

Using the DeepSee Connector

Version 2010.1
17 February 2010

Using the DeepSee Connector

Caché Version 2010.1 17 February 2010

Copyright © 2010 InterSystems Corporation

All rights reserved.

This book was assembled and formatted in Adobe Page Description Format (PDF) using tools and information from the following sources: Sun Microsystems, RenderX, Inc., Adobe Systems, and the World Wide Web Consortium at www.w3c.org. The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.



Caché WEBLINK, Distributed Cache Protocol, M/SQL, M/NET, and M/PACT are registered trademarks of InterSystems Corporation.



InterSystems Jalapeño Technology, Enterprise Cache Protocol, ECP, and InterSystems Zen are trademarks of InterSystems Corporation.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Customer Support

Tel: +1 617 621-0700

Fax: +1 617 374-9391

Email: support@InterSystems.com

Table of Contents

About This Book	1
1 Introduction to the DeepSee Connector	3
1.1 Purpose of the DeepSee Connector	3
1.2 Logging Into DeepSee	4
1.3 Accessing the DeepSee Connector	5
1.4 Switching to Another Namespace	5
2 Tutorial	7
2.1 Tutorial Part 1: Setup	7
2.2 Tutorial Part 2: Generating Classes	9
2.3 Tutorial Part 3: Defining Relationships Among the Classes	12
2.4 Tutorial Part 4: Checking Your Work	14
3 Creating Classes Based on Loaded Data	17
3.1 Preparing to Use an External Database	17
3.2 Creating Classes for Multiple Tables	18
3.2.1 Advanced Options	19
3.3 Creating a Class for a Single Table or a Query	20
3.4 Creating a Class for a File	21
3.4.1 Specifying the Fields in a Fixed-width File	22
3.5 Specifying the Class Basics	23
3.6 Reopening the Class Wizard	25
3.7 Defining Relationships Among the Classes	25
3.7.1 Modifying a Property Type via Drag and Drop	25
3.7.2 Loading Foreign Key Relationships from Microsoft Access	26
3.7.3 Adding a Foreign Key	26
3.8 Modifying the Class Definitions	28
3.9 Compiling the Classes	28
4 Modifying the Classes	29
4.1 Modifying a Class	29
4.2 Overview of the Class ETL Details Area	30
4.2.1 Basics	30
4.2.2 Tabs	31
4.3 Adding Standard Indices	32
4.4 Specifying Advanced Class Options	33
4.5 Deleting a Class	34
5 Modifying the Class Properties	35
5.1 Adding a Property	35
5.2 Modifying the Data Type of a Property	35
5.3 Specifying the Value of a Property	36
5.4 Deleting a Property	36
6 Using Multiple Databases for a Single Class	37
6.1 Enabling the Multi Site Feature	37
6.2 Setting Up a Site Group	37
6.3 Creating a Class That Uses a Site Group	38
6.4 Differences in the Class Definition	39

7 Loading, Browsing, and Analyzing the Data	41
7.1 Analyzing the Data	41
7.1.1 String	43
7.1.2 Number	43
7.1.3 Date Time	43
7.2 Loading or Deleting Data	44
7.3 Browsing the Data	45
7.4 Exporting Data	47
7.5 Creating Saved Queries for Local Data	48
7.5.1 Saving a Query	48
7.5.2 Opening a Saved Query	48
7.5.3 Modifying a Saved Query	49
7.5.4 Deleting a Saved Query	50
8 Fine-tuning the Data Loading Methods	51
8.1 Overview	51
8.1.1 Overall Flow of Data Loading	51
8.1.2 Summary of Connector Options for Customizing Data Loading	52
8.2 Defining Load All Scripts	52
8.3 Defining Cleaning Scripts	53
8.4 Defining Cleaning Rules	54
8.4.1 Other Options	55
8.4.2 Testing Cleaning Rules	55
8.4.3 Viewing the Cleaning Rule Logs	55
8.5 Using the Script and Expression Editor	56

List of Tables

Table 4–1: Basic Information in Class ETL Details	30
Table 4–2: Tabs in Class ETL Details	31
Table 8–1: Custom Processing within Data Loading	52

About This Book

This book describes how an implementer can use the DeepSee Connector to import externally stored data, in preparation for using the DeepSee Architect.

Note: The DeepSee Connector is available only with InterSystems Ensemble.

This book contains the following sections:

- [Introduction to the DeepSee Connector](#)
- [Tutorial](#)
- [Creating Classes Based on Loaded Data](#)
- [Modifying the Classes](#)
- [Modifying the Class Properties](#)
- [Using Multiple Databases for a Single Class](#)
- [Loading, Browsing, and Analyzing the Data](#)
- [Fine-tuning the Data Loading Methods](#)

For a detailed outline, see the [table of contents](#).

For more information, see the following books:

- *Overview of DeepSee*, an introductory guide for all users who are interested in learning about DeepSee.
- *DeepSee Model Design Guide*, an introductory guide for implementers and business users.
- *DeepSee Developer Tutorial*, a tutorial for implementers who are creating DeepSee models, pivot tables, and dashboards.
- *Using the DeepSee Architect*, a guide for implementers who are setting up a DeepSee model for use in the Analyzer.
- *Using the DeepSee Analyzer*, a guide for implementers and advanced users who want to create pivot tables to embed in applications — or who simply want to explore their data.
- *Using the DeepSee Dashboard Designer*, a guide for implementers who are using the Dashboard Designer to create dashboards.
- *Expressions and Scripts in DeepSee*, an implementer guide that describes the syntax and options for all formulas, expressions, and scripts supported in DeepSee. This book also lists all the locations where you can use these expressions and scripts.
- *DeepSee Site Configuration and Maintenance Guide*, a guide for implementers and system administrators. This book describes how to configure and maintain a DeepSee site. It also includes a chapter on troubleshooting.
- *DeepSee User Guide*, a user manual for your end users. This book describes how to work with deployed dashboards and pivot tables.

For general information, see the *InterSystems Documentation Guide*.

1

Introduction to the DeepSee Connector

This chapter introduces the DeepSee Connector, which you use to import external data for use in DeepSee. This chapter discusses the following topics:

- [Purpose of the DeepSee Connector](#)
- [How to log into DeepSee](#)
- [How to access the DeepSee Connector](#)
- [How to switch to another namespace](#)

Note: The DeepSee Connector is available only with InterSystems Ensemble.

Be sure to consult *InterSystems Supported Platforms* for information on system requirements.

1.1 Purpose of the DeepSee Connector

You use the DeepSee Connector during an implementation process whose overall goal is to embed pivot tables in existing or new applications. Pivot tables provide real-time business intelligence (BI) — interactive tables and graphs with which your users can explore the data used in or generated by their applications.

The purpose of the Connector is to generate Caché classes that model data in external sources and that include class methods for loading that data into Caché.

When you compile these classes, Caché generates class methods for loading the data. You use the Connector to apply transformation rules of various kinds to the data as it is being loaded. You can also add custom processing to perform before or after loading data or at other points. The generated class methods then contain all the transformation and processing logic.

You can execute these data loading methods from within the Connector, from the DeepSee Scheduler, or programmatically from your own code.

In a typical implementation process, you use the Connector to do any or all of the following:

- Add other classes, based on other data sources, including files
- Fine-tune which fields to use
- Add additional properties, perhaps based on multiple fields of the external tables
- Define cleaning rules, cleaning scripts, and so on

- Browse the data
- Test the data loading methods

1.2 Logging Into DeepSee

To log into DeepSee:

1. Click the InterSystems Launcher.

When you do so, the system displays a menu.

2. Click **DeepSee**.

If you have not yet specified a namespace, the system displays a page that prompts you for a namespace.

Otherwise, the system displays the DeepSee login page.

3. If you are prompted for a namespace, type the name of the namespace you want to work in and then click **Logon to DeepSee**.

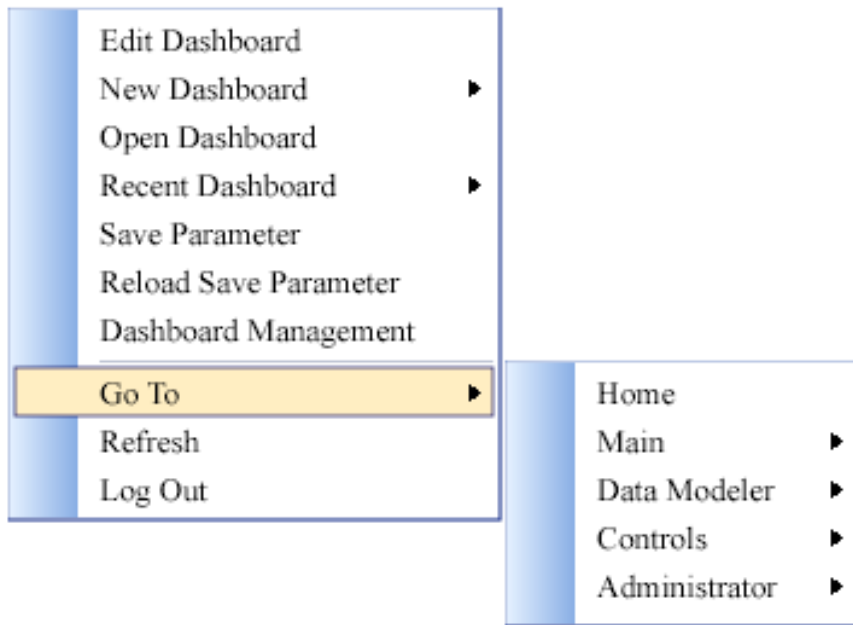
The system then displays the DeepSee login page.

4. On the DeepSee login page, enter a DeepSee username and password. For example, you can use the username `demo` with the password `demo`.
5. For **Role**, select `demo`.
6. Click **Login**.

DeepSee displays the home page, which depends upon the user ID you used to log in. The home page is either a DeepSee module or a dashboard. If the page is a DeepSee module, it has a row of buttons at the top as follows:



If the page is a dashboard, the context menu provides access to all the same options provided by these buttons, in addition to options that apply to dashboards:

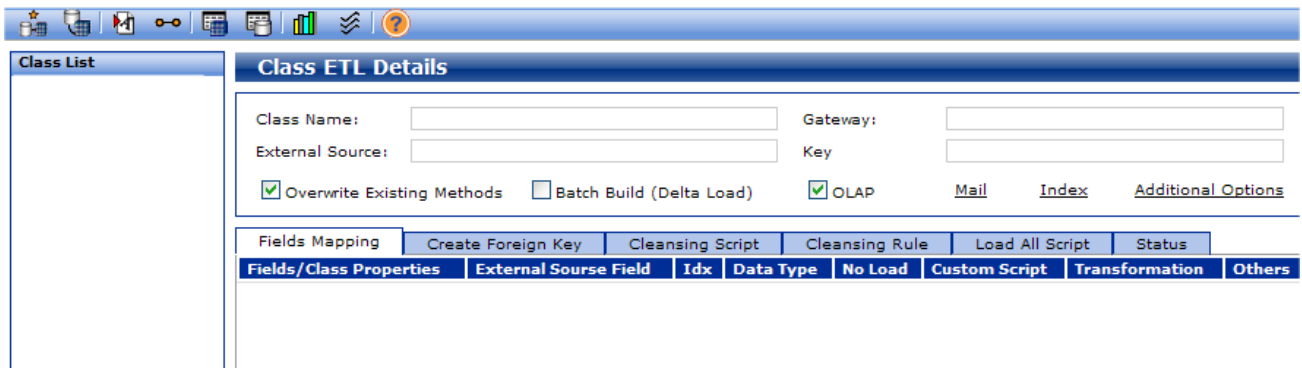


1.3 Accessing the DeepSee Connector

To access the Connector:

- If you are currently viewing a DeepSee module, click **Data Modeler > Connector**.
- If you are currently viewing a dashboard, right-click and then click **Go To > Data Modeler > Connector**.

DeepSee then displays the following page:



1.4 Switching to Another Namespace

To switch to a different namespace:

1. Log out. To do so, do one of the following, depending on what you are currently viewing:
 - If you are currently viewing a DeepSee module, click **Log Off** in the upper right.
 - If you are currently viewing a DeepSee dashboard, right-click and then click the **Log Out** option.

The system logs you out of DeepSee.

2. Click **Switch Namespace**.
3. For **Namespace**, type the name of the namespace you want to work in.
4. Click **Logon to DeepSee**.

This displays a login page.

5. Log in as usual.

2

Tutorial

This chapter presents a tutorial that starts in the DeepSee Connector with an external sample database and ends in the DeepSee Architect with a view of the classes that you created.

This tutorial is divided into the following parts:

1. [Setup work](#)
2. [Generating classes](#)
3. [Defining relationships among the classes](#)
4. [Checking your work](#)

This tutorial uses Microsoft Access and its sample database, the Northwinds Trade Company database. If you do not have Microsoft Access, you may be able to download a trial version. If you do have Microsoft Access but do not have the Northwinds (NWIND) database, you can download this common sample from the internet.

Or you can use some other database type and database sample and adapt the steps given here.

2.1 Tutorial Part 1: Setup

We start the tutorial as follows:

1. Download the English-version Northwinds Trade Company database (NWIND.mdb) and place it in C:\InterSystems\sample-data or some other appropriate location, depending on your operating system and preferences.

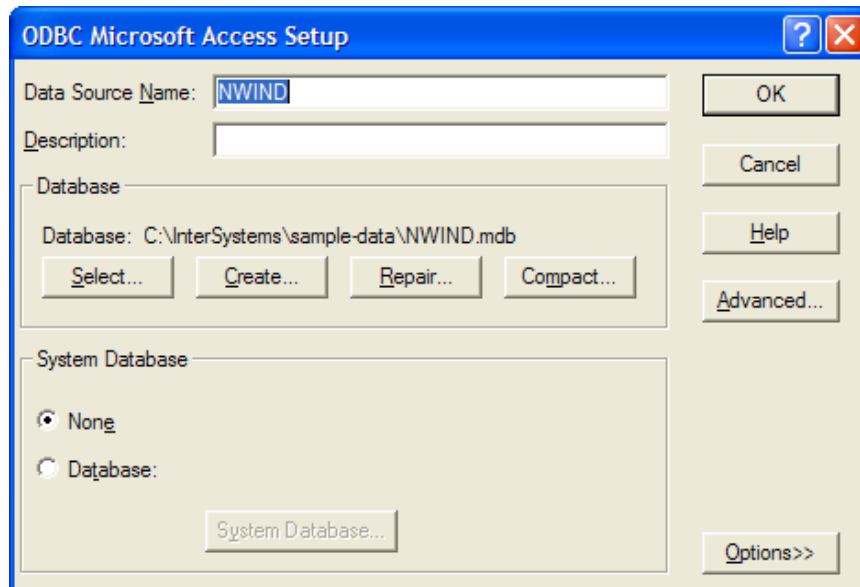
To find this Microsoft Access sample, use any internet search engine. (No link is given here because such links are often not permanent.)
2. Create an ODBC data source name (DSN) for this database. The technique depends upon your operating system (and on the database type). On Windows, assuming you are using NWIND.mdb:
 - a. Click **Start > Control Panel**.
 - b. Double-click **Administrative Tools**.
 - c. Double-click **Data Sources (ODBC)**.
 - d. Click the **System DSN** tab.
 - e. Click **Add**.

