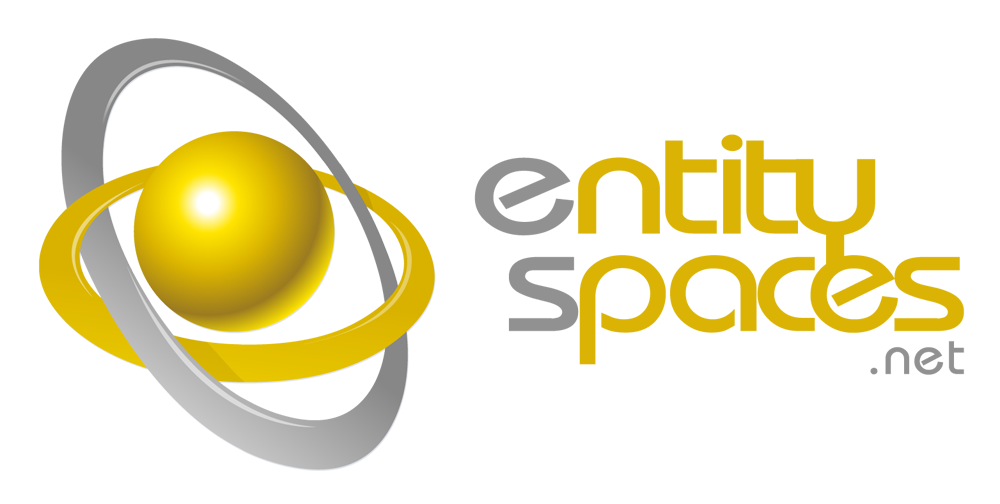
EntitySpaces 2009 Q3 Beta Release Notes  
2009.2.0831.0

**[](http://www.entityspaces.net/)**  
  
  
Persistence Layer and Business Objects for Microsoft .NET

Contents

[Installation Instructions 3](#_Toc239466773)

[Breaking Changes 3](#_Toc239466774)

[The New EntitySpaces.DynamicQuery assembly 3](#_Toc239466775)

[Enums that changed Namespaces 3](#_Toc239466776)

[DynamicQuery Sub Operator Order Differences 3](#_Toc239466777)

[Query "es" Property Change 4](#_Toc239466778)

[DynamicQuery esWhereItem and esSelectItem Renamed 4](#_Toc239466779)

[Enhancements and Fixes 4](#_Toc239466780)

[Conversion of your UserData.xml file 4](#_Toc239466781)

[Editing User Metadata 5](#_Toc239466782)

[Sybase SQL Anywhere Support 6](#_Toc239466783)

[Connection Configuration Information Enhancements 6](#_Toc239466784)

[The Having Clause is now Supported 6](#_Toc239466785)

[DynamicQuery Supports “Raw SQL” Everywhere 7](#_Toc239466786)

[Full Expressions in OrderBy and GroupBy 7](#_Toc239466787)

[Silverlight Support 8](#_Toc239466788)

[DynamicQuery Helper Methods for Serialization 8](#_Toc239466789)

[The EntitySpaces Generic Concurrency Mechanism 8](#_Toc239466790)

[Proxy Stub Enhancements 9](#_Toc239466791)

[VistaDB Password Bug Fix 9](#_Toc239466792)

[DynamicQuery Daisy Chaining Syntax has been fixed 10](#_Toc239466793)

[DynamicQuery Order of Values More Flexible 10](#_Toc239466794)

[There is now a Twitter toolbar button on the “Whats New” page 10](#_Toc239466795)

[AND NOT / OR NOT Added To DynamicQuery Syntax 11](#_Toc239466796)

# Installation Instructions

You cannot run this beta side by side with the production version or any other version of EntitySpaces 2009. You MUST uninstall any prior versions before installing. You **WILL NEED** to regenerate all your generated classes and we recommend you regenerate your stored procedures if you are using those for your CRUD operations.

# Breaking Changes

## The New EntitySpaces.DynamicQuery assembly

There is a new assembly named EntitySpaces.DynamicQuery.dll. You will have to add a reference to it in your project(s).

## Enums that changed Namespaces

The enums listed below moved from the EntitySpaces.Interfaces namespace to the EntitySpaces.DynamicQuery namespace. For the most part just adding a using or Imports statement to the “EntitySpaces.DynamicQuery” namespace will solve any compile errors. However, if you used the full namespace to access these enums such as “EntitySpaces.Interfaces.esQueryType” you may need to do a find/replace. These are the enums that have changed:

* esQueryType
* esQuerySubOperatorType
* esOrderByDirection
* esComparisonOperand
* esConjunction
* esParenthesis
* esSubquerySearchCondition
* esJoinType
* esArithmeticOperator
* esCastType
* esSystemType

You may only use one or two of these enums in your code, so the impact probably is not that great.

