

DevForce 2010

Release Notes

Table of Contents

Introduction.....	3
Why Read the Release Notes	3
What's in a Note.....	3
Read the Installation Guide	4
Customer Support	5
DevForce 2010 Version 6	6
Version 6.0.8	6
Summary.....	6
Breaking Changes	6
Defect Repairs	6
New and Improved Product Features	7
Version 6.0.7	9
Summary.....	9
Breaking Changes	10
Defect Repairs	10
New and Improved Product Features	12
Known Limitations.....	13
Version 6.0.6	13
Summary.....	13
Breaking Changes	14
Defect Repairs	14
New and Improved Product Features	16
Obsolescence	17
Version 6.0.5	17
Summary.....	17
Breaking Changes	17
Defect Repairs	18
New and Improved Product Features	20
Known Issues.....	22
Version 6.0.4	22
Summary.....	22
Breaking changes	22
Defect Repairs	22
New and Improved Product Features	24
Learning Resource Updates	26
Known Issues.....	26
Version 6.0.3	26
Summary.....	26
Breaking changes	27
Defect Repairs	27
New and Improved Product Features	29
Learning Resource Updates	33
How DevForce Initializes New Entity Properties	33

Known Issues.....	35
Version 6.0.1	36
Summary.....	36
Breaking changes	37
Entity Framework	38
Entity Data Model Wizard and Designer.....	39
Entities and the Entity Manager	45
Validation	51
Business Object Server	53
Configuration and Deployment	54
Enjoy your new DevForce 2010!.....	55

Introduction

These notes describe the history of changes to the DevForce product and provide important information before you upgrade. Each DevForce “Feature” release earns its own chapter with more minor “Maintenance” releases listed as topics within the chapter.

We indicate a “Feature” release by changing in the second digit of the DevForce version number (e.g., 3.1 to 3.2); a “Maintenance” release is a change in the third digit (e.g., 3.1.4 to 3.1.5). The fourth digit varies by build and does not typically merit a release note.

Why Read the Release Notes

While the notes may be just a curiosity to the first-time DevForce installer, they are **essential reading** if you are upgrading your DevForce application or thinking about doing so.

The steps you must take every time you upgrade are covered in the **DevForce Installation Guide**. There may be additional, **release-specific upgrade steps**. These you can only learn about here.

Each release topic in these notes calls out the additional steps required to transition from the previous release. There may be none. There may be just a few that can be handled by re-running a DevForce tool and recompiling the application. On rare occasions, you may have to make manual changes to your code.

You will only know if you read these notes. If you are upgrading across several versions, please read the notes for all intermediate versions.

What’s in a Note

Upgrade Advice

We describe what you have to do to migrate to this version.

New Features

We add new features almost every release. New features are often described here in great detail when first introduced so that you can discover and learn about them in one convenient place. We migrate that detail into the **Developers Guide** over time as the “new” becomes the “mainstream”. That’s where most developers will go to learn about programming with DevForce.

Fixes

These notes cover our corrections to defects in the product. You’ll want to check here to confirm we’ve addressed a bug that you submitted.

Backward Compatibility and Signature Obsolescence

We strive to avoid changes that could break your code. Backward compatibility within the DevForce product is a cherished value.

However, we learn from experience and sometimes we find some functionality that needs to be redesigned. The new design may result in method names or signatures that compete with the prior design. We preserve the previous signatures if we can, marking them obsolete so that you can gradually remove them from your code.

IdeaBlade DevForce Release Notes

No one wants to be stuck with poor, unused signatures that clutter your code and confuse the developer. Accordingly, when we mark a method signature obsolete, we often announce that we will remove that signature at a specific future date. Typically, that is six months from the obsolescence date.

We will accommodate any Enterprise Edition customer's need for more time to adapt. Please contact your sales representative when this is an issue and we will make accommodations for you.

Breaking Changes

On rare occasions, we choose to remove the old signature entirely. Any application that depends upon that signature requires some re-coding and recompilation.

Perhaps we determined that there is no way to safely implement the method's intended behavior while that signature exists. Rather than permit your code to proceed in error we'll "force" you to fix it – if you choose to upgrade.

Sometimes we change a method in a dark, infrequently visited corner of the product. We may decide that the risk of deleting or altering a signature that almost no one sees is less than the risk that someone might come along later and depend upon it. We hope we are right and we'll revert quickly if we learn that we were wrong.

In all cases, we will tell you what we did and how to adjust your code to recover from the break. We apologize in advance for this inconvenience; we trust you'll perceive the gain as worth the pain.

Read the Installation Guide

The Installation Guide contains important information about how to install and upgrade your DevForce application to a new version.

Customer Support

For support, you can visit our support forums at www.ideablade.com/forum. Your question may have already been answered there by a DevForce engineer or by someone in our development community.

If you have an active Support Subscription, you can also submit a direct support case via our web site at

<http://www.ideablade.com/CustomerSupportRequestForm.aspx>

When submitting a support case, please try to simplify and isolate your problem as much as possible. If you are submitting a bug report, creating a simple test case using the NorthwindIB database will help us to quickly identify, correct, and test the issue.

IdeaBlade also offers Enterprise Support options as well as a full range of Professional Services to help you build your application. Please contact your sales representative or email info@ideablade.com for more information.

DevForce 2010 Version 6

Version 6.0.8

Summary

This release of DevForce 2010 includes a large number of enhancements and bug fixes.

You must regenerate your model if you are upgrading to this release. Please refer to [F1523] in version 6.0.7 Features section for more details.

DevForce 2010 version 6.0.8 now provides the ability to construct completely dynamic queries. See F1606 for more details.

Breaking Changes

- *CompositionHost.Instance.AddCatalog* was removed. This method had been available as an adjunct to MEF initialization. If you were using this method, you can instead use *CompositionHost.Instance.Catalog.Catalogs.Add*. If you are using the new Silverlight on-demand content download available with this release, you will use one of the *CompositionHost.Add* overloads. See F1589 for more information.
- The *PropertySortSelector* no longer implements *IEnumerable<PropertySortSelector>*. In order to get the list of 'chained' selectors within a *PropertySortSelector* use the new

IEnumerable<PropertySortSelector> GetChainedSelectors()

method instead.

Defect Repairs

- Silverlight: Loosened the rules for placement of models and links within desktop and Silverlight assemblies. You can now have: 1) multiple EDMX in a single desktop project with links in different Silverlight assemblies, and 2) a single Silverlight project holding links to entity models in multiple desktop assemblies. [B1606]
- The 'failureType' parameter of the *EntityServerInterceptor.OnError* method was not always getting set properly. [B1652]
- Fixed a composition issue where the *NullLoginManager* could be used as the *IEntityLoginManager* even when a better match was available. [B1659]
- The *EntityServerSaveInterceptor.OnError* method was not receiving concurrency exceptions. [B1660]
- *EntityAspect.RejectChanges* was not clearing the *EntityAspect.ValidationErrors* collection. [B1662]
- *INotifyDataErrorInfo.ErrorsChanged* event was not getting fired after an *EntityAspect.Clear* method call. [B1663]

IdeaBlade DevForce Release Notes

- Silverlight *ObjectDataSource* – Fix a *NullReferenceException* which occurred when paging was off and filters were not in use. Improved error messages. Allow filter changes when the query is not an *EntityQuery* and/or paging is off. [B1664]
- Many to many relations that were defined between detached entities were getting lost when the entities later got attached to an *EntityManager*. [B1666]
- Silverlight *EntityQueryPagedCollectionView*: Do not set *PageIndex* to -1 at initialization. Improved error handling. [B1668]
- Exclude interfaces from known type lookup. [B1669]
- Allow *AspNetAuthenticatingLoginManager*, and sub-types, to be used in 2-tier applications. [B1670]
- Fixed an issue where a Silverlight shortcut to the generated code of a model could be duplicated in multiple projects. [B1672]
- Allow store-generated ID to be in any column of a multi-column Primary Key. [B1673]
- Saving multiple new entities containing *DateTimeOffset* fields was failing. [B1674]
- Allow *DataSourceKeyName* to contain underscores. Underscores are used within DevForce to append a data source extension, and inclusion in the base key name caused an unhandled exception. [B1678]
- Support *Contains* query with *List<byte>* or *List<decimal>*, in addition to other numeric types already supported. [B1680]
- Intellisense and Help Reference information are now available for the *EdmKey*. [B1681]
- The *EntityManager.RefetchEntities* call would fail with a stack overflow when called with very large lists of entities. *RefetchEntities* calls are now batched so that even very large lists of entities can be handled. This does not change the API in any way but the default batch size can be controlled via a setting on the *EntityManager*. See F1607. [B1684]
- Silverlight Business Application templates – The revised templates contain a fix for a multi-threading problem causing intermittent errors related to verifier resources. [B1686]
- Silverlight: Fixed an issue with *INotifyDataErrorInfo* showing duplicate messages when adding a free form *VerifierResult* constructed with property names to *EntityAspect.ValidationErrors* collection. [B1687]
- Don't issue warning message for a remote service method with no return value. [B1688]
- Allow for the same short type name, for example “Customer”, in different entity models within an assembly. Improved error messages for metadata errors (including the dreaded “... must inherit from Entity and does not”). [B1689]
- Fixed a composition issue with how class and interface types get discovered. [B1696]