This displays a dialog box that lists the ODBC database drivers installed on this machine. There are different database drivers for each database type, and the driver names indicate the types. Sometimes for a given type, there are different drivers for different locales as well.

- f. Click the appropriate database driver.
- g. Click **Finish**.

This displays another dialog box where you choose the data source and specify a name for this DSN.

- h. For **Data Source Name**, specify a convenient name for the DSN. For example: NWIND
- i. For **Database**, click **Select**, navigate to the NWIND.mdb file, and click **OK**.



- j. Click **OK** to save the new DSN and close this dialog box.
 - k. Click **OK** again to exit the DSN setup tool.
3. Create a Caché/Ensemble SQL gateway connection so that the Caché database can find this external data source. To do so:
- a. Open the Caché System Management Portal.
 - b. Click **[Configuration] > [Object/SQL Gateway Configurations]**.
 - c. Click **Object/SQL Gateway Connections**.
 - d. Click **Create New Connection**.
 - e. Click **ODBC**.
 - f. For **Name**, specify a convenient name for the connection. For example: NWIND
 - g. For **Select an existing DSN**, click the DSN you defined earlier.

Create a new SQL Gateway connection using the form below:

The screenshot shows a form with the following fields:

- Type of connection:** Radio buttons for JDBC and ODBC. ODBC is selected.
- Connection name:** Text input field containing 'NWIND'.
- Select an existing DSN:** Dropdown menu with 'NWIND' selected.

- h. Click **Save**.

2.2 Tutorial Part 2: Generating Classes

1. Log into the DeepSee Connector [as described earlier](#).

2. Click the data dictionary wizard button () in the menu bar.

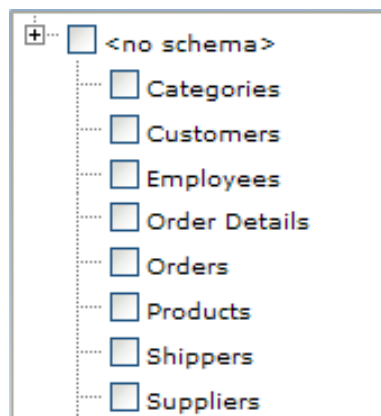
The system displays the first screen of a wizard.

3. Click **Next**.

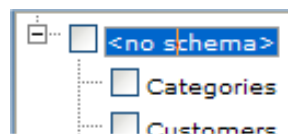
The next screen of the wizard displays a list of SQL Gateway connections.

4. Click the name of the SQL Gateway connection you created previously and then click **Next**.

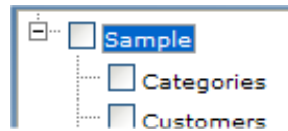
The next screen of the wizard lists the tables, views, or both within this database, grouped by the schemas (if any) to which they belong. Microsoft Access does not use schemas, so for the Northwinds sample, we see the following:



5. Click within the <no schema> label. The cursor is displayed within the name:



Then edit that name to be `Sample` (without angle brackets). This becomes the name of the package that contains the classes we are generating.



6. Click the check box next to Sample.

DeepSee selects the check box for each table in this database.

7. Click **Finish**.

The wizard now displays, in succession, one screen for each class definition that it will create (one per table). It processes the tables in alphabetical order. You can reopen this wizard later if needed to make changes, and most of the information can be edited in other ways as well.

8. The first table, Categories, is displayed as follows:

Table: <no schema>.Categories

Select	Field/Property Name	>>Key
<input checked="" type="checkbox"/>	CategoryID	>>>>>
<input checked="" type="checkbox"/>	CategoryName	>>>>>
<input checked="" type="checkbox"/>	Description	>>>>>
<input checked="" type="checkbox"/>	Picture	>>>>>

Select All

Primary Key (ID Key)

Delete
Delete All

Slowly Change Dimension

Not Applicable ▼

Foreign Version Derived Field

Multi Sites

Overwrite Use Wide

System Default ▼

Copy Data
 No Validation
 OLAP

Process
Cancel

This screen lists the fields in the Categories table.

For this table, the primary key is CategoryID, which is the identifier for the categories in this database.

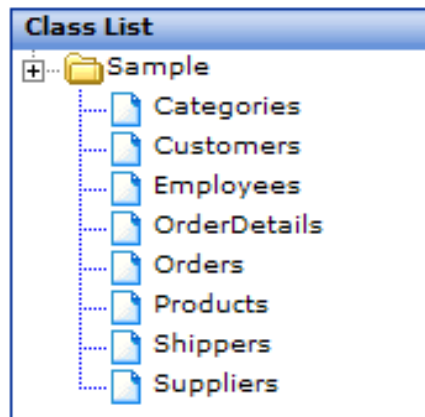
9. For Categories, do the following:
 - a. Click >>>>> next to CategoryID. This field is then added to the **Primary Key** list.
 - b. Click **Process**. The system then generates the class definition.

- c. Click **OK** to continue. The system then displays the next table.
10. Work through the remaining tables in the same way. The following table summarizes what to do for each table. If you make mistakes, you can correct them easily afterwards.

Table Name	Primary Key	Additional Steps
Categories	CategoryID	
Customers	CustomerID	
Employees	EmployeeID	
OrderDetails	OrderID and ProductID	
Orders	OrderID	Select the OLAP check box.
Products	ProductID	
Shippers	ShipperID	
Suppliers	SupplierID	

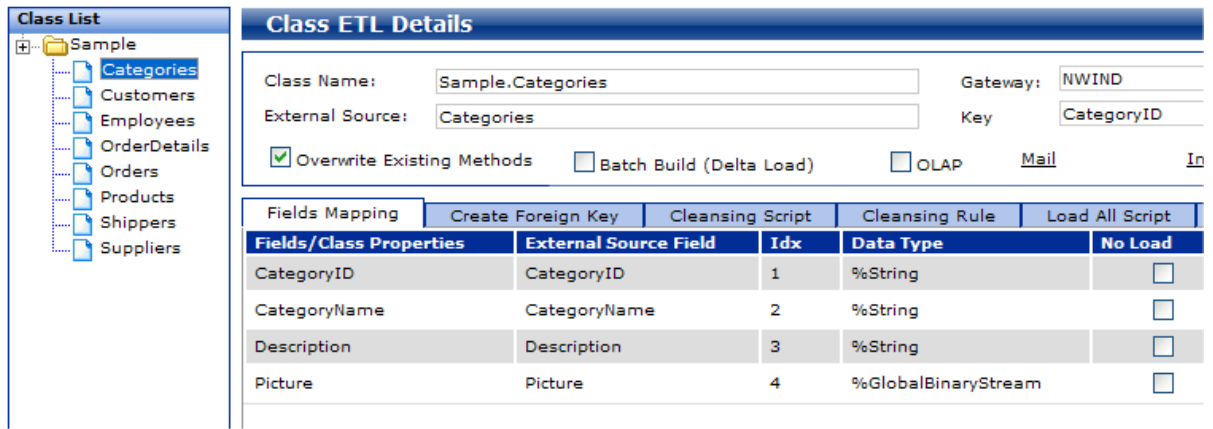
These details would of course be different if you are using a different sample.

11. When you have completed all the tables, the Connector displays a list of the classes it has generated:



The Connector also loads data from the external tables into the Caché database.

12. The next step is to review the classes and make any corrections. For each class:
- a. Click the class name within **Class List**. The right area then displays details for the class.



- b. View the property names in the **Fields Mapping** table. Each property corresponds to a field.
- c. Look at the **Key** field. This must be a property name (or a comma-separated list of property names) or null.
Make edits here if needed.
- d. For Orders, look at the **OLAP** check box and select it if you forgot to do so earlier.
- e. Click **Update** to save the changes to the class.

2.3 Tutorial Part 3: Defining Relationships Among the Classes

DeepSee works primarily by using the property relationships among your classes. Such properties correspond to foreign keys between tables, so we start by listing the relevant relationships in the Northwinds sample:

In This Table ...	This Field ...	Refers to a Row in the Table ...
Orders	CustomerID	Customers
Orders	ShipperID	Shippers
Orders	EmployeeID	Employees
Employees	ReportsTo	Employees (this is not a foreign key, strictly speaking, but it is a relationship we should model)
OrderDetails	OrderID	Orders
OrderDetails	ProductID	Products
Products	CategoryID	Categories
Products	SupplierID	Suppliers

If you are using a different sample, be sure that you know the relationships between the tables in your sample.

We will use this information when we define relationships among our classes:

1. Click the `Orders` class. The **Fields Mapping** tab displays the following.

Class List

- Sample
 - Categories
 - Customers
 - Employees
 - OrderDetails
 - Orders
 - Products
 - Shippers
 - Suppliers

Class ETL Details

Class Name: Gateway:

External Source: Key

Overwrite Existing Methods Batch Build (Delta Load) OLAP

Fields/Class Properties	External Source Field	Idx	Data Type	No L
CustomerID	CustomerID	1	%String	[
EmployeeID	EmployeeID	2	%Numeric	[
Freight	Freight	3	%Numeric	[
OrderDate	OrderDate	4	%TimeStamp	[
OrderID	OrderID	5	%String	[
RequiredDate	RequiredDate	6	%TimeStamp	[
ShipAddress	ShipAddress	7	%String	[
ShipCity	ShipCity	8	%String	[
ShipCountry	ShipCountry	9	%String	[
ShipName	ShipName	10	%String	[
ShipPostalCode	ShipPostalCode	11	%String	[
ShipRegion	ShipRegion	12	%String	[
ShipVia	ShipVia	13	%Numeric	[
ShippedDate	ShippedDate	14	%TimeStamp	[

2. Notice that CustomerID is of type %String, and EmployeeID and ShipVia are of type %Numeric.
3. Drag Customers from **Class List** and drop it onto the **Data Type** column in the CustomerID row.

Class List

- Sample
 - Customers
 - Categories
 - Employees
 - OrderDetails
 - Orders
 - Products
 - Shippers
 - Suppliers

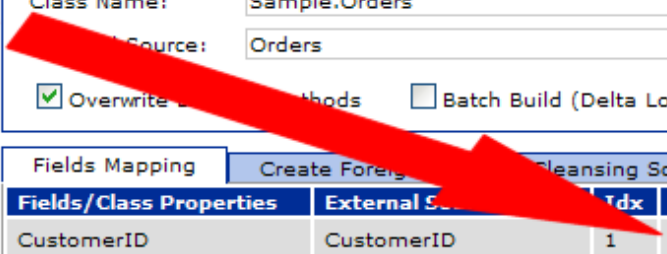
Class ETL Details

Class Name: Gateway:

External Source: Key


Overwrite Existing Methods Batch Build (Delta Load) OLAP

Fields/Class Properties	External Source Field	Idx	Data Type	No L
CustomerID	CustomerID	1	%String	[
EmployeeID	EmployeeID	2	%Numeric	[
Freight	Freight	3	%Numeric	[
OrderDate	OrderDate	4	%TimeStamp	[
OrderID	OrderID	5	%String	[



The type for CustomerID is modified as follows:

Fields/Class Properties	External Source Field	Idx	Data Type
CustomerID	CustomerID	1	Sample.Custome
EmployeeID	EmployeeID	2	%Numeric
Freight	Freight	3	%Numeric
OrderDate	OrderDate	4	%TimeStamp
OrderID	OrderID	5	%String
RequiredDate	RequiredDate	6	%TimeStamp


4. Click **Update** to save this change. Now we have established a property relationship between `Orders` and `Customers`. We could repeat this technique for each of the other relationships. Because we are using Microsoft Access, however, a quicker technique is available.
5. Click the load relationships button () in the menu bar. The system displays a dialog box.
6. For **DSN**, click the name of the SQL gateway connection to the database.
7. Click **Load Relationship**. The Connector now uses a Microsoft Access system table that lists all foreign key relationships and then uses that information to update the **Data Type** settings of any affected properties for all classes. These changes are immediately saved.
8. Click **OK**.
9. Click **Exit** to close the dialog box.
10. Click the `Employees` class. Notice that for the `ReportsTo` property, the type is `%String`. This type was not updated because, as noted earlier, this relationship is not a foreign key relationship.
11. Drag the `Employees` class from **Class List** and drop it onto **Data Type** in the `ReportsTo` row.
12. Click **Update** to save this change.

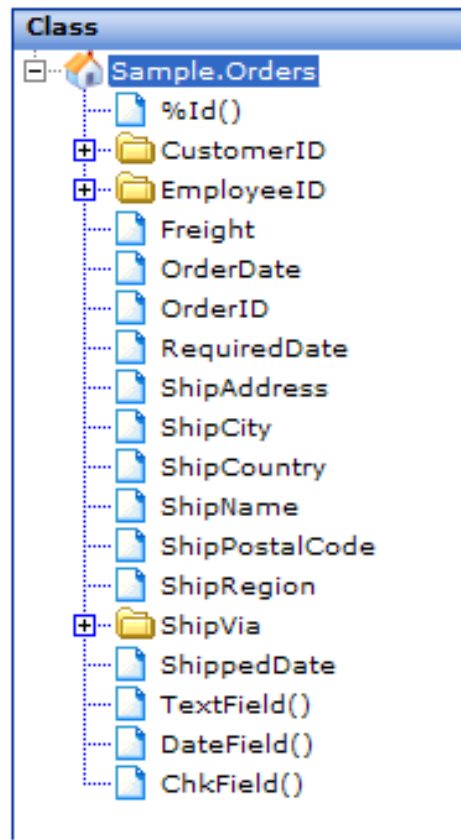
2.4 Tutorial Part 4: Checking Your Work

The primary purpose of the Connector is to define classes that can be used in the Architect, so let us check what the Architect sees.