## DynamicQuery Sub Operator Order Differences

We were spitting out the sub operators in the reverse order, which typically was not noticeable. For instance

q.Where( q.LastName.ToLower().Trim() );

would first do the Trim, then the ToLower, as of this release we now do the ToLower first then the Trim. This change will not cause compile errors, but depending on the sub operators used, may cause a different result set to be returned from the generated SQL query.

## Query "es" Property Change

There is a minor change in the DynamicQuery API with the ".es" property. This is only for the DynamicQuery class, the Entity and Collection classes have no change in this area.

Before the Alpha you could do this:

EmployeesQuery query = new EmployeesQuery();  
 query.es.Connection.Name = "SomeName";

Now the connection is accessible this way ...

EmployeesQuery query = new EmployeesQuery();  
 query.es2.Connection.Name = "SomeName";

\*\* Notice we use "es2" to get to the Connection. All of the other properties are still available under the ".es" property.

## DynamicQuery esWhereItem and esSelectItem Renamed

esWhereItem was renamed to esComparison and esSelectItem was renamed to esExpression. They both now live in EntitySpaces.DynamicQuery.dll. Unless you are manually creating these classes, which is rare, you will not be affected.

# Enhancements and Fixes

## Conversion of your UserData.xml file

If you use user metadata to for aliases, oracle sequences, or other reasons, we suggest you back up your file. Typically, you would have this checked into your source control system. We do make a backup for you before conversion however. The conversion will take place upon the first access of the metadata.

The old format looked like this:

**<esUserData>**

**<Databases>**

**<Database p="MYGENERATION" n="">**

**<Tables>**

**<Table p="OracleTest" n="OracleTest">**

**<Properties>**

**<Property k="AUTOKEY:Id" v="seq\_OracleId" />**

**<Property k="CONCURR:ConcurrencyCheck" v="1" />**

**</Properties>**

**</Table>**

**</Tables>**

**</Database>**

**</Databases>**

**</esUserData>**

The New Format (below) looks like this. Notice the Driver= “SQL” and the EntitySpaces Version number.

**<esUserData Version="2009.2.0831.0">**

**<Drivers>**

**<Driver Name="SQL">**

**<Databases>**

**<Database Name="MYGENERATION">**

**<Tables>**

**<Table Name="OracleTest" Alias="OracleTest">**

**<Columns>  
 <Column Name="Id" IsAutoKey="True" AutoKeyText="seq\_OracleId" />**

**<Column Name="ConcurrencyCheck" Alias="ConcurrencyCheck"   
 IsEntitySpacesConcurrency="True" />**

**</Columns>**

**</Table>**

**</Tables>**

**</Database>**

**</Databases>**

**</Driver>**

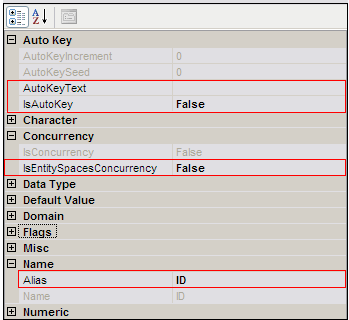
**</Drivers>**

**</esUserData>**

This allows you to work with many different database systems and not worry about having conflicts if the Database names match between systems and so on. Customers who are using the EntitySpaces Provider Independence features will find this very handy.

## Editing User Metadata

On the Metadata tab we now use the .NET Property Grid to display the metadata for columns, tables, views and so on. You also now edit your User Metadata through the property grid. The values you can edit are shown below. The columns highlighted in red are the only ones you can edit, there are some values under the “Flags” section that are editable but they are not implemented and may be removed before the production release.



Your Oracle Sequences will show up in the **AutoKeyText** and **IsAutoKey** columns.

## Sybase SQL Anywhere Support

We now support Sybase SQL Anywhere. All of our development was done with Sybase SQL Anywhere 11. For more on this support, and how to switch version numbers, see this [post on our forums](http://community.entityspaces.net/forums/thread/15560.aspx).

## Connection Configuration Information Enhancements

This enhancement will be much appreciated. In the past, when you used an App.Config or Web.Config file for your EntitySpaces connection information, it was read only. This is no longer true. Now you can modify all of the connection information at runtime. We also added two new optional attributes on the connection, “schema” and “catalog”.