New and Improved Product Features

- Added the *EntityServerErrorInterceptor* class to provide server-side interception and filtering of exceptions before sending to the client. Sub-type this class to override default processing. With a custom interceptor

you can include centralized logging of exceptions, and disguise the exceptions sent to the client for security purposes. See the [DevForce Resource Center](#) for more information. [F1565]

- A *Tag* property was added to the *SaveOptions* class. Setting this property on the client to any ‘WCF serializable’ value allows this value to be inspected during the execution of any *EntityServerSaveInterceptor*, via the *SaveOptions* property, and can be used to mediate the save operation. Note that the *Tag* property on the *QueryStrategy* performs the same service within an *EntityServerQueryInterceptor*. An associated *SaveOptions.WithTag* method has also been added and may be used as follows [F1581]:

```
var saveResult = myEntityManager.SaveChanges(myEntityManager.DefaultSaveOptions.WithTag("my custom information"));
```

- Silverlight: Support multi-XAP Silverlight applications with models in separate XAPs. DevForce now supports on-demand loading of entity models. See the [DevForce Resource Center](#) for a complete description and sample code. [F1589]
- The *PredicateDescription* class now supports nested properties. This means that you can now create *predicateDescriptions* such as [F1592]:

```
var pd = new PredicateDescription(typeof(Product), "Supplier.CompanyName",  
FilterOperator.StartsWith, "B");
```

```
var query = myEntityManager.Products.Where(pd);
```

- Added support to allow entity metadata classes to participate in code generation. Support for the .NET *MetadataTypeAttribute* has been supported within DevForce since DF2010 was first released. However, as this attribute was originally designed by Microsoft, it was only intended to be used at design time to support code generation. This means that virtually none of the .NET base class libraries support any extended metadata at runtime. DevForce did offer this runtime support but it could only be provided to DevForce’s own classes.

As of this release, we now automatically apply any Metadata markup directly to our generated code. This process occurs whenever an EDMX file is modified or when the custom tool associated with the generated code is executed manually. In other words, you will need to regenerate your model if you have used the *[MetadataType]* attribute in your existing code. This is required in order for your existing metadata markup to be picked up and regenerated into your domain model. Metadata markup for any given attribute on a per property basis will suppress whatever attribute DevForce’s code generation would have provided. In the case of metadata markup that adds either .NET Validation or DevForce Verification attributes, this markup will replace all of Devforce’s default generation of Validation or Verification attributes for that property. The idea here being that validation should either come from the metadata markup or from EDMX metadata but not both for the same property.

Within the EDMX designer, you can also suppress the generation of any attribute via the ‘*AttributesToSuppress*’ property. As of this release an additional ‘Metadata’ item now appears in this list, and can be used to tell the DevForce code generator to ignore any metadata attributes for the selected property. [F1599]

- Silverlight: Enable application library caching of DevForce Silverlight assemblies. When the “Reduce XAP size by using application library caching” checkbox is selected on the Visual Studio property page of a Silverlight application, the referenced DevForce assemblies will be packaged separately as “zip” files. Library caching can improve startup performance of an application, and is also useful in applications supporting on-demand download of assemblies. See the [DevForce Resource Center](#) for more information. [F1601]

IdeaBlade DevForce Release Notes

- A *RefetchEntitiesAsync* overload that takes an *EntityState* has been added. This was to provide parity with the synchronous API. [F1603]
- The *PropertySortSelector* now supports nested properties. This means that you can now create *propertySelectors* such as [F1605]:

```
var ps = new PropertySortSelector(typeof(Product), "Category.Name", ListSortDirection.Descending);  
var query = _em1.Products.OrderBySelector(ps);
```

- New extension methods that take a loosely typed *IQueryable* or *IEntityQuery* (as opposed to *IQueryable<T>* or *IEntityQuery<T>*) have been added to support additional dynamic query capabilities. [F1606]

```
IQueryable IQueryable.Where(IPredicateDescription);  
IEntityQuery IEntityQuery.Where(IPredicateDescription)
```

```
IQueryable IQueryable.OrderBySelector(IPropertySortSelector);  
IEntityQuery IEntityQuery.OrderBySelector(IPropertySortSelector)
```

This now allows for the construction of completely dynamic queries such as the following:

```
var entityType = typeof(Product);  
var baseQuery = EntityQuery.Create(entityType);  
  
var pd1 = new PredicateDescription(entityType, "UnitPrice", FilterOperator.IsGreaterThanOrEqualTo, 24);  
var pd2 = new PredicateDescription(entityType, "Discontinued", FilterOperator.IsEqualTo, true);  
var pd3 = new PredicateDescription(entityType, "Category.Name", FilterOperator.StartsWith, "P");  
var pd = pd1.And(pd2).And(pd3);  
  
var ps1 = new PropertySortSelector(entityType, "Category.Name");  
var ps2 = new PropertySortSelector(entityType, "ProductName");  
var ps = ps1.ThenBy(ps2);  
  
var query = baseQuery.Where(pd).OrderBySelector(ps);  
var results = myEntityManager.ExecuteQuery(query);
```

- Added a new *EntityManager.Options* property that returns an instance of a new *EntityManagerOptions* class. At present this class contains a single property *RefetchEntitiesBatchSize*, that can be set to control the number of entities being queried with each batch that is submitted during a *RefetchEntities* call. Batching is necessary to avoid excessively large queries from being sent to the server in a single call. By default, this batch size is set to 200 and will not need to be changed in most cases. [F1607]

Version 6.0.7

Summary

This release of DevForce 2010 includes a large number of enhancements and bug fixes.

DevForce2010 6.0.7 now checks for version compatibility between the generated business object code and the current runtime. This check will throw an exception at runtime during the construction of any *EntityManager* if your locally generated code is out of date. **You must regenerate your model** to avoid this message. Please refer to [F1523] in the *New and Improved Product Features* section for more details.

Breaking Changes

- The *TransactionSettings.Default* now defaults to using a *TransactionScope* with all queries and saves. If this causes any problems you may modify the default by setting *TransactionSettings.Default = new TransactionSettings(System.Transactions.IsolationLevel.ReadCommitted, new TimeSpan(0, 1, 0), false);*
- If you are sub-typing the *AspAuthenticationLoginManager*: the *LogoutCore* virtual method now takes an *IPrincipal*. See B1638.
- If you are sub-typing the *DomainModelTemplate* for custom code generation: the *MaxClassesPerFile* property was removed; use the *TemplateSettings.MaxClassesPerFile* property instead. The *OMFileType* enumeration was changed to a *Flags* enumeration.
- If you enable OData in your entity model in a VB project – To avoid a collision with the default parameterless constructor needed by the OData provider, the first parameter to the following constructor is no longer optional:
Public Sub New (ByVal shouldConnect As Boolean, Optional ByVal dataSourceExtension As String=Nothing, Optional ByVal entityServiceOption As EntityServiceOption=EntityServiceOption.UseDefaultService, Optional ByVal compositionContextName As String=Nothing)

Defect Repairs

- The Null Entity could be added to a *RelatedEntityList* - and shouldn't have been able to. [B1605]
- Improved Intellisense and exception text for the *EntityManager InvokeServerMethod* and *InvokeServerMethodAsync* calls if it fails due to a badly formed type name parameter. The async method now “fails fast” if the type name is invalid, and returns the error to the async handler. [B1607]
- The *DiscoverableKnownType* attribute can now be applied to an enum or struct, in addition to a class. [B1611]
- An async navigation involving a scalar 1-1 relation would throw if a pending entity was not found. [B1612]
- After updating the *ProductKey*, running the setup and choosing *Modify/Repair* will throw an error/not install Silverlight Assemblies. [B1613]
- Fixed a problem where reference fixup was not performed when an *ImportEntities* was called, resulting in unnecessary fetches to the database. [B1615]
- An *Entity.RejectChanges* call on an entity with many-to-many changes did not revert any many-to-many changes. [B1618]
- *ObjectDataSource* (ODS) filtering can produce a duplicate user state error. [B1620]
- Serialization problems would occur when querying with an anonymous projection that included custom data transfer types with static properties. Static properties are now ignored on these types for serialization purposes. [B1622]
- Additional fixes to address the DTC escalation problem. Changed the default to use a *TransactionScope* with all queries and saves. [B1623]

