11.3. Testing With Your VDB

In Teiid Designer you can execute a VDB to test/query actual data.

The requirements for VDB execution are:

- A deployed VDB backed by valid deployed Data Sources
- An instance of a Teiid Connection Profile configured for the deployed VDB

Teiid Designer simplifies this process via Deploy VDB and Execute VDB actions. Deploy VDB does just that, deploy a selected VDB to a running Teiid instance. Execute VDB performs the VDB deployment, creates a Teiid Connection Profile, opens the Database Development perspective and creates a connection to your VDB.

11.3.1. Creating Data Sources

The mechanism by which VDBs are able to query actual data sources is the Data Source. These are deployed configurations backed by database or source connection jars. Each source model referenced within a VDB requiries a JNDI name representing a deployed Data Source.

When creating VDBs you do not need to have deployed data sources on your Teiid server, but if you wish to test your VDB, the data sources need to be present.

Teiid Designer provides a **Create Data Source** action so you can create compatible data sources for your source model. If you wish to create a data source for a specific model you can select that source model in your workspace and select the **Modeling > Create Data Source** action. This will extract the connection profile data from your source model and create a corresponding data source on your default Teiid server.

You can also create data sources from the Servers view. Select a Teiid server instance in the Servers view and right-click select the **Create Data Source** action. This will launch the Create Data Source Dialog shown below.

9		Cre	ate Data Sour	ce	×
Create Data	a Soui	ce			
All inputs are v	valid. S	elect Finish to d	create data sou	rce.	
Teiid Server:	mm://	ocalhost: 31443	3		
Name	Books	DB2			
Connection	n Sou	rce			
Use Mode	el Conn	ection Info			
Model					0
✔ Use Conn	nection	Profile Info			
Connection P	Profile	BooksDB2		0	New Edit
Connection	n Proj	perties			
Name			Value		
password			*otototototot		
user-name			books		
connection-u	url		jdbc:db2://db	000255.org.my	dbs.com :50000
driver-class			com.ibm.db2.j	cc.DB2Driver	
?				Cancel	<u>Finish</u>

Figure 11.12. Create Data Source Dialog

You can either select and existing Connection Profile from the drop-down list (Use Connection Profile Info option) or check the Use Model Info option and select an existing source model containing connection info.

After creating your new data source it should now be shown in the **Data Sources** folder of the corresponding Teiid server.

11.3.2. Execute VDB from Model Explorer

- If you have a Teiid instance defined and connected in your Servers view you can:
 - Step 1 : Right-click a VDB in your Model Explorer select Modeling > Execute VDB action. This action will insure your selected VDB is deployed to Teiid, create a Teiid Connection Profile specific for that VDB, open the Database Development perspective and create a connection to your VDB.

▽ 🛃 PartsProject			
🕨 🗁 data			
👂 🗁 readme			
📋 Parts.v	Ne <u>w</u>	>	
👂 宿 PartsS	Modeling	\rightarrow	n Deploy
👂 📬 PartsS(🗌	_		😭 Execute VDB
👂 💼 PartsVi	<u>O</u> pen		😭 Generate JBossWS-CXF War
👂 📬 Sample	Open Wit <u>h</u>	>	AL CONTRACT JOURNAL OF MAIL

Figure 11.13. Execute VDB Action

 Step 2: Select your new Teild connection profile and right-click select Open SQL Scrapbook, enter your designer SQL (i.e. SELECT * FROM TableXXXX), select all text and right-click select Execute Selected Text

Connec		book 0 🖇	3 [Parts.	vdb									
	- Teiid	-				•	<u>N</u> ame:	Parts -	Teiid Cor	necti 🗘	Data <u>b</u> as	e: 📃	٥	Sta
SEL	FCT	* EDON	Dante	sSourc	eA.Par	ts								
6		- I RUN	rai L											
	LUI	TRUP	Fart	Jooure										
		* TROP	Fart	Jooure										
	Lei	- TROP	Fart											
	Lei	* TROP	Fai L											

Figure 11.14. SQL Scrapbook Editor

• Step 3 : Results of query should be displayed in the SQL Results view on the Result1 tab.

Туре	query exp	ression here					
Stat	us C	Operation		Date	Connection Profile		
v	Succee S	SELECT * FROM Parts	SourceA.Parts	Oct 15, 2010	Parts - Teiid Connectio	n	
							_
<							
	PART_ID		PART_NAME		PART_COLOR		PA
1	PART_ID P300		PART_NAME		PART_COLOR Red		PA
1 2	-		-				
	P300		Nut		Red		12
2	P300 P301		Nut Bolt		Red Green		1

Figure 11.15. SQL Results View

11.3.3. Deploy VDB from Model Explorer

You can also deploy your VDB first by selecting it in the Model Explorer and dragging/dropping it onto a connected Teiid instance in the Servers view, or right-click select **Modeling > Deploy** action.

Once deployed, you can select the VDB in the Teiid View and right-click select the **Execute VDB** action there. This will create a Teiid Connection Profile specific for that VDB, open the Database Development perspective and create a connection to your VDB. Continue with Step's 2 and 3 above.



Note

You can deploy a dynamic VDB to the server, but the VDB may end up being INACTIVE if any referenced data source (JNDI name) is not already be deployed. Dynamic VDBs do not contain enough connection or data source information to automatically find and/or deploy a data source.



Note

If you do not have a Teiid instance defined or your default Teiid instance is disconnected, the following dialog will be displayed if the **Modeling > Deploy** action is launched.

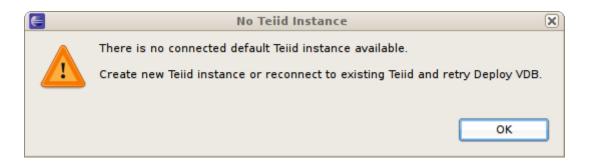


Figure 11.16. No Teiid Instance Defined

11.3.4. Executing a Deployed VDB

To execute a VDB, that's been deployed manually, follow the steps below:

- To execute a VDB, that's been deployed manually, follow the steps below:
 - Step 1 : Open the Database Development perspective.
 - Step 2 : Select the Database Connections folder and choose the New action to display the New Connection Profile dialog.

New Connection Profile	×
Connection Profile	
Create a Teiid connection profile.	
Connection Profile Types:	
type filter text	
Ingres	
MaxDB	
III ModeShape	
₩ySQL	
🔮 Oracle	
PostgreSQL	
SQL Server	
SQLite	
Sybase ASA	
Sybase ASE	
🔮 Teiid	
Na <u>m</u> e:	
Products	
Description (optional):	
?	Finish

Figure 11.17. New Connection Profile Dialog

- Step 3 : Enter unique name for your profile, select an existing connection profile type and hit Next.
- Step 4 : In the Teiid Profile Wizard page, select the New Driver Definition button to locate and select the Teiid client jar on your file system. Configure your URL using your VDB Name, Host, Port, Username (default = "admin") and Password (default = "teiid").

🧐 Teiid Profile Wizard 🛛 🕅
Specify a Driver and Connection Details Select a driver from the drop-down and provide login details for the connection.
Drivers: Teiid Server JDBC Driver
Properties G <u>e</u> neral <u>O</u> ptional
VDB Name
Products
H <u>o</u> st localhost
Port
31000
Username
admin
P <u>a</u> ssword
SSL Connection
☑ Save password
✓ Connect when the wizard completes Test Connection
Connect every time the workbench is <u>s</u> tarted
? < <u>Back</u> <u>Next</u> > Cancel <u>Finish</u>

Figure 11.18. Teiid Connection Profile Dialog

• Step 5 : Select Next to view a summary of your new Teiid Connection Profile.

Property	Value
Name	Products
Description	
Auto connect at startup	false
Auto connect on finish	true
Database	Products
Host	localhost
Port	31000
Username	admin
SSL protocol	false
Save password	true
URL	jdbc:teiid:Products@mm://localhost:31000
	Jabc.tella.Products@mm.//localnost.31000

Figure 11.19. Teiid Connection Profile Summary

- Step 6 : Select Finish.
- Step 7 : Select your new Teild connection profile and right-click select Open SQL Scrapbook, enter your designer SQL (i.e. SELECT * FROM TableXXXX), select all text and right-click select Execute Selected Text.

SQL Scrapbook 0 🛛	🗍 Parts.vdb				
Type: Teiid_7.1		≎ <u>N</u> ame:	Parts - Teiid Connecti 🗘	Data <u>b</u> ase:	
SELECT * FROM Par	rtsSourceA.Part	s			
STELLET THEFT I					

Figure 11.20. SQL Scrapbook Editor

• Step 8 : Results of query should be displayed in the SQL Results view on the Result1 tab.

ype	e query expression here					
Stat	Operation		Date	Connection Profile		
•	/ Succee SELECT * FROM	PartsSourceA.Parts	Oct 15, 2010	Parts - Teiid Connection		
	PART_ID	PART_NAME		PART_COLOR		P
	PART_ID P300	PART_NAME Nut		PART_COLOR Red		P
	-			_		
2	P300	Nut		Red		
1 2 3 4	P300 P301	Nut Bolt		Red Green	-	1

Figure 11.21. SQL Results View

• Step 9 : The query Execution Plan should also be displayed on the Teiid Execution Plan view tab. The Execution Plan can also be generated without running the query. In the SQL scrapbook, rt-click then select Teiid_7.x > Get Execution Plan.

SQL Results 🐙 Teiid Execution Plan 🖾
Preview Data for : myTbl
SQL select * from "vModel"."myTbl"
Execution Plan
Text Plan Tree
xml version="1.0" encoding="UTF-8" standalone="yes"? <node name="AccessNode"> <property name="Output Columns"> <value>INTKEY (integer)</value> <value>STRINGKEY (string)</value> <value>STRINGKEY (string)</value> <value>INTNUM (integer)</value> <value>STRINGNUM (string)</value> <value>FLOATNUM (double)</value> <value>LONGNUM (long)</value> <value>DOUBLENUM (double)</value></property></node>

Figure 11.22. Teiid Execution Plan View

Materialization Support

Designer provides modeling support for server-side materialization including:

- Extension properties for materialized view and referenced materialized tables
- · General Create POJO feature for selected table or view
- Materialize action and wizard to set required properties and generate optional JDG module corresponding to the generated materialized table
- JDG connection profile definition and wizard

12.1. Materialization Extension Properties

The Teiid runtime supports materialization which is identified at VDB deployment time through metadata properties. A couple base properties are provided by Designer's relational table properties. These include a **Materialized** boolean property and a **Materialized Table** reference property as shown below. In addition, there are extension properties for a materialized table to help set-up your materialized table behavior. (See Teiid Runtime documentation for details)

🚰 Model Explorer 🕱 🛛 📴 Outline 陷	Project Explorer	
	월 - 🗣 🖧 🚸 ⇔ ⇔ 🚇 🖻 🔩	∇
 MarketData.xmi PortfolioVDB.vdb StockJDGSource.xmi Stocks.xmi * ≅ import declarations (5) Rackage Diagram StockPrices Stock 		
✓ Default Server		\$
Name: 🖷 Teiid 9.0 Teiid Version: 🕋 9.0.0		
Description Properties 🛛 🎆	Status 📑 🕆 🔽 🗢 🗖	
Property	Value	
 Extension relational:Action On Error relational:Allow Teiid based management relational:Commands To Run On VDB Drop relational:Commands To Run On VDB Start 		
relational:Global Temp Table	false	
relational:Materialize After Load Script relational:Materialize Before Load Script relational:Materialized Share Scope relational:Materialized Staging Table relational:Materialize Load Script relational:Status Table Name relational:Time To Live (ms) Info	 execute StockJDGSource.native("swap cache names"); execute StockJDGSource.native("truncate cache"); NONE StockJDGSource.ST_StockMatSource 	
▼ Misc		
Cardinality Logical Relationships Materialized	-1	
Materialized Table Name	StockMatSource (Path=/PortfolioExample/StockJDGSource	e.xmi)
Name In Source Supports Update	u≣ Isk false	
System	🖙 false	

Figure 12.1. Materialized View Properties

12.2. Create POJO

As part of the new 11.0 Designer feature for supporting JDG and materialization, we added a separate **Create POJO** action and wizard. This feature will generate a simple java class that results in getter and setter methods for each column. There is also an option to generate an JDG module containing the compiled class.

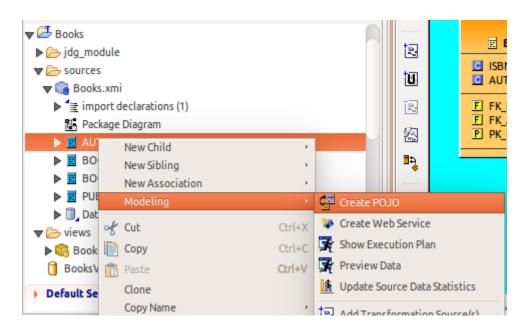


Figure 12.2. Create POJO Action

😕 💷 Create POJO an	d JDG Module
Define Java and JDG Mod Click Finish to generate Po	-
Properties	
Package name	org.teiid.jdg.pojo
Java class name	AUTHORS
Annotation type	Protobuf v
Generate JDG Mo	odule
Module zip file name	AUTHORS_JDG_Module.zip
Output Options	
🚰 Location Books	
?	Cancel Finish

Figure 12.3. Create POJO Dialog

12.3. Materialize a View

You can now materialize a view and generate the necessary related artifacts to support external materialization through your JDG cache. The generated artifacts include:

- New source model containing both the materialized table and companion staging table.
- POJO java file and compiled java class
- JDG zip module containing packaged jar file and module.xml file for deployment/extraction into your JBoss server

Follow the steps below to materialize your view....

 Step 1- In the Model Explorerselect the view you wish to materialize and right-click select the Modeling > Materialize action

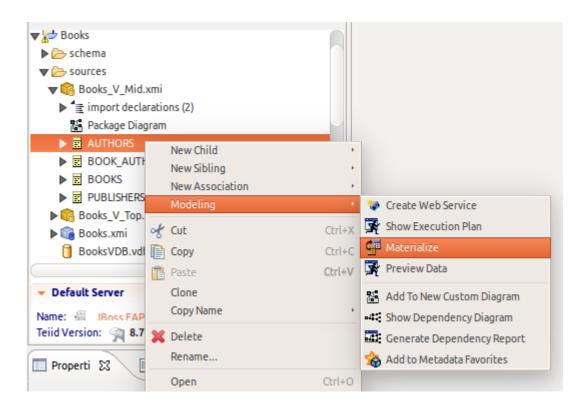


Figure 12.4. Materialize Action

• Step 2- On the first wizard page, define your generated source model name and location within the same model project. Click **NEXT>** to continue.

😣 🗆 Generate	Materialized JDG Module
Materialized Source	ce Model Definition
Click Finish to gene	arate JDG module
View Model Info	rmation
🕞 Name 🛛 🖪	Books_V_Mid.xmi
E View	AUTHORS
Source Model In	nformation
👔 Name	AUTHORS
🔁 Location	Books/sources
\bigcirc	< Back Next > Cancel Finish

Figure 12.5. Materialized Source Model Definition

• Step 3- On the second wizard page, select your options for generating a POJO and a JDG module as well as the output folder location in your model project. Click **Finish** perform to complete the operations.

😣 💷 Generate Materialized JDG Module	
Define Java and JDG Module Options Click Finish to generate JDG module	
Generate POJO Properties	
Package name org.teiid.jdg.pojo	
Java class name AUTHORS	
Annotation type Protobuf 💌	
Generate JDG Module	
Module zip file name authors Jdg Module	
Output Options	
Eocation Books/authors-jdg-module	
Cancel Finish	

Figure 12.6. Define Java and JDG Module Options

The figure below shows the file structure of the generated POJO, class file, jar and jdg module zip file

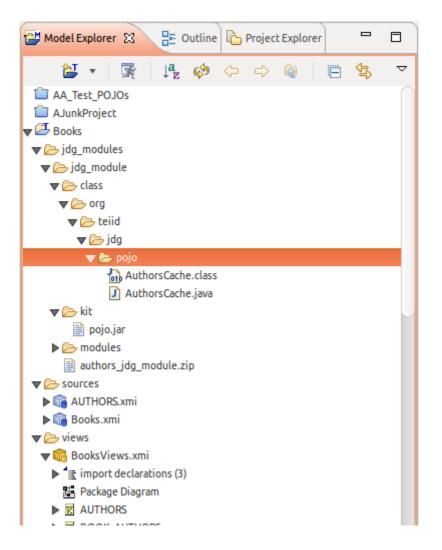


Figure 12.7. Generated JDG module files

12.4. JDG Connection Profile

JDG connection profile wizard can be used to define JDG-specific connection information required to create a server-side resource adapter. By setting the JDG connection profile on your generated materialized source model you can use the **Create Data Source** feature to create and deploy a resource adapter that supports your materialized view.

To create a JDG connection profile, open the **Database Development** perspective and select the **New Connection Profile** action and **JDG** profile option as shown below.

New Conne	ction Profile		
onnection Profile			
reate a JDG connect	ion profile.		
Connection Profile T	ypes:		
type filter text			
A Flat File URL So	urce		
JBoss DataSour	ce		
🚛 JDG			
🕞 LDAP			
X SalesForce			
A Web Services D	ata Source (OData)	
A Web Services D	ata Source (SOAP)		
AML File URL So	ource		
A XML Local File S	Source		
lame:			
My_MaterializedSto	ock		
	.):		
escription (optional)			
Description (optional			
escription (optional)			
Description (optional			

Figure 12.8. JDG Connection Profile Page 1

Enter your unique JBoss JNDI name and various JDG-specific resource adapter properties and click **Finish**

Chapter 12. Materialization S...