1. At the top of the Connector screen, click **Data Modeler > Architect**.

This displays the Architect.

2. Click the open class button () in the menu bar.
The Architect displays all the BI-enabled classes in this namespace. This should include `Sample.Orders`.
If you do not see this class, go back to the Connector and make sure you clicked **OLAP** for this class.
3. Click `Sample.Orders` and click **OK**.
The **Class** area now displays an expandable folder labeled `Sample.Orders`.
4. Click the plus sign (+) beside `Sample.Orders`. DeepSee now displays the following:



If you do not see the subfolders as shown here, that indicates that you have not defined property relationships among the classes as described in the previous section. Go back to the Connector and double-check the types for the properties, especially in the `Orders` class.

If you see the `Sample.Orders` class with subfolders as shown here, you can use this class within the Architect, and you have successfully completed this tutorial. For a tutorial on the Architect and other DeepSee components, see the [DeepSee Developer Tutorial](#).

3

Creating Classes Based on Loaded Data

This chapter describes how to generate classes that model data loaded from external databases or files. It discusses the following:

- [Steps to perform before you can access an external database](#)
- [How to generate Caché class definitions for multiple external tables](#)
- [How to generate a Caché class for a single external table or for a query](#)
- [How to generate a Caché class for an external file](#)
- [How to specify the basic information for the Caché class in all cases](#)
- [How to reopen and rerun the wizard used to define any of these classes](#)
- [How to define the relationships between classes needed by DeepSee](#)
- [How to modify the classes further](#)
- [How to compile the classes](#)

As you perform the tasks in this chapter, you may want to load and examine data, both the locally stored data as well as the externally stored data. See the chapter “[Loading, Browsing, and Analyzing the Data](#),” later in this book.

3.1 Preparing to Use an External Database

Before you can work with an external database from the DeepSee Connector, perform the following setup work:

1. Configure the DeepSee Connector to use the appropriate type of external database:
 - a. Click **Administrator > Site Configuration**.
 - b. Click **ETL** on the left.
 - c. For **Database?**, type one of the following values:
 - 0 — For Microsoft Access (the default)
 - 1 — For Oracle
 - 2 — For Postgres SQL
 - d. Click **Save**.

2. Define an ODBC data source name (DSN) for the external database. See the documentation for the external database for information on how to do this.
3. Use the SQL Gateway to create an SQL Gateway connection that uses this DSN. See the section “Creating an ODBC Gateway Connection” in *Using Caché with ODBC*.

3.2 Creating Classes for Multiple Tables

To generate classes for multiple external tables, do the following:

1. Do the preparatory work described in “[Preparing to Use an External Database.](#)”

2. Click the data dictionary wizard button () in the menu bar.

The system displays the first screen of a wizard.

3. On the welcome screen of the wizard, do the following:
 - Select or clear the **View** check box, depending on whether you want to see views. The default is not to see views.
 - Select or clear the **Table** check box, depending on whether you want to see tables. The default is to see tables.
 - For **Schema**, optionally type the name of the schema that you want to view. The default is to see all schemas.

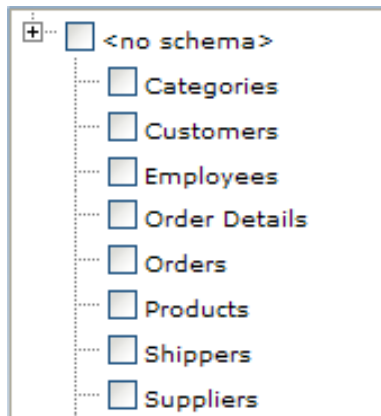
Note that not all databases use schemas.

4. Click **Next**.

The next screen of the wizard displays a list of SQL Gateway connections.

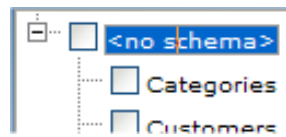
5. Click the name of a SQL Gateway connection and then click **Next**.

The next screen of the wizard displays a list of the tables, views, or both within this database, grouped by the schemas (if any) to which they belong. For example:



6. On this screen, do the following:
 - Select the check box next to each table that you want to model as a class.
 - Optionally specify the package that will contain the class that you are generating. By default, the package name is the same as the schema name. If there is no schema name, the default package is User.

To specify the package name, click the name of the schema in the hierarchy. The cursor is displayed within the name:



Then edit that name.

7. Click **Finish**.

The wizard now displays, in succession, one screen for each class that it will create (one class per table).

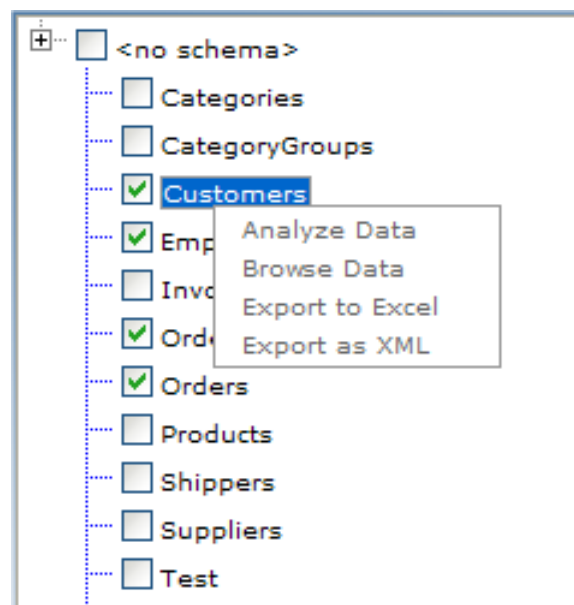
8. On each screen, specify basic details of the class such as its primary key. For details, see “[Specifying the Class Basics](#),” later in this chapter.
9. When you are done with a screen, click **Process**.

The wizard then creates the class and displays the next class, if any.

After the wizard has run, the classes are shown in the **Class List** window. You can reopen and rerun this wizard later again if needed. For each class, you can also modify all the details outside of the wizard, apart from the package and class name.

3.2.1 Advanced Options

This wizard provides easy access to data browsing and analysis functions that are discussed later in this book. When you are viewing the list of tables in the external database, you can right-click a table name to access these functions.




- **Analyze Data** gives you information about the uniqueness of data, the range of values represented, the presence of nulls, and so on. This information is useful for determining the primary key and for determining data cleaning needs. See “[Analyzing the Data](#).”
- **Browse** enables you to browse either the local data or the externally stored data. See “[Browsing the Data](#).”
- **Export to Excel** and **Export to XML** export the local data or the externally stored data, See “[Exporting Data](#).”

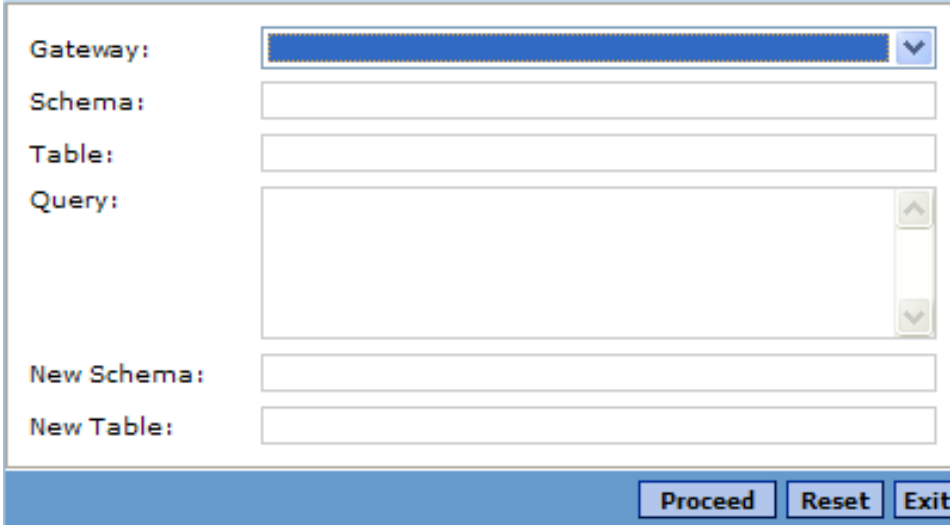
3.3 Creating a Class for a Single Table or a Query

To generate a class for a single external table or for the data returned by a query, do the following:

1. Do the preparatory work described in [“Preparing to Use an External Database.”](#)

2. Click the connect table button () in the menu bar.

The system displays the following dialog box:



The dialog box contains the following fields and controls:

- Gateway:** A dropdown menu.
- Schema:** A text input field.
- Table:** A text input field.
- Query:** A text area with scrollbars.
- New Schema:** A text input field.
- New Table:** A text input field.
- Buttons:** Proceed, Reset, and Exit.

3. For **Gateway**, click the name of a SQL Gateway connection.
4. Specify the source table or query. To do so:
 - For **Schema**, type the name of the schema to which the table belongs, if any.
 - For **Table**, type the name of the table.

Or leave these fields blank and type an SQL query into **SQL Query**.

5. Specify the target class to create. To do so:
 - For **New Schema**, type the name of the package to which the class belongs. The default is the name of the original schema. If there is no schema name, the default package is User.
 - For **New Table**, type the name of the class. The default is the name of the original table.

6. Click **Proceed**.

The wizard now displays a screen where you specify some details of the class.

7. On this screen, specify basic details of the class such as its primary key. For details, see [“Specifying the Class Basics,”](#) later in this chapter.
8. When you are done, click **Process**.


The wizard then creates the class.

After the wizard has run, the class is shown in the **Class List** window. You can reopen and rerun this wizard later again if needed. You can also modify all the details outside of the wizard, apart from the package and class name.

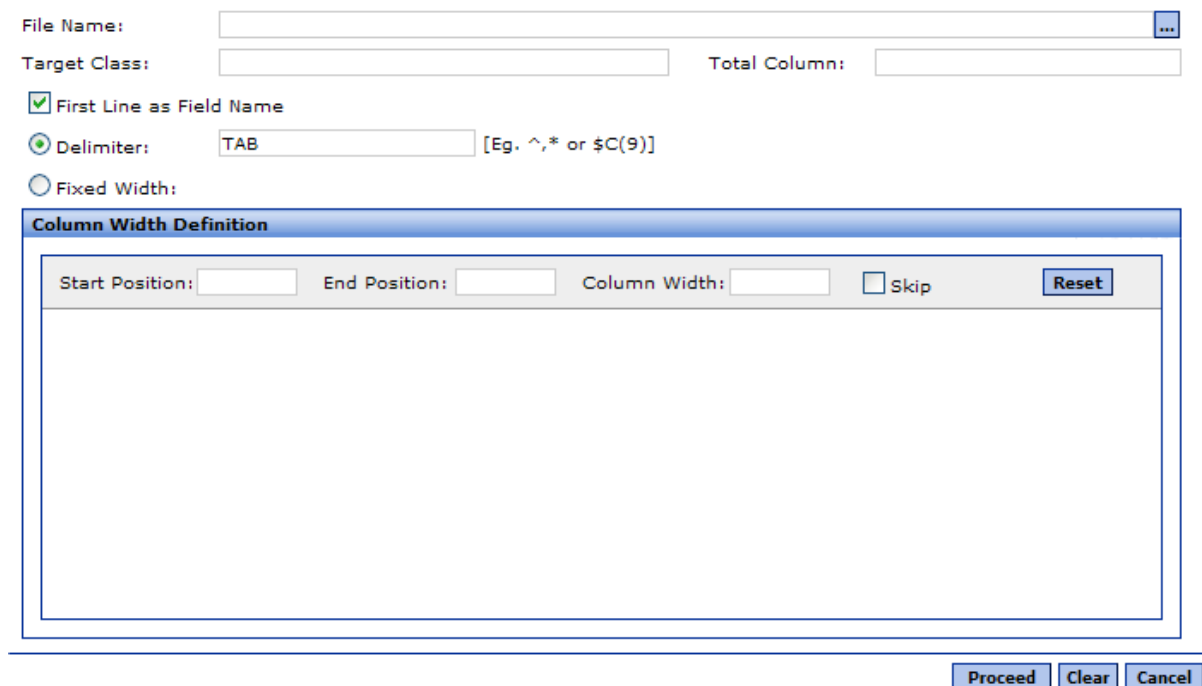
3.4 Creating a Class for a File

To access a file and create a class for it, do the following.

Note: The file must be encoded as UTF-8.

1. Click the load text file table button () in the menu bar.

The system displays the following dialog box:



File Name: ...

Target Class: Total Column:

First Line as Field Name

Delimiter: [Eg. ^, * or \$C(9)]

Fixed Width:

Column Width Definition

Start Position: End Position: Column Width: Skip

2. Click the browse button (...) next to **File Name**.
3. Navigate to the file and click **OK**. The dialog box automatically updates **Target Class** to display the default package and class name in the form *package.class*. The default package is User and the default class name is the name of the file, without the extension.
4. Optionally edit **Target Class**.
5. Optionally type the number of columns in the external file into **Total Columns**.
6. If the first line of file does not consist of header information, clear **First Line as Field Name**.
7. Specify how the file is delimited. Either:
 - Click **Delimiter** and then type the delimiter into the field next to **Delimiter**.
 - Click **Fixed Width**.
8. If the table has fixed-width columns, indicate the boundaries between fields. To do so, click between dots in the position reference bar. See the following subsection for details.
9. Click **Proceed**.

The wizard now displays a screen where you specify some details of the class.

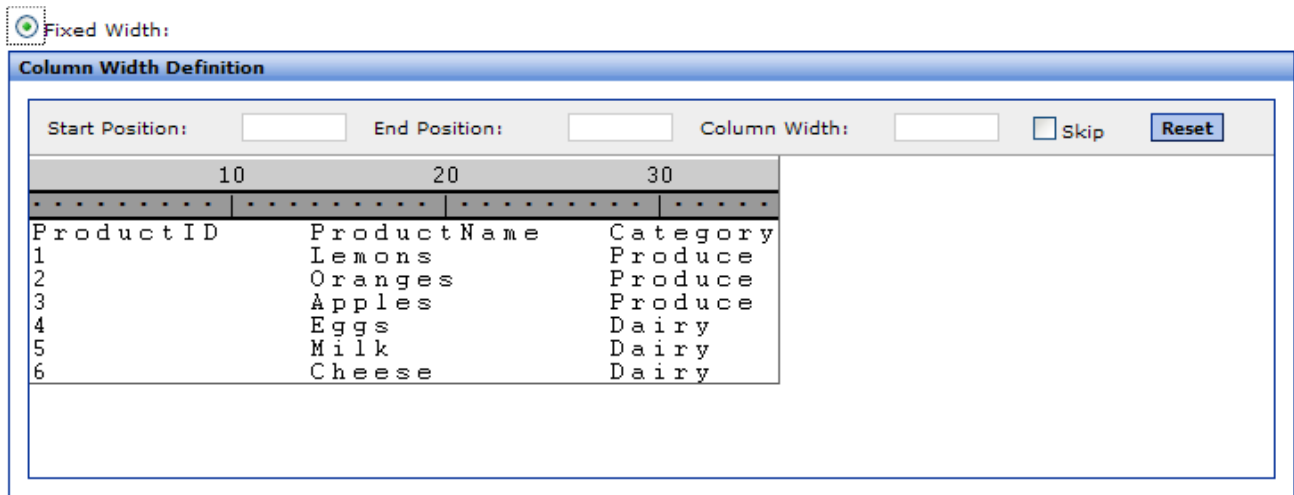
10. On this screen, specify basic details of the class such as its primary key. For details, see “[Specifying the class Basics](#),” later in this chapter.
11. When you are done, click **Process**.