<add name="SQLDynamic"   
   providerMetadataKey="esDefault"   
   sqlAccessType="DynamicSQL"   
   provider="EntitySpaces.SqlClientProvider"   
   providerClass="DataProvider"   
   connectionString="yada yada;"   
   databaseVersion="2005"   
**schema="dbo"   
   catalog="Northwind"**   
/>

## The Having Clause is now Supported

This has been added for all providers. Here is a sample query (this might not make sense as a real query, this was just for testing the syntax. The query did return data however).

EmployeeQuery q = new EmployeeQuery();  
q.Select(q.EmployeeID, q.Age.Sum().As("TotalAge"));  
q.Where(q.EmployeeID.IsNotNull());  
q.GroupBy(q.EmployeeID);  
**q.Having(q.Age.Sum() > 5);**  
q.OrderBy(q.EmployeeID.Descending);

EmployeeCollection coll = new EmployeeCollection();  
if (coll.Load(q))  
{

}

This resulting SQL is shown below:

SELECT [EmployeeID] AS 'EmployeeID',SUM([Age]) AS 'TotalAge' FROM   
[ForeignKeyTest].[dbo].[Employee]   
WHERE[EmployeeID] IS NOT NULL   
GROUP BY [EmployeeID]   
**HAVING SUM([Age]) > @Age2**   
ORDER BY [EmployeeID] DESC

## DynamicQuery Supports “Raw SQL” Everywhere

This acts as an escape hatch when you run into things we might not have added support for yet. You can inter-mix raw SQL code with our DynamicQuery. This was partially implemented before, but now it is possible everywhere. This query is not really how you write EntitySpaces queries. It is just trying to demonstrate injecting raw SQL. You can easily write this query in EntitySpaces without the raw SQL. (We do not want to encourage bad practices.)

EmployeesQuery q = new EmployeesQuery();  
q.Select( "<FirstName>", q.HireDate);  
q.Where ("<EmployeeID = 1>" );   
q.GroupBy("<FirstName>" , q.HireDate);  
q.OrderBy("<FirstName ASC>" );

EmployeesCollection coll = new EmployeesCollection();  
if (coll.Load(q))  
{  
}

The SQL Generated is as follows (and works)

SELECT FirstName,[HireDate] AS 'HireDate'   
FROM [Employees] WHERE (EmployeeID = 1)   
GROUP BY FirstName,[HireDate]   
ORDER BY FirstName ASC

## Full Expressions in OrderBy and GroupBy

This query does not really make sense, but we wanted to show you what is possible in this release.

EmployeesQuery q = new EmployeesQuery();   
q.Select(q.LastName.Substring(2, 4).ToLower());   
**q.OrderBy(q.LastName.Substring(2, 4).ToLower().Descending);   
q.GroupBy(q.LastName.Substring(2, 4).ToLower());**

EmployeesCollection coll = new EmployeesCollection();   
if (coll.Load(q))   
{   
 string s = q.es.LastQuery;   
}

The SQL Generated is as follows:

SELECT SUBSTRING(LOWER([LastName]),2,4) AS 'LastName'   
FROM [Employees]   
GROUP BY SUBSTRING(LOWER([LastName]),2,4)   
ORDER BY SUBSTRING(LOWER([LastName]),2,4) DESC

Not all database vendors support expressions everywhere. For example, aggregates are not allowed in an ORDER BY for SqlCe and VistaDB.

## Silverlight Support

A lot of the release is geared toward supporting Silverlight. In fact, there is a new assembly named “EntitySpaces.DynamicQuery.Silverlight.dll” which allows you to create and execute full EntitySpaces DynamicQueries within your Silverlight Application. The assembly above will be downloaded and run fine in the browser under Silverlight. You just serialize your query through XML back to the server to execute it. The XML packet is very tiny.

This is very big topic. For more information on this, see this [blog post](http://www.entityspaces.net/blog/2009/07/17/The+EntitySpaces+Silverlight+Demo+Online.aspx). In addition, our full Silverlight demo is installed with this Beta using all of the latest features, including our Compact XML.