IdeaBlade DevForce Release Notes

- Validation of an instance of a complex type was not informing parent entity of any errors encountered. [B1625]
- RangeVerifier constructor parameters were incorrectly documented. [B1626]
- The removal of an entity followed by requery for that same entity could cause previously detached children to show up as related entities. [B1627]
- Changing a navigation property, via its foreign key, within an EntityServerSaveInterceptor could cause the corresponding navigation property to be incorrectly updated. [B1628]
- Entities deleted in an *EntityServerSaveInterceptor* were not being merged back into client cache correctly. [B1629]
- Completion was never signalled on a parallel Coroutine when all operations completed synchronously. [B1630]
- A query with a PreserveChanges QueryStrategy could overwrite navigation properties under certain conditions. [B1631]
- Improved error messages for connection and login failures due to problems with custom types (Silverlight). [B1634]
- Building up a detached entity graph with more than a two level hierarchy involving multipart primary keys could cause an 'Invalid operation exception' when entities were finally added to an EntityManager. [B1635]
- Improved each of sub-typing the *AspNetAuthenticatingLoginManager*. Added new virtual methods *CreatePrincipalCore* and *CreateIdentityCore*; exposed additional properties which were previously private; improved Intellisense. [B1638]
- A query with an 'Include' of a parent relation was bypassing the Include when the child was in a modified state. [B1639]
- *SaveChangesAsync* would throw an exception immediately if an *EntityManager.Saving* event handler threw an exception. The exception now propagates to the async callback like all other exceptions. [B1641]
- Validation metadata properties on a nested Metadata class were required to be static; they no longer are. [B1642]
- A removal followed by a re-add of the same related object in many to many relation would cause a save failure. [B1644]
- Changing a navigation property, via its Foreign Key, was not updating the corresponding *EntityReference.IsLoaded* flag when the navigated to entity was not present in the cache. There was a related bug involving requerying for the entity when in this state. [B1645]/[B1646]
- Silverlight Business Application Templates - *AuthenticationManager.Current.User* was fixed to never return null. If the user is not logged in the anonymous user is returned. The data type of this property was also changed from *IPrincipal* to *UserBase*. [B1647]
- The *StringVerifierAttribute.Required* attribute was causing property setters to fail with an exception when the corresponding property was set to null. The correct behavior is to allow the 'exception' caused by the set to be handled by the DevForce validation engine instead. [B1648]

- When using a ‘Fake’ EntityServerBackingStore, the *AsScalarAsync().Count()* query against a many-many relation would always return 0. [B1649]

New and Improved Product Features

- Beta – An EntityManager may now be used as a WCF Data Services (OData) data source for both read and write access. You must set the “OData Enabled” flag in the EDM designer to enable this feature. When set, additional attribute markup is added to types in your generated model, references to Data Services assemblies are added to the model project, and a file named <EntityManager>_IUpdatable.cs (.vb) will be added to the project. This file contains the OData IUpdatable implementation allowing for CUD access, and will be generated once only. Refer to the [DevForce Resource Center](#) for more information. [F1359]
- Simplified installation process from six screens to three. [F1485]/ [F1558]
- New *PropertySortSelector* class and new *Queryable* and *EntityQuery* extension methods that allow construction of queries with dynamic property based sorting. Where our existing *PredicateBuilder* and *PredicateDescription* classes allow dynamic construction of a LINQ Where clause, the *PropertySortSelector* allows dynamic construction of OrderBy, OrderByDescending, ThenBy and ThenByDescending clauses. For more information, please see the [DevForce Resource Center](#). [F1491]
var ps1 = new PropertySortSelector(typeof(Customer), "Country", ListSortDirection.Descending);
var q1 = myEntityManager.Customers.OrderBySelector(ps1);
- Set Silverlight version in a project generated by a DevForce Silverlight template to the version installed on the developer’s machine. You’ll see this in the “minimumRuntimeVersion” set in the Silverlight plugin definition in the default.aspx. [F1493]
- Added ability to get list of all EntityTypes involved in an EntityQuery via the *GetReferencedEntityTypes()* extension method on IEntityQuery. This will return any of the types referenced by the query itself, as well as those involved in any joins or subselects. This does not include “Included” types at this time. [F1495]
- Added a check for version compatibility between generated business object code and the current runtime. This check will emit the following message at runtime during the construction of any EntityManager if your locally generated code is out of date: [F1523]
An entity model in the '{0}' assembly was generated by a different version of DevForce that is not compatible with the current version.
Please regenerate the entity model base classes by resaving the EDMX in Visual Studio.
Model template version: {1}
Current template version: {2}
- An *IEntityQuery.OfType<T>* extension was added to allow simpler use of the OfType extension without having to cast the result back into an EntityQuery<T>. [F1526]
- A free form *VerifierResult* can now be created and associated with a specific property independently of any verifier. These results can then be added to any entity’s ValidationErrors collection. [F1539]
var vr = new VerifierResult(VerifierResultCode.Error, "Bad birthdate", "BirthDate");
- Support to a new InList FilterOperator in the PredicateBuilder (Contains in LINQ). [F1540]
PredicateBuilder.Make(typeof(Customer), "Id", FilterOperator.InList, listOfValidIds);
- Additional extension methods and overloads to make use of the PredicateDescription class simpler. [F1541]
EntityQueryFilterCollection
public void AddFilter(IPredicateDescription pd, bool canReplaceIfAlreadyExists = false);
IEntityQuery extension method

IdeaBlade DevForce Release Notes

```
public static IEntityQuery<TSource> Where<TSource>(this IEntityQuery<TSource> source1, IPredicateDescription predicateDescription);  
Queryable extension method  
public static IQueryable<TSource> Where<TSource>(this IQueryable<TSource> source1, IPredicateDescription predicateDescription);
```

- Added support for the *IEditableObject* interface on Detached entities. [F1542]
- Improved sort order for DevForce Visual Studio templates displayed in the New Project dialog. [F1543]
- VB versions of Silverlight Business Application templates are now available. [F1547]
- Added *AsScalarAsync.Any()* and *AsScalarAsync.All()* methods. [F1548]
- Changes to the Silverlight Business Application templates: fixed issues where resource errors caused pages not to load in a designer; added *MultipleSiteBindings* enabled flag to web.config; added Forms cookie name in web.config; modified *VerifiableObject* class to support instance-level verification; XML comment cleanup. [F1551]
- Support filtering in the *EntityQueryPagedCollectionView* (Silverlight). Use one of the *SetQueryFilter* overloads to pass filter criteria to the view. [F1556]
- We now offer a full implementation of *INotifyDataErrorInfo* and *IDataErrorInfo* for Complex Types. This means that any validation errors that occur on a complex type will perform notification thru both interfaces on both the complex type as well as its parent entity. [F1557]
- All server-side validation errors that occur during a Save now flow back through to the client. Validation errors will now show up on the individual entities and will cause UI updates if data-bound. [F1561]
- *VerifierResults* are now serializable. [F1561]
- An optimization was added that internally caches compiled local queries and reuses the compiled plan for subsequent executions. This means that if you hang onto a query object, any executions of that query **against the local cache** after the first may be much faster. The amount of improvement will be related to how long compilation of the query takes versus the time taken to execute the compiled *Func<>*. Expressions that involve small datasets, hence small execution times, tend to benefit the most. [F1570]

Known Limitations

Anonymous projection queries to an OData service backed by the *EntityManager* are not currently supported.

Version 6.0.6

Summary

This release of DevForce 2010 includes a large number of enhancements and bug fixes.

It includes a signature obsolescence that replaces *AsyncSerialTask* and *AsyncParallelTask* with *Coroutines* and new DevForce Silverlight Business Application Templates. Please see [Obsolescence](#) and [Features](#) section below for more info.

Breaking Changes

- Related to D1598: The TransactionSettings.UseDTC parameter has been replaced with a UseTransactionScope parameter that defaults to 'false', as of release 6.0.6.1. In release 6.0.6.0 the default was 'true', but we found this to be a problem with some databases.
- The BaseOperation.IsCompletedSynchronously property is now named CompletedSynchronously.