New conne	ction profile			
DG Connection Prot Click Next or Finish	file Definition			
Profile Name Profile Description	My_MaterializedStoc	k		
JNDI Name	My_MaterializedSto	ck		
Cache Type Map	addressbook_indexe	d:org.teiid.jdg.pojo.Stock;m_ProductId		
Configure Connec	tion to Cache			
O Remote Server	List	127:0.0.1:11322		
Cache JNDI Nan	ne			
O Hot Rod Client I	Properties File Name			
JDG Cache Names Staging Cache Name	addressbook_inde	xed_mat		
Alias Cache Name alias Cache				
JDG Schema Optio	ns			
Annotations	O Pr	otobuf		
Protobuf Definition	File Location			
Message Ma	rshallers Map			
Messag	ge Descriptor			
Mod	ule (optional) org.te	iiid.jdg.pojo		
?	< Back	Next > Cancel Finish		

Figure 12.9. JDG Connection Profile Page 2

Searching

Designer provides multiple search actions located via Teiid Designer sub-menu in Eclipses Search menu. **Search** menu.

🔗 Se <u>a</u> rch	Ctrl+H
🐶 <u>F</u> ile	
ኛ <u>R</u> emote	
🛷 Pointcut Matches	
🛷 Beans	
💫 Find Seam References	
Find Seam Declarations	
Text	>
Teiid Designer	>

Figure 13.1. Search Options

- The individual actions in the Teiid Designer sub-menu are described below:
 - 🚀

Transformations... - Launches the Transformation Search dialog. User can search models in the workspace for matching SQL text. Search results appear in the dialog and user can select and view SQL as well as open desired transformations for editing.

8

Metadata... - Launches the Search dialog. User can search for models in the workspace by specifying an Object Type, and/or a Data Type, and/or a property value. Search results appear in the Section D.2.8, "Search Results View" view, and double-clicking a result will open that model in the appropriate editor.

• 🔎

Find Model Object - Launches the Find Model Object dialog, which can be used to find an object in the workspace by specifying all or part of its name. Selecting the object will open it in the appropriate editor.

13.1. Finding Model Objects

The Teiid Designer provides a name-based search capability to quickly locate and display model objects.

• To find a model object:

• Step 1 - Open the Find Model Object dialog by either selecting the

action on the main Teiid Designer tool-bar.

		Teiid De	signer	- PartsPi	roject/PartsSour
Se <u>a</u> rch	<u>P</u> roject	<u>M</u> etadata	<u>R</u> un	<u>W</u> indow	<u>H</u> elp
🔗 Se <u>a</u> r	ch		Ctrl+H	a 9.	in 'n 👂
将 <u>F</u> ile				9 33] 🎟 📼 🧨 (
ኛ <u>R</u> em	ote				
🛷 Point	tcut Matcł	nes			~
🛷 Bear	1S				
💫 Find	Seam Ref	ferences			
😕 Find	Seam De	clarations			
Text			>		
Teiid	Designer				ationship
Icitu	Designer				-
				Nr Trai	nsformations
				🔗 Met	adata
				🥭 Find	d Model Object
	 ✓ Sear ✓ File ✓ Rem ✓ Point ✓ Point ✓ Bear Sind ✓ Find ✓ Find 	 Se<u>a</u>rch File Eemote Pointcut Match Beans Find Seam Ref Find Seam De Text 	Search Project Metadata Image: Search Image: Search Image: File Image: Search Image: Search Image: Search Image: Search <th>Search Project Metadata Run Search Ctrl+H Pile Remote Pointcut Matches Beans Find Seam References Find Seam Declarations Text Y</th> <th>Search Ctrl+H Pile File Pointcut Matches Pointcut Matches Pointcut Matches Pointcut Matches Point Seam References Point Seam Declarations Text Image: Search Se</th>	Search Project Metadata Run Search Ctrl+H Pile Remote Pointcut Matches Beans Find Seam References Find Seam Declarations Text Y	Search Ctrl+H Pile File Pointcut Matches Pointcut Matches Pointcut Matches Pointcut Matches Point Seam References Point Seam Declarations Text Image: Search Se

Figure 13.2. Find Model Object Action In Toolbar

or select the same action via the main menu's **Search > Find Model Object**

action.

Find Model Object	×
Type Object Name (? = any character, * = any String);	
Part	
Matching Model Objects:	
A PART_COLOR	-
PART_ID	
A PART_NAME	
A PART_WEIGHT	
D PARTS	
A parteBuColor	<u> </u>
Locations:	
/PartsProject/PartsSourceA.xmi/PARTS/PART_COLOR	
/PartsProject/PartsSourceB.xmi/PARTS/PART_COLOR	
/PartsProject/PartsVirtual.xmi/PARTS/PART_COLOR	
/PartsProject/PartsVirtual.xmi/partsByColor/NewProcedureR	e
	_
	>
Cancel OK	

Figure 13.3. Find Model Object Dialog

- Step 2 Begin typing a word or partial word in the Type Object Name field. Wild-card (*) characters will be honored. As you type, the objects which match the desired name will be displayed in the Matching Model Objects list. If there are more than one objects with the same name, the locations or paths of the objects are displayed in the Locations list.
- Step 3 If more than one object exists with the desired name, select the one of the locations.
- Step 4 Click OK. If editor is not open for the object's model, an editor will open. The desired object should end up displayed in a diagram (if applicable) and selected.

13.2. Search Transformation SQL

The Teiid Designer provides a search capability to string values present in transformation SQL text.

• To search for string values in your transformations SQL:

• Step 1 - Select Search > Transformations... action on the Teiid Designer main menu

which opens the Search Transformations dialog.

	Search Tranformations 🛛 🗙
Inpu	ıt
	Parts 🗸
	Case Sensitive
Perfo	orm Search
Sea	rch Status
Sea	rch complete for: [Parts]
Virt	ual Tables, Views, Procedures _Operations
[2] 9	SQL matches for: [Parts]
	SupplierInfo <table> : /PartsProject/PartsVirtual.xmi</table>
	VE SELECT
	partsByColor <procedure> : /PartsProject/PartsVirtual.xmi</procedure>
	R SELECT
SEL	ECT SQL String
	ATE VIRTUAL PROCEDURE BEGIN SELECT * FROM PartsSourceB.PARTS WHERE
Part	sSourceB.PARTS.PART_COLOR = PARTSVIRTUAL.PARTSBYCOLOR.colorin; END
	Edit Close

Figure 13.4. Search Transformations Dialog

- Step 2 Specify a string segment in the Find: field and specify/change your case sensitive preference.
- Step 3 Select Perform Search button. Any transformation object containing SQL text which contains occurances of your string will be displayed in the results section.

You can select individual objects and view the SQL. If a table or view supports updates and there is insert, update or delete SQL present, you can expand the object and select the individual SQL type as shown below.

)	Search Tranformations	(
Input-		
Find: Pa	arts	~
Cas	e Sensitive	
Perform	Search	
Search	n Status	
	complete for: [Parts]	
Virtual	l Tables, Views, Procedures _Operations	
[3] SQL	. matches for: [Parts]	
🕨 🖻 S	SupplierInfo <table> : /PartsProject/PartsVirtual.xmi</table>	A
▶ <u>f</u> , p	partsByColor <procedure> :/PartsProject/PartsVirtual.xmi</procedure>	
🗢 🖪 P	PARTS <table> : /PartsProject/PartsVirtual.xmi</table>	
E	E SELECT	
Q	E INSERT	
E	UPDATE	
E	E DELETE	·
NSERT	r sqL string	
CREATE (PartsS PartsSo	E PROCEDURE BEGIN ROWS_UPDATED = INSERT INTO PartsSourceB.PARTS SourceB.PARTS.PART_ID, PartsSourceB.PARTS.PART_NAME, ourceB.PARTS.PART_COLOR, PartsSourceB.PARTS.PART_WEIGHT) VALUES S.PART_ID, INPUTS.PART_NAME, INPUTS.PART_COLOR, INPUTS.PART_WEIGHT); EN	D
	Edit	Close

Figure 13.5. Insert SQL Example

If you wish to view the selected object and its SQL in a **Model Editor**, you can click the **Edit** button. An editor will be opened if not already open. If an editor is open its tab will be selected. In addition, the **Transformation Editor** will be opened and you can perform **Find/Replace** (Ctrl-F) actions to highlight your original searched text string and edit your SQL if you wish.

13.3. Search Models Via Metadata Properties

The Teiid Designer provides a search capability to find model objects that are characterized by one or more metadata property values.

- To search your models using metadata:
 - Step 1 Select Search > Metadata... action on the main Teild Designer toolbar

which opens the **Search** dialog.

9	Search 🕺
🕻 漀 Remote Sea	rch 💱 File Search 🔗 Metadata Search 🔯 Relationship Search 🕽
Object Type	
Any	
O Object Class:	
Data Type	
Any	
◯ Simple Type:	
🗌 Include su	btypes
O Runtime Type:	bigdecimal 🗘
Properties	
✔ Where:	Name
Contains	O Does NOT Contain C Exact Match
	case insensitive, * = any string, ? = any character):
	ase insensitive, * = any string, ? = any character).
Part	▼
Scope	
<u>W</u> orkspace	○ Selecte <u>d</u> resources
○ Wor <u>k</u> ing set:	Choose
Customize	Cancel <u>S</u> earch

Figure 13.6. Metadata Search Dialog

- Step 2 Specify desired search options for Object Type, Data Type and Properties.
- Step 3 Click Search. The search will be performed and the results will be displayed in the Section D.2.8, "Search Results View". If the view is not yet open, it will be opened automatically.

Appendix A. Supported Data Sources

The matrix indicates for a given data source how a model can be created (Desginer Import Option) and how the data source is integrated (Translator) for data access.



Note

The DDL Import option is an available option to build a source model for any data source. Its only indicated below when there's no specific importer created for that specific data source type.

Table A.1. Teiid Designer Supported Data Sources

Data Source	Translator Type	Designer Import Option
Apache Derby	JDBC - derby	JDBC Importer
Files	file	File Importer
General JDBC	JDBC - jdbc-simple	JDBC Importer
HDFS (Hadoop)	hive	use DDL importer or perform modeling manually
HSQL	JDBC - hsql	JDBC Importer
H2	JDBC - h2	JDBC Importer
Ingres	JDBC - ingres (Ingres 2006 or later) JDBC - ingres93 (Ingres 9.3 or later)	JDBC Importer
IBM DB2	JDBC - db2	JDBC Importer
Informix	JDBC - informix	JDBC Importer
LDAP/ActiveDirectory	Idap	LDAP Importer
LoopBack	JDBC - loopback	use DDL importer or perform modeling manually
MetaMatrix	JDBC - metamatrix	JDBC Importer
ModeShape/JCR	JDBC - modeshape	JDBC Importer
Mondrian	olap	use DDL importer or perform modeling manually
MS Access	JDBC - access	JDBC Importer

Data Source	Translator Type	Designer Import Option
MS Excel	JDBC - excel-odbc	JDBC Importer
MS SQL Server	JDBC - sqlserver	JDBC Importer
MySQL	JDBC - mysql5 (mysql)	JDBC Importer
Netezza	JDBC – netezza	JDBC Importer
Oracle	JDBC - oracle	JDBC Importer
PostgreSQL	JDBC - postgresql	JDBC Importer
Salesforce.com	salesforce	SalesForce Importer
SAP Gateway	WS	File Source (XML) Importer
SAP R/3	WS	
SAP Services Registry	WS	WSDL Importer
Sybase ASE	JDBC - sybase	JDBC Importer
Teradata	JDBC - teradata	JDBC Importer
Teiid	JDBC - teiid	JDBC Importer
Web Services (SOAP/WSDL)	WS	WSDL or URL Importer
Web Services (Rest/OData)	WS	File Source (XML) Importer

Appendix B. Designer Metadata Usage Requirements In Teiid Runtime

Based on the metadata exposed by the Teiid Desinger the below table shows which fields are required and how that information is being used in Teiid runtime currently as of Teiid version 7.5. A ODS file attached if you like to modify.

TABLE	Туре	In Designer	In Metadata API	Required	Description
FullName	String	Yes	Yes	Yes	Name of the Table
NameInSource	String	Yes	Yes	Yes	Name of Table in the source system, for view this can be empty, also used on variety of use cases
Cardinality	Integer	Yes	Yes	Yes	Cardinality is used to calculate the cost of source node access
ableType	Integer	Yes	Yes	Yes	Table,View,Do
Virtual	Boolean	Yes	Yes	Yes	Used to find if this is source table Vs view
sSystem	Boolean	Yes	Yes	No	Only used for System metadata
sMaterialized	Boolean	Yes	Yes	Yes	To identify that the table is materialized

Table B.1. Data Usage for Tables

TABLE	Туре	In Designer	In Metadata API	Required	Description
SupportsUpdat	eBoolean	Yes	Yes	Yes	To allow updates on the table
PrimaryKeyID	String	Yes	KeyRecord	Yes	Used for creating indexes on temp tables and to create default update/delete procedures
ForeignKeyIDs	Collection	Yes	List <foreignke< td=""><td>ey⊁es</td><td>Used in Planning of query (rule raise access)</td></foreignke<>	ey⊁es	Used in Planning of query (rule raise access)
IndexIDs	Collection	Yes	List <keyrecord< td=""><td>d¥es</td><td>Used for creating indexes on temp tables and in planning (estimate predicate cost)</td></keyrecord<>	d¥es	Used for creating indexes on temp tables and in planning (estimate predicate cost)
UniqueKeyIDs	Collection	Yes	List <keyrecor< td=""><td>d¥es</td><td>Used for query planning</td></keyrecor<>	d¥es	Used for query planning
AccessPattern	Dection	Yes	List <keyrecor< td=""><td>d¥es</td><td>Used for enforcing the criteria on query</td></keyrecor<>	d¥es	Used for enforcing the criteria on query
MaterializedTa	b &thD ng	Yes	Table	Yes	Reference to Materialization table
insertEnabled	Boolean	**	Yes	Yes	Flag for checking insert procedure is enabled for view

TABLE	Туре	In Designer	In Metadata API	Required	Description
deleteEnabled	Boolean	**	Yes	Yes	Flag for checking delete procedure is enabled for view
updateEnabled	Boolean	**	Yes	Yes	Flag for checking update procedure is enabled for view
Select Transformation	String	**	Yes	Yes	Transformation for Select in case of View
Insert Plan	String	**	Yes	Yes	Transformation for Insert in case of View
Update Plan	String	**	Yes	Yes	Transformation for Update in case of View
Delete Plan	String	**	Yes	Yes	Transformation for Delete in case of View
Bindings	Collection	**	Yes	Yes	XML Document
SchemaPaths	Collection	**	Yes	Yes	XML Document

Table B.2. Data Usage for Columns

COLUMN	Туре	In Designer	In Metadata API	Required	Description
FullName	String	Yes	Yes	Yes	Name of the column
NameInSource	String	Yes	Yes	Yes	Name of the column in source system

COLUMN	Туре	In Designer	In Metadata API	Required	Description
IsSelectable	Boolean	Yes	Yes	Yes	Column is allowed in select
IsUpdatable	Boolean	Yes	Yes	Yes	Column is allowed in Update/ Insert/Delete
NullType	Integer	Yes	Yes	Yes	Used for validation if null value allowed
IsAutoIncreme	n fabte ean	Yes	Yes	Yes	During insert used to validate if a value is required or not
IsCaseSensitiv	eBoolean	Yes	Yes	??	??
IsSigned	Boolean	Yes	Yes	??	Used in System Metadata
IsCurrency	Boolean	Yes	Yes	No	Only used for System metadata
IsFixedLength	Boolean	Yes	Yes	No	Only used for System metadata
IsTranformation	n Booltean amete	r Yes	??	??	??
SearchType	Integer	Yes	Yes	Yes	Used for defining the capability of the source
Length	Integer	Yes	Yes	??	Used in System Metadata
Scale	Integer	Yes	Yes	??	Used in System Metadata

COLUMN	Туре	In Designer	In Metadata API	Required	Description
Precision	Integer	Yes	Yes	??	Used in System Metadata
CharOctetLeng	gtInteger	Yes	Yes	No	only used for System metadata
Radix	Integer	Yes	Yes	??	Used in System Metadata
DistinctValues	Integer	Yes	Yes	Yes	Used for cost calculations, System metadata
NullValues	Integer	Yes	Yes	Yes	Used for cost calculations, System metadata
MinValue	String	Yes	Yes	Yes	Used for cost calculations, System metadata
MaxValue	String	Yes	Yes	Yes	Used for cost calculations, System metadata
Format	String	Yes	Yes	No	Only used for System metadata
RuntimeType	String	Yes	DataType	Yes	Data Type
NativeType	String	Yes	Yes	Yes	Translators can use this field to further plan
DatatypeObjec	t lS tring	Yes	??	??	
DefaultValue	String	Yes	Yes	Yes	Used for Insert and procedure execute operations