The wizard then creates the class.

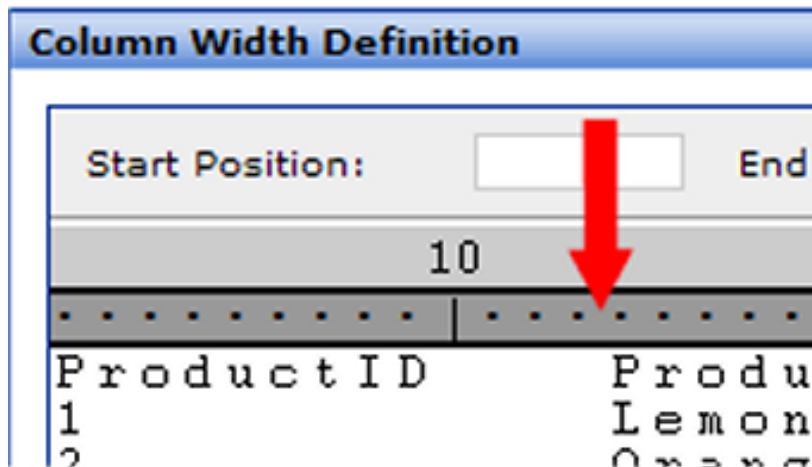
After the wizard has run, the class are shown in the **Class List** window. You can reopen and rerun this wizard later again if needed. You can also modify all the details outside of the wizard, apart from the package and class name.

3.4.1 Specifying the Fields in a Fixed-width File

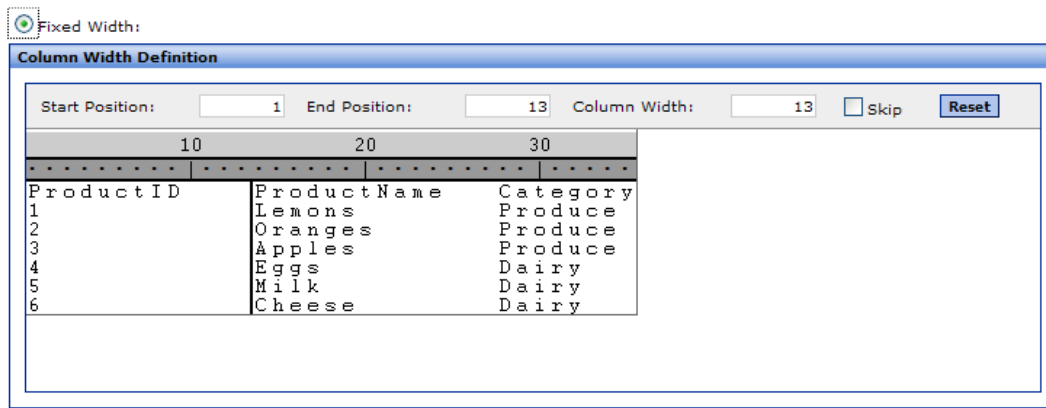
The **Column Width** window shows a preview of the file that you are using. Above the data, the window displays an reference bar that shows column positions:



To specify the fields in a fixed-width file, click on between the dots on this bar to indicate each boundary between fields. To specify the boundary between the first and second fields in this example, we click on the reference bar just before the start of the second field:

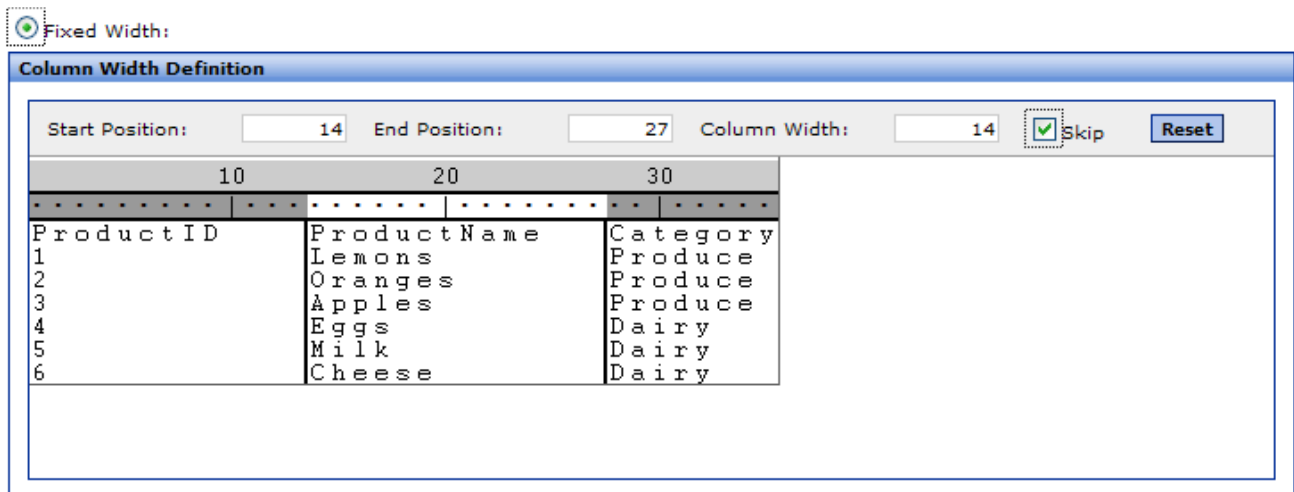


As soon as you do this, the system displays a vertical line at that position, selects the field to the left of the boundary, and updates the **Start Position**, **End Position**, and **Column Width** fields accordingly:



This action is a toggle; if you click again in the same position, the boundary is removed and its information is cleared.

Repeat for each field in the file. If you want to ignore a given field, mark it as described here and click **Skip**. When you do so, the reference bar displays the field in white rather than gray. For example, suppose that we skip the second field:



To clear all boundaries (for example, so that you can start over), click **Skip**.

3.5 Specifying the Class Basics

In all cases, when you use the Connector to generate a Caché class, you specify the basic information on a dialog box that lists the fields that it has identified in the data.

You specify which fields to load, which fields make up the primary key, and other basics. An example is as follows:

All Fields		
Select	Field/Property Name	>>>>>Key
<input checked="" type="checkbox"/>	OrderID	>>>>>
<input checked="" type="checkbox"/>	ProductID	>>>>>
<input checked="" type="checkbox"/>	UnitPrice	>>>>>
<input checked="" type="checkbox"/>	Quantity	>>>>>
<input checked="" type="checkbox"/>	Discount	>>>>>

Primary Key (ID Key)

Delete Delete All

Slowly Change Dimension

Not Applicable ▼

Foreign Version Derived Field

▼

Multi Sites

▼

Overwrite Use Wide

System Default ▼

Copy Data

No Validation

OLAP

Process Cancel

On this screen, do the following:

- Identify the primary key for this table. To do so, click >>>>> next to each field that is in the primary key. As you do so, the field is added to the **Primary Key** list:

Primary Key (ID Key)

OrderID

ProductID

To remove a field from the primary key, click it in the **Primary Key** list and click **Delete**. To remove all fields from the primary key, click **Delete All**.

- Specify which fields to use. By default, all fields in the table are used (and properties are created for them). To ignore a field, clear the check box next to the field name. To select all fields, click **Select All**.
- Copy the external data into the database class. To do so, click the **Copy Data** check box.
- BI-enable this class. To do so, click the **OLAP** check box.

When you BI-enable a class, all its properties and (recursively) their properties are automatically available in the DeepSee Architect; therefore you do not typically need to directly BI-enable every class.

When you are done, click **Process**. The wizard then creates or modifies the class, copies the data if requested, and displays the next class, if any.

Or click **Cancel** to ignore this table. This wizard then displays the next class, if any.

3.6 Reopening the Class Wizard

After you have defined a class, you can reopen the class wizard and make changes.

To do so, right-click the class name in the **Class List** window and click **Modify ETL Class**. The system displays the appropriate wizard, showing the details that currently apply to this class.

3.7 Defining Relationships Among the Classes

DeepSee works by using the property relationships among your classes. That is, when you BI-enable a class, the Architect has easy access to all properties of that class and (recursively) all properties of those properties. Therefore, after you generate the classes as described earlier in this chapter, it is necessary to define the relationships between them. To do this, you must start by understanding the relationships among the tables in the external data; these may be implicit or may be reinforced by foreign key constraints.

For example, suppose the table `Orders` includes a field called `Customer`. In this table, `Customer` is an internal identifier that refers to a row in another table, `Customers`. In an object model, `Orders` is the *parent* and `Customer` is the *child*.

If you use the Connector to generate classes in a package called `Sample`, the result is two classes named `Sample.Orders` and `Sample.Customers`. In `Sample.Orders`, the property `Customer` is initially of type `%String`. In an object model, you would modify the type of the `Customer` property to be the `Sample.Customers` class.

There are three equivalent ways to modify a property type to be a class name:

- Edit the class in the Studio.
- Use the drag-and-drop technique in the Connector to update the property type. This technique is discussed in the [next subsection](#) (and is demonstrated in the [tutorial](#) earlier in this book).
- If the tables were originally in Microsoft Access, load the relationship information from Access, automatically updating all the affected class. This technique is discussed in a [later subsection](#).

A different approach is to add a foreign key relationship; this adds a calculated property and does not change the definitions of the existing properties. This approach is discussed in a [later subsection](#).

3.7.1 Modifying a Property Type via Drag and Drop

To modify the type of an existing property to be a class name:


1. In the **Class List** window, click the parent class (for example, `Sample.Orders`).
2. If necessary, scroll the **Fields Mapping** tab so that you can see the row that defines the property you want to modify.
3. Drag the child class (for example, `Sample.Customers`) from **Class List** and drop it onto the **Data Type** column in that row (for example, `Customer`).

The property type is modified to equal the name of the class you dragged (including package).

4. Click **Update** to save this change.

3.7.2 Loading Foreign Key Relationships from Microsoft Access

Microsoft Access stores foreign keys in a system table that can be queried. If you are using Access tables, you can retrieve that information and use it to automatically update the class. To do so:

1. Click the load relationships button () in the menu bar.
The system displays a dialog box.
2. For **DSN**, click the name of the SQL gateway connection to the database.
3. Click **Load Relationship**. If possible, the relationships are loaded. This immediately modifies the **Data Type** settings of any affected properties for all classes.
4. Click **OK**.
5. Click **Exit** to close the dialog box.

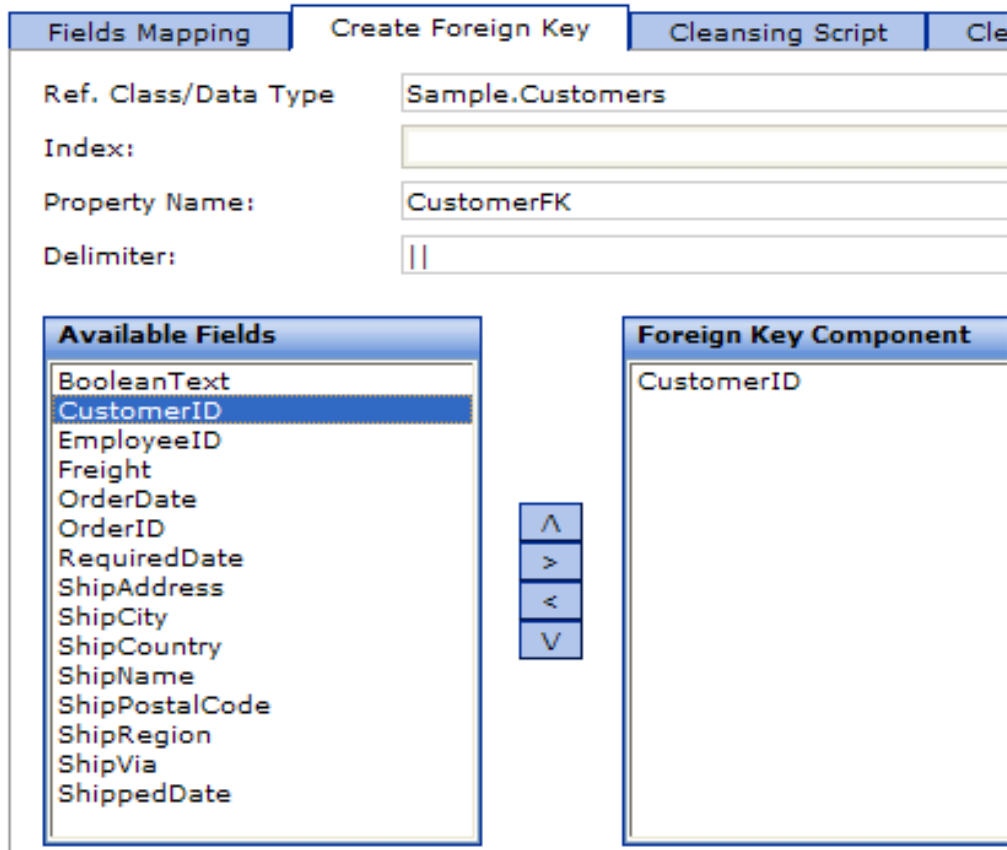
Note: This potentially affects all classes created by the Connector, not just the currently selected class.

3.7.3 Adding a Foreign Key

If you want to define a property relationship between two classes but you do not want to change the types of any existing properties, then add a foreign key instead. You can do this in the Studio as usual, or you can do it in the Connector.