Defect Repairs

- Exception thrown with UseAsyncNavigation and one-to-one navigations. Fixed. [B1551]
- InvalidCastException thrown when RelatedEntityList used as source for ListCollectionView. Fixed. [B1553]
- AsyncSerialTask and AsyncParallelTask do not honor EntityManager.EntityServerError marking of an error as 'Handled'. Fixed. [B1557]
- CompositionContext.Fake queries with scalar results fail. Fixed. [B1558]
- DataSourceExtensions and CompositionContexts with invalid names were not throwing exceptions. Fixed. [B1559]
- Inconsistent behavior with entity navigation on detached entities. Fixed. [B1561]
- With CompositionContext.Fake, FirstOrNullEntity throws null reference exception when entity not found. Fixed. [B1563]
- Exception thrown for bad synchronous query inside ExecuteAsync rather than passed thru to callback. Fixed. [B1568]
- In a VB Project, placing or creating .edmx and .tt files in a project subfolder generates faulty code files. Fixed. [B1569]
- Guid.CompareTo(Guid) method not serializing properly in expression tree. Fixed. [B1570]
- Poco missing method should throw a more explanatory exception. Fixed. [B1571]
- Difficulty with updating navigational properties inside a server method. Fixed. [B1574]
- CancelEdit not restoring "Added" entities to their state before BeginEdit. Fixed. [B1575]
- Unable to serialize IGrouping<K, T> where T is an anon type. Fixed. [B1576]
- RefetchEntitiesAsync without an Action and only a Completed event fails. Fixed. [B1581]
- ChangedEntities in EntityQueriedEventArgs was returning more entities than necessary. Fixed. [B1583]
- EntityCacheState methods that return an in-memory entity cache state should not be live (should be snapshots). Fixed. [B1586]
- PropertyInterceptorManager.RemoveAction not handling null parameter properly. Fixed. [B1587]

IdeaBlade DevForce Release Notes

- Issue with principal entity being added to an EntityManager after its dependents and not being fixed up. Fixed. [B1588]
- Deleted and removed entities (detached) that are later reattached to the same EntityManager were not always being correctly relinked. Fixed. [B1589]
- Codegen for an Edmx nested more than 1 subfolder deep does not work. Fixed. [B1591]
- Entities detached from an EntityManager using Clear do not report the “Detached” entity state even though they are. Fixed. [B1596]
- Async navigation from child to parent incorrectly marks the child as modified. Fixed. [B1597]
- Saves involving deletes are not transactional when UseDTC=false. Fixed. The UseDTC flag has been removed, all saves are now transactional. [B1598]
- Entities imported from one EntityManager to another not handling IEditableObject.RejectChanges properly. Fixed. [B1599]
- Model First code generation missing namespace in generated code. Fixed. [B1601]
- Scalar async queries that return a 'null' are incorrectly treated as having gone async even when they complete synchronously. Fixed. [B1603]
- Fixed several design-time issues in Cider and Blend with constructing an EntityManager and restoring an EntityCacheState. [B1554]
- EntityServiceApplication.OnServiceShutdown event now fires consistently when the EntityService is stopping. [B1555]
- Multiple instances of the EntityQueryPagedCollectionView can now be used without causing a duplicate UserState exception. [B1556]
- Fixed a problem with finding the IdeaBlade.EntityModel.Push.SL assembly. [B1560]
- The StoreGeneratedIdGenerator used in the EntityServerSaveInterceptor now correctly assigns temporary ids. [B1567]
- Fixed a MEF probing issue related to supporting anonymous logins. [B1572]
- Fixed an error in parsing the URL for the default Silverlight configuration. [B1573]
- Fixed a problem in configuring EntityServer endpoints for all base addresses. [B1577]
- Fixed problem with the startup of a non-IIS BOS using net.tcp protocol. [B1578]
- The EntityServerSaveInterceptor Principal now remains the same for the request life cycle. [B1579]
- Now add the Silverlight fault behavior for endpoints defined in the config file. [B1580]
- Better error messages for connection-related failures. Unhandled failures for an “implicit” connection will be thrown, consistent with the async API. [B1590]
- EntityManager.LinkForAuthentication now sets the thread Principal. [B1592]

- Fixed a `NullReferenceException` occurring during save processing with a Model-First model. [B1593]
- Improved warning and error messages for “bad” return types from an `InvokeServerMethod`. [B1600]
- Quickie Silverlight movie can't be played on a 64-bit machine using Getting Started shortcut.

On a 64-bit machine using IE 8, the `GettingStarted.exe` uses the 64-bit version of IE. This doesn't work with Silverlight Quickie (or WPF Quickie) movie, and you just get a prompt telling you to install Silverlight (even though Silverlight is already installed). [B1604]

New and Improved Product Features

- Made `IdGeneration` translation map available on the client after a successful save. See `TemporaryIdMap` in `SaveResult`. [F1336]
- We now support a `Distributed (Remote) EntityServerFakeBackingStore` as well as a local one. [F1494]
- Made `GroupBy` return strongly typed `EntityQuery` so that group bys with anonymous projections can be handled without casting the query. [F1497]
- We now explicitly throw a more meaningful exception when trying to recursively call `Save` within a `Save` `Interceptor`. Previously exception was not terribly clear. [F1502]
- Cleanup of signatures for `FakeBackingStore` operations. [F1503/F1505/F1506]
- Added ability to register a `CompositionContext` without using custom `CompositionContextProvider`. See the [DevForce Wiki](#) and the *DevForce API Documentation* (installed with the product; available via the Windows Start menu) for more information. [F1504]
- Support a separate `Fake` backing store per fake composition context. Previously only one shared ‘fake’ backing store for all fake contexts. [F1511]
- New overloads for `EntityManager.AddEntities` and `AttachEntities` that take an `IEnumerable` instead of `IEnumerable<T>`. [F1516]
- New overloads for `EntityManager.RefetchEntities` and `RefetchEntitiesAsync` that take an `IEnumerable` instead of `IEnumerable<T>`. [F1521]
- Add `Coroutine` support to simplify implementation of async serial and parallel tasks. See the [DevForce Wiki](#) and the *DevForce API Documentation* (installed with the product; available via the Windows Start menu) for more information. [F1522]
- Added a static settable `"TransactionSettings.Default"` property to be used as the default for all transaction settings if not otherwise specified. [F1537]
- In a 2-tier application whose config file does not contain an `ideablade.configuration` section, the `connectionStrings` will be found if present and the application will run with default configuration options. [F1414]
- Visual Studio templates for the “DevForce Silverlight Business Application” are now available. Currently these templates are provided in C# only; VB templates will be provided in the next release. These templates closely mirror the “Silverlight Business Application” templates from Microsoft RIA Services, and provide the same navigation structure and themes. The templates also provide user authentication and

registration using ASP.NET Membership, and provide a sample implementation of the new *IAuthenticationManager* interface. The *EntityManager* will now search for an implementation of this interface when performing an implicit login, so that credentials entered once can be automatically used by other *EntityManager* instances [F1467/F1468]

- The *EntityCacheState* can now be serialized in either binary (the default) or text format. All *SerializationFns* methods now accept a flag for either binary or text format. [F1496]
- Use default browser to view Getting Started shortcut.

In previous releases, the Getting Started shortcut would only display the splash page in Internet Explorer. Now it will display the splash page in your default browser (which could be Firefox or Chrome). [F1525]

Obsolescence

As of this release, we are deprecating the *AsyncSerialTask* and the *AsyncParallelTask* classes. The new *Coroutine* class supports all of the same features but is much easier to use and understand. We will be removing the *AsyncSerialTask* and the *AsyncParallelTask* from the product after April 1st 2011. We will provide these libraries as open source after that time.

Version 6.0.5

Summary

This release of DevForce 2010 includes a large number of enhancements and bug fixes.

Breaking Changes

- **Exceptions Thrown by Asynchronous Operations Must Be Handled.** All asynchronous persistence operation exceptions now conform with Microsoft RIA Service style of exception handling. This requires that any exceptions thrown within an asynchronous operation be handled, by calling *MarkErrorAsHandled()* within the callback method or one of the Completed event handlers. If an exception is thrown and *MarkErrorAsHandled* is not called, an exception will be thrown upon the completion of the last *Completed* event handler.