COLUMN	Туре	In Designer	In Metadata API	Required	Description
					when the values are not supplied
Position	Integer	Yes	Yes	Yes	Used in the index calculations

Table B.3. Data Usage for Primary Keys

PRIMARY KEY	Туре	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				
ForeignKeyIDs	Collection				Extends KeyRecord

Table B.4. Data Usage for Unique Keys

UNIQUE KEY	Туре	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				
ForeignKeyIDs	Collection				

Table B.5. Data Usage for Indexes

INDEX	Туре	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				

ACCESS PATTERNS	Туре	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				

Table B.6. Data Usage for Access Patterns

Table B.7. Data Usage for Result Sets

RESULT SET	Туре	In Designer	In Metadata API	Required	Description
FullName	String				See DataType
NameInSource	String				
ColumnIDs	Collection				

Table B.8. Data Usage for Foreign Keys

FOREIGN KEY	Туре	In Designer	In Metadata API	Required	Description
FullName	String				See the KeyRecord, See Table
NameInSource	String				
ColumnIDs	Collection				
UniqueKeyID	String				

Table B.9. Data Usage for Data Types

DATA TYPE	Туре	In Designer	In Metadata API	Required	Description
FullName	String			No	Only used for System metadata
NameInSource	String			No	Only used for System metadata
Length	Integer			No	Only used for System metadata

DATA TYPE	Туре	In Designer	In Metadata API	Required	Description
PrecisionLengt	hInteger			No	Only used for System metadata
Scale	Integer			No	Only used for System metadata
Radix	Integer			No	Only used for System metadata
IsSigned	Boolean			No	Only used for System metadata
IsAutoIncreme	nBoolean			No	Only used for System metadata
IsCaseSensitiv	eBoolean			No	Only used for System metadata
Туре	Integer			No	Only used for System metadata
SearchType	Integer			No	Only used for System metadata
NullType	Integer			No	Only used for System metadata
JavaClassNam	eString			Yes	Maps to runtime type based on java class name
RuntimeTypeN	a 8te ng			No	Only used for System metadata
DatatypeID	String			No	Only used for System metadata

DATA TYPE	Туре	In Designer	In Metadata API	Required	Description
BaseTypeID	String			No	Only used for System metadata
PrimitiveTypeII	DString			No	Only used for System metadata
VarietyType	Integer			No	Only used for System metadata
VarietyProps	Collection			No	Only used for System metadata

Table B.10. Data Usage for Procedures

PROCEDURE	Туре	In Designer	In Metadata API	Required	Description
FullName	String	Yes	Yes	Yes	Name of the column
NameInSource	String	Yes	Yes	Yes	Name of the column in source system
IsFunction	Boolean		Yes	Yes	Determines if this function
IsVirtual	Boolean		Yes	Yes	If Function then UDF else stored procedure
ParametersIDs	Collection		Yes	Yes	Parameter List
ResultSetID	String		Yes	Yes	Result set columns
UpdateCount	Integer		Yes	Yes	Update count defines the number of sources being updated, only applicable

PROCEDURE	Туре	In Designer	In Metadata API	Required	Description
					for virtual procedures

Table B.11. Data Usage for Procedure Parameters

PROCEDURE PARAMETER	Туре	In Designer	In Metadata API	Required	Description
ObjectID	String				Same as Column
FullName	String				Same as Column
nameInSource	String				Same as Column
defaultValue	String				Same as Column
RuntimeType	String				Same as Column
DatatypeObjec	tl S tring				Same as Column
Length	Integer				Same as Column
Radix	Integer				Same as Column
Scale	Integer				Same as Column
NullType	Integer				Same as Column
Precision	Integer				Same as Column
Position	Integer				Same as Column
Туре	String			Yes	Defines parameter is IN/OUT/ RETURN
Optional	Boolean			No	Defines if the parameter is optional or not, only

PROCEDURE PARAMETER	Туре	In Designer	In Metadata API	Required	Description
					used system metadata

Table B.12. Data Usage for SQL Transformations

SQL NSFORMATIO	Туре	In Designer	In Metadata API	Required	Description
VirtualGroupNa	ar 8e ring	Yes	No	Yes	See Table, the properties defined on Table
TransformedOl	Djêtrillø	Yes	No	Yes	See Table, the properties defined on Table
Transformation	(Shijëngi D	Yes	No	Yes	See Table, the properties defined on Table
Transformation	SSa‡tring	Yes	No	Yes	See Table, the properties defined on Table
Bindings	Collection	Yes	No	Yes	See Table, the properties defined on Table
SchemaPaths	Collection	Yes	No	Yes	See Table, the properties defined on Table

Table B.13. Data Usage for VDBs

VDB	Туре	In Designer	In Metadata API	Required	Description
FullName	String	Yes	vdb.xml	Yes	Name of the VDB
NameInSource	String	??	No	No	Not required
Version	String	Yes	vdb.xml	Yes	VDB version

VDB	Туре	In Designer	In Metadata API	Required	Description
Identifier	String	Yes	No	No	Not required
Description	String	Yes	vdb.xml	No	Used by System metadata
ProducerName	e String	Yes	No	No	Not required
ProducerVersio	orString	Yes	No	No	Not required
Provider	String	Yes	No	No	Not required
TimeLastChan	g ett ring	Yes	No	No	Not required
TimeLastProdu	ucentaring	Yes	No	No	Not required
ModelIDs	Collection	Yes	vdb.xml	Yes	Defines the model list in a VDB

Table B.14. Data Usage for Annotations

ANNOTATION	Туре	In Designer	In Metadata API	Required	Description
FullName	String	Yes	Yes	No	System metadata, as description on procedure parameter
NameInSource	String	Yes	No	No	Not required
Description	String	Yes	No	No	Not required

Appendix C. User Preferences

The Teiid Designer provides options or preferences which enable customization of various modeling and UI behaviors. Preferences can be accessed via the Edit > Preferences action on the Main toolbar.

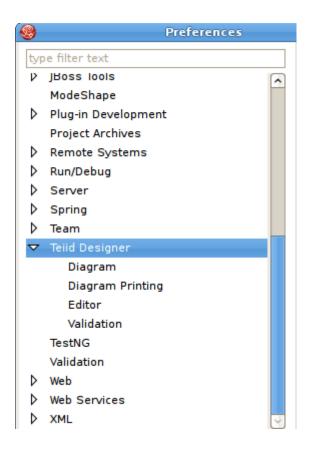


Figure C.1. Preferences Dialog

C.1. Teiid Designer Preferences

General Teiid Designer preferences include.

- Enable auto-creating of a source model's data source on Teiid Server indicates if data sources that match the default name should be auto-created if they do not exist on Teiid server.
- Enable Preview If the Designer Runtime feature is installed and a Teiid Instance is defined, Teiid Designer will automatically keep the preview artifacts (VDBs) in sync with the workspace models. Unchecking this preference will disable preview feature and not create preview artifacts.
- Enable Preview Teiid Cleanup If operating Designer with Enable Preview = TRUE, then this preference will result in automatic clean-up of your preview artifacts from your Teiid servers.

Any preview VDBs or preview data sources will be undeployed from your servers as part of Eclipse's shut-down process.

- Always open editor without prompting To change/edit a model, it must be opened for editing. Checking this box will automatically open the model in an editor if the user attempts to perform a change in a model. If unchecked, the user will be informed that an editor will be opened before the operation is completed.
- Open Designer perspective when model is opened If a model is opened via importing projects, the New > Teiid Metadata Model menu and the Teiid Designer perspective is not open, you may want to automatically open the perspective and beging working on your model. This preference has 3 settings. Always open, which means always open the perspective without prompting; never open, which means do not open the Teiid Designer perspective, or prompt, which will always ask you if you wish to open the Teiid Designer perspective.
- Check and update imports during save occasionally editing a model may add or remove objects in one model that reference objects in another model. Model Imports keep track of these dependencies within each model. A validation error or warning may appear during a build. Checking this box will automatically check and update imports during the save process. This will result in any unneeded imports being removed from the model or any required imports added to the model. If unchecked, no updating of imports will be performed.
- Default Teiid Server Version determines what version of server modelling will be targeted at, if no teiid server has been defined in the Servers View. Thus, it is possible to still design models without the need to define and connect to a server. However, possible values are confined to the teiid runtime clients installed.

Teiid Designer 🔶 👻 🗣	
Specify values for general Designer preferences.	
Preview Data/VDB Execution	
☑ Enable auto-creating of a source model's data source on Teiid Instance (VDB Execution)	
Enable auto-toggling children of the checked model object	
C Enable Preview	
Generate Teiid Query Plan on execution	
✓ Enable Preview Teiid Cleanup	
120 Teiid Connection Importer Timeout (sec)	
Teiid Designer	
Always open editor without prompting	
Open Designer perspective when a model is opened	
○ Always open ○ Never open Prompt	
Check and update imports during save	
8.4.x Default Teiid Instance Version (applicable if no teiid instance defined)	
Restore Defaults Apply	

Figure C.2. General Teiid Designer Preferences Panel

C.1.1. Diagram Preferences

Several diagram preferences are available to customize your diagrams.

- **Notations** Standard diagram notation for Teild Designer is based on UML notation. Future releases may include alternate notations.
- Routers The relationship link type for Package and Custom diagrams (Foreign Key Primary Key relationships) can be customized. Available options include Orthogonal (default), Direct or Manual (user defined breakpoints).
- Font Settings Select font type, style and size.
- **Background Color Settings** Select a unique background color for each diagram type to help differentiate between types.

- **Model Size** Displaying very large diagram may take a considerably long time. This preference allows users to set an upper limit on the number of objects to display in a diagram. If this limit is exceeded, a warning is displayed to the user and the diagram is not constructed.
- **Relationship Options** UML-type relationships can be customized in a couple of ways. Role Names and Multiplicity labels can be shown or hidden using the check-boxes labeled **Show Role Names** and **Show Multiplicity**.

Diagram	⇔ • ⇔~ •
General settings for Diagrams	
Notations	
Default Notation: UML	•
Routers	
Default Router Style: Orthogonal	•
Font Settings	
Font Type: Sans - Regular - 8	ew / Change
Background Color Settings	
Dependency Diagram: Custom Diagram:]
Mapping Diagram: Relationship Diagram:]
Transformation Diagram: Custom Relationship Diagram:	
Package Diagram:	
Model Size	
Warn If Model Size is Larger Than 20000	
Relationship Options	
Show Role Names	
Show Multiplicity	
Restore <u>D</u> efaults	Apply

Figure C.3. Diagram Preferences Panel

C.1.2. Editor Preferences

C.1.2.1. XML Document Preferences

XML Document Mapping Preferences provide ways to customize *Section D.3.1.2.4, "Mapping Diagram*" and *Section 6.3.4, "Recursion Editor (XML)*" behavior.

Editor 🗘 ~	⇔~ ▼
XML Document Mapping Settings Transformation Table VDB	
XML Mapping Diagram Settings]
Auto-expand tree when model has fewer than 21 mapping class	ises.
Auto-expand XML document tree to level:	
Hide Mapping Class Columns by Default	
XML Document Mapping	
Upper XML Recursion Limit: 10	
Remove Duplicate Attributes on Merge	

Figure C.4. XML Document Preferences Panel

C.1.2.2. Table Editor Preferences

Section D.3.1.3, "Table Editor" Preferences provide a way to customize the order and the information content for each model object type.

ML Document Mapping Settings	Transformation Table	VDB					
Preferences for a given table will only be present if the table editor tab for it has been viewed. After restoring defaults, the table editor must be							
iewed again for the default prefe		ble.					
Table	Columns						
Base Tables	Column	Visible					
Columns	Location	Yes					
Foreign Keys	Name	Yes					
Function Parameters	Name In Source	Yes					
Inputs	Native Type	Yes					
Interfaces	Length	Yes					
Operations	Length Fixed	Yes					
Outputs	Numeric Precision	Yes	Up				
Primary Keys	Numeric Scale	Yes	Down				
Procedures	Nullable	Yes					
Procedure Parameters	Auto Incremented	Yes					
Procedure Results	Default Value	Yes					
Return Parameters	Minimum Value	Yes					
Sample Messages	Maximum Value	Yes					
Scalar Functions	Format	Yes					
Schemas	Character Set Name	Yes					
Views	Collation Name	Yes					

Figure C.5. Table Editor Preferences Panel

C.1.2.3. Transformation Editor Preferences

Section 6.3.1, "Transformation Editor" Preferences provide a way to customize SQL formatting, diagram layout, and default view entity properties.

Editor	⇔ ~ ⇔~ ▼
XML Document Mapping Settings Transformation Table VDB	
SQL Clauses ✓ Start Clauses on New Line ✓ Indent Clause Content Auto-expand default SELECT * FROM on construction	
 Diagram Layout View and Sources Tree Layout 	
Miscellaneous Default String Length: 10	

Figure C.6. Transformation Editor Preferences Panel

C.1.2.4. VDB Editor Preferences

Section D.3.2, "VDB Editor" Preferences provide a way to customize VDB editor behavior.

Editor	⇔∽ ⇔∽ ▼
XML Document Mapping Settings Transformation Table Image: Synchronize without showing warning dialog Image: Synchronize without showing warning dialog Image: Synchronize without showing warning dialog	VDB

Figure C.7. VDB Editor Preferences Panel

C.1.3. Validation Preferences

Validation Preferences provide a way to customize the severity of some of the rules checked during model validation.

The Validation preference pages, shown below, include the validation preferences for **Core**, **Relational**, **XML and XSD** (**XML Schema**) models.

Validation	⇔ ⊷ ⇔∼ ▼
Specify the Severity Level for validation-related occurrences. Core Relational XSD XML	
Core Validation Control Settings	
Multiple EObjects with same uuid.	Ignore 🗘
String functions (SUBSTRING, LOCATE, and INSERT) 0 to 1 base check:	Warning \$

Figure C.8. Core Model Validation Preferences Panel

Validation 🖓	• 🔿 • •						
Specify the Severity Level for validation-related occurrences.							
Core Relational XML XSD							
Relational Validation Control Settings							
Missing length property on columns with string or character types:	Warning 🌲						
Missing precision property on columns with numeric types:	Warning 🌲						
Columns of built-in type integer:	Warning 🌲						
Table missing name in source value:	Warning 🌲						
Name in source conflicts with sibling entities:	Ignore 🜲						
Name conflicts with sibling entities:	Error 🔹						
Indexes with columns from more than one table:	Warning 🜲						
Empty transformations:	Error 🌲						
Restrict name characters to valid unquoted SQL identifiers:	Ignore 🌲						
Groups are not joined either directly or transitively to other groups, possible cross join:	Warning 🌲						

Figure C.9. Relational Model Validation Preferences Panel

Validation	⇔
Specify the Severity Level for validation-related occurrences. Core Relational XSD XML	
XML Validation Control Settings	
XML Document Elements/Attributes not referencing an XML Schema component:	Warning 😂
Excluded Element from XML Document required by XML Schema:	Warning 😂
XML document entity violates max occurs specified by its schema component:	Warning 😂
Unmapped required XML element or attribute:	Error 😂
Excluded Elements/Attributes from XML Document are mapped:	Warning \$
Mapped XML Element/Attribute has fixed or default value:	Warning 😂
Mapped XML Elements/Attributes with zero minimum occurrences:	Ignore 😂
Mapped XML Elements/Attributes with one maximum occurrence:	Warning \$
XML Root Element mapped to Mapping Class:	Warning 😂
Mapped XML Elements/Attributes Nillable:	Ignore 😂
Incompatible Datatypes for Column-to-Element/Attribute Mappings:	Warning 😂
Restore <u>D</u> efaults	Apply

Figure C.10. XML Document Model Validation Preferences Panel

Validation	, → , → , →
Specify the Severity Level for validation-related occurrences.	
Core Relational XSD XML	
XSD Validation Control Settings	
XML Schema Document validation problems	lgnore ≎

Figure C.11. XSD Schema Model Validation Preferences Panel



Note

Increasing the severity level to error will prevent you from testing your VDB or deploying a web service if violations of that preference are found during validation.

Appendix D. Teiid Designer Ui Reference

D.1. Teiid Designer Perspectives

Teiid Designer utilizes the *Eclipse* [http://www.eclipse.org] Workbench environment which controls visual layout via perspectives. A **Perspective** defines the initial set and layout of views and editors. Within the application window, each perspective shares the same set of editors. Each perspective provides a set of functionality aimed at accomplishing a specific set of tasks.

Perspectives also control what appears in certain menus and toolbars. They define visible action sets, which you can change to customize a perspective. You can save a perspective that you build in this manner, making your own custom perspective that you can open again later.