To add a foreign key by using the Connector:

1. In the **Class List** window, click the parent class (for example, `Sample.Orders`).
2. Click the **Create Foreign Key** tab.
3. For **Ref. Class/Data Type**, type the name of the child class (for example, `Sample.Customers`).
4. For **Property Name**, type the name of the property that you are adding to the parent class (for example, `CustomerFK`).
5. For each property in the parent class that participates in the foreign key relationship, do the following:
 - a. Click the property name in **Available Fields**.
 - b. Click **>** to add this property to **Foreign Key Component**.



6. If you selected multiple properties in the previous step, optionally modify **Delimiter**. This is the delimiter used to combine these properties into a unique identifier.
7. Click **Create Foreign Key** to add this foreign key to the parent class. The foreign key is added immediately and this tab is cleared.

Now the **Fields Mapping** list includes another property (for example, `CustomerFK`) whose type is the child class (for example, `Sample.Customers`). In the Analyzer, this class becomes an expandable folder that contains the properties of the child class.

If you view this class in the Studio, notice that the new property is marked with the `Calculated`, `SqlComputed`, and `SqlComputeCode` keywords. The parent class also includes getter methods for this property.

3.7.3.1 Details on the Foreign Key Created by the Connector

You may find it helpful to see how this Connector option alters the class definition. For the previous example, the parent class (`Sample.Orders`) is updated to include one new property and three new methods:

```
Property CustomerFK As Sample.Customers [ Calculated,
    SqlComputeCode = {s {CustomerFK}=##class(Sample.Orders).
                        CustomerFKGetSQLID({CustomerID})},
    SqlComputed ];

Method CustomerFKGet() As Sample.Customers
{
    quit ##class(Sample.Customers).%OpenId(..CustomerIDGetObjectID())
}

Method CustomerFKGetObjectID() As %String
{
    quit ..CustomerIDGetObjectID()
}

ClassMethod CustomerFKGetSQLID(p1 As %Library.String) As %String
{
    q p1
}
```

3.7.3.2 Other Options for Accessing Properties in Other Classes

DeepSee makes it very easy to base dimensions and measures on properties of your BI-enabled class and (recursively) properties of its properties. However, you are not limited to using only those properties. When you use the Architect to define dimensions and measures, you can write Caché ObjectScript expressions to access any properties of any classes, in the same way that you would within your application itself.

3.8 Modifying the Class Definitions

If the preceding steps do not result in the exact class definitions that you need, you can modify the class definitions in the Studio. For example, you can add properties and methods.

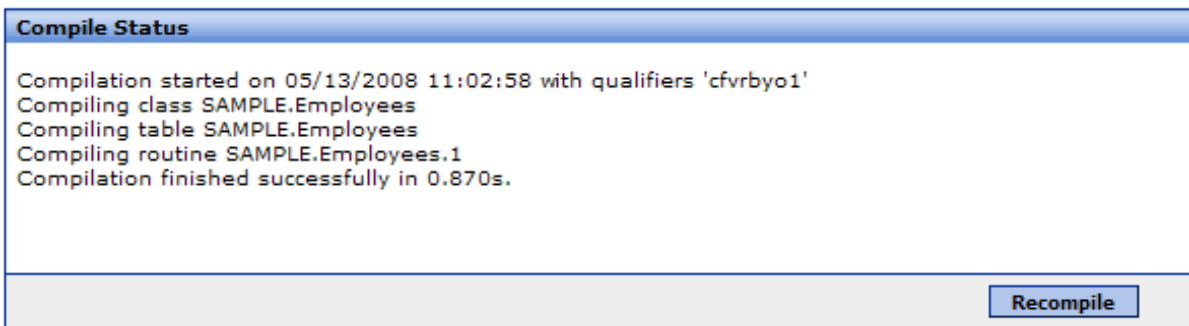
Depending on the complexity of your needs, the classes generated by the Connector may be only a starting point.

If you do modify the class definitions after generating them via the Connector wizards, do not use the Connector wizards again, unless you intend to overwrite your modifications.

3.9 Compiling the Classes

To compile the classes:

1. Click the **Status** tab.
2. Click **Recompile**. The **Compile Status** section shows the status of the compilation process:



You can also compile the classes in Studio, just as you would compile any other classes.

4

Modifying the Classes

After you have generated a class as described in the previous chapter, you can use the Connector to modify it. This chapter discusses the following topics:

- [How to modify a class](#)
- [An overview of the options](#)
- [How to add a standard index to a class for your own use](#)
- [How to specify advanced class options](#)
- [How to delete a class](#)

You can also modify the classes within Studio.

Because class properties have many options, they are discussed in the [next chapter](#).

As you perform the tasks in this chapter, you may want to load and examine data, both the locally stored data as well as the externally stored data. See the chapter “[Loading, Browsing, and Analyzing the Data](#),” later in this book.

If you do modify the class definitions after generating them via the Connector wizards, do not use the Connector wizards again, unless you intend to overwrite your modifications.

4.1 Modifying a Class

To modify a class in the Connector:

1. Click the class name in the **Class List** window.

The right side of the screen then displays details for that class.

2. Make changes as needed.
3. When you are done modifying a class, click **Update**.

When you do so, the Connector saves your changes and then displays the **Status** tab.

4. Optionally click **Recompile**.

You can also reopen the class wizard and make changes there. To do so, right-click the class name in the **Class List** window and click **Modify ETL Class**. The system displays the appropriate wizard, showing the details that currently apply to this class. For details, see the chapter “[Creating Classes Based on Loaded Data](#).”

4.2 Overview of the Class ETL Details Area

When you select a class in the left area of the Connector, the **Class ETL Details** area displays details for that class. The details are slightly different for classes based on tables and classes based on files. The following shows details for a class based on a table:

Class ETL Details

Class Name:

External Source:

Overwrite Existing Methods Batch Build (Delta Load)

Gateway:

Key:

OLAP [Mail](#) [Index](#) [Additional Options](#)

Fields Mapping

Create Foreign Key

Cleansing Script

Cleansing Rule

Load All Script

Populate

Status

Fields/Class Properties	External Source Field	Idx	Data Type	No Load	Custom Script	Transformation	Others
Discount	Discount	1	%String	<input type="checkbox"/>			Click
OrderID	OrderID	2	User.Orders	<input type="checkbox"/>			Click
ProductID	ProductID	3	User.Products	<input type="checkbox"/>			Click
Quantity	Quantity	4	%String	<input type="checkbox"/>			Click
UnitPrice	UnitPrice	5	%Numeric	<input type="checkbox"/>			Click

In contrast, for a class based on a file, the top area looks like this:

Class ETL Details

Class Name:

File Name: ...

Overwrite Existing Methods Batch Build (Delta Load)

Delimiter:

Key:

OLAP [Mail](#) [Index](#) [Additional Options](#)

This section provides an overview of this area.

4.2.1 Basics

The top area of **Class ETL Details** displays the following information, all of which applies to the class itself. You can edit most of this information.

Table 4–1: Basic Information in Class ETL Details

Field	Applies to ...	Purpose
Class Name	All classes generated by the Connector	Displays the package and class name for the class you selected. This information is read-only.
Gateway	Classes based on tables	Specifies the name of the SQL gateway connection used to access the data for this class.
External Source	Classes based on tables	Name of the table on which this class is based. You can load data from additional tables into this class, if they all have the same structure; see “ Specifying Advanced Class Options ” in the next chapter.
Key	All classes generated by the Connector	Comma-separated list of fields in the primary key.
Delimiter	Classes based on files	Delimiter that separates fields in the file.

Field	Applies to ...	Purpose
File Name	Classes based on files	Name of the file on which this class is based.
Overwrite Existing Methods	All classes generated by the Connector	Specifies whether the Connector should overwrite the existing internal BI-related methods when you click Update . In most cases, it is best to overwrite these methods to agree the current class. If you have modified the methods, however, you would want to carefully control when they were replaced.
OLAP	All classes generated by the Connector	Specifies whether this class is BI-enabled. If you select this, the class %BI.Adaptor is added to the superclass list.
Mail	All classes generated by the Connector	Allows you to send an email message that contains information about this class. Before you can use this, make sure the DeepSee email options have been set up, as described in the chapter “ Configuring the DeepSee Site ” in the <i>DeepSee Site Configuration and Maintenance Guide</i> .
Index	All classes generated by the Connector	<p>Specifies any standard (non-BI) indices in this class. DeepSee does not use these indices.</p> <p>You can click this link to display a dialog box where you can define and rebuild these indices. Here:</p> <ul style="list-style-type: none"> • Name is the name of the index. • Index is the name of the property that is indexed by this index. • Data lists any properties whose data is included in this index. <p>You can, of course, also edit the class in Studio. If an asterisk is displayed next to this link, that indicates that this class includes standard indices.</p>
Additional Options	All classes generated by the Connector	<p>Specifies advanced options; see “Specifying Advanced Class Options.”</p> <p>If an asterisk is displayed next to this link, that indicates that this class uses a non-default setting for at least one of these options.</p>

4.2.2 Tabs

The tabs in **Class ETL Details** are as follows:

Table 4–2: Tabs in Class ETL Details

Tab	Purpose
Fields Mapping	Defines the properties in this class. See the next chapter for details.
Cleaning Script	Specify optional processing to perform while loading each row. See the chapter “ Fine-tuning the Data Loading Methods .”
Cleaning Rule	Specify optional rules to use while loading each row. See the chapter “ Fine-tuning the Data Loading Methods .”
Load All Script	Specify optional processing to perform before and after loading all rows. See the chapter “ Fine-tuning the Data Loading Methods .”

Tab	Purpose
Populate	Enables you to run the Populate class method for this class. This tab is shown only if this class inherits from %Populate and is useful only in demo and test situations. To specify the number of records to create, type a number into No. of Records . To create the records, click Populate .
Status	Displays the status of the most recent data load and compilation actions.

4.3 Adding Standard Indices

When you define dimensions (in the Architect — see the book *Using the DeepSee Architect*), DeepSee automatically adds DeepSee bitmap indices. However, these indices are for use only by DeepSee. You can add standard indices for your own use; DeepSee does not use these indices.

To add a standard index in the Connector:

1. Click the class name in the **Class List** window.
The right side of the screen then displays details for that class. The **Index** link is displayed with an asterisk next to it if this class includes any standard indices; otherwise, no asterisk is shown.
2. Click **Index**. The system displays a dialog box that lists the standard indices defined in this class, if any.
3. Right-click and then click **Add Index**. The system adds a new, empty row.
4. In the new row, specify the following:
 - **Name** is the name of the new index. For example: `EmployeeIndex`
 - **Index** is the name of the existing property to which this index applies. For example: `EmployeeID`
 - **Data** lists any existing properties to include in this index. For example: `Region`

In this example, the Connector adds the following index to the class:

```
Index EmployeeIndex On EmployeeID [ Data = Region ];
```

5. Click **Save**.
6. When you are done modifying a class, click **Update**.

You can, of course, also edit the class in Studio.

On the same dialog box, you can also do the following:

- Build these indices. To do so, click **Build Indices**. This does not affect the DeepSee bitmap indices.
- Edit the details of an existing index. To do so, edit the row and then click **Save**.
- Delete an index. To do so, right-click it and then click **Delete Index**.
- Delete all standard indices. To do so, click **Clear** and then click **Save**.
- Discard any unsaved changes you have made on this dialog box. To do so, click **Clear**.
- Exit this dialog box without making any changes. To do so, click **Cancel**.

4.4 Specifying Advanced Class Options

The DeepSee Connector provides a set of advanced options that affect a class; these affect how data is loaded. To view and modify these options:

1. Click the class name in the **Class List** window.
The right side of the screen then displays details for that class. The **Additional Options** link is displayed with an asterisk next to it if any non-default values are used here; otherwise, no asterisk is shown.
2. Click **Additional Options**. The system displays a dialog box that shows the options.
3. Make changes as needed; see the table after the steps.
4. Click **OK**.
5. When you are done modifying a class, click **Update**.

The following list describes these additional class options:

- **Additional Filter** — Specifies a filter to apply when retrieving external data to load. Specify a boolean expression that refers to fields in the same external table; the system retrieves only records for which this expression is true.

This expression is added as a WHERE clause to the query that retrieves data from the external source. Therefore use literal values as appropriate for that source. For example, the following is suitable for Microsoft Access:

```
OrderDate < #2006-06-01#
```

If your expression uses any double quotes, double each of those so that they are properly escaped. That is, instead of " use ""

You can combine multiple expressions with AND or OR.

- **Asc Index Field** — Specifies the property in this class for the ascending incremental load option to use.
The ascending incremental load feature uses a specific field such as `OrderID`. First it checks the highest value in the local database for that field. When it selects external data to load, it ignores any records with lower values for that field. Then it loads the remaining records in ascending order by that field value.
To perform this kind of load, click **Asc Index** at the bottom of the screen.
- **Multiple Table Load** — Semi-colon-separated list of additional tables to load into this Cache table/class. These tables must be in the database used by this class, and must all have the same metadata as the basic table on which this class is based.
- **SQL** — Controls how records are inserted. If you select this option, records are inserted via an SQL INSERT. If this option is cleared (the default), records are inserted by direct update of the globals.
- **No Validation** — Applies only if you have selected **SQL**. If you select **SQL** and **No Validation**, the insert query uses the NOLOCK SQL directive and the compiler uses COMPILEMODE = NOCHECK.
- **Query** — Shows the query that this class is based on, if any. See “[Creating a Class for a Single Table or Query](#),” earlier in this book.
- **'Load All' Default No Deletion** — By default, when you run the load all option, all existing records for the class are deleted before any data is loaded. Select this check box if you do not want to first delete those records.
- **CDC** — CDC (Change Data Capture) is an Oracle feature, which captures changes to records, writing that information in supplemental table (a CDC table name). You can use this feature when loading data from an Oracle database.

To enable this feature, select **CDC** and optionally edit the **CDC Table**, which specifies the name of the table into which to write CDC data.

If you enable this feature, then instead of a full load, the system loads only the changed records as captured in the given CDC table.

4.5 Deleting a Class

To delete a class:

1. Right-click the class name in the **Class List** area and click **Delete Class**.
2. Click **OK** to confirm this action.
3. In the System Management Portal, delete the global that stores the data for this class. The name of the global is *Package.ClassD*, for example: `User.OrdersD`

5

Modifying the Class Properties

With the Connector, you can fine-tune the generated classes and modify the properties defined in them. This chapter discusses the following topics:

- [How to add a property to a generated class](#)
- [How to modify the data type of a property](#)
- [How to specify the value of a property](#)
- [How to delete a property](#)

You can also modify the classes within Studio; see *Using Caché Studio*.