## DynamicQuery Helper Methods for Serialization

There is a new helper when using the DynamicQuery features:

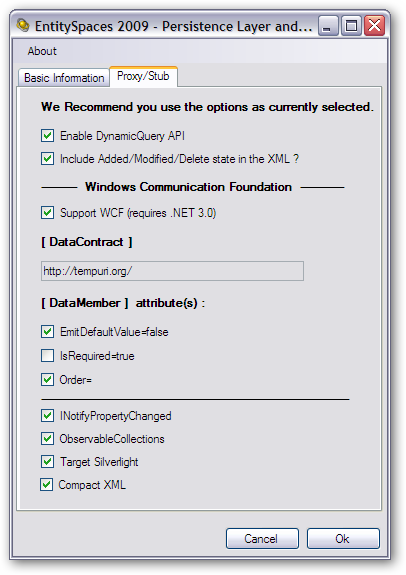
EmployeesQuery query = new EmployeesQuery();  
 query.es.Serializer.[helper methods]

## The EntitySpaces Generic Concurrency Mechanism

This really comes in handy for database independent applications. However, you can use this feature even if you are not creating a database independent application. You can now declare an integer or bigint column in all your tables and then set the **IsEntitySpacesConcurrency** flag to true on the Metadata tab for that column. This is supported in both DynamicSQL and StoredProcedure mode. This feature is not implemented in our Compact Framework providers.

## Proxy Stub Enhancements

Both our Server and Proxy stub templates have been enhanced. Below is a screen shot of our “Client Side Proxy Stub” template UI as we generated it for our Silverlight demo. Notice that we selected almost all of the options. The Compact XML feature is valid in normal WCF scenarios as well. You will notice that the” Compact XML” checkbox is on the Proxy Stub tab of the Generated Master template as well.



## VistaDB Password Bug Fix

You no longer have to worry about having passwords on your VistaDB databases during code generation, just add the password to your connection string and everything will work fine. We did not get the same fix in for SQLCE but it will be done by the production release. We are also bound to the “3.5.1.84” version of VistaDB.

## DynamicQuery Daisy Chaining Syntax has been fixed

At one point we broke the DynamicQuery syntax a few releases ago. Now you can always using the daisy chaining syntax.

EmployeeCollection collection = new EmployeeCollection();

EmployeeQuery emp = new EmployeeQuery("e");  
EmployeeTerritoryQuery empTerr = new EmployeeTerritoryQuery("et");  
TerritoryQuery terr = new TerritoryQuery("t");  
TerritoryExQuery terrEx = new TerritoryExQuery("tx");

emp  
 .Select(emp.FirstName, emp.LastName, terr.Description.As("Territory"), terrEx.Notes)  
 .LeftJoin(empTerr).On(emp.EmployeeID == empTerr.EmpID)  
 .LeftJoin(terr).On(empTerr.TerrID == terr.TerritoryID)  
 .LeftJoin(terrEx).On(terr.TerritoryID == terrEx.TerritoryID)  
 .Where(emp.FirstName.Trim().Like("J%"));

collection.Load(emp);

## DynamicQuery Order of Values More Flexible

You have much more flexibility in the order values in your queries, for instance.

OrderDetails details = new OrderDetails();  
details.Query.Select(10 \* details.Query.Quantity);

Notice that the number 10 comes before the “QueryItem”. This was not possible before. This can come into play when you need your column to come later in a formula.

OrderDetails details = new OrderDetails();  
details.Query.Select( (10 \* details.Query.Quantity) / 2);

## There is now a Twitter toolbar button on the “Whats New” page

Not really much to say here, but you can get more detailed “behind the scenes” information and listen into the EntitySpaces development team chatter. You can find out about things that might not be posted on our blog as well.

## AND NOT / OR NOT Added To DynamicQuery Syntax

Basically, we had the SQL conjuctions AND and OR but we didn’t have any way to indicate **AND NOT** or **OR NOT** but that is no longer true, see the example below.

**VB.NET**

Dim q As New EmployeesQuery()  
q.es.Top = 1  
q.Where(q.EmployeeID = 1 And Not (q.LastName = "goofy"))  
q.Load()

Notice you can now use And Not/Or Not in your query syntax. This will yield the following SQL syntax.

SELECT TOP 1 \*   
FROM [Employees]   
WHERE ([EmployeeID] = @EmployeeID1 AND NOT [LastName] = @LastName2)

**C#**

You can do this in C# too, however this is one case where the VB.NET syntax reads better than the C# syntax. Take a look at the sample below.

EmployeesQuery q = new EmployeesQuery();  
q.es.Top = 1;  
q.Where(q.EmployeeID == 1 && ! (q.LastName == "googy"));  
q.Load();

Granted, a C# developer would probably never think of that syntax, but it yields the exact same SQL as above. Probably a C# developer would write it as below

EmployeesQuery q = new EmployeesQuery();  
q.es.Top = 1;  
q.Where(q.EmployeeID == 1 && (q.LastName != "googy"));  
q.Load();

The above query would product the following SQL.

SELECT TOP 1 \*   
FROM [Employees]   
WHERE ([EmployeeID] = @EmployeeID1 AND [LastName] <> @LastName2)