D.1.1. Teiid Designer Perspective

The Teiid Designer perspective provides access to fundamental model editing and management capabilities. This perspective includes 6 main UI components (or groups of components) as shown below. They include:

- Section D.2.1, "Model Explorer View" Teiid 'tree' view of Model Objects.
- ??? Teiid Server instance view. Provides view of contents for connected instances of installed Teiid runtime.
- Section D.3.1, "Model Editor" Custom editors targeted for ".xmi" metadata model files.
- Section D.2.5, "Properties View" Standard property values for selected workbench objects.
- Section D.2.14, "Guides View" Guides and Status Views which enhance usability.
- **Miscellaneous Views** Includes the Problems view, Message Log view and the SQL Results view (opened if Preview Data action is performed)

ces/P	roductsSQL.xmi - Red Hat JBoss Developer Studio						
*							
•	📬 ProductsSQL.xmi 🔀 👘 ProductViews.xmi	🗍 Product	sVDB.vdb				🔲 Connections 🔅
-	ProductsSQL.xmi Model Editor Model Editor Model Editor Model Editor Contents Image: ProductsSQL.xmi Image: ProductsSQL.xmi <td>ductsProject/sou</td> <td>irces</td> <td>Profile ource Name ame Overrides ons</td> <td>cription</td> <td></td> <td>Connections X Local and Deploye X X X X X X X X X X X X X</td>	ductsProject/sou	irces	Profile ource Name ame Overrides ons	cription		Connections X Local and Deploye X X X X X X X X X X X X X
	0 items						
	Description	Resource	Path	Location	Туре		

Writable

Figure D.1. Teiid Designer Perspective Layout

D.1.2. Opening a Perspective

There are two ways to open a perspective:

Using the Open Perspective button

on the shortcut bar.

- Choosing a perspective from the **Window > Open Perspective** menu.
- To open a perspective by using the shortcut bar button:
 - Step 1 Click on the Open Perspective button
 - Step 2 A menu appears showing the same choices as shown on the Window > Open Perspective menu. Choose Other from the menu.

	er 🐉	
Out	🕸 Debug	-
utli	💱 Java Browsing	t
	<u>O</u> ther	

Figure D.2. Perspectives Menu

• Step 3 - In the Select Perspective dialog choose Teiid Designer and click OK.

	Open Perspective	×
	🐉 Java	
	🔊 Java Browsing	^
	Pava EE	
	Java Type Hierarchy	
	ava specific and the second se	
	JBoss AS	
	BPM jPDL 3	
	a [©]]MX	
	↓ JPA	
	↓ Plug-in Development	
	Remote System Explorer	
	Seam (default)	
	F ⁰ Team Synchronizing	
	😭 Teiid Designer	
	Web	
	🛞 Web Development	
	X XML	-
L		Ū
	Cancel OK	

Figure D.3. Select Perspective Dialog

The Teiid Designer perspective is now displayed.

There are few additional features of perspectives to take note of.

• The title of the window will indicate which perspective is in use.



Figure D.4. Workbench Window Title Bar

- The shortcut bar may contain multiple perspectives. The perspective button which is pressed in, indicates that it is the current perspective.
- To display the full name of the perspectives, right click the perspective bar and select **Show Text** and conversely select **Hide Text** to only show icons.
- To quickly switch between open perspectives, select the desired perspective button. Notice that the set of views is different for each of the perspectives.

🖹 🞯 Teiid	d Desig 🛅 Database	D
-----------	--------------------	---

Figure D.5. Workbench Window Title Bar

D.1.3. Further information

For more details on perspectives, views and other Eclipse workbench details, see formal *Eclipse Documentation* [http://help.eclipse.org/ganymede/index.jsp].

D.2. Teiid Designer Views

Views are dockable windows which present data from your models or your modeling session in various forms. Some views support particular *Section D.3.1, "Model Editor"* and their content is dependent on workspace selection. This section summarizes most of the views used and available in Teiid Designer. The full list is presented in the main menu's *Window > Show View > Other...* dialog under the **Teiid Designer** category.

🔲 Show View	
type filter text	4
👂 🗁 Team	$\mathbf{}$
🗢 🗁 Teiid Designer	
Datatypes	
Description	
☆ Metadata Favorites	
Model Explorer	
 Relationship Navigator 	
ở System Catalog	
Jags 🖅 Tags	
Teiid Model Classes	~
Cancel OK	

Figure D.6. Eclipse Show View Dialog

D.2.1. Model Explorer View

Teiid Designer allows you manage multiple projects containing multiple models and any corresponding or dependent resources. The *Model Explorer* provides a simple file-structured view of these resources.

The Model Explorer (shown below) is comprised of a toolbar and a tree view.

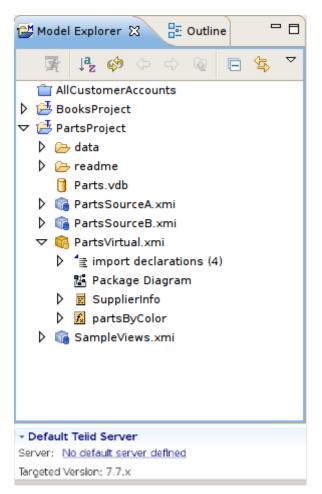


Figure D.7. Model Explorer View

The toolbar consists of nine common actions:

• 록

Preview Data - Executes a simple preview query (SELECT * FROM).

• 1<mark>a</mark>

Sort Model Contents - Sorts the contents of the models based on object type and alphabetizing.

ଂ 🏟

Refresh Markers - Refreshes error and warning markers for objects in tree.

• 🔶

Back - Displays the last "Go Into" location. (See Eclipse Help)

• 🔶

Forward - Displays the next "Go Into" location. (See Eclipse Help)

ଂ 🧔

Up - Navigates up one folder/container location. (See Eclipse Help)

```
• 🖪
```

Collapse All - Collapses all projects.

• 🔁

Link with Editor - When object is selected in an open editor, this option auto-selects and reveals object in Model Explorer.

• 🗸

Additional Actions

The additional actions are shown in the following figure:

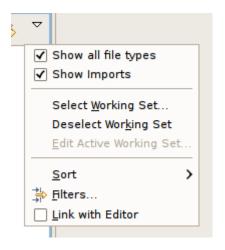


Figure D.8. Additional Actions

If **Show Model Imports** is checked, the imports will be displayed directly under a model resource as shown below.

```
    ✓ PartsVirtual.xmi
    ✓ ▲ import declarations (4)
    ▲ /PartsProject/PartsSourceA.xmi
    ▲ /PartsProject/PartsSourceB.xmi
    ▲ http://www.w3.org/2001/XMLSchema
    ▲ /PartsProject/SampleViews.xmi
    ▲ Package Diagram
    ♦ SupplierInfo
```

Figure D.9. Show Model Imports Action

D.2.1.1. Selection-Based Action Menus

Selecting specific objects in the **Model Explorer** provides a context from which Teiid Designer presents a customized menu of available actions.

Selecting a **view model**, for instance, results in a number of high-level options to manage edit model content, perform various operations and provides quick access to other important actions available in Teiid Designer. These may include specialized actions based on model type.

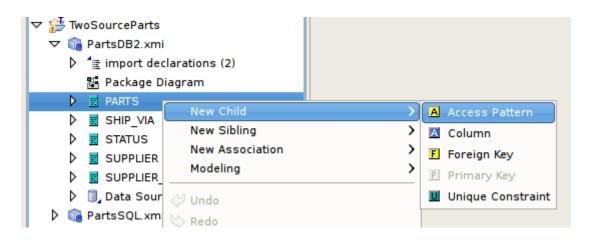


Figure D.10. Sample Context Menu

D.2.2. Outline View

The Outline View is a utility view which provides both at tree view dedicated to a specific model (open in an editor) and a scaled thumbnail diagram representative of the diagram open in the corresponding Diagram Editor.

You can show the Outline View by clicking on its tab. If there is no open editors, the view indicates that Outline is not available. If a Model Editor is open, then the root of the displayed tree will be the model for the editor that is currently in focus in Teiid Designer (tab on top).

D.2.2.1. Outline Tree View

This tree view provides the same basic editing and navigation behavior as the Model Explorer. One additional capability is the drag and drop feature which provides re-ordering and re-parenting of objects in a model.

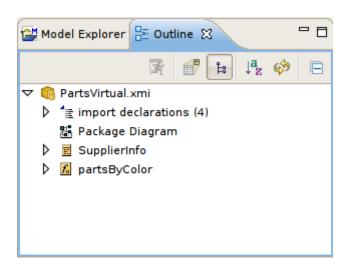


Figure D.11. Outline View

D.2.2.2. Outline Thumbnail View

The Outline View also offers you a way to view a thumbnail sketch of your diagram regardless of its size. To view this diagram thumbnail from the Outline panel, click the Diagram Overview button

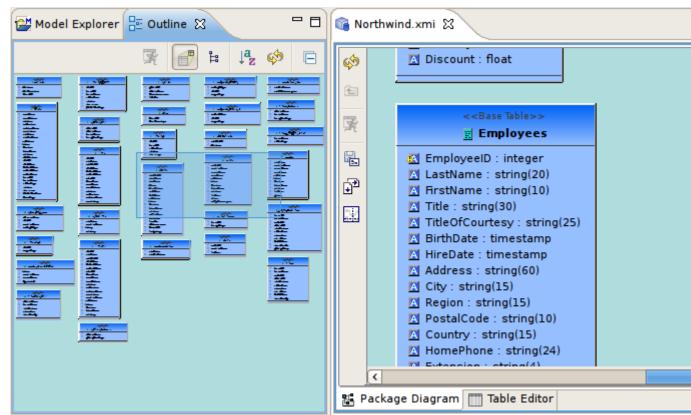
at the top of the view. The diagram overview displays in the Outline View.

🚰 Model E	xplorer	🗄 Outline	z	- 8
		R	¦a ↓a	ø 🖻

Figure D.12. Outline View

The view contains a thumbnail of your entire diagram. The shaded portion represents the portion visible in the Diagram Editor view.

To move to a specific portion of your diagram, click the shaded area and drag to the position you want displayed in the Diagram Editor view.



D.2.3. Server View

The Server View provides a means to display and manage server instances and their contents within the Eclipse environment. Since Teild is installed as part of a JBoss server, its contents is displayed as part of its JBoss parent.

To show the Server View click "Window > Show View > Other..." to display the Show View dialog. Choose "Servers > Server" view and click OK.

	 _	· ·	-	P		-

Figure D.13. Server View

To create your Teiid instance:

- Select the New... action in the Server view and this will launch the JBoss Server wizard. Configuring the JBoss Server instance (with Teiid installed) will enable connection to the Teiid Server.
- In the New Server wizard. Select the JBoss AS 7.1.1 server type under the JBoss Community category and click Next>.

New Server	-
Define a New Server	
Choose the type of server to cr	reate
	Download additional server adapters
Select the server type:	
type filter text	
JBoss AS 5.1	
JBoss AS 6.x	
JBoss AS 7.0	
JBoss AS 7.1	
JBoss Enterprise Middl	leware
JBoss Application Server 7.1	
Server's host name:	localhost
Server name:	JBoss 7.1 Runtime Server
?	< Back Next > Cancel Finish

Figure D.14. New Server Dialog

• On the **JBoss Runtime** page, click the top **Browse..** button to select the installation folder of your **JBoss AS 7.1** server.

😣 🔲 New Server	
JBoss Runtime	•**
JBoss Application Server 7.1	BOSS
A JBoss Server runtime references a JBoss i It can be used to set up classpaths for projec as well as by a "server" which will be able to	cts which depend on this runtime,
Name	
JBoss 7.1 Runtime	
Home Directory	Download and install runtime
fond/Testing/Servers/AS_7.1.1_Teiid_8.2	_Beta2_10_19_2012 Browse
JRF	
Default JRE for JavaSE-1.6	‡ JRE
Configuration file: standalone-teiid.xml	Browse
< Back Next >	Cancel Finish

Figure D.15. JBoss Runtime Definition

• Then click the bottom **Browse...** button to select the **standalone-teiid.xml** configuration file located under the **standalone/configuration/** folder on your file system. Then click **Finish** to return the the **New Teiid Instance** dialog.

Name v	Size	Modified
standalone_xml_history		11:14
application-roles.properties	634 bytes	03/10/2012
application-users.properties	812 bytes	03/10/2012
logging.properties	2.0 KB	03/10/2012
mgmt-users.properties	836 bytes	03/10/2012
🧊 standalone.xml	14.6 KB	11:11
🐻 standalone-full.xml	20.3 KB	03/10/2012
💿 standalone-full-ha.xml	26.4 KB	03/10/2012
🐻 standalone-ha.xml	19.9 KB	03/10/2012
🐻 standalone-teiid.xml	23.1 KB	11:14
teiid-security-roles.properties	106 bytes	10/19/2012
teiid-security-users.properties	95 bytes	10/19/2012

Figure D.16. Teiid Configuration File Selection

• Click **Finish** and your new Teiid server configuration will be opened in the **JBoss / Teiid Editor** for viewing. In this editor you can test both Teiid admin and JDBC connections.

EAP6.1+Teiid8.7FIN	VAL 🖾								
Teiid Instance									
▼ Overview									
10-10-10-10-10-10-10-10-10-10-10-10-10-1	5 4 5 4 · T - ¹¹ / 9 7 5								
JBoss Server	EAP6.1+Teiid8.7FI	INAL							
Host	localhost								
Teiid Runtime Version	8.7.0								
👻 Administration Cor	✓ Administration Connection								
Administration is perf	ormed via the jboss	management config	uration						
Port			9999						
Test Administration C	onnection								
➡ JDBC Connection									
Username									
User name			user						
Password			••••						
Port			31000)					
JDBC connection uses	SSL								
Test JDBC Connection									
<u>researce connection</u>	<u>-</u>								
Overview Deployment	t Teiid Instance								
🛱 Servers 🔀 🛐 P	roblems 📮 Consc	ole 🔲 SQL Results	🐑 Error Log	🖉 Teiid Execution Plan					
				Ē	🤣 🏇		ý= 🗖	p p	
DV6_V4 [Stopped]	1							-	
EAP6.1+Teiid8.6									
EAP6.1+Teiid8.76		hronized]							
X XML Configurat									
Filesets									
🕨 🗁 Server Details									
🔻 🗁 Teiid Instance (
▼ 🙊 mm://localho									
Data Source									
Translators	5								
VDBs									
🕨 🏭 JBoss 5.1 Runtim	e Server [Stopped]								

Actions available in this view include:

A

Teiid Server Properties - View and edit properties of an existing Teiid instance

ଂ 🦚

- Reconnect and refresh contents of the selected Teiid instance

• 📴

Execute VDB - Creates a JDBC Teild connection profile and opens the Data Tools Database Development perspective

• 🗙

Undeploy VDB - Removes the selected VDB from the Teiid instance

• 🐳

Create Data Source - Launches the New Data Source wizard

• 🗙

Delete Data Source - Removes the selected Data Source from the Teiid instance



Note

Once the server is started, the **Server** view will now display the data source, translator and VDB content for your running Teiid server as shown below.

🖭 Problems 🥺 Error Log	👭 Servers 🛛	📮 Console	SQL Results	Teiid Execution Plan	
				E	3 (
👻 🤀 localhost 7.1 server [Started, Synch	nronized]			
XML Configuration					
> 🗎 Filesets					
> 🗁 Server Details					
🗸 🗁 Teiid Server Configu	iration				
🗸 🙀 mm://goshawk:99	999				
> 🗁 Data Sources					
> 🗁 Translators					
> 🗁 VDB s					

Figure D.18. Teiid Contents in Server View

D.2.4. Connections View

The **Connections View** allows editing and management of your local connection profiles as well as deployed data source configurations in one view.

The figure below shows 2 panels labeled **Local and Deployed Connections** and **Default Server**. The Connections panel contains a tree view containing locally defined connection profile definitions and a list of data sources configurations that are deployed on your default server.

The toolbar on the left contains create, delete and edit actions as well as a refresh button.

Connections 🕄 🦂 Servers		
Local and Deployed Connections		
Image: Second symplect of the second symplech symplech second symplech second symplech second symplech second symplech symplech symplech second symplech second symplech sympl		
✓ Default Server	¢0	
Name 🙀 JDV 6.4 ER1		
Teiid Version 🕋 8.12.11		

Figure D.19. Connections View

In addition to managing your connections, you can also create new source models (i.e. import) from selecting an existing profile or deployed data source and launching the corresponding import wizard via a **Generate Source Model** action (see below).

Local and	Deployed Connection	s	
-	▼ 🕲 Local Profiles ▶ 🗁 JDBC		
*	▶ 🗁 ODA ▶ 彦 Teiid		
*	▼ 🎕 Deployed 📋 ExampleDS		
	📋 PartsSS 🔄		
NA	Books	% 1	Create Data Source
v	ProductsSS	8	Generate Source Model
		*	Edit
		-	Delete

Figure D.20. Available Actions

The **Default Server** section includes a useful toolbar that includes start, and stop actions. Note that the default Designer perspective includes the standard **Servers** view as a tab behind the **Connections View**.

D.2.5. Properties View

The **Properties View** provides editing capabilities for the currently selected object in Teild Designer. The selection provided by whichever view or editor is currently in focus will determine the its contents.

To edit a property, click a cell in the Value column. As in the **Table Editor**, each cell provides a UI editor specific to the property type.

🔲 Properties 🕱 📄 De	scription 🔚 🛱 🖾 🖻 🗖 🗄	3
Property	Value	-
▼ Info		
Object URI		
Searchability	E SEARCHABLE	
Selectable	⊡⁄k true	
Updateable	🖙 true	
∀ Misc		-
Distinct Value Count	L11 -1	
Name	LA SUPPLIER_NAME	
Name In Source	A SUPPLIER_NAME	
Null Value Count	L11 -1	
⊽ Type		
Datatype	string (Path=http://www.w3.org/2001/XMLSc	
Native Type	🖪 varchar	
Nullable	E NULLABLE	
57 Tine Detail	i b	1

Figure D.21. Properties View

If the model for the object being edited is not open in an editor, a dialog may appear confirming the attempt to modify the model and asking the user to confirm or cancel. This dialog can be prevented by checking the preference Always open editor without prompting. You can re-set/uncheck this property via Teild Designer's main preference page.