As you perform these tasks, you may want to load and examine data, both the locally stored data as well as the externally stored data. See the next chapter, “[Loading, Browsing, and Analyzing the Data](#).”

5.1 Adding a Property

To add a property:

1. Right-click in the **Fields Mapping** tab and click **Add Field**.
The system add a new row to the table.
2. Type the details into the new row.
3. Click **Update** to save your changes.

5.2 Modifying the Data Type of a Property

On the **Fields Mapping** tab, the **Data Type** column specifies the data type of each property.

You can modify the data type of a property in either of the following ways:

- Drag a class name from the **Class List** hierarchy and drop it onto the **Data Type** field for a given property. The existing data type is automatically replaced with that class name.

For an example, see the [tutorial](#) earlier in this book.

- Click the **Data Type** field and select a new class (in the %Library package) from the drop-down list.

5.3 Specifying the Value of a Property

To specify the value of a property, specify one or more of the following options on the **Fields Mapping** tab:

- **External Source Field** — Optionally specifies the name of the field in the external table or file on which this property is based. Note that if **Custom Script** is specified, that overrides **External Source Field**.
- **Custom Script** — Optional Caché ObjectScript expression that specifies the value for the property, instead of the value in **External Source Field**. To refer to the current value of any field in the external data, use the following syntax:

```
{fieldname}
```

For example:

```
{FirstName} _ " " _ {LastName}
```

You can either type into this field or double-click it to display an editor that helps you build the expression. See “[Using the Script and Expression Editor](#),” later in this book.

- **Transformation** — Optionally select a transformation to apply to this property. For information on transformations, see [Using the DeepSee Architect](#).

5.4 Deleting a Property

To delete a property:

1. Right-click the property name in the **Fields Mapping** tab and click **Delete Field**.
2. Click **Update** to save your changes.

6

Using Multiple Databases for a Single Class

In some cases, the data that you want to use in DeepSee is stored in multiple databases. You can use the DeepSee multi site feature to load the data from each of those sources and combine it locally into one class. This chapter describes how to set up and use this feature:

- [How to enable the multi site feature](#)
- [How to set up a group of multiple sites](#)
- [How to create a class that uses a site group](#)
- [Differences in the generated class](#)

6.1 Enabling the Multi Site Feature

To enable the multi site feature:

1. Click **Administrator > Site Configuration**.
2. Click **ETL**.
3. Click **Multi Site?**.
4. Click **Save** to save your changes.

Your changes take effect immediately.

6.2 Setting Up a Site Group

A site group consists of a set of databases from which you intend to extract data in a common form to use as a single source for one or more classes within the Connector. The databases must contain a table that is defined in the same way in each of the databases; that table must have the same field definitions in each of the databases.

To set up a site group:

1. Do the following for each database that you want to use in this group:

- a. Define an ODBC data source name (DSN) for the external database. See the documentation for the external database for information on how to do this.
- b. Use the Caché SQL Gateway to create an SQL Gateway connection that uses this DSN. See the section “Creating an ODBC Gateway Connection” in *Using Caché with ODBC*.

When you are done, you might have two SQL Gateway connections like this:

Connection Name	DSN	User		
ordersdb1	OrdersDB1		Edit	Delete
ordersdb2	OrdersDB2		Edit	Delete

2. Click **Data Modeler > Multi Sites**.
3. For **Group Name**, type a string such as `Orders`. You use this name later to refer to the site group.
4. Right-click in the empty area of the **Sites** table and then click **Add Site**.
DeepSee adds a new, empty line to the table.
5. For **Name**, type a name that identifies a data source.
6. Double-click in the **Gateway** column of the same row and then click the gateway connection for that data source.
7. Repeat the previous steps until you have added all needed data sources to this group.
8. For **Index**, type an index value for this data source.
9. Click **Add** in the lower right.

The result might be as follows:

The screenshot shows the 'Multi Sites Definition' dialog box. On the left, a 'Site Group' list contains 'OrdersDatabases'. The main area is titled 'Multi Sites Definition' and contains a 'Group's Detail' section with 'Group Name' set to 'OrdersDatabases' and an empty 'Description' field. Below this is a 'Sites' table with the following data:

Name	Gateway	Index
OrdersDB1	ordersdb1	1
OrdersDB2	ordersdb2	2

6.3 Creating a Class That Uses a Site Group

To create a class that uses a site group, follow the steps described in the chapter “[Creating Classes Based on Loaded Data](#).” You can use any of the SQL gateways as a starting point. When the Connector displays the wizard where you can modify the class basics, notice that the **Multi Sites** option is no longer grayed out:

All Fields		
Select	Field/Property Name	>>>>Key
<input checked="" type="checkbox"/>	OrderID	>>>>>
<input checked="" type="checkbox"/>	CustomerID	>>>>>
<input checked="" type="checkbox"/>	EmployeeID	>>>>>
<input checked="" type="checkbox"/>	OrderDate	>>>>>
<input checked="" type="checkbox"/>	RequiredDate	>>>>>
<input checked="" type="checkbox"/>	ShippedDate	>>>>>
<input checked="" type="checkbox"/>	ShipVia	>>>>>
<input checked="" type="checkbox"/>	Freight	>>>>>
<input checked="" type="checkbox"/>	ShipName	>>>>>
<input checked="" type="checkbox"/>	ShipAddress	>>>>>
<input checked="" type="checkbox"/>	ShipCity	>>>>>
<input checked="" type="checkbox"/>	ShipRegion	>>>>>
<input checked="" type="checkbox"/>	ShipPostalCode	>>>>>
<input checked="" type="checkbox"/>	ShipCountry	>>>>>

Select All

Primary Key (ID Key)

OrderID

Delete Delete All

Slowly Change Dimension

Not Applicable

Foreign Version Derived Field

Multi Sites

Overwrite Use Wide

System Default

Copy Data
 No Validation
 OLAP

Process Cancel

Click in **Multi Sites** and then select the site group to use for this class.

6.4 Differences in the Class Definition

When you define a class that uses the multi site feature, note the following differences compared to a class defined without this feature:

- The ID property contains a string of the following form:

```
gateway_name|original_ID
```

Where *gateway_name* indicates the data source that is the source of the given record, and *original_ID* is the ID of that record as given in the original data source.

- The class also has a SiteID property, which indicates the data source that is the source of the given record.

7

Loading, Browsing, and Analyzing the Data

The Connector provides a set of tools to help you load and examine data, both the locally stored data as well as the externally stored data. These tools are useful during implementation, especially when you work with tables that are unfamiliar to you.

This chapter discusses the following:

- [How to use the Connector data analysis tool](#), which can help you determine the primary key for a table, determine whether a given field contains null values, and so on
- [How to load data into Caché and how to delete data from Caché](#)
- [How to browse either the local data in Caché or the externally stored data](#)
- [How to export data to Microsoft Excel or to XML documents](#)
- [How to save and reuse queries](#)

You can also load data by using the Scheduler, which is documented in the *DeepSee Site Configuration and Maintenance Guide*.

7.1 Analyzing the Data

As you define data cleaning and other processing, it is useful to have information about the range of values represented, the presence of nulls, and so on. The Connector provides an analysis tool that gives you such information. To use it:

1. Right-click the class name in the **Class List** area.
2. Then click **Data Analysis**.


The Connector then displays a dialog box like the following:

Table			
DSN:	<input type="text"/>		
Schema:	<input type="text" value="SAMPLE"/>		
Table:	<input type="text" value="Orders"/>		
		<input type="button" value="Load Field"/>	<input type="button" value="Analyze"/>
		<input type="button" value="Cancel"/>	

All Fields			
Select	Field/Property Name	Data Type	Analyze As
<input checked="" type="checkbox"/>	CustomerID	%String	String
<input checked="" type="checkbox"/>	EmployeeID	%Numeric	Number
<input checked="" type="checkbox"/>	Freight	%Numeric	Number
<input checked="" type="checkbox"/>	OrderDate	%TimeStamp	DateTime
<input checked="" type="checkbox"/>	OrderID	%String	String
<input checked="" type="checkbox"/>	RequiredDate	%TimeStamp	DateTime
<input checked="" type="checkbox"/>	ShipAddress	%String	String
<input checked="" type="checkbox"/>	ShipCity	%String	String
<input checked="" type="checkbox"/>	ShipCountry	%String	String
<input checked="" type="checkbox"/>	ShipName	%String	String
<input checked="" type="checkbox"/>	ShipPostalCode	%String	String
<input checked="" type="checkbox"/>	ShipRegion	%String	String
<input checked="" type="checkbox"/>	ShipVia	%Numeric	Number
<input checked="" type="checkbox"/>	ShippedDate	%TimeStamp	DateTime

Select All

Tip:

To display this dialog box, you can also click the data analysis button (). If you do, however, you must manually specify the query and the DSN (if needed).

For each field in this class, this dialog box lists the data type.

- Optionally clear the check box next to any field in which you are not interested.
- Optionally modify the analysis type for any fields, which controls how the fields are analyzed. For example, if you analyze a field as a number, DeepSee shows its maximum and minimum values; if you instead analyze it as a date, DeepSee shows its earliest and latest values. See the following subsections for details.

To modify the analysis type for a field, click in the **Analyze As** column and then click **String**, **Number**, or **DateTime**.

- Click **Analyze**.

The Connector then displays a report about the data in these fields. The top area indicates the schema and table name, the number of fields analyzed, and the number of records analyzed. Below that, the report contains one or all of the following sections:

7.1.1 String

The **String** section displays the following information for all fields analyzed as strings:

- **Field Name** — Name of the field analyzed in this row of the report.
- **Null** — Number of records that contain a null value for this field.
- **Unique** — Number of unique values in this field, for all records.
- **Uniqueness** — A score (from 0 to 100) that indicates how nearly unique the values are in this field. If this score is 100, this field is uniquely valued (at least within the data retrieved by the query). If **Uniqueness** is 100, that strongly suggests that this field is the primary key field for this table.

The following shows an example:

String			
Field Name	Null	Unique	Uniqueness
CustomerID	0	89	10.72
OrderID	0	830	100.00
ShipAddress	0	89	10.72
ShipCity	0	70	8.43
ShipCountry	0	21	2.53
ShipName	0	89	10.72
ShipPostalCode	19	85	10.24
ShipRegion	507	20	2.41

7.1.2 Number

The **Number** section displays the following information for all fields analyzed as numbers:

- **Field Name** — Name of the field analyzed in this row of the report.
- **Minimum** — Lowest value contained in this field.
- **Maximum** — Highest value contained in this field.
- **Median** — Median value of this field.
- **Mean** — Mean value of this field.
- **Var** — Variance of this field.
- **SD** — Standard deviation of the values in this field.
- **Sk** — Skewness of the values in this field.
- **Kur** — Kurtosis of the values in this field.
- **SE** — Standard error of the values in this field.
- **COV** — Coefficient of variance of the values in this field.

7.1.3 Date Time

The **Date Time** section displays the following information for all fields analyzed as date-time values:

- **Field Name** — Name of the field analyzed in this row of the report.

- **First Date** — Earliest date from all the values in this field.
- **Last Date** — Latest date from all the values in this field.

7.2 Loading or Deleting Data

When you use the Connector wizards to generate classes for external data, those wizards copy the data into the Caché database by default. (To control this, you use the **Load Data** check box in the wizards.)

In a production system, you use the DeepSee Scheduler to automate data loading. During implementation, however, you often need to delete and reload data. You can do this within the Connector.

Within the Connector, you can load data into Caché or delete data from Caché for a given class as follows:

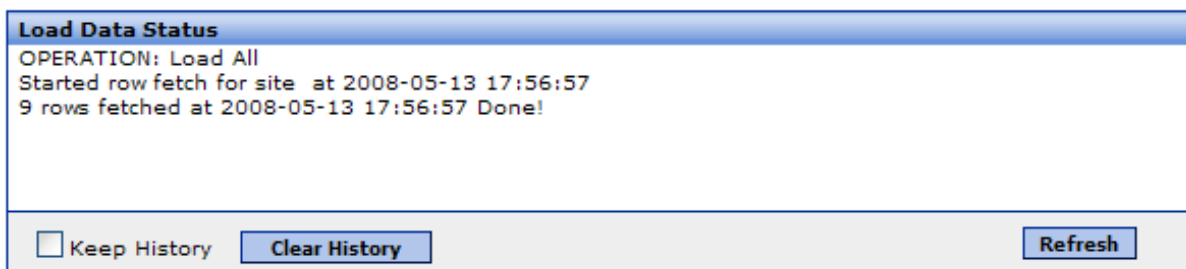
1. Click the class name in the **Class List** area.
2. Then, to load or delete records in Caché for this class:
 - To load all records from the external source, click **Load All**. The system first deletes all existing records for this class in Caché.
 - To load records using the ascending incremental load feature, click **Asc Index**.
The ascending incremental load feature uses a specific field such as `OrderID`. First it checks the highest value in the local database for that field. When it selects external data to load, it ignores any records with lower values for that field. Then it loads the remaining records in ascending order by that field value.

To enable this feature, see “[Specifying Advanced Class Options](#).”

- To load only records that have changed, click **Delta Load**.
- To load a single record, click **Load One Data**.
The system displays a dialog box. For **Load One Data ID**, specify the ID of the record to load. Then click **Load**. Repeat as necessary. When you are done, click **Exit**.
- To delete all existing records, click **Delete Data**. You are prompted to confirm this action.

None of these actions affect any externally stored data.

3. If you click **Load All** or **Delta Load**, the Connector displays the **Status** tab and updates the **Load Data Status** section. For example:



By default, this section shows only the results for the most recent load operation. To enable the Connector to keep history, click **Keep History**.

To clear the history, click **Clear History**.

To refresh the display (for example, during a lengthy load operation), click **Refresh**.