The impact of this change can be mitigated somewhat by adding an *EntityManager.EntityServerError* event handler that internally sets the *EntityServerEventArgs.Handled* property to *true*. This event handler will be called before any async callback methods or Completed event handlers, and will internally mark any exceptions as handled. [F1408]

- **EntityManager.AuthorizedThreadId Property.** There is a new *AuthorizedThreadId* property on the *EntityManager* that checks to see that the *EntityManager* is used in a thread-safe manner. This property is set when the *EntityManager* is first created, and is used internally to check that no operations on the *EntityManager* are initiated from another thread. For certain scenarios (described within the *EntityManager* API), this property may be set to null or to another thread id to either suppress the checking or to move the *EntityManager* conceptually onto a different thread. This may be a breaking change; but if your code *does* break, there is a good possibility that you are using the *EntityManager* in a non thread-safe manner. [F1471]
- **The ComplexObject now has a ComplexAspect that is analogous to the Entity-EntityAspect relationship.** All DevForce-specific methods have been moved off the *ComplexObject* and onto the *ComplexAspect*. [F1475]

- **Constructors for the *EntityManager* have been changed to use default parameters and to support the new *EntityManagerContextClass* and the *CompositionContextName* parameter.** Code-generated, subclassed *EntityManagers* have been modified as well. We tried to keep backward-compatibility with all previous constructor calls via the use of default parameters and by keeping the same order to the parameters. However, this may be a breaking change for dynamically invoked constructor calls, which are not aware of default parameters. This feature requires that all generated code be regenerated. [F1480]
- **An “Implicit” connect is no longer performed on a disconnected *EntityManager*.** If your Silverlight code relied on this behavior, you will have to change it to connect explicitly at the desired time.

This change was made to make working with a disconnected *EntityManager* more intuitive, and to align behavior between synchronous and asynchronous connection behaviors. A “connection” is the handshake which occurs between an *EntityManager* and an *EntityServer* before other requests are sent; usually the connection will occur during *EntityManager* construction, based on arguments passed, or later via an *EntityManager.Connect* or *EntityManager.ConnectAsync* call. [F1219/F1470]

- ***EntityQueryOp<T>.IsCompleted* Behavior Changed.** Previously in DevForce, *EntityQueryOp<T>.IsCompleted* was never set to *true* when the server encountered an exception. We have changed this to adhere to Microsoft’s guidelines for asynchronous behavior. This is a breaking change because many developers, including us, viewed the *IsCompleted* flag as indicating that an asynchronous operation completed *successfully*. With this fix, it no longer means that. It simply indicates that the asynchronous operation has completed -- with or without errors. To determine if the asynchronous operation completed successfully, use the *HasErrors* property. [B1512]
- **Some DevForce DLLs and EXEs moved.** Because two new folders (*Tools* and *DesktopAssemblies*) were added to the installation directory and certain files were moved into these folders, applications that make reference to a relocated file will fail until the reference is updated. For example, a DevForce 2010 application that makes a .NET reference to *WPFTraceViewer.exe* or *WinTraceViewer.exe* in the *C:\Program Files\DevForce 2010* folder won’t work until that reference is updated to point to the file in its new location, typically the *C:\Program Files\DevForce 2010\Tools* folder. [B1430]

Defect Repairs

- Under certain circumstances, *AcceptChanges()* was not being called after a save. Fixed. [B1488]
- In a multi-model project, a static *EntityRelations()* constructor could be duplicated on primary and secondary *EntityManagers*. Fixed. [B1480]
- Custom verifier error messages were not being displayed for null property values on custom subclasses of the *PropertyValueVerifier*. Fixed. [B1489]
- QueryInversion was not working properly for some queries. Fixed. [B1496]
- Differences between the way that EF and the .NET CLR evaluated the same LINQ expression could cause string equality (casing, whitespace) and *Guid* ordering issues on queries. Fixed. See the [DevForce Wiki](#) and the *DevForce API Documentation* (installed with the product; available via the Windows Start menu) on the topic *EntityQuery.CacheQueryOptions* for more information. [B1491]
- In a multi-model project, multiple models using the same *DatasourceKey* would cause an exception to be thrown. Fixed. [B1498]

- In a multi-model project it was possible to create a project with no primary *EntityManager*. Fixed. [B1500]
- In a multimodel project, there was an attribute collision when using same injected base class on multiple models. Fixed. [B1501]
- *EntityAspect.AddToManager()* would run in $O(n^2)$ time. Fixed. [B1502]
- Custom verifier error messages could be superceded by the default attribute verifier messages depending on the verifier discovery method. Fixed. [B1507]
- Linq 'Contains' clauses with arrays did not work properly in n-tier environments without adding to the known type collection because of *Array* serialization issues. Fixed. [B1510]
- Verifiers defined via an *IVerifierProvider* were not being rediscovered after a *VerifierEngine.Clear()*. Fixed. [B1511]
- Saves with entities containing multiple properties of the same complex type set to the same value could fail. Fixed. [B1514]
- *EntityManager.RefetchEntities()* with *MergeStrategy.PreserveChanges* would incorrectly remove a deleted entity from the entityManager. Fixed. [B1517]
- Save of a deleted entity after a refetch would use the wrong concurrency value. Fixed. [B1518]
- Deletes were not using the correct original value of their concurrency column under some circumstances. Fixed. [B1519]
- Setting the 'Default Value' on a boolean property in an EDMX would cause compile to fail due to capitalization error. Fixed. [B1522]
- Entities removed within the *EntityManager.Queried* handler were being given an *EntityState* of *Added* instead of *Detached*. Fixed. [B1527]
- *AsyncParallelTasks* with no tasks defined were not invoking the callback method. Fixed. [B1530]
- Code generation of the *EntityRelations* class would reverse *Role1* and *Role2* under some circumstances. Fixed. [B1537]
- Relationships where only one side had navigation properties defined were failing. Fixed. [B1540]
- Concurrency bug with deletes & uninitialized concurrency fields. Fixed. [B1542]
- The *ICloneable.Clone()* implementation on *Entity* previously caused the cloned entity to share its fields with the original entity. Fixed. [B1545]
- Probing of unmanaged DLLs previously threw an exception. Fixed. [B1499]
- Fixed a problem with *RefetchEntities()* when using the *PreserveChangesUpdateOriginal* merge strategy which occurred when the "remote" entity had been deleted. [B1503]
- Previously an infinite loop occurred if an exception was thrown in an async completion handler when the query completed synchronously. Fixed. [B1504]

IdeaBlade DevForce Release Notes

- Previously the presence of a signed assembly in the Silverlight web project caused an exception to be thrown. Fixed. [B1505]
- Fixed a problem with the MEF CompositionContainer not being entirely thread-safe. [B1506]
- The error message received by the client when the IIdentity/IPrincipal returned by a remote login is not serializable has been improved. [B1509]
- The *NullLogger* (used when a debug log is not enabled and no other *ITraceLogger* is found) now writes Trace messages (useful in Azure diagnostics). [B1524]
- All BOS services now use AddressFilterMode = Any, as required for load balancing in Azure. [B1525]
- Added a setting, multipleSiteBindingsEnabled = “true”, to the web.config files generated by DevForce templates, for easier Azure deployment. [B1528]
- DevForce no longer adds an unnecessary reference to the *WindowsBase* assembly when auto-generating the domain model. [B1533]
- Virtual directory names containing underscores are now supported. [B1539]
- Virtual directory names containing slashes (sub-folders) are now supported by the default Silverlight configuration. [B1541]
- Previously some DevForce 2010 templates that should have been removed during a DevForce uninstall were missed. Now when you uninstall DevForce (whether as a separate operation or as part of the install of a newer version), all DevForce 2010 templates previously installed for the user who initiated the uninstall are now removed. This includes templates installed (after product installation) using the Template Installer utility on the DevForce Start menu.

Note that DevForce 2010 templates installed for *other* users on the machine are *not* removed. [B1487]

- Previously the OM Designer Extension did not get installed if VS2010 was newly installed and had not been exercised. The extension is now installed correctly even in that circumstance. [B1515]
- Previously the Template Installer did not always work in some non-U.S. environments. This problem has been corrected. [B1526]
- Previously, .NET interfaces did not show up in class definitions in the *API Help Reference* (the .chm file). This has been corrected. [B1534]

Breaking Change Defects

For detail on the Breaking Change defects listed below, see the discussion in the **Breaking Changes** section of these 6.0.5 Release Notes.