Figure D.22. Open Model Editor Dialog

Properties can also be edited via a right-click menu presented below.

	Value		
nt	5 1		
	LA SUPPLIER_NA		
	K SUPPLIER_NA	Cut	:ka
	<u>u.</u> -1	Eopy	
		🗎 <u>P</u> aste	lei
	string (Path=	🗑 <u>D</u> elete	
	🖪 varchar		_
	E NULLABLE	📄 Select <u>A</u> ll	ge
		Input <u>M</u> ethods	> ^{ior}
	🏎 false	Insert Unicode Control Character	> Int
	🙀 true		Jnł

Figure D.23. Open Model Editor Dialog

The **Properties** toolbar contains the following actions:

• 🗄

Show Categories - toggles between categorized properties and flat alphabetical properties list.

• 🚔

Show Advanced Properties - shows/hide advanced properties (if available).

•

Restore Default Value - for a selected property, this action will reset the current to a default value (if available).

D.2.6. Description View

The **Description View** provides a means to display and edit (add, change or remove) a description for any model or model object. To show the Description View click **Window > Show View >**

Other... to display the *Figure D.6, "Eclipse Show View Dialog"* dialog. Choose **Teiid Designer** > **Description** view and hit **OK**.

Properties	Description	×		ľ		- 8
			ious stock attribute siness type, etc	valu	es	6
Primary key k	S PK_PD_INSTR_	ID				

Figure D.24. Description View

You can click the edit description action in the toolbar or right-click select "Edit" in the context menu to bring up the Edit Description dialog. edit actions.

Properties	Description	×	Ľ		- 0
			us stock attribute valu iness type, etc	ies	
Primary key k	s PK_PD_INSTR_	ID			
			<u>C</u> opy		
			Select <u>A</u> ll		
			🔗 Find		_
			🛃 Edit		
			🔒 Clear		
			Input Methods	>	
BaseTa	ble: Products/PR	ODU	CIDATA	-	

Figure D.25. Description View Context Menu

Edit Description for PRODUCTDATA
Description
PRODUCTDATA table contains various stock attribute values including basic name, primary business type, etc
Primary key ks PK_PD_INSTR_ID
Cancel OK

Figure D.26. Edit Description Dialog

D.2.7. Problems View

The **Problems View** displays validation errors, warnings, or information associated with a resource contained in open projects within your workspace.

ems 🖾				
warning, 0 others				
ion	Resource	Path	Location	Тур
rors (1 item)				
SQL statement is empty.	Books.xmi	/BooksProject	bookCollection	Pro
rnings (1 item)				
The type referenced by column book_edition must be marked as an enterprise datatype	Books.xmi	/BooksProject	bookCollection/t	Pro

Figure D.27. Problems View

By default, the **Problems View** is included in Teiid Designer perspective. If the **Problems View** is not showing in the current perspective click **Window > Show View > Other > Teiid Designer > Problems**.

There are 5 columns in the view's table which include:

1. Description - A description of the problem preceded by a severity icon (i.e., error



- 2. **Resource** The name of the resource.
- 3. Path The project name.
- 4. Location The object within the resource that has a validation error.
- 5. type Type of validation item.

D.2.7.1. Toolbar Menu

Click	the	upside-down	triangle
\bigtriangledown			

icon to open the View Menu icon to see various options including sorting, filtering, displayed columns and much more.

D.2.7.2. Context Menu

Additional actions are available by selecting a problem and right-click to open a context menu.

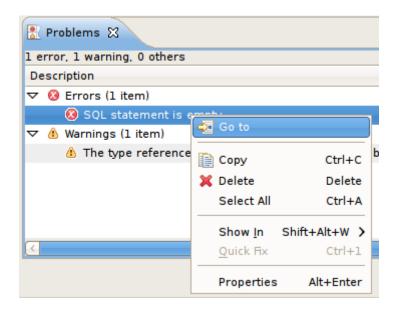


Figure D.28. Problems View Context Menu

- Go To will open the appropriate editor and select the affected/referenced object.
- Show In Navigator Opens the Basic > Navigator view (if not open) and expands file system tree and reveals applicable resource.
- Copy Copies the problem information to the system clipboard.
- **Paste** Pastes the problem information located in the system clipboard (if applicable) into the curor location for a specified text editor.
- **Delete** Deletes the selected problem rows (if applicable).
- Select All selects all problems in the table.
- Quick Fix (Not yet implemented in Teiid Designer).
- Properties displays a dialog containing additional information.

D.2.8. Search Results View

Below is an example set of search results. The view contains rows representing matches for your search parameters. You can double-click a entry and the object will be opened and selected in an editor and/or the Model Explorer if applicable.

🖹 Problems 🔗 Search 🕱	Ŷ	Û	ж	*	Ŧ	Ē	>>		! ?~	
feature=Name, pattern=*Part* - 26 matches										
Found 8 matches - /PartsProject/PartsSourceA.xmi										
Found 1 match - /PartsProject/SampleViews.xmi										
▽ 🕞 Found 7 matches - /PartsProject/PartsVirtual.xmi										
PART_NAME : PART_NAME - http://www.metamatrix.com	m/meta	mode	els/Re	lation	al#//C	olum	n			
PART_COLOR : PART_COLOR - http://www.metamatrix.	com/m	etam	odels/	/Relati	ional#	t//Colu	umn			
PART_ID : PART_ID - http://www.metamatrix.com/meta	models	s/Rela	tional	#//Co	lumn					
PART_ID : PART_ID - http://www.metamatrix.com/meta	models	s/Rela	tional	#//Co	lumn					
PART_WEIGHT : PART_WEIGHT - http://www.metamatri	x.com/r	netar	nodel	s/Rela	tional	#//Co	lumn			
f PartsVirtual : PartsVirtual - http://www.metamatrix.co	om/met	amod	lels/C	ore#/	/Mode	lAnno	tation			
🔏 partsByColor : partsByColor - http://www.metamatrix		netan	nodels	s/Rela	tional	#//Pro	ocedui	re		
Found 10 matches - /PartsProject/PartsSourceB.xmi										
K										

Figure D.29. Search Results View

The toolbar actions for the Search Results view are:

• 🕂

Show Next Match - Navigates down one row in the view.

• 🔶

Show Previous Match - Navigates up one row in the view.

• 🕱

Remove Selected Matches - Removes selected results from the view.

• 🍇

Remove All Matches - Clears the view.

• 🔏

Search - Launches the MoTeiid Designerearch Dialog.

• 🐶

Previous Search Results - Select previous search results from history.

You can also perform some of these actions via the right-click menu:

	rch ⊠ n=*Part* - 26 matches s - /PartsProject/PartsSourceA.	×mi
	Show In S	hift+Alt+W > ationa Ctrl+.
■ PARTS : http * PartsSource ▲ PART_WEIGH ■ SUPPLIER_PA	Previous Match Expand All Copy Remove Match	Ctrl+. al# al# Ctrl+C ela s/F
 General Found 1 match General Found 7 match General Found 10 match 	Remove Selected Matches	Delete
<	Search Again	F5

Figure D.30. Search Results Context Menu

D.2.9. Datatype Hierarchy View

To open **Teiid Designer's Datatype Hierarchy** view, select the main menu's **Window > Show View > Other...** and select the **Teiid Designer > Datatypes** view in the dialog.

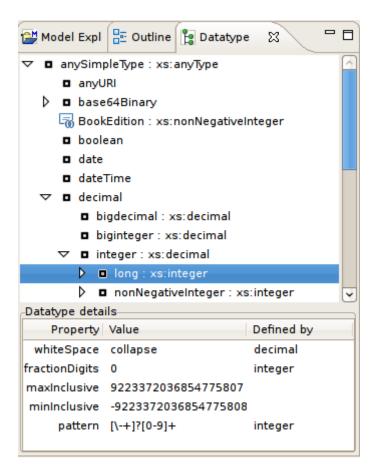


Figure D.31. Datatype Hierarchy View

D.2.10. Teiid Model Classes View

The Model Classes View provides a hiearchical EMF-centric view of the various metamodel classes available within Teiid Designer. This view is primarily for informational purposes, but can be used as a reference if creating relationships or searching your workspace for specific metamodel constructs.

الار	Teiid	Model Classes	ß	🚰 Model I	Explorer	E Outline	Project Explorer		
<u>کې</u>	😫 ht	tp://www.eclip	se.or	g/emf/2002/	/Ecore				
▶ ∰	😫 ht	tp://www.eclip	se.or	g/emf/2002/	/Mapping				
▶ ∰	😫 ht	tp://www.eclip	se.or	g/xsd/2002/X	XSD				
▶ ∰	😫 ht	tp://www.meta	amatri	x.com/meta	models/0	lompare			
▶ ∰	😫 ht	tp://www.meta	amatri	x.com/meta	models/0	lore			
▶ ∰	😫 ht	tp://www.meta	amatri	x.com/meta	models/D	ependency)			
▶ ∰	😫 ht	tp://www.meta	amatri	x.com/meta	models/D	Diagram			
▶ ∰	😫 ht	tp://www.meta	amatri	x.com/meta	models/E	xtension			
▶ ∰	😫 ht	tp://www.meta	amatri	x.com/meta	models/J	DBC			
▶ ∰	😫 ht	tp://www.meta	amatri	x.com/meta	models/N	/letaMatrixFur	nction		
▼ ∰	🐉 ht	tp://www.meta	amatri	x.com/meta	models/F	Relational			
⊳		AccessPattern	-> Rel	ationalEntity	/ [org.teii	d.designer.me	etamodels.relational.A	ccessPat	tern]
⊳		BaseTable -> Ta	able [o	org.teiid.des	igner.me	tamodels.rela	tional.BaseTable]		
⊳		Catalog -> Rela	tional	Entity [org.te	eiid.desig	gner.metamod	lels.relational.Catalog]		
⊳		Column -> Rela	tional	Entity [org.te	eiid.desi	gner.metamod	lels.relational.Column]		
⊳		ColumnSet -> R	elatio	nalEntity [or	g.teiid.d	esigner.metar	nodels.relational.Colu	mnSet]	
⊳	Ŷ	DirectionKind							
⊳		ForeignKey -> I	Relati	onship [org.t	eiid.desi	gner.metamo	dels.relational.Foreign	Key]	
⊳		Index -> Relatio	onalEr	ntity [org.teii	d.design	er.metamodel	ls.relational.Index]		
⊳		LogicalRelation	nship	Relationsh	ip [org.te	eiid.designer.r	metamodels.relational	LogicalR	elatior
⊳		LogicalRelation	nshipE	End -> Relatio	onalEntity	/ [org.teiid.de	signer.metamodels.re	lational.L	ogicalI
	<i>(</i>)	and the second second							

Figure D.32. Datatype Hierarchy View

D.2.11. System Catalog View

To open **Teiid Designer's System Catalog** view, select the main menu's **Window > Show View > Other...** and select the **Teiid Designer > System Catalog** view in the dialog..

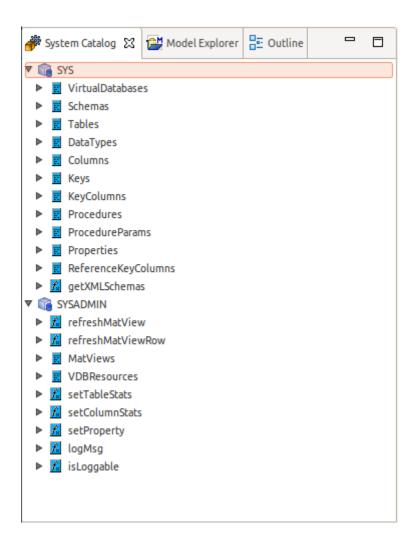


Figure D.33. System Catalog View

D.2.12. SQL Reserved Words View

To open **Teiid Designer's SQL Reserved Words** view, select the main menu's **Window > Show View > Other...** and select the **Teiid Designer > SQL Reserved Words** view in the dialog.

ilter Text (? = any character, * = any String):	
Se	
1atching Reserved Words:	
SEARCH	
SECOND	
SELECT	
SENSITIVE	
SESSION_USER	
SET	

Figure D.34. SQL Reserved Words View

You can also display the view by selecting the the main menu's **Metadata > Show SQL Reserved Words** action as shown below.

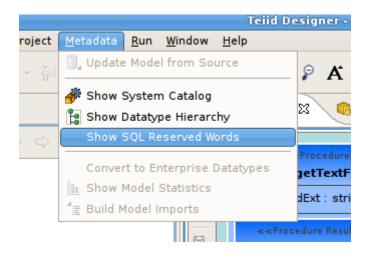


Figure D.35. SQL Reserved Words Action

D.2.13. Model Extension Definition Registry View (MED Registry View)

To open **Teiid Designer's MED Registry** view, select the main menu's **Window > Show View** > **Other...** and select the **Teiid Designer > Model Extension Registry** view in the dialog. You can also open the MED Registry view from the MED Editor - by selecting the MED Editor toolbar action in the right corner of the MED Editor header section.

The **Model Extension Registry** view shows the currently registered MEDs. Registered MEDs can be applied to models in the workspace (see *Section 6.4, "Managing Model Object Extensions*"). The Model Extension Registry view looks like this:

Туре	Namespace Prefix	Namespace URI	Model Class	Supported Model Types	Versio
mported	accumulo	http://www.teiid.org/translator/accumulo/	Relational	Source Model	1
mported	cassandra	http://www.teiid.org/translator/cassandra	Relational	Source Model	1
Built-In	core	http://www.teiid.org/ext/core/2012	Core	<all supported=""></all>	3
mported	couchbase	http://www.teiid.org/translator/couchbase	Relational	Source Model	1
mported	excel	http://www.teiid.org/translator/excel/201	Relational	Source Model	1
Built-In	ext-custom	http://www.jboss.org/teiiddesigner/ext/d	Relational	<all supported=""></all>	1
Built-In	function	http://www.teiid.org/ext/function/2012	Function (Deprecated)	User Defined Function	2
Built-In	infinispan	http://www.teiid.org/translator/object/20	Relational	Source Model	1
mported	infinispan-hotrod	http://www.teiid.org/translator/infinispan	Relational	Source Model	1
mported	jpa2	http://www.teiid.org/translator/jpa/2014	Relational	Source Model	1
mported	mongodb	http://www.teiid.org/translator/mongodb/	Relational	Source Model	1
.ocal	my_custom_med	http://www.teiid.org/translator/custom/20	Relational	Source Model	1
mported	odata	http://www.jboss.org/teiiddesigner/ext/o	Relational	Source Model	1
mported	odata4	http://www.jboss.org/teiiddesigner/ext/o	Relational	Source Model	1
mported	osisoft-pi	http://www.teiid.org/ext/relational/2012	Relational	Source Model	1
Built-In	relational	http://www.teiid.org/ext/relational/2012	Relational	Source Model ,View Mode	8
Built-In	REST	http://teiid.org/rest	Relational	View Model	3
Built-In	salesforce	http://www.teiid.org/translator/salesforce	Relational	Source Model	1
moorted	salesforce-34	http://www.teiid.org/translator/salesforce	Relational	Source Model	1

Figure D.36. MED Registry View

For each registered MED, the namespace prefix, namespace URI, extended model class, version, and description is shown. In addition, a flag indicating if the MED is built-in is shown. The Model Extension Registry view has toolbar actions that register a workspace MED file, unregister a userdefined MED, copy a registered MED to the workspace, or view the MED. All of these actions are also available via a context menu.

A MED registry keeps track of all the MEDs that are registered in a workspace. Only registered MEDs can be used to extend a model. There are 2 different types of MEDs stored in the registry:

- Built-In MED these are registered during Designer installation. These MEDs cannot be updated or unregistered by the user.
- User-Defined MED these are created by the user. These MEDs can be updated, registered, and unregistered by the user.



(-

Note

When a workspace MED is registered it can be deleted from the workspace if desired. The registry keeps its own copy. And a registered MED can always be

copied back to the workspace by using the appropriate toolbar or context menu action.

D.2.14. Guides View

To open **Teiid Designer's Guides** view, select the main menu's **Window > Show View > Other...** and select the **Teiid Designer > Guides** view in the dialog.

The **Guides** view provides assistance for many common modeling tasks. The view includes categorized Modeling Actions and also links to 'Cheat Sheets' for common processes. 'Cheat Sheets' are an eclipse concept for which Teiid Designer has provided contributions (see *Section D.2.16, "Cheat Sheets View"*). The Guides view is shown below:

🍓 Guides 🛱 🎆 Status	' 🗆
Modeling Actions 🖻	Â
- Action Sets	
Execute selected action	
Model JDBC Source	\$
🚔 Create Teiid Model Project	
🐟 Create JDBC connection	
Create source model for JDBC data source	
🔀 Preview Data	
💦 Create VDB	
🚱 Execute VDB	
Description	
<no action="" selected=""></no>	=
⑦ Create Model from JDBC Source	
⑦ Create Model from Flat File Source	
⑦ Consume a SOAP Web Service	
Create Model from XML Local File Source	
⑦ Create Model from XML Remote Source	
<u>Create multi-source VDB</u>	
⑦ Create and test a VDB	

Figure D.37. Guides View

The upper 'Action Sets' section provides categorized sets of actions. Select the desired category in the dropdown, then the related actions for the selected category are displayed in the list below it. Execute an action by clicking the 'Execute selected action' link or double-clicking on the action.