7.3 Browsing the Data

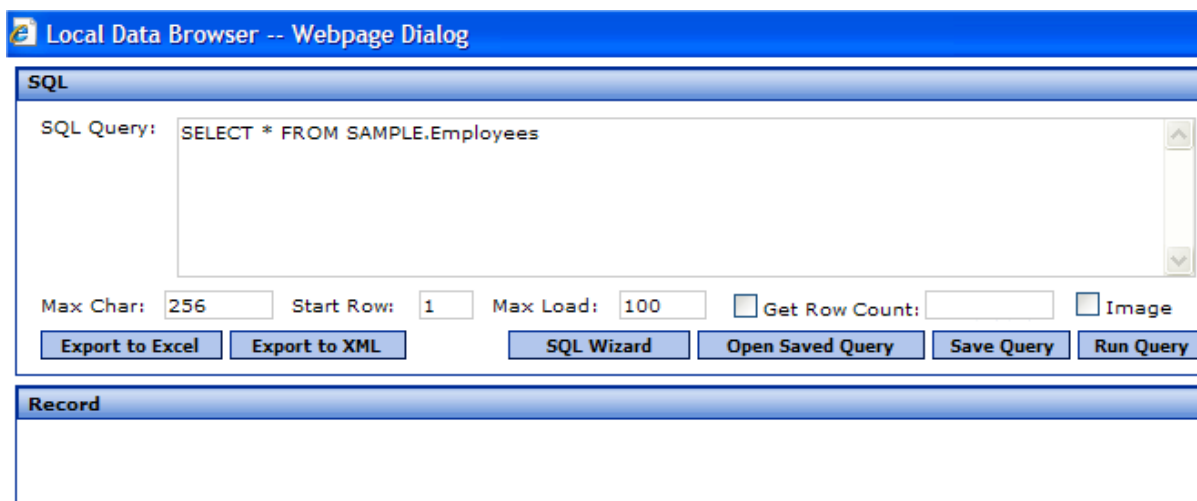
The Connector provides an easy way to browse the local data as well as the external data (unless the external data is in a file). This is particularly useful when you are adjusting data cleaning scripts and so on. The user interface is slightly different for local data and external data; this section discusses both versions.

To browse the data corresponding to a given class:

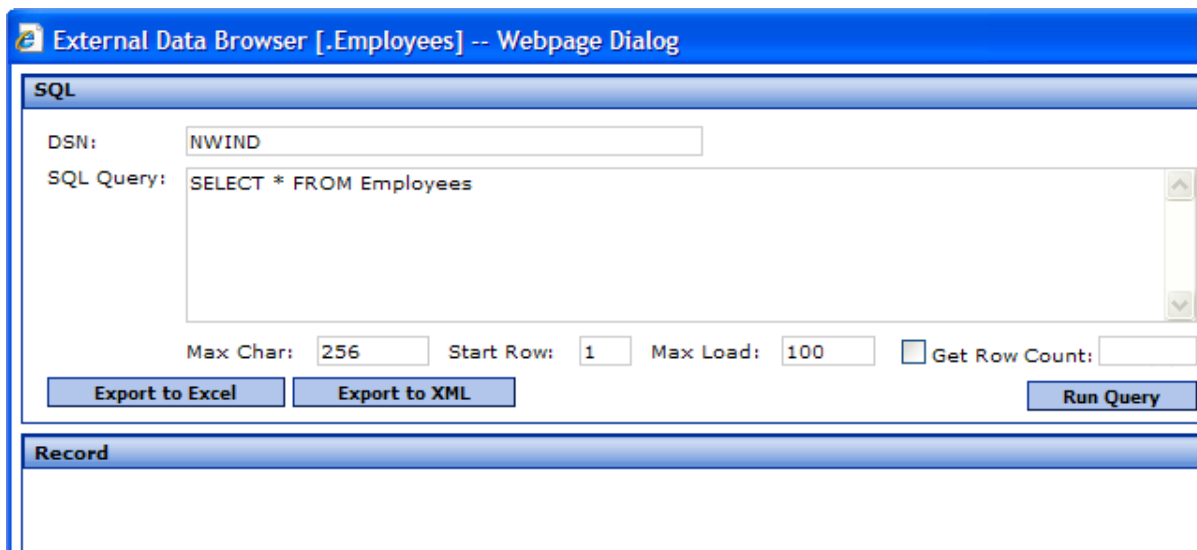
1. Right-click the class name in the **Class List** area.
2. Then click **Browse Local Data** or **Browse External Data**.

(If the context menu does not list **Browse External Data**, this class is based on data from a file. To browse the external data, open that file with a text editor.)

The Connector then displays a dialog box like the following:



If the data is external, the dialog box is slightly different:



3. Optionally edit the following details:
 - **SQL Query** — Specifies the SQL query to execute.

- **Max Char** — Specifies the maximum number of characters to retrieve for any given field. Any characters after that are not shown. The default is 256.
- **Start Row** — Specifies the row ID of the first row to retrieve. The default is 1.
- **Max Load** — Specifies the maximum number of rows to retrieve. The default is 100.
- **Get Row Count** — Specifies that the system should display the total number of rows in this table. If you select this, the row count is shown in the field to the right of this check box, after you run the query. This option is cleared by default.
- **Image** — Affects the display of fields that contain images stored as %GlobalBinaryStream. If you select this, the images are shown in the results. Otherwise, the string %GlobalBinaryStream is shown. This option is cleared by default.

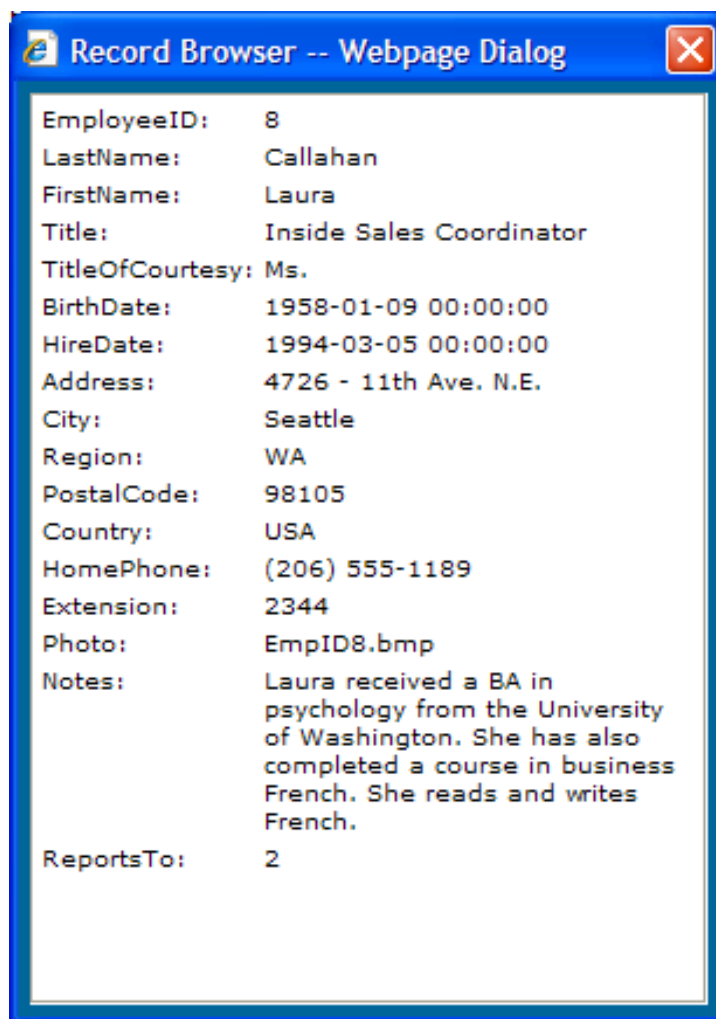
4. Click **Run Query**.

The bottom area then displays results of the query in tabular format.

Record						
Row	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate
1	1	Davolio	Nancy	Sales Representative	Ms.	1968-12-08 00:00
2	2	Fuller	Andrew	Vice President, Sales	Dr.	1952-02-19 00:00
3	3	Leverling	Janet	Sales Representative	Ms.	1963-08-30 00:00
4	4	Peacock	Margaret	Sales Representative	Mrs.	1958-09-19 00:00
5	5	Buchanan	Steven	Sales Manager	Mr.	1955-03-04 00:00
6	6	Suyama	Michael	Sales Representative	Mr.	1963-07-02 00:00
7	7	King	Robert	Sales Representative	Mr.	1960-05-29 00:00
8	8	Callahan	Laura	Inside Sales Coordinator	Ms.	1958-01-09 00:00
9	9	Dodsworth	Anne	Sales Representative	Ms.	1969-07-02 00:00

You can scroll horizontally to see all the fields. If the query returns too much data to display in a single screen, you can page through it.



Also, if you double-click a row, the system displays a dialog box showing the data for just that row.



For information on **Export to Excel** and **Export to XML**, see “[Exporting Data.](#)”

For information on **Open Saved Query** and **Save Query**, see “[Creating Saved Queries.](#)”

Tip:

To display this dialog box, you can also click the browse local data button () or the browse DSN data button () in the menu bar. If you do, however, you must manually specify the query and the DSN (if needed).

7.4 Exporting Data

If you want to take a closer look at either the local data or the external data (except for data in files), you can export it from the Connector. To do so:

- Browse the local or external data as described in the [previous section](#). Then click **Export to Excel** or **Export to XML**.
- Right-click the class name in the **Class List** area and then click **Export to Excel** or **Export to XML**.

The data is exported to the DeepSee working directory, which is typically *install-dir\CSP\sys\bi\work\namespace*. See the [DeepSee Site Configuration and Maintenance Guide](#).

7.5 Creating Saved Queries for Local Data

So that you can easily explore data during implementation (for example, to check your data cleaning), the Connector provides options you can use to save and reuse SQL queries. These options apply only to local data.

This section discusses the following:

- [How to save a query](#)
- [How to open a saved query and rerun it](#)
- [How to modify a saved query](#)
- [How to delete a saved query](#)

7.5.1 Saving a Query

You define and save SQL queries within the local data browse option.

1. Right-click a class name in the **Class List** area.
2. Then click **Browse Local Data**.
3. Optionally edit the query displayed in the **SQL Query** field.
4. Click **Save Query**.

The system then displays a dialog box, which shows the default query used for this class. For example:

```
SELECT * FROM Sample.Orders
```

5. Optionally edit the query displayed in the **Query** field.
6. Type a name into **Name**.
7. For **Folder**, click the browse button (...) and select a folder in which to store this query.
8. Optionally type a description into **Description**.
9. Click **Save** to save the query and close this dialog box.
10. Optionally click **Run Query**.

The bottom area then displays results of the query.

11. When you are done, click the X in the upper right to close the data browse dialog box.

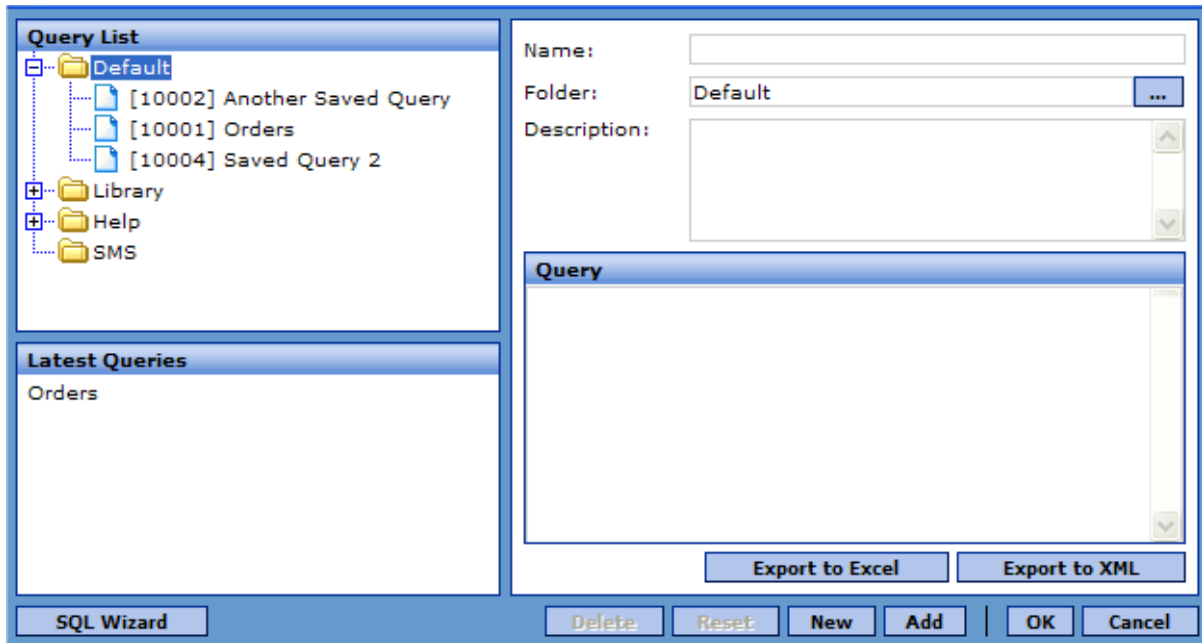
Tip: You can also create a new query from scratch on the dialog box where you modify saved queries. See “[Modifying a Saved Query](#).”

7.5.2 Opening a Saved Query

To open a saved query and run it:

1. Right-click a class name in the **Class List** area.
2. Then click **Browse Local Data**.
3. Click **Open Saved Query**.

The system then displays a dialog box where you can open a saved query, modify a saved query, or define a new saved query.



4. Select a query in either of the following ways:
 - Expand folders as needed in **Query List** and then click a query name.
 - Click the name of a recently run query in the **Latest Queries List** list.
5. Click **OK** to close this dialog box and return to the data browse dialog box.
6. Click **Run Query**.

The bottom area then displays results of the query.

7. When you are done, click the X in the upper right to close the data browse dialog box.

7.5.3 Modifying a Saved Query

To modify a saved SQL query:

1. Right-click a class name in the **Class List** area.
2. Then click **Browse Local Data**.
3. Click **Open Saved Query**.
4. Select a query in either of the following ways:
 - Expand folders as needed in **Query List** and then click a query name.
 - Click the name of a recently run query in the **Latest Queries List** list.
5. Edit the query by changing any or all of the following in the upper right:
 - **Name**
 - **Folder**
 - **Description**

- **Query**

Tip: If you want to cancel your changes, click **Reset**.

6. Click **Update**. The query is immediately saved with the new information.
7. Click **OK** to close this dialog box and return to the data browse dialog box.
Or click **Cancel** to simply close this dialog box.
8. If you selected the modified query, optionally click **Run Query**.
The bottom area then displays results of the query.
9. Click the X in the upper right to close the data browse dialog box.

Tip: You can also create a new query from scratch on this dialog box. To do so, click **New**, enter information in the upper right and then click **Add**.

7.5.4 Deleting a Saved Query

To delete a saved query:

1. Right-click any class name in the **Class List** area.
2. Then click **Browse Local Data**.
3. Click **Open Saved Query**.
4. Select a query in either of the following ways:
 - Expand folders as needed in **Query List** and then click a query name.
 - Click the name of a recently run query in the **Latest Queries List** list.
5. Click **Delete**. The query is immediately deleted.
6. Click **Cancel** to close this dialog box and return to the data browse dialog box.
7. Click the X in the upper right to close the data browse dialog box.