- EntityQueryOp<T>.IsCompleted Behavior Changed [B1512].
- Some DevForce DLLs and EXEs moved. [B1430]

New and Improved Product Features

IdeaBlade DevForce Release Notes

- Navigation properties that contain collections can now be marked as read-only. All navigation properties will now appear in the EDMX designer window with an “Is Collection ReadOnly” choice. By default this value is *false*, but if set to *true*, then any collections returned by this property will be read-only. Furthermore, since every navigation property that returns a collection does so with an instance of a *RelatedEntityList<T>*, the *RelatedEntityList* now has an *IsReadOnly* property that will automatically be set based upon the EDMX design-time setting. It can, however, be changed at runtime to allow modifications of the collection. [F1326]
- Support for *CompositionContext*, *Fakes*, and *Mocking*. See the [DevForce Wiki](#) and the *DevForce API Documentation* (installed with the product; available via the Windows Start menu) for more information. [F1351]
- Additional signatures have been added for *EntityManager.RefreshEntitiesAsync()* to make it more consistent with its synchronous counterpart. [F1464]
- We now support code generation when the EDMX model has no mappings or datastore ("Model-First code generation"). This may be very useful in conjunction with the *EntityServerFakeBackingStore* discussed in the [DevForce Wiki](#) and the *DevForce API Documentation* (installed with the product; available via the Windows Start menu) for more information. [F1469]
- Two additional values have been added to the *EntityState* enumeration: *AnyAddedModifiedOrDeleted* and *AllButDetached*. [F1474]
- Within code generation templates, we now allow the *PropertyDef* and *MethodDef* classes to describe virtual properties and methods. [F1478]
- Error messages issued when DevForce discovers the developer's machine to be using an unsupported (too early) version of the Entity Framework and .NET have been improved. [F1487]
- It is now possible to configure DevForce to perform Required Value validations in the same manner that .NET validation facilities do.
We have added a *TreatEmptyStringAsNull* Boolean property to the *VerifierOptions* class to permit the developer to force DevForce validation behavior to match that of .NET for required value verifiers defined using attributes.

This was necessary because DevForce and .NET validation facilities treat empty strings differently when performing Required Value checks. By default, DevForce treats an empty string as a null for the purposes of validation;.NET does not.

We have kept our current default behavior but added the new property to permit you to alter DevForce's behavior. The default value for *TreatEmptyStringAsNull* is *true*. When set to *false*, DevForce validation behavior will match that of .NET. The property can be set for individual verifiers, or set once for the entire environment (via the *VerifierEngine.DefaultVerifierOptions*). [F1488]

- It is now possible to configure DevForce to perform Required Value validations in the same manner that .NET validation facilities do.

By default, the DevForce Verification engine treats empty strings as nulls when performing Required Value checks. The .NET validation engine does not.

We have kept our current default behavior but added a new property, *TreatEmptyStringAsNull*, to the *VerifierOptions* type. Its default value is *true*. When set to *false*, DevForce validation behavior will match that of .NET.

The property can be set for individual verifiers, or set once for the entire environment via the

IdeaBlade DevForce Release Notes

VerifierEngine.DefaultVerifierOptions. [F1488]

- Additional folders (“Tools” and “Desktop Assemblies”) were added within the main DevForce installation folder more helpfully to organize the content. [F1430]

Breaking Change Features

For detail on the Breaking Change features listed below, see the discussion in the **Breaking Changes** section of these 6.0.5 Release Notes.

- Exceptions Thrown by Asynchronous Operations Must Be Handled. [F1408]
- *EntityManager*. *AuthorizedThreadId* Property. [F1471]
- The *ComplexObject* now has a *ComplexAspect* that is analogous to the *Entity-EntityAspect* relationship. [F1475]
- Constructors for the *EntityManager* have been changed to use default parameters and to support the new *EntityManagerContextClass* and the *CompositionContextName* parameter. [F1480]
- Added a using (Imports) statement for the IdeaBlade.EntityModel namespace to all code files created by the DevForce templates. [F1465]
- An “Implicit” connect is no longer performed on a disconnected *EntityManager*. [F1219/F1470]

Known Issues

The NorthwindIB database (used with most of the code samples) will not work with versions of SQL Server prior to SQL Server 2008. If you need a version compatible with SQL Server 2005, contact [IdeaBlade Support](#).

Version 6.0.4

Summary

This release of DevForce 2010 is the first to include support for Push Notification in Silverlight applications. It also includes a number of other minor enhancements bug fixes.

Breaking changes

Implementation of Push Notification for Silverlight resulted in breaking changes for current users of the Push feature. These are documented in the section “New and Improved Product Features”.

Defect Repairs

- The “source” of trace messages written to the debug log was corrected to show the true calling method name. [B1411]
- Null parameters can now be passed to *StoredProcQueries*. [B1423]
- DevForce WinForm project templates no longer display a trust warning when used. [B1427]

IdeaBlade DevForce Release Notes

- The `GettingStarted` link no longer throws an exception for non-administrative users. [B1430, B1434]
- The error messages shown when *IIdGenerator* and *IConcurrencyValueSetter* implementations are required but not found have been corrected. [B1438]
- *ProbeAssemblyName* elements in the `ideablade.configuration` are now loaded even if not fully-qualified. [B1439]
- Error handling during MEF probing has been improved. Previously, an assembly which could not be loaded for probing would cause an error and require that a filter (*CompositionHost.SearchPatterns*) be used as a workaround. Now, probing errors are logged and probing continues. [B1440, B1457]
- DevForce now ensures that *EdmKeys* are initialized prior to issuing any queries or saves. Previously, the keys were not initialized in some situations when either a *PassthruEsqQuery* or *StoredProcQuery* was used, resulting in “Entity not in context” errors. [B1454]
- Projects created with the DevForce Silverlight Application template, and the n-tier DevForce WinForms and WPF application templates, now always prompt for the project location when created. Previously, if the “save new projects when created” setting in Visual Studio was off, the web application project could not be saved to a permanent location. [B1471]
- DevForce now provides more descriptive error messages for failures during save processing. [B1476]
- Failures during execution of a scalar async query (e.g., *entityManager.Customers.AsScalarAsync().First()*) are now correctly returned to the specified callback and completion handler. [B1479]
- The attempt to save an entity involved in a many-to-many association after removing and re-adding it to that association no longer results in a Save exception. [B1477]
- Saves no longer fail on a many-to-many entity after retrieving its M-2-M siblings via a collection navigation property. [B1456]
- Detached entities and Null Entities now always return a null value for the *EntityAspect.EntityManager* property path. [B1466]
- *ServerEncryptionKeys* can now be set on all versions of DevForce regardless of license. (However, shared Load Balanced keys still require the DataCenter Server edition.) [B1478]
- Setting *DomainModelTemplate.NamingService.FullyQualifySystemTypes* = true now fully qualifies all System type references in the generated code. [B1475]
- In `DomainTemplate.cs`, the *WriteEntityDataPropertyDisplayNameAttribute()* method is now marked ‘protected virtual’ instead of private. [B1472]
- A Threading issue with *CompositionHost* (MEF configuration) has been fixed. [B1477]
- Guid (*uniqueIdentifier*) properties/columns are now handled correctly when marked as (StoreGenerated) Identity columns. [B1464]
- An exception is no longer thrown by Stored Procedure calls that returns nothing. [B1444]
- High resolution *DateTime* properties in a Complex Type can no longer cause a save to fail. [B1452]

IdeaBlade DevForce Release Notes

- A non-public navigation property no longer prevents the use of the corresponding public property on the related entity (Silverlight only) [B1450]
- *RejectChanges()* calls can no longer cause incorrect child-to-parent navigation issues. [B1448]
- Code generation now fully qualifies references to the *System.Reflection.Assembly* methods in generated code. [B1445]
- A problem with instance verification not always reporting validation from cross-object verifiers has been fixed. [B1403]
- *IEditableObject.Cancel* now works properly with *INotifyDataErrorInfo.GetErrors()*. Previously, errors were not always reset properly. [B1420]
- *EntityManager.ImportEntities()* can no longer break navigation properties. [B1404]
- Some templates previously not installed with Enterprise EF license, are now properly installed. [B1486]
 - If you use a Universal license, you will get six Templates. (1 BOS + 1 Silverlight + 2 WinClient + 2 WPF)
 - If you use a Enterprise EF license, you will get five Templates. (1 BOS + 2 WinClient + 2 WPF)
 - If you use a Silverlight license, you will get two Templates. (1 BOS + 1 Silverlight)
- Fixed issue with *Entity.RejectChanges()* not working properly if called immediately after *EntityManager.SaveChanges()*. [B1488]
- DevForce now probes just once for known types (e.g., *IIdGenerator*), caching the results (previously it repeated the probing under various circumstances). This improves the performance of *ImportEntities* and 2-tier *SaveChanges*. [B1490]

New and Improved Product Features

- Push Notification is now supported in DevForce Silverlight applications. Previously this feature had been available to non-Silverlight DevForce applications only. The Push feature allows client applications to “subscribe” to server-side code in order to receive periodic user-defined notifications.