The lower 'Cheat Sheets' section provides a list of available 'Cheat Sheet' links, which will launch the appropriate Cheat Sheet to guide you step-by-step through the selected process.

D.2.15. Status View

To open **Teiid Designer's Status** view, select the main menu's **Window > Show View > Other...** and select the **Teiid Designer > Status** view in the dialog.

The **Status** view provides a quick overview status of the selected project. A sample Status view for a project is shown below:

🍓 Guides	🎆 Status ස	
Project	Status	Enable Cnange Project
Project:	TestProj	
	Source Connections	
	Sources	
<u>4</u>	XML Schema	
×	Views	
	VDBs	
*	Model validation (OFF)	
	Test	

Figure D.38. Status View

The status view is broken down into common project areas:

- Source Connections all Source Connections are fully defined.
- Sources Source Models exist.
- XML Schema XML Schemas exist.
- Views View Models exist.
- VDBs VDBs exist and are deployable.
- Model Validation (Status) all Models pass validation.
- Test all defined VDBs pass validation.

The status of each area is denoted by an icon: A green check indicates OK, a red 'x' indicates errors and a 'warning' icon indicates potential problems. The project can be changed by selecting the 'Change Project' button.

D.2.16. Cheat Sheets View

To open **Cheat Sheets** view, select the main menu's **Window > Show View > Other...** and select the **Help > Cheat Sheets** view in the dialog.

The **Cheat Sheets** view is a standard Eclipse Help concept. Cheat Sheets provide step-bystep assistance for common process workflows. Teiid Designer has contributed to the Eclipse help framework to provide assistance for many common modeling tasks. The Guides View (see *Section D.2.14, "Guides View*") provides links to these Cheat Sheets, as previously described. A sample Cheat Sheet is shown below:



Figure D.39. Cheat Sheet Sample

D.3. Editors

Editors are the UI components designed to assist editing your models and to maintain the state for a given model or resource in your workspace. When editing a model, the model will be opened in a **Model Editor**. Editing a property value, for instance, will require an open editor prior to actually changing the property.

Any number of editors can be open at once, but only one can be active at a time. The main menu bar and toolbar for Teiid Designer may contain operations that are applicable to the active editor (and removed when editor becomes inactive).

Tabs in the editor area indicate the names of models that are currently open for editing. An asterisk (*) indicates that an editor has unsaved changes.



Figure D.40. Editor Tabs

By default, editors are stacked in the editors area, but you can choose to tile them vertically, and or horizontally in order to view multiple models simultaneously.

🏻 PartsSourceA.xmi 🛛					😭 Pa	rtsVirtual.xmi	щ
	: string(255) R : string(30)	SHIPPER_ID	ME : string(Table>> IER_PARTS) : string(10 ring(50) short			A SUPPL A PART A QUAN A SHIPP A SUPPL A SUPPL A SUPPL	<base table=""/> SupplierIn IER_ID : string ID : string(50) TITY : short ER_ID : short IER_NAME : st IER_STATUS : IER_CITY : str IER_STATE : st
Package Diagram	Table Editor				E P	∢ ackage Diagrar	m Table E
Models							
Model	Path	Synchronized?	Visible?	Source Name		Translator	JNDI Name
PartsSourceA.xmi	/PartsProject	√	Z	PartsSourceA		db2	PartsSourc
PartsSourceB.xmi	/PartsProject /PartsProject	N N	V	PartsSou	rceB	sqlserver	PartsSourc
Other Files						VDB Data Ro	oles
File Path Sy	nchronized? D	escription			dd	Data Role	Descriptio
	Syr	nchronize All					

Figure D.41. Viewing Multiple Editors

The Teiid Designer provides main editor views for XMI models and VDBs.

The Model Editor contains sub-editors which provide different views of the data or parts of data within an XMI model. These sub-editors, specific to model types are listed below.

- Diagram Editor All models except XML Schema models.
- Table Editor All models.
- Simple Datatypes Editor XML Schema models only.
- Semantics Editor XML Schema models only.
- Source Editor XML Schema models only.

The *VDB Editor* is a single page editor containing panels for editing description, model contents and data roles.

In addition to general Editors for models, there are detailed editors designed for editing specific model object types. These "object" editors include:

- **Transformation Editor** Manages Transformation SQL for Relational View Base Tables, Procedures and XML Web Service Operations.
- Choice Editor Manages properties and criteria for XML choice elements in XML Document View models.
- Input Editor Manages Input Set parameters used between Mapping Classes in XML Document View models.
- Recursion Editor Manages recursion properties for recursive XML Elements in XML Document View models.
- Operation Editor Manages SQL and Input Variables for Web Service Operations.

D.3.1. Model Editor

The Model Editor is comprised of sub-editors which provide multiple views of your data. The Model Editor provides the primary means for editing model content, the Diagram Editor provides a graphical means and a graphical while the Table Editor provides spreadsheet-like editing capabilities. This section describes these various sub-editors.

D.3.1.1. Model Editor

The Model Editor includes a detailed summary of the contents of a model and provides a centralized panel for editing the model contents and properties.

The editor page provides for editing the model contents via:

• vertical toolbar containing the following operations



New Table

· 🔁

New View

• 📆

New Procedure or Function

• 📆

New View Procedure or Function

• 宜

New Index

• 🕅

Generate Data Service - Generate and export a deployable *-vdb.xml as a simple data service for the selected table or view. (Also available when view or table is selected via the **Modeling.... > Generate Data Service** action)

- Model Contents tree view (same information shown in Model Explorer tree)
 - Simplified action context menus compared to **Model Explorer** and limited to applicable model object actions.
- Actions panel
 - Contains applicable operations you can perform for the reference model
 - Also available in **Model Explorer** tree view via model selection and context menu **Modeling...** action category
- Properties panel
 - Provides the same information that can still be edited in Eclipse's Properties View
 - · Properties are driven based on selection in tree view in this editor
- Description panel
 - Provides the same description that can still be edited in Designer's Description View
 - · Description editing is driven by selection in tree view in this editor

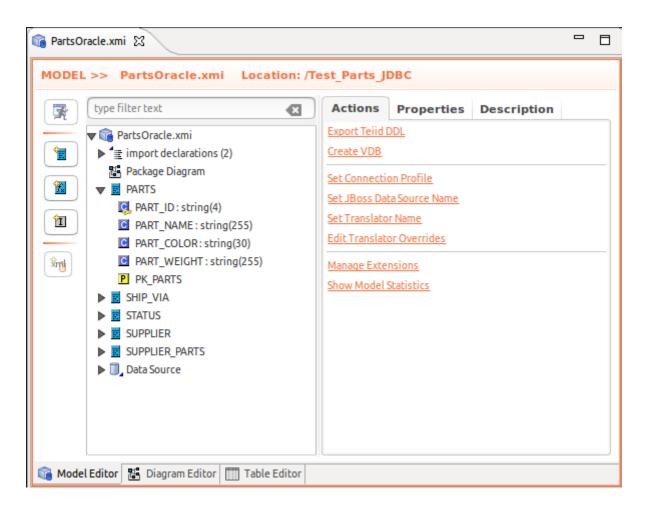


Figure D.42. Model Editor Tab

▼ 📬 PartsOracle11.xmi	Actions Properties D	escription
import declarations (2)	Property	Value
Package Diagram		Volue
DIVIDED_ROUTES	Extension	
LAKES	▼ Info	
PARTS	Object URI	
SHIP_VIA	▼ Misc	
STATUS	Cardinality	-1
▶ E SUPPLIER	Logical Relationships	
SUPPLIER_PARTS	Materialized	🔤 false
Data Source	Materialized Table	UE .
	Name	LA STATUS
	Name In Source	"PARTSSUPPLIER"."STATUS
	Supports Update	🙀 true
	System	<u>⊾</u> false

Figure D.43. Model Editor Tab

D.3.1.2. Diagram Editor

The Diagram Editor provides a graphical view of the a set of model components and their relationships.

Several types of diagrams are available depending on model type. They include:

• 蹯

Package Diagram

Custom Diagram

• ⊮{

Transformation Diagram

•

Mapping Diagram

• ⊮

Mapping Transformation Diagram

You can customize various diagram visual properties via Diagram Preferences.

Each diagram provides actions via the Main toolbar, diagram toolbar and selection-based context menus. These actions will be discussed below in detail for each diagram type.

When a **Diagram Editor** is in focus, a set of common diagram actions is added to the application's main toolbar.



Figure D.44. Main Toolbar Diagram Actions

· The actions include:

' 🗜 Zoom In



Zoom to Level

 Zoom Out

• A Increase Font Size

A Decrease Font Size

• Perform Diagram Layout

D.3.1.2.1. Package Diagram

The Package Diagram provides a graphical view of the contents of a model container, be it the model itself, a relational catalog or schema.

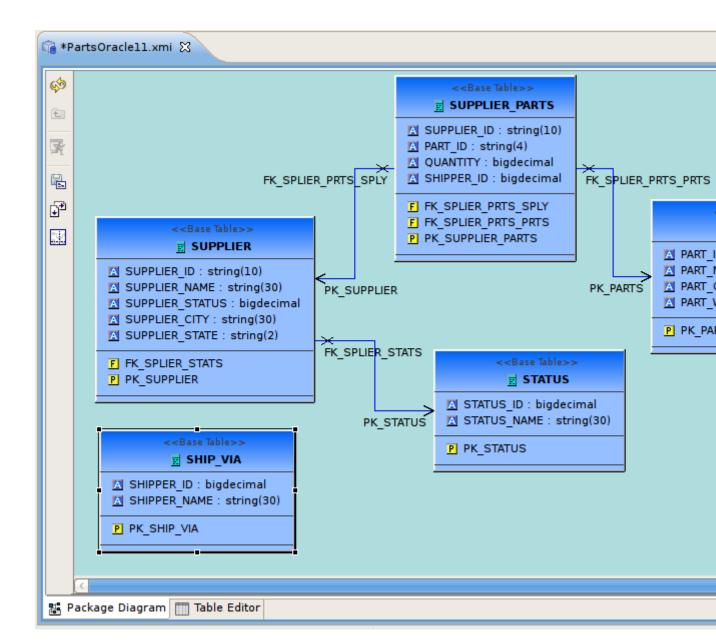


Figure D.45. Package Diagram Example

- Package Diagram toolbar actions include:
 - 🏟

Refresh Diagram - Re-draws diagram.

ં 🖻

Show Parent Diagram - Navigates to diagram for parent object (if available).

• 록

Preview Data - Executes a simple preview query (SELECT * FROM).

• 🖫

Save Diagram as Image - Save the diagram image to file in JPG or BMP format.



Show/Hide Page Grid - Show current page boundaries as grid in diagram.

Context menus provide a flexible means to edit model data, especially from Package Diagrams. Each Package Diagram represents the contents of some container (i.e. Model, Category, Schema, etc...), so New Child, New Sibling and New Association actions are almost always available in addition to standard Edit actions (Delete, Cut, Copy, Paste, Rename, Clone).

A sample context menu for a relational base table is shown below.

New Child	>
New Sibling	>
New Association	>
Modeling	>
💛 Undo Delete multiple objects	
崎 Redo	
of Cut	Ctrl+X
📑 Сору	Ctrl+C
👕 Paste	Ctrl+V
Clone	
Copy Name	>
💢 Delete	
Rename	
Edit	
Link Types	>
🟟 Refresh Diagram	
🔁 Show Parent Diagram	
🖳 Save Diagram To File	
 ∳ [⊉] Page Setup	
Show Page Grid	
Input Methods	>

Figure D.46. Package Diagram Context Menu

D.3.1.2.2. Custom Diagram

The **Custom Diagram** represents a view of user-defined model objects. Unlike **Package Diagrams**, **Custom Diagrams** can contain objects that are not only unrelated, but can be from different containers and even models.

📬 Pa	artsOra	cle11.xmi 🔀 🕼 PartsView.xmi 🖇	3	
69				
£		< <base table=""/> >		< <base table=""/> >
R		Partsoracie11		SUPPLIER_PARTS PartsOracle11
Þ		PART_NAME : string(255) PART_COLOR : string(30)	PK_PARTS	SUPPLIER_ID : string(10) PART ID : string(4)
B		PART_WEIGHT : string(255)	FK_SPLIER_PRTS_PRTS	QUANTITY : bigdecimal
43		P PK_PARTS		F FK_SPLIER_PRTS_SPLY
R.		< <base table=""/> >	i	F FK_SPLIER_PRTS P PK_SUPPLIER_PARTS
€ [≠]		PARTS		
		A PART_ID : string(4)		
		PART_NAME : string(255) PART COLOR : string(30)		
		A PART_WEIGHT : string(255)		
		P PK_PARTS		
		· · · · · · · · · · · · · · · · · · ·		
	<			
25 C	ustom	Diagram 🛅 Table Editor		

Figure D.47. Package Diagram Example

- Custom Diagram toolbar actions include:
 - ଂ 🏟

Refresh Diagram - Re-draws diagram.

• 🖻

Show Parent Diagram - Navigates to diagram for parent object (if available).

• 록

Preview Data - Executes a simple preview query (SELECT * FROM).

• 🖪

Add To Diagram - Add objects selected in Model Explorer to diagram.

• 🖪

Remove From Diagram - Removed objects selected in diagram from diagram.

• 🕼

Clear Diagram - Remove all objects from diagram.

• 🖫

Save Diagram as Image - Save the diagram image to file in JPG or BMP format.

ų.	
Ø	

Show/Hide Page Grid - Show current page boundaries as grid in diagram.

Since **Custom Diagrams** do not represent represents the contents of container objects (i.e. Model, Category, Schema, etc...) its **context menus** are limited to adding/removing objects from diagram and basic diagram-related display options.

D.3.1.2.3. Transformation Diagram

The **Transformation Diagram** represents a view of the relationships defined by the source inputs described in a view table's SQL transformation.

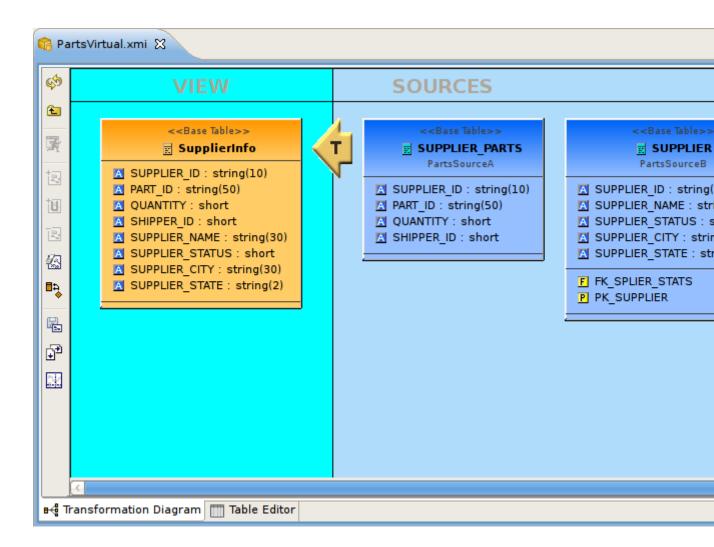


Figure D.48. Transformation Diagram Example

- Transformation Diagram toolbar actions include:
 - ଂ 🏟

Refresh Diagram - Re-draws diagram.

• 🔁

Show Parent Diagram - Navigates to diagram for parent object (if available).

• 록

Preview Data - Executes a simple preview query (SELECT * FROM).

• 🕄

Add Transformation Sources - Add selected sources to transformation.

· 🔃

Add Union Transformation Sources - Add selected sources as union sources.

• 🖪

Remove Transformation Sources - Removed sources selected in diagram from transformation.

• 🧖

Clear Transformation - Remove all sources from transformation.

• 🖬 🎝

Open Transformation Reconciler dialog

•

Save Diagram as Image - Save the diagram image to file in JPG or BMP format.

• 🔛 Ø

Show/Hide Page Grid - Show current page boundaries as grid in diagram.

Context menus for the

D.3.1.2.4. Mapping Diagram

The **Mapping Diagram** represents a view of the mapping between virtual mapping class columns and XML document elements. This mapping defines how source data is transformed from row-based results into XML formatted text.

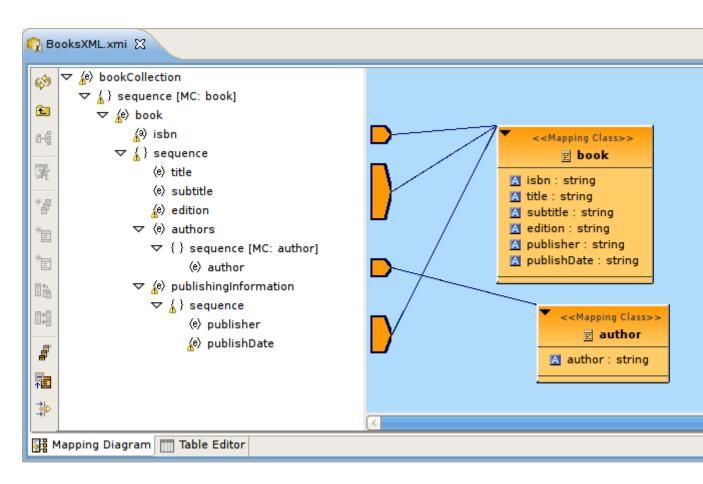


Figure D.49. Mapping Diagram Example

- Mapping Diagram toolbar actions include:
 - ଂ 🏟

Refresh Diagram - Re-draws diagram.