8

Fine-tuning the Data Loading Methods

This chapter discusses the following topics:

- [An overview of how you can modify the behavior of the data loading methods](#)
- [How to run scripts before or after loading data for a class](#)
- [How to run scripts on each row as it is being loaded](#)
- [How to apply cleaning rules that either modify or flag data as it is being loaded](#)
- [How to use the script and expression editor](#)

Also see “[Specifying Advanced Class Options](#)” earlier in this book; many of these advanced options affect data loading.

You can execute these methods by using the Scheduler, which is documented in the [DeepSee Site Configuration and Maintenance Guide](#).

8.1 Overview

When the Connector generates or updates a class, by default it generates or updates class methods that load and delete data for this class. These methods are used by the load and delete options in the Connector, as well as by the DeepSee Scheduler.

You can fine-tune the behavior of these methods for each class by adding Caché ObjectScript to key points. To do so, you specify configuration options within the Connector.

8.1.1 Overall Flow of Data Loading

When data is loaded for a given table, the overall flow is as follows:

1. Transaction mode is disabled, if it was enabled.
2. Journaling is disabled, if it was enabled.
3. If you have defined custom processing to occur before all data loading, that processing is performed.
4. The system initializes the connection to the external database, if applicable.
5. The system then iterates through the rows. For each row:
 - a. If you defined any pre-transformation processing, the system performs it.
 - b. If you defined any cleaning rules, the system uses them.

- c. If you defined transformations to apply to this class, the system uses them.
 - d. If you defined any post-cleaning rule processing, the system performs it.
6. The system closes the connection to the external database, if applicable.
 7. If you have defined custom processing to occur after all data loading, that processing is performed.
 8. Journaling is re-enabled, if appropriate.
 9. Transaction mode is reestablished, if appropriate.

8.1.2 Summary of Connector Options for Customizing Data Loading

The following table summarizes the locations of the relevant configuration options, which are discussed later in this chapter (except where noted):

Table 8–1: Custom Processing within Data Loading

Custom Processing	Where to Specify	Relevant Documentation
Custom processing before data loading (step 3)	Load All Script tab, Pre Load All Script option	“Defining Load All Scripts”
Pre-transformation processing, per row (step 5a)	Cleaning Script tab, Pre Transformation Code option	“Defining Cleaning Scripts”
Cleaning rules, per row (step 5b)	Cleaning Rule tab	“Defining Cleaning Rules,” earlier in this book
Transformations, per row (step 5c)	Fields Mapping tab, Transformation option	Using the DeepSee Architect
Post-cleaning rule processing , per row (step 5d)	Cleaning Script tab, Post Cleaning Rule Code option	“Defining Cleaning Scripts”
Custom processing after all data is loaded loading (step 7)	Load All Script tab, Post Load All Script option	“Defining Load All Scripts”

8.2 Defining Load All Scripts

For each class, you can define a script to run before any rows are loaded and another script to run after all rows are loaded.

To define these scripts for a given class:

1. Click the class name in the **Class List** window.
2. Click the **Load All Script** tab.
3. Type into one of the following fields:
 - **Pre Load All Script** — Specify Caché ObjectScript statements to execute just before starting to load the data for this class.
 - **Post Load All Script** — Specify Caché ObjectScript statements to execute just after loading all the data.

Or click the **Edit** option for the field; this displays the editor [described later in this chapter](#).

You can use multiple statements but do not include manual line breaks. (That is, type on a single line. The text wraps automatically.)

These statements cannot refer to data, because there is no access to data at these points.

4. When you are done, click **Update**.

When you do so, the Connector saves your changes and then displays the **Status** tab.

5. Optionally click **Recompile**.

The following shows a simple example:

```
write "***Write this before loading data for this class", !
```

Note: These options affect the `zzLoadAll()` method.

8.3 Defining Cleaning Scripts

For each class, you can define two cleaning scripts that are run for each row as it is being loaded:

- A script that runs before any transformations that are applied
- A script that runs after the transformations and after the cleaning rules (see “[Defining Cleaning Rules](#),” earlier in this book)

To define the cleaning scripts for a given class:

1. Click the class name in the **Class List** window.
2. Click the **Cleaning Script** tab.
3. Type into one of the following fields:
 - **Pre Transformation Code** — Specify Caché ObjectScript statements to execute before the transformations are applied for this class.
 - **Post Cleaning Rule Code** — Specify Caché ObjectScript statements to execute after the transformations are applied for this class and after the cleaning rules are used.

Or click the **Edit** option for the field; this displays the editor [described later in this chapter](#).

In both cases, the script must consist of one or more Caché ObjectScript statements.

To refer to the current value of a given property of this class (for the current row), use the syntax `{propertyname}`, where *propertyname* is the name of a property in this class.

Or, to refer to a field, use the syntax `F(i)`, where *i* is field position as shown in the Connector.

4. When you are done, click **Update**.

When you do so, the Connector saves your changes and then displays the **Status** tab.

5. Optionally click **Recompile**.

Note: These options affect the `zzCleanse()` method.

8.4 Defining Cleaning Rules

For each class, you can define a set of cleaning rules, which are all used when any row is being loaded for that class. DeepSee uses two categories of cleaning rules:

- A code-type cleaning rule executes Caché ObjectScript statements, which typically modify the data for this row.
- A cleaning rule of any other type evaluates a boolean Caché ObjectScript expression and then, if the statement is true, logs a message and optionally rejects the row. Note that this kind of cleaning rule does not affect the data, apart from possibly rejecting it.

Note: By default, DeepSee can send an automatic email message when using a cleaning rule. This option works only if the DeepSee email options have been set up, as described in the chapter “[Configuring the DeepSee Site](#)” in the *DeepSee Site Configuration and Maintenance Guide*.

To define these rules for a given class:

1. Click the class name in the **Class List** window.
2. Click the **Cleaning Rules** tab.
3. Right-click and then click **Add Condition**.
4. To create a code-type cleaning rule:

- For **Type**, select **Code**.
- For **Expression**, type a Caché ObjectScript statement. Or double-click this field to display the editor [described later in this chapter](#).

To refer the current value of a given property; use the syntax `{propertyname}`, where *propertyname* is the name of a property in this class. Remember that this statement is executed once for each row as it is being loaded.

5. Or to create other types of rules:

- For **Type**, select **Reject**, **Accept**, **Warning**, or **Mail**. The type of rule affects how the record is handled:
 - **Reject** — The data is not saved to the Caché database, and a reject-type message is logged.
 - **Accepted** — The data is saved to the Caché database, and an accept-type message is logged.
 - **Warning** — The data is saved to the Caché database, and a warning-type message is logged.
 - **Mail** — The data is saved to the Caché database, and a mail-type message is logged, and email is sent to the default send-to address, if that has been configured.

For information on logged messages, see “[Viewing the Cleaning Rule Logs](#),” later in this chapter.

- For **Expression**, type a boolean-valued Caché ObjectScript expression. Or double-click this field to display the editor [described later in this chapter](#).

To refer the current value of a given property; use the syntax `{propertyname}`, where *propertyname* is the name of a property in this class. Remember that this expression is evaluated once for each row as it is being loaded.

6. Optionally type a description into the **Description** field.

For non-code-type rules, this description is used in the log.

7. Repeat the preceding steps to add as many rules as needed.
8. When you are done, click **Update**.

When you do so, the Connector saves your changes and then displays the **Status** tab.

- Optionally click **Recompile**.

Note: These options affect the `zzCleanse()` method.

8.4.1 Other Options

You can also perform the following tasks on this tab:

- Disable the automatic email sent when this rule is executed. To do so, clear the check box in **Mail** column.
- Deactivate a cleaning rule. To do so, clear the **Active** check box next to it. To activate it, select the check box.
- Deactivate all cleaning rules in this class. To do so, right-click and then click **Deactivate All Conditions** check box next to it. To activate them all, right-click and then click **Activate All Conditions**.
- Insert a new cleaning rule before a given rule. To do so, right-click a rule and then click **Insert Condition**.
- Delete a cleaning rule. To do so, right-click it and then click **Delete Condition**.
- Test the active cleaning rules. For details, see the next subsection.
- View the cleaning rule logs. This is discussed in a [later subsection](#).

After you add, delete, modify, activate, or deactivate a rule, you must recompile the class.

Ignore the **Business Rule** option.

8.4.2 Testing Cleaning Rules

To test the active cleaning rules, load some rows for this class:

- Optionally edit **Start Row** at the bottom of the screen. This specifies the row ID of the first row to load. The default is 1.
- Optionally edit **Max Load**. This specifies the maximum number of rows to load.
- Click **Test Run**. The system deletes existing rows for this class and loads rows as specified.

Note: The **Status** tab does not show the status of this processing.

- Do some or all of the following to verify that your rules behaved as expected.
 - Browse the loaded data. For information on browsing local data, see “[Browsing the Data](#)” earlier in this book.
 - View the cleaning rule logs, as described in the next subsection.

8.4.3 Viewing the Cleaning Rule Logs

For convenience during development, the cleaning rule logs display messages of different types for the most recent time that you used **Load All** or **Test Run**.

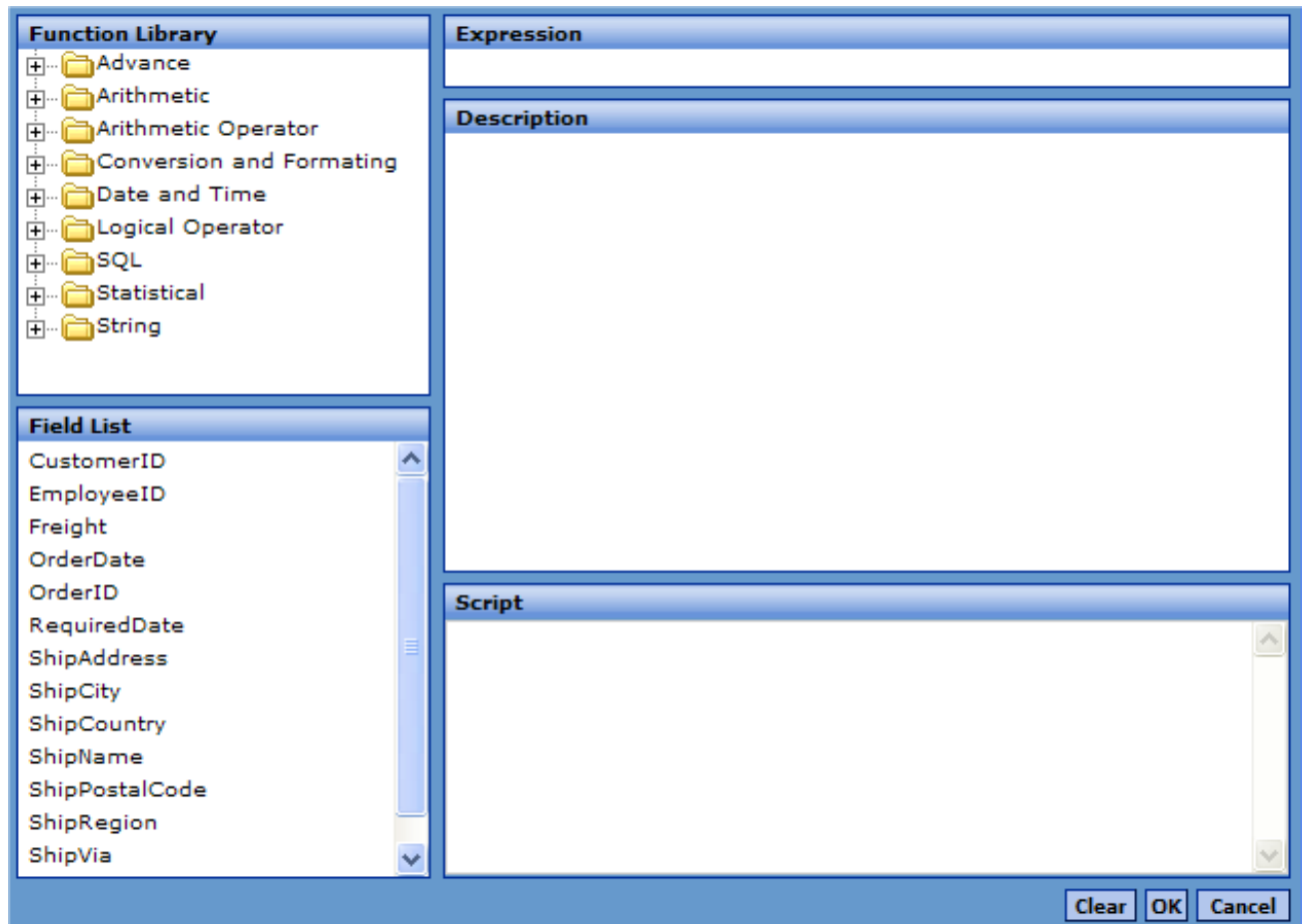
To see these logs, use the links at the bottom of the **Cleaning Rules** tab:

- Reject** — Displays a message for each row that was rejected and flagged due to a cleaning rule of type **Reject**.
- Accept** — Displays a message for each row that was flagged by a cleaning rule of type **Accept**.
- Mail** — Displays a message for each row that was flagged by a cleaning rule of type **Mail**.

- **Warning** — Displays a message for each row that was flagged by a cleaning rule of type **Warning**.
- **Error** — Displays errors detected in other ways.

8.5 Using the Script and Expression Editor

The Connector provides the following dialog box (or a variation of it) to help you create Caché ObjectScript expressions and scripts:



On this dialog box, you can do the following:

- Expand the folders and scroll in the **Function Library** panel to see functions that you can use in the expression.
- Click a function in the **Function Library** panel. When you do, **Expression** shows the synopsis for that function and **Description** shows help for it.
- Double-click a function in **Function Library**. When you do so, it is automatically added to the **Script** panel.
- Scroll in the **Field List** panel to see the properties of this class.
This panel is not displayed for load all scripts, which cannot use properties.
- Double-click a field in **Field List**. When you do so, it is automatically added to the **Script** panel, enclosed by curly braces ({}).
- Edit directly in the **Script** panel.

- Clear the contents of the **Script** panel. To do so, click **Clear**.
- Accept the current contents of the **Script** panel and close the dialog box. To do so, click **OK**.
- Close the dialog box without making any changes. To do so, click **Cancel**.