Implementation of this feature also resulted in breaking changes for current users of the Push feature, as follows:

- The *serverPort* attribute on the *NotificationService* element has been removed. This attribute allowed the *EntityService* and *NotificationService* to listen on different ports, while still using DevForce “programmatic” setup, i.e., setup not requiring the *serviceModel* section of the config file. Programmatic setup now defaults to using the same port number for the *NotificationService* as is used by the *EntityService*. You can still use different port numbers if you define these endpoints via the *serviceModel* configuration.
- The *RegisterCallback* and *CancelCallback* methods (defined on the *EntityManager*) are now always executed asynchronously. Instead of passing a *ClientNotifyDelegate* to the *RegisterCallback* you now supply an *Action<SubscriptionOperation>*. If an error occurs during the subscription process it will be passed to the callback action via the *SubscriptionOperation.Error* property.

Additional information on the Push feature, with code samples, is available via the online DevForce Resource Center. [F1367]

- Support was added for async *FirstOrDefaultEntity* and *SingleOrDefaultEntity* methods [F1426]. Example:

```
C# var op = _entityManager.Employees.AsScalarAsync().FirstOrDefaultEntity(e => e.Id
```

IdeaBlade DevForce Release Notes

```
== 1);  
op.Completed += (o, e) => {  
    var emp = e.Result;  
};
```

- The client portion of n-tier applications can now be built using the .NET Client Profile. Note that 2-tier applications, in which a BOS is not used, still require the full .NET profile. [F1442]
- Silverlight only - Support filtering of the assemblies probed. (This feature was previously available only to non-Silverlight applications.) By default all files in the XAP with a “.dll” extension will be probed. To change this, set `CompositionHost.SearchPatterns` in your `Application_Startup` logic to ensure it occurs before other DevForce initialization. Search patterns are in Regex format [F1450]. For example:

```
C# private void Application_Startup(object sender, StartupEventArgs e) {  
    IdeaBlade.Core.Composition.CompositionHost.SearchPatterns.Clear();  
    IdeaBlade.Core.Composition.CompositionHost.SearchPatterns.Add(@"MyApp\.dll");  
    ... other startup logic  
}
```

- Push Notification is no longer restricted to the DataCenter edition: it is now available in all DevForce editions. [F1451]
- We added a partial workaround to handle an Entity Framework bug with setting the CSDL `StoreGenerationPattern`. The EF bug is described at the following link:

<https://connect.microsoft.com/VisualStudio/feedback/details/505178/storegeneratedpattern-property-in-ado-net-entity-model-designer-sets-cdsl-annotation-but-not-ssdl-attribute?wa=wsignin1.0>

The workaround is that we now offer a setting at the top level of the EF Designer labeled ‘Handle Mapping Issues/Mismatches’ with three settings possible ‘Fix’, ‘Warn’ or ‘Skip’. ‘Fix’ will attempt to fix these mismatches if possible by modifying the SSDL section of the EDMX or by adding a warning comment to the EDMX if the schema is too difficult for DevForce to fix automatically. ‘Warn’ will simply add warnings (and suggestions about how to fix the issues) to the EDMX as XML comments. ‘Skip’ will simply not attempt to do any ‘fixup’. [F1460]

- We now allow ‘paired’ Silverlight and Web Model Assemblies to have different assembly names. This capability is important because Microsoft’s WPF and Silverlight tooling has problems when a Silverlight and a desktop assembly both have the same name within a single solution. Type correspondence between Silverlight and Desktop types is now performed by comparing ‘namespace’-qualified type names. [F1423]
- Property interceptors can now be marked as ‘private’ or ‘internal’ in both Desktop and Silverlight applications. [F1455]
- MEF probing informational messages have been added to the Debug log. [F1458]
- DevForce will now throw a compiler error if the Entity Framework Foreign Key support is not turned on. We have always required this in order for the model to execute, but users often did not discover this until runtime. We felt that failing earlier was better. [F1446]
- Nongeneric *GetValue* and *SetValue* methods have been added to the *EntityProperty* class. [F1433]
- *EntityAspect* indexers (index methods) have been replaced with *GetValue* and *SetValue* methods. This was done to accommodate shortcomings in Microsoft’s Blend and Cider designers where the existence of certain forms of indexers would cause features of both designers to fail. [F1424]

- *EntityAspect* now exposes a new mutable *ValidationErrors* property that is coordinated with the *INotifyDataErrorInfo* interface and the DevForce Verification engine. Any validation errors encountered on an entity can be viewed and modified directly by examining the *EntityAspect.ValidationErrors* property, and any errors added or removed from this collection will cause the appropriate *INotifyDataErrorInfo* events to fire. [F1415]
- The *BaseOperation.Completed* event is now raised when a derivative Operation raises its *Completed<SomeEventArgs<T>>* event. [F1431]
- Code generation: Four additional overridable methods have been added to the *DomainModelTemplate.cs* for code generation to allow attributes to be placed directly on property getters and setters. *WriteEntityDataPropertyGetAttributes*, *WriteEntityDataPropertySetAttributes*, *WriteEntityNavigationPropertyGetAttributes*, *WriteEntityNavigationPropertySetAttributes*. [F1457]
- Code generation: Many additional members of the *DomainModelTemplate* class have now been made protected to allow subclasses to access them. [F1454]
- Code generation: DevForce now generates *Editable(false)* & *ReadOnly* UI hint attributes if a setter is less accessible than *public*. [F1453]
- Code generation: A 'Tag' property has been added at each level of the EF designer and is editable in the EF designer property window. Any values set here are available for use in generated code via a *Tag* property on the associated DevForce EDM context object passed into *DomainModelTemplate Write* method. [F1434]
- A new shortcut, "Template Installer", was added to the DevForce 2010 / Tools start menu. During installation, the DevForce Project Templates are installed only for the user who is logged in. This is also true for the Template Installer utility; but since it is run *after* DevForce is installed it can be used to install an additional set of DevForce templates for any user logged into the machine. [B1428]

Learning Resource Updates

The "DevForce Learning Resources" consist of technical documentation, articles, papers, code samples, and videos.

We used to ship some of these resources in the DevForce product download. As of version 6.0.4, you will find all such resources on our web-based [DevForce Resource Center](#).

This cycle we put most of our effort into building the wiki. We also improved existing content and added new material. We expect to add and improve content continuously now that we have a convenient home for it online.

Known Issues

None.

Version 6.0.3

Summary

Version "603" is our RTM ("Release To Manufacturing") version, following closely on the heels of Microsoft's Silverlight RTM.

We had three primary objectives:

1. Make DevForce easier to learn, understand, and use (a work-in-perpetual-progress)
2. Make breaking changes to fix fundamental mistakes in the RC before going RTM
3. Fill some of the feature gaps such as Entity Framework derived entities backed by stored procedures and async scalar queries.

Breaking changes

The DevForce server throws a `LoginException` when credentials fail ASP.NET Membership validation. [1371]

The `EntityManager.EntityServerError` event is now raised for async query and login failures. The event was previously raised only for async connect and save failures. Your `EntityServerError` event handler will now be called when an async query or login fails ... which shouldn't come as a surprise but might disrupt your handler's workflow. [1406]

DevForce raises an exception if the persistence operation (e.g., a query) failed or was cancelled and you attempt to access the results of that operation either through the operation value or the pertinent event args. Previously the results returned an empty list without complaint. Your query would appear to have "succeeded" ... and found nothing.

You should test the `HasError` and `Cancelled` properties before reading any `Result(s)` property. Your code will break if you expected always to be able to access `EntityQueryOperation.Results`, `EntityQueriedEventArgs.Results`, or `InvokeServerMethodEvents.Result`. [1407]

We "hid" or moved two members of `Entity` that were public and directly accessible in version 6.01. [1360]

We believe the public API of your entity should speak to the business purpose of the entity; the consumer of your entity shouldn't have to bother with internal infrastructure concerns unless and until those concerns become important to the caller. Accordingly:

- `ErrorsChanged` is now implemented explicitly as `INotifyDataErrorInfo.ErrorsChanged`; you must cast the entity to `INotifyDataErrorInfo` to access it.
- `PendingEntityResolved` moved to `EntityAspect.PendingEntityResolved`.

We could not move `ToQuery<T>` - the method that constructs a query returning this entity. We wanted to move it to `EntityAspect` but could not because it is generic and `<T>` must refer to the type of the entity. The utility of the method trumped principle.