° 🔁

Show Parent Diagram - Navigates to diagram for parent object (if available).

• ₽+<mark>8</mark>

Show Mapping Transformation Diagram - Show detailed mapping transformation diagram for selected mapping class.

• 醒

Preview Data - Executes a simple preview query (SELECT * FROM).

• *₽

Generate Mapping Classes - Generate mapping classes for the selected XML document root element.

•*=

New Mapping Class - Insert new mapping class referenced to the selected XML document element or attribute..

• * 🗉

New Staging Table - Insert new staging table referenced to the selected XML document element or attribute.

• 🛯 🖥

Merge Mapping Classes - Merge selected mapping classes.

• 🔡

Split Mapping Class - Split selected mapping class.

• 🗗

Display All Mapping Classes

•

Show Mapping Class Columns

‡‡⊳ ∔T

Filter Displayed Mapping Classes with Selection

Context menus for **Mapping Diagrams** provide Edit capability to the mapping class in addition to mapping class manipulation actions (i.e. Merge Mapping Classes, Split Mapping Class, etc..)

D.3.1.2.5. Mapping Transformation Diagram

The **Mapping Transformation Diagram** is identical to a Transformation Diagram except for displaying an Input Set and possibly Staging Tables as sources for the Mapping Class's transformation.

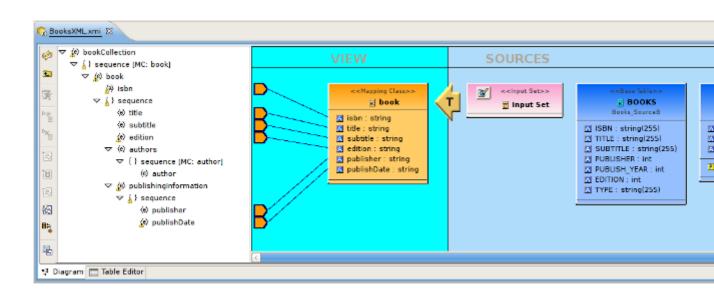


Figure D.50. Mapping Transformation Diagram Example

- Mapping Transformation Diagram toolbar actions include:
 - ଂ 🧒

Refresh Diagram - Re-draws diagram.

' 🛍

Show Parent Diagram - Navigates to diagram for parent object (if available).

• 록

Preview Data - Executes a simple preview query (SELECT * FROM).

• •*

New Mapping Link - Create a mapping link between selected mapping extent (i.e. XML element or attribute) and mapping class column.

• 🛰

Remove Mapping Link - Delete mapping link between selected mapping extent (i.e. XML element or attribute) and mapping class column.

• 🖪

Add Transformation Sources - Add selected sources to transformation.

· 🔃

Add Union Transformation Sources - Add selected sources as union sources.

• 🖪

Remove Transformation Sources - Removed sources selected in diagram from transformation.

• 🐼

Clear Transformation - Remove all sources from transformation.

• 🖬 🛓

Open Transformation Reconciler dialog

• 🖫

Save Diagram as Image - Save the diagram image to file in JPG or BMP format.

Context menus for **Mapping Transformation Diagrams** identical capabilities to the Transformation Diagram with the addition of managing and editing Input Sets.

D.3.1.3. Table Editor

The **Table Editor** provides a table-based object type structured view of the contents of a model. The figure below shows a relational model viewed in the **Table Editor**. Common object types are displayed in individual folders/tables. All base tables, for instance, are shown in one table independent of their parentage.

	oles 🖪 Column						
Location	Name	Name In Source	System	Cardinality	Supports Up	Materialized	Logical Rel
	PARTS	PARTS	false	16	true	false	
	SHIP_VIA	SHIP_VIA	false	3	true	false	
	STATUS	STATUS	false	3	true	false	
	SUPPLIER	SUPPLIER	false	16	true	false	
	SUPPLIER_PA	SUPPLIER_PARTS	false	227	true	false	
C.							

Figure D.51. Table Editor Example

You can customize Table Editor properties via Table Editor Preferences.

These are the primary features of the Table Editor:

- Edit existing properties.
- Add, remove or edit objects, via the main Edit menu and context menu (Cut, Copy, Paste, Clone, Delete, Rename, Insert Rows).
- Paste information from your clipboard into the table.
- Print your tables.

When a **Table Editor** is in focus, the **Insert Table Rows** action

is added to the application's main toolbar.

A few Table Editor actions are contributed to the right-click menu for selected table rows. These actions, described and shown below include:

• 🖺

R

Paste - Paste common spreadsheet data (like Microsoft Excel) to set object properties.

- Table Editor Preferences Change table editor preferences, including customizing visible properties.
- Insert Rows Create multiple new sibling objects.
- ଂ 🏟

Table - Refreshes the contents of the current Table Editor to insure it is in sync with the model.

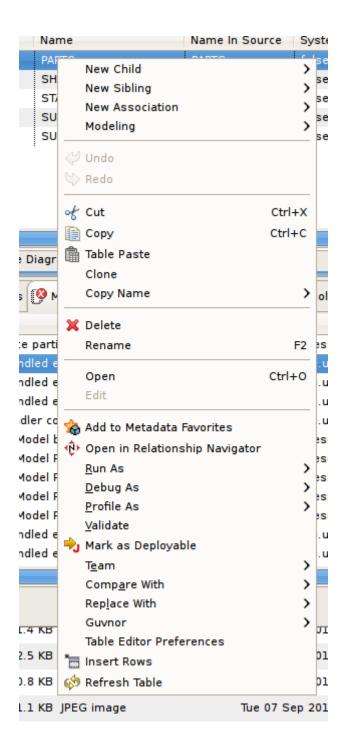


Figure D.52. Table Editor Example

D.3.1.3.1. Editing Properties

You can edit properties for an object by double-clicking a table cell.

For String properties, the table cell will become an in-place text editor field.

Base	Tables 🛛 🖪 Columns 🗍 🖪 F	oreign
ation	Name	Na
	CATEGORIES	C
	CUSTOMERCUSTOMERDEMO) (L
	CUSTOMERDEMOGRAPHICS	CL
	CUSTOMERS	CL
	EMPLOYEES	EN

Figure D.53. Editing String Property

If a property is of a boolean (true or false) type or has multiple, selectable values, a combo box will be displayed to change the value.

Support	s U
true	
true	-
true	
Frite	

Figure D.54. Editing Boolean Value

Currenc
false
false
false
false
KE
false
false

Figure D.55. Editing Multi-Value Property

For multi-valued properties where the available values are dynamic (i.e. can change based on available models or data), a picker-button ("....") will be displayed.

Datatype		1
int : xs:lo	ng	
short : xs	:int	
long : xs:	integer	
string (Pat	h=http://www.w3.org/200	
long : xs:	integer	1
string		
string		
long : xs:	integer	
chring	(h)	

Figure D.56. Editing Multi-Value With Picker

An example of of this type is the relational column datatype property. Editing via the table cell and clicking the "..." button for datatype will display the following dialog.

Datatype	(
int : xs:long	
short : xs:int	
long : xs:integer	
string (Path=http://www.w3.org/200	
long : xs:integer	
string	
string	
long : xs:integer	
string	

Figure D.57. Editing Datatype Values

D.3.1.3.2. Inserting Table Rows

The Insert Rows action provides an additional way to create objects in a model. Insert Rows action performs the same function as Insert Sibling action, but allows you to create multiple children at the same time. All new rows will correspond to an object of the same type as the selected object and be located under the same parent as the selected object.

To Insert Rows in a table:

Step 1: Select a table row to insert rows after.

Step 2: Right-click select "Insert Rows" action or select the Insert Rows action on the main toolbar. The following dialog will be displayed.

8	Insert Table Row	s 🕺
Insert Rows (M 20 rows will be in	laximum of 1000) serted	
Number of Rows:	20	
?		Cancel OK

Figure D.58. Editing String Property

Step 3: Edit the Number of Rows value in the dialog, or use the up/down buttons to change the value.

Step 4: Select OK in dialog.

The desired number of rows (new model objects) will be added after the original selected table row.

D.3.1.4. Simple Datatypes Editor

The Simple Datatype Editor provides a form-based properties view of XML Schema data.

BookDatatypes.xsd			
New Delete	Simple Datatype Editor		
type filter text Clear	e filter text Clear Target Namespace: http://www.metamatrix.com/XMLSchema/DataSets/Books/Bo		
	(change)		
Datatypes in BookDatatypes.xsd:	▼ Identification		
BookType	Name: ISBN <u>(change)</u>		
CommaSeparatedList	Description ISBN, using very simple formatting rules.		
ISBN PublicationDate			
PublicationYear	▼ Inheritance		
	Base datatype: <u>string</u> (http://www.w3.org/2001/XMLSchema) (<u>change)</u> <u>Create subtype of this datatype</u> Prevent Restrictions:		
	Open in Datatype Hierarchy view		
	 Enterprise Information in this section is required in order to use this type in other models. 		
	Enterprise Datatype		
	Runtime type: string Fixed V Use Default Inherited from string		
Hide Built-in Datatypes			
Built-in Datatypes:	▼ Length		
anySimpleType	Length 0 🛱 🗌 Fixed 🗹 Use Default Add Description		
anyURI	Minimum Length 🛛 🗘 🗔 Fixed 🗹 Use Default Add Description		
base64Binary	Maximum Length 0 🖨 🗆 Fixed 🗹 Use Default Add Description		
bigdecimal biginteger	▼ Format		
blob			
boolean	Whitespace Collapse 🗘 🗌 Fixed 🗹 Use Default		
byte char	Enumerations Value Description Add V Use Default		
clob	Edit		
date	Remove		
dateTime	Patterns Value Description Add V Use Default		
decimal double	[0-9.\X]{10.13}		
duration	Remove		
ENTITY	Keniove		
float 🗸			
📄 Simple Datatypes 🥅 Ta	ble Editor 💲 Semantics 🗖 Source		
,			

Figure D.59. Editing String Property

D.3.1.5. Semantic Editor

The Semantic Editor is a tree based editor for XML Schema elements and attributes.

D.3.1.6. Source Editor

The Source Editor is a simple text editor which is aware of XML Schema formatting rules.

D.3.1.7. Model Object Editors

The Model Object Editors represent specialized sub-editors which are available for specific model object types.

· For details, select a specific editor listed below:

 Section 6.3.1, "Transformation Editor" Section 6.3.2, "Input Set Editor (XML)" Section 6.3.3, "Choice Editor (XML)" Section 6.3.4, "Recursion Editor (XML)" Section 6.3.5, "Operation Editor"

D.3.2. VDB Editor

A VDB, or virtual database is a container for components used to integrate data from multiple data sources, so that they can be accessed in a federated manner through a single, uniform API. A VDB contains models, which define the structural characteristics of data sources, views, and Web services. The VDB Editor, provides the means to manage the contents of the VDB as well as its deployable (validation) state.

The VDB Editor, shown below, contains a number of tabs for setting and editing the various allowable properties and contents of your VDB. These tabs enable management of Models, UDF (User Defined Functions) jars and Other Files you wish to include in your VDB, tabs for managing Data Roles, VDB Properties, Description and Translator Overrides.

In our 11.1 release, we've improved usability by adding a couple more options to the toolbar.

- Deploy Performs the same operation as Modeling.... > Deploy action.
- Test Performs the same operation as Modeling.... > Execute VDB action.
- Save as XML Performs the same function as the previous Modeling... > Generate Dynamic VDB action by generating an XML version of the *.vdb archive and saving it to a workspace location or to your local file system.

/ersion 1	Synchronize All Show	Import VDBs Deploy Test Save as	s XML
Models Schemas UDF J	ors Other Files Descriptio	n 🗟 Data Roles Properties User Proper	ties Translator Override
ProductViews.xmi ProductsSQL.xmi	2 2 2		
		Delete	Translotor Nam

Figure D.60. VDB Editor

Note that Deploy and Test actions require you've defined and connected to a default server.

You can manage your VDB contents by using the *Add* or *Remove* models via the buttons below the Models panel.

Set individual model visibility via the *Visibility* checkbox for each model. This provides low level data access security by removing specific models and their metadata contents from schema exposed in GUI tools.

In order for a VDB to be fully queryable the "Source Name", "Translator" and "JNDI Names" must have valid values and represent deployed artifacts on your Teiid server.

If you have Designer runtime plugins installed, you have a default Teiid server instance defined and connected, the translator and JNDI table cells will contain drop-down lists of available translator and JNDI names available on that server.

	ails Source Binding De		
Add	Source Name	Translator Name	JNDI Name
Delete	Customer_Accounts	sqlserver 🛛 👻	Customer_Accounts
Dente		jpa2	
		ldap	
		loopback	
		map-cache	
		metamatrix	-
		madachana	

Figure D.61. Change Translator or Data Source Actions

If you have a default Teiid server instance defined and connected the translator and JNDI table cells will contain drop-down lists of available translator and JNDI names available on that server.

D.3.2.1. Editing Data Roles

Teiid Designer provides a means to create, edit and manage data roles specific to a VDB. Once deployed within a Teiid server with the security option turned on (by default) any query run against this VDB via a Teiid JDBC connection will adhere to the data access permissions defined by the VDB's data roles.

The VDB Editor contains a VDB Data Roles section consisting of a List of current data roles and *New...*, Edit... and Remove action buttons.

Data Roles VDB Description	
Data Role	Description
NoAccountHoldingsAccess	Client cannot access customer account holdings (financial product) info
VirtualOnly	Client users can only access virtual layer. Access requests to physical sources wi
å₀ å⁄ å <mark>×</mark>	

Figure D.62. VDB Data Roles Panel

Clicking *New...* or *Edit...* will launch the *New VDB Data Role* editor dialog. Speicify a unique data role name, add a optional description and modify the individual model element CRUD values by check or unchecking entries in the models section.

😣 🗇 Edit VDB Data Role								
Edit VDB Data Role Select Finish to save data role								
Name CustomerSupport								
Permissions Options Mapped Enterprise Role of Set object level permissions (i.e. tables, procedu Model Row Filter Column Masking Allowed I	res, columns							
Model	Security	Create	Read	Update	Delete	Execute	Alter	
▼ 😭 CustomerAccounts.xmi	•	1	1			-2		
ACCOUNT	ŏ	₫	1			⊻		
ACCOUNTHOLDINGS	0		√			√		
CUSTOMER	9		⊻					
CustomerViews.xmi	۲		⊻					
CustomerViews_WS.xmi	۲		✓					
CustomerViews_Output_View.xmi	۲		\checkmark					
<								•
?						Cancel	Fini	sh

Figure D.63. VDB Data Roles Tab

D.3.2.2. Editing Translator Overrides

Teiid Designer provides a means to create, edit and manage translator override properties specific to a VDB via the Tranlator Overrides tab. A translator override is a set of non-default properties targeted for a specific source model's data source. So each translator override requires a target translator name like "oracle", db2, mysql, etc. and a set of non-default key-value property sets.

The VDB Editor contains a Tranlator Overrides section consisting of a List of current tranlator overrides on the left, a properties editor panel on the right and Add (+) and Remove (-) action buttons on the lower part of the panel.

Data Roles Properties Description	Translator Overrides		
Overridden Translators sqlserver-1	Description:		
	Property	Value	
	Addtional Driver Properties		
	allow-multiple-users	false	
	background-validation	false	
	background-validation-millis		
	blocking-timeout-wait-millis		
۰			

Figure D.64. VDB Translator Overrides Tab

To override a specific translator type, select the add translator action (+). If a default Teiid server instance is connected and available the Add Translator Override dialog (below) is presented, the user selects an existing tranlator type and clicks OK.



Note

The override is only applicable to sources within the VDB, so be sure and select a translator type that corresponds to one of the VDB's source models. The properties panel on the right side of the panel will contain editiable cells for each property type based on the data-type of the property. (i.e. boolean, integer, string, etc.).

loopback		
metamatrix		
modeshape		
mysql		
mysql5		
olap		
oracle		
postgresql		
salesforce		
sqlserver		
sybase		

Figure D.65. Add Translator Override Dialog

If no default Teiid server instance is available, the "Add New Translator Override" dialog is presented. Enter a unique name for the tranlator override (i.e. "oracle_override"), a valid translator type name (i.e. "oracle") and click OK. The properties panel on the right side of the panel will allow adding, editing and removing key-value string-based property sets. When editing these properties all values will be treated as type string.

	Add New Translator Override	\otimes					
?	A Teiid server is not available so translator types are not available. A translator name and translator type are both required when creating a translator override.						
	Press "OK" when finished.						
Name:							
Type:							
	Cancel OK						

Figure D.66. Add New Translator Override Dialog

D.3.3. Model Extension Definition Editor

The **MED Editor** is a multi-tabbed editor and is used to create and edit user-defined MEDs (*.mxd files) in the workspace. The MED Editor has 3 sub-editors (**Overview, Properties, and Source**) which share a common header section. Here are the MED sub-editor tabs:

• Overview Sub-Editor - this editor is where the general MED information is managed. This information includes the namespace prefix, namespace URI, extended model class, and the description. The Overview sub-editor looks like this:

🦻 mymxd.mxd 🕱	
🕆 Overview 🔹	
Namespace Prefix:	mymodelextension
Namespace URI:	mymodelextension
Model Class:	Relational
Version:	1
Description:	This is my model extension
Overview Properties	Source

Figure D.67. Overview Tab

Properties Sub-Editor - this editor is where the MED extension properties are managed. Each
extension property must be associated with a model object type. The Properties sub-editor is
divided into 2 sections (Extended Model Objects and Extension Properties) and looks like
this:

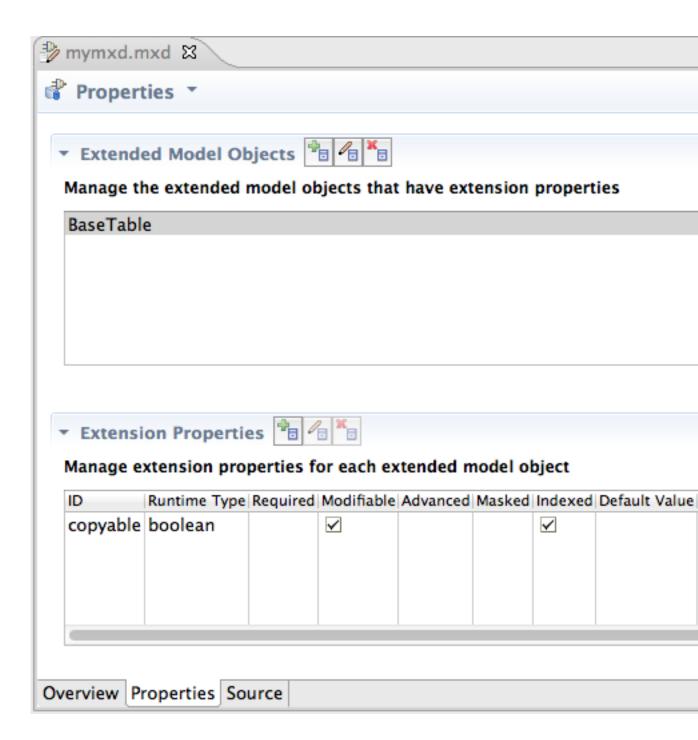


Figure D.68. Properties Tab

• **Source** - this tab is a read-only XML source viewer if you wish to view the details of your MED. This source viewer is NOT editable.

The GUI components on the Overview and Properties sub-editors will be decorated with an error icon when the data in that GUI component has a validation error. Hovering over an error decoration displays a tooltip with the specific error message. Those error message relate to the error messages shown in the common header section. Here is an example of the error decoration:

Namespace URI:

Figure D.69. Text Field With Error

The MED sub-editors share a header section. The header is composed of the following:

- Status Image an image indicating the most severe validation message (error, warning, or info). If there are no validation messages the model extension image is shown.
- Title the title of the sub-editor being shown.
- **Menu** a drop-down menu containing actions for (1) adding to and updating the MED in the registry, and (2) for showing the Model Extension Registry View.
- Validation Message this area will display an OK message or an error summary message.
 When a summary message is shown, the tooltip for that message will enumerate all the messages.
- Toolbar contains the same actions as the drop-down menu.

Below is an example of the shared header section which includes an error message tooltip.

Ø Overview ▼	1 error detected	÷.
Č.	Namespace URI: The "namespace U	IRI" cannot be empty.

Figure D.70. Shared Header Example

D.4. Teiid Designer Main Menu

There are 8 categories of actions on Teiid Designer's main menu bar.

- These categories include:
 - Section D.4.1, "File Menu" Resource management actions.
 - Section D.4.2, "Edit Menu" Standard edit actions including undo/redo.
 - Section D.4.3, "Refactor Menu" Resource actions (i.e. Rename, Move, etc...).
 - Section D.4.5, "Search Menu" Find data within your workspace.
 - Section D.4.6, "Project Menu" Model level actions.
 - Section D.4.7, "Metadata Menu" Custom metadata-related actions.
 - Section D.4.9, "Window Menu" Change perspectives or add/remove views to your perspective.

• Section D.4.10, "Help Menu" - Access available Teiid Designer help documents, Teiid Designer SQL Support Guide and Eclipse Overview information.

 Teiid Designer - PartsProject/PartsSourceA.xmi - JBoss Developer Studio

 File
 Edit
 Refactor
 Navigate
 Search
 Project
 Metadata
 Run
 Window
 Help

Figure D.71. Application Main Menu

D.4.1. File Menu

The File menu provides actions to manage your workspace resources.

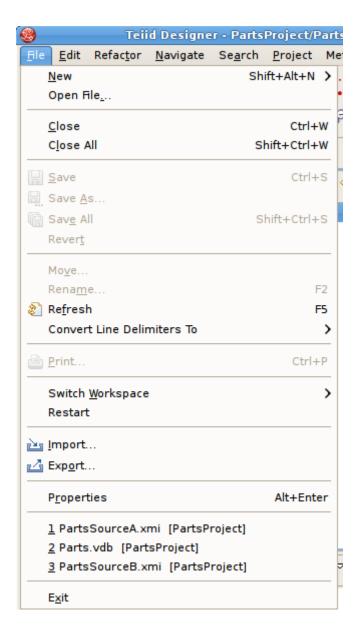


Figure D.72. File Menu

The **New** > sub-menu provides specific actions to create various generic workspace resources as well as Teiid Designer models and VDBs.

<u>File</u> <u>E</u> dit Refac <u>t</u> or	<u>N</u> avigate Se <u>a</u> rch	<u>P</u> roject M	etadata	<u>R</u> un	<u>W</u> indow	<u>H</u> elp
<u>N</u> ew	Sh	ift+Alt+N >	🔺 Teii	d Mode	el Project	
Open File <u>.</u>			📑 P <u>r</u> oj	ject		
<u>C</u> lose		Ctrl+W	🔐 Fold	ler		
C <u>l</u> ose All	s	hift+Ctrl+W			data Mode	el
Save		Ctrl+S	💦 Teii	d VDB		
📓 Save <u>A</u> s			<mark>≣</mark> ∲ E <u>x</u> a	mple		
ि Sav <u>e</u> All	S	hift+Ctrl+S				
Revert			런 <u>O</u> th	er		Ctrl+N

Figure D.73. File Menu

- The File menu contains the following actions:
 - 者

New > Model Project - Create user a new model project.

• 😭

New > Folder - Create new folder within an existing project or folder.

· 🎸

New > Model - Create a new model of a specified model type and class using the *Chapter 4, New Model Wizards*.

• 💦

New > Virtual Database Definition - Create a new VDB, or Virtual Database Definition.

- Open File Enables you to open a file for editing including files that do not reside in the Workspace.
- Close (Ctrl+W) Closes the active editor. You are prompted to save changes before the file closes.
- Close All (Shift+Ctrl+W) Closes all open editors. You are prompted to save changes before the files close.
- 📳

Save (Ctrl+S) - Saves the contents of the active editor.

• 🔡

Save As - Enables you to save the contents of the active editor under another file name or location.

•

Save All (Shift+Ctrl+S) - Saves the contents of all open editors.

- Move... Launches a Refactor > Move resource dialog...
- **Rename... (F2)** Launches a Refactor > Rename resource dialog if resource selected, else in-line rename is preformed.
- Refresh Refreshes the resource with the contents in the file system.
- **Convert Line Delimiters To** Alters the line delimiters for the selected files. Changes are immediate and persist until you change the delimiter again you do not need to save the file.

٠ r Eh

Print (Ctrl+P) - Prints the contents of the active editor.

- Switch Workspace Opens the Workspace Launcher, from which you can switch to a different workspace. This restarts the Workbench.
- Restart Exits and restarts the Workbench.

• 迠

Import - Launches the **Import Wizard** which provides several ways to construct or import models..

• 🗳

Export - Launches the Export Wizard which provides options for exporting model data.

- **Properties (Alt+Enter)** Opens the **Properties** dialog for the currently selected resource. These will include path to the resource on the file system, date of last modification and its writable or executable state.
- Most Recent Files List Contains a list of the most recently accessed files in the Workbench. You can open any of these files from the File menu by simply selecting the file name.
- Exit Closes and exits the Workbench.

D.4.2. Edit Menu

The **Edit** menu provides actions to manage the content, structure and properties of your model and project resources. The figure below represents the Edit menu presented when a metadata model is selected.

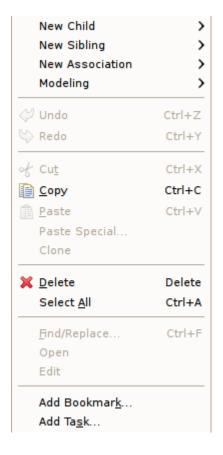


Figure D.74. Edit Menu

- The Edit menu contains the following actions:
 - New > Child This menu is created dynamically to support the creation of whatever types of child objects can be created under the selected object.
 - New > Sibling This menu is created dynamically to support the creation of whatever types of sibling objects can be created under the same parent as the selected object
 - New > Association This menu is created dynamically to support the creation of whatever types of associations can be created with the selected object.
 - **Modeling** > This menu is created dynamically. Various modeling operations are presented based on selected model object type.
 - 🥑

Undo - Reverses the effect of the most recent command.

• 🍫

Redo - Reapplies the most recently undone command.

• • t

Cut - Deletes the selected object(s) and copies it to the clipboard.

• 🗈

Copy - Copies the selected object(s) to the clipboard.

• 💼

Paste - Pastes the contents of the clipboard to the selected context.

- Paste Special... Provides additional paste capabilities for complex clipboard objects.
- **Clone** Duplicates the selected object in the same location with the same name. User is able to rename the new object right in the tree.

• 🗙

Delete - Deletes the selected object(s).

- Select All Select All objects in current view.
- Rename Allows a user to rename an object in the tree.
- Find/Replace Launches dialog that can be used to search in the current text view, such as a Transformation Editor.
- **Open** Opens the selected object in the appropriate editor.
- Edit Opens the selected object in the appropriate specialized editor, such as the Choice Editor or Recursion Editor..
- Add Bookmark... This command adds a bookmark in the active file on the line where the cursor is currently displayed.
- Add Task... This command adds a task in the active file on the line where the cursor is currently displayed.

D.4.3. Refactor Menu

The Refactor menu provides Teiid Designer specific actions for file-level changes to the models.

-	Undo	
	Redo	
1	Rename	
	Move	

Figure D.75. Refactor Menu

- The Refactor menu contains the following actions:
 - Undo Undo the last refactor command.
 - Redo Redo the last undone refactor command.
 - Move Move a model from one container (folder or project) to another.
 - Rename Rename a model.

D.4.4. Navigate Menu

Teiid Designer currently does not contribute actions to the Navigate menu. See Eclipse documention for details.

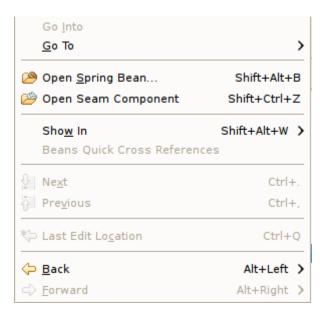


Figure D.76. Navigate Menu

D.4.5. Search Menu

The Search menu presents several specific search options.

🔗 Se <u>a</u> rch	Ctrl+H
📝 <u>F</u> ile	
💖 <u>R</u> emote	
🛷 Pointcut Matches	
🛷 Beans	
💫 Find Seam References	
🖄 Find Seam Declarations	
Text	>
Teiid Designer	>

Figure D.77. Search Menu

Teiid Designer contributes a sub-menu (i.e. Teiid Designer >) to the main search menu, as shown above.

Text	>
Teiid Designer	🗾 📌 Transformations
, (0)	🔗 Metadata
	ind Model Object 👰

Figure D.78. Search Menu

- The individual actions in the Teiid Designer sub-menu are described below:
 - 🚀

Transformations... - Launches the Transformation Search dialog. User can search models in the workspace for matching SQL text. Search results appear in the dialog and user can select and view SQL as well as open desired transformations for editing.

• 🔗

Metadata... - Launches the Search dialog. User can search for models in the workspace by specifying an Object Type, and/or a Data Type, and/or a property value. Search results appear in the *Section D.2.8, "Search Results View"* view, and double-clicking a result will open that model in the appropriate editor.

' 🥟

Find Model Object - Launches the Find Model Object dialog, which can be used to find an object in the workspace by specifying all or part of its name. Selecting the object will open it in the appropriate editor.

D.4.6. Project Menu

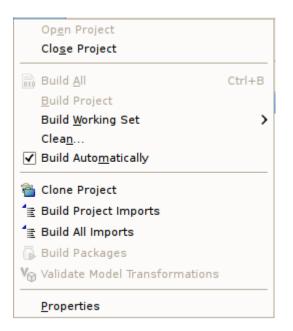


Figure D.79. Project Menu

- The individual actions in the Project menu are described below:
 - Open Project Launches the Open Project dialog.
 - Close Project Closes the currently selected project(s).

Build All - Validates the contents of the entire workspace. Any errors or warnings will appear in the **Problems View**.

- **Build Project** Validates the contents of the selected project(s). Any errors or warnings will appear in the **Problems View**.
- Build Working Set Validates the contents of the selected working set. Any errors or warnings will appear in the Problems View.
- Clean.. Launches the Clean dialog.
- Build Automatically Sets the Build Automatically flag on or off. When on, a check-mark appears to the left of this menu item. When this is turned on, validation of changes is done automatically each time a **Save** is done.
- Clone Project Launches the Clone Project dialog.
- **Build Project Imports** Reconciles all model import dependencies for models contained within the selected project.

[•]

- **Build All Imports** Reconciles all model import dependencies for models contained within the workspace.
- Build Packages TBD
- Validate Model Transformations Revalidates all transformations for the selected view model.
- Properties Displays the operating system's file properties dialog for the selected file.

D.4.7. Metadata Menu

The Metadata menu provides Teiid Designer-specific actions.

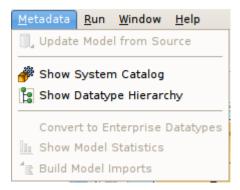


Figure D.80. Metadata Menu

- The Metadata menu contains the following actions:
 - Update Model from Source If the selected model is a relational source model that was originally created via JDBC Import, then the model will be updated based on changes in the database schema.
 - Show System Catalog Opens the Section D.2.11, "System Catalog View".
 - Show Datatype Hierarchy Opens the Section D.2.11, "System Catalog View".
 - Re-resolve References Analyzes references within models to other model components.
 - **Convert to Enterprise Datatypes** Adds an additional property to simple datatypes within your selected schema model to label them as enterprise datatypes.
 - Show Model Statistics Opens the Model Statistics dialog for the selected model.
 - Build Model Imports Reconciles all model import dependencies for the selected model.

D.4.8. Run Menu

Teiid Designer currently does not contribute actions to the Run menu. See Eclipse documention for details.

💁 External Tools 🔹 🔉

Figure D.81. Window Menu

D.4.9. Window Menu

The **Window** menu shown below contains no Teiid Designer-specific actions. See Eclipse Workbench documentation for details.

<u>N</u> ew Window	
New <u>E</u> ditor	
Open Perspective	>
Show View	>
Customize Perspectiv	e
Save Perspective <u>A</u> s.	
Reset Perspective	
<u>C</u> lose Perspective	
Close All Perspectives	s
Navi <u>g</u> ation	>
<u>P</u> references	

Figure D.82. Window Menu

The **Preferences...** action launches the Preferences dialog, which can be used to set preferences and default values for many features of Teiid Designer.



If you wish to customize a perspective to include one or more Teiid Designer views, select the **Show View > Other...** action and expand the Teiid Designer category to show the available views.

Show View	
type filter text	4
 Team Teiid Designer Datatypes Description Metadata Favorites Model Explorer Relationship Navigator System Catalog Tags 	
Teiid Model Classes	~
Cancel OK	

Figure D.83. Show View Dialog

D.4.10. Help Menu

The **Help Menu** shown below contains no Teiid Designer-specific actions. See Eclipse Workbench documentation for details.



Figure D.84. Help Menu

• The individual actions are described below:

• **Welcome** - Shifts to the Welcome perspective, which contains links to documentation, examples and 'how-to' starting points.

• Help

Contents

- Launches the Help Window. All of Designer's online documentation is accessible from there as well.

Search 💯

- Launches the Help Search view, which can be used to search for phrases in the documentation.

- Dynamic Help Opens the docked dynamic help view.
- Key Assist (Ctrl-Shift-L) ... Launches a dialog describing existing key assist bindings.
- Tips and Tricks... Launches a dialog to select one of any contributed "Tips and Tricks" help pages.
- Cheat Sheets... Launches a dialog to select one of any contributed Eclipse cheat sheets.
- **Project Examples...** A JBoss contributed action which provides quick access to import various project examples into your workspace.
- Report Problem A JBoss contributed action which provides simple problem reporting.
- Check for Updates... provides access to retrieve updates to installed Eclipse software.
- Install New Software... provides access to install new software into your workbench.
- About JBoss Developer Studio Launches the About dialog.