We debated reimplementing `PropertyChanged` explicitly as `INotifyPropertyChanged.PropertyChanged`. While this is "the right thing" in principle, we opted against doing so because we felt many of you expect to see and handle this well-known event.

Defect Repairs

No longer throwing MEF exception when using sub-typed `AspAuthenticatingLoginManager`. [1415]

EDM metadata discovery no longer throws an unhandled exception in multi-tenant applications. [1413]

`EntityQueriedEventArgs.ChangedEntities` no longer throws an `ArgumentNullException` when the list of entities added or removed from cache is null. [1409]

ASP.NET Windows authentication mode no longer requires that `AllowAnonymousLogin` be enabled. [1340]

Updated our Visual Studio Silverlight templates to use the Silverlight RTM version number [1341]

The *Required* verifier produces one validation error message (rather than two) in the Silverlight `DataGrid`. [1344]

IdeaBlade DevForce Release Notes

Data source extensions now work in Silverlight. [1347]

Resolved `EntityManager.SaveChangesAsync` signature conflict (visible only in VB) [1348]

Fixed an invalid cast exception thrown while processing the non-generic `EndExecuteQuery`. [1350]

Factor IIS 'application' name into generation of server URL for Silverlight applications. [1353]

Corrected a problem in determining which of multiple EDMXs "*IsPrimary*" when adding a second EDMX. [1355]

Support readonly properties on a POCO type. [1359]

The internal `EntityTypePropertyNames` classes inherit from an injected custom base type. Previously inherited from the DevForce `Entity` class. [1362]

`EntityMetadata.DevForceDefaultValueFunction` provides a default value for properties returning the `DateTimeOffset` type. [1365]

Custom `UserBase` implementations are discovered. [1368]

Rationalized behavior of validation setting made in the EDM. [1373; see also feature 1362]

We could not find a good use case for DevForce entities to use the Silverlight validation engine. To do so anyway was causing confusion, would require updating the T4 template to call Silverlight validation within the property setters, and would necessitate adding the `ValidateInstance` and `ValidateProperty` attributes.

`ValidateInstance` and `ValidateProperty` attributes are no longer generated. They remain available and will be honored if the developer decides to *both* call the Silverlight validation engine *and* combine .NET validation rules and DevForce verifiers.

Improved `IdeaBlade.Validation` XML reference documentation for Intellisense and API Help. [1374]

`Skip()` is only meaningful for a `DataSourceOnly` or `CacheOnly` query. [1377]

Can't ensure reliable results of a skip query applied to both cache and the persistent data store. Therefore, when specifying skip:

- `FetchStrategy.Optimized` is resolved to `FetchStrategy.DataSourceOnly`
- `FetchStrategy.DataSourceThenCache` - Throw exception.

Exception message asks you to specify a meaningful `QueryStrategy/FetchStrategy`.

`INotifyDataErrorInfo.GetErrors` collection is cleared after `IEditableObject.Cancel` [1378]. Note that this will remove validation results that were in the collection before editing began.

Support Silverlight client queries returning anonymous types containing a `RelatedEntityList` or complex type. [1379]

Cross field validations raise `INotifyDataErrorInfo` events in the manner expected by the Silverlight `DataForm`. [1381]

Resolved serious performance issue with queries for inherited types that returned many entities. [1382]

Queries with both recursive and non-recursive `Includes` now mark the recursive entities as `Loaded` only when they actually are loaded. [1383]

`QueryCache` considers `Includes` when determining if a query has already been executed. [1389]

Generate .tt file in same physical location as the Entity Framework EDMX file, even when the EDMX is in a subdirectory. [1391]

`AuthorizationQueryResult` considers `Included` entities in the evaluation. [1393]

ComplexTypes with null subproperties no longer cause binding exceptions. [1394]

Queries with a `QueryStrategy` of `DataSourceOnly` are added to the `QueryCache` correctly [1395, 1396]

`UserBase` no longer uses a static field initializer. [1399]

New and Improved Product Features

Support entities backed by stored procedures. [1353]

Support asynchronous execution of scalar (immediate) queries. [1279]

Entity Framework LINQ provides “reduce” methods such as `Count`, `First`, and `Sum` that return a single value (scalar) result. These methods execute immediately (like “`ToList`” and unlike the compositional methods “`Where`”, “`OrderBy`”, etc.). In the synchronous EF LINQ world, they block until the query execution completes.

We have added asynchronous, non-blocking equivalents especially for Silverlight developers. We added the **`AsScalarAsync()`** extension method to the `EntityQueryExtension` class. It takes an `IEntityQuery<T>` object and returns an `IEntityScalarQuery` object. `IEntityScalarQuery` is extended in turn by the `EntityScalarAsyncExtensions` class which exposes asynchronous versions of `Average`, `Count`, `First`, `FirstOrDefault`, `LongCount`, `Max`, `Min`, `Single`, `SingleOrDefault`, and `Sum`.

Each is structured like `ExecuteAsync<T>()`. Each returns an `EntityScalarQueryOperation<T>` and takes a callback that receives an `EntityScalarQueryOperation<T>`. A handler of the `Completed` event receives an `EntityScalarQueriedEventArgs<T>`.

Both the operation and the event args have a strongly-typed **`Result`** property; note the singular property name in contrast to the plural `EntityQueriedEventArgs<T>.Results`.

The following test demonstrates querying for the first ‘D’ employee:

```
var op = _eml.Employees.AsScalarAsync()  
    .FirstOrDefault(e => e.LastName.StartsWith("D"));  
op.Completed += (o, e) => {  
    var emp = e.Result;  
    Assert.IsNotNull(  
        emp, "Did not retrieve a first employee beginning 'D'.");  
};
```

We’ll implement `FirstOrDefaultEntity` and `SingleOrDefaultEntity` in a future release.

Added the read-only `EntityRelationQuery.Entities` property to reveal the query source entity (or entities). With this property you can log which `Order` (or list of `Orders`) were the source of an `EntityRelationQuery` that returned related `Customer(s)`. [1315]

No longer throw when query condition tests a nullable string property [1320].

Until this change, a query for `Customer` with the `Where` predicate:

```
x => x. Name.Contains("Acme")
```

would throw a `NullReferenceException` if there existed any customers in cache with a null `Name`.

The query would succeed on the server because SQL tolerates null field values. But in .NET, “`x.Name`” must be non-null before executing “`Contains()`”.

The workaround was to rewrite the predicate as:

IdeaBlade DevForce Release Notes

```
x => x.Name != null && x.Name.Contains("Acme")
```

DevForce now accepts the coalescing operator so we can write:

```
x => (x.Name ?? String.Empty).Contains("Acme")
```

Because most developers are unlikely to recognize or remember this issue, DevForce automatically converts

```
x => x.Name.Contains("Acme")
```

to

```
x => (x.Name ?? String.Empty).Contains("Acme")
```

Auto-coalescing is also applied to non-boolean methods:

```
x => x.Name.ToUpper().Contains("ACME")
```

becomes:

```
x => (x.Name ?? String.Empty).ToUpper().Contains("ACME")
```

We auto-coalesce the following supported String properties: Contains, EndsWith, Length, ToLower, ToUpper, and StartsWith.

Considering implementation for IndexOf, IndexOfAny, LastIndexOf, LastIndexOfAny, ToLowerInvariant, ToUpperInvariant, Trim... if they are supported by EntityFramework.

We have found no comparable issue with the other nullable primitive properties which are not coalesced.

Improved error message returned when a connection string is not found. [1390]

Improved error message returned when the metadata definition in a connection string is invalid. [1411]

Added missing *HasError* property to AsyncEventArgs; equivalent to `args.Error != null`. [1417]

Added *Cancelled* flag to AsyncResult. [1350]

Added the "Start Menu | DevForce 2010 | Tools | OM Designer Extension Installer" shortcut to facilitate re-installation of DevForce Object Mapper Visual Studio extension in case the developer de-installed it for some reason. [1412]

RestoreStrategy.Normal no longer overrides EntityManager defaults. [1274] [RestoreStrategy](#).Normal returns `new RestoreStrategy(false, false, MergeStrategy.PreserveChanges)`.

Custom DefaultValue logic for entity initialization. [1351]

DevForce initializes entity data properties of a newly created entity with the value specified each property's [DefaultValue] attribute ... if it has one. Otherwise, it turns to the [EntityMetadata.DefaultValueFunction](#) to get the default initialization value for the property's target type. You can replace the DevForce function with a custom function. See "[How DevForce Initializes New Entity Properties](#)" below.

Code generation and performance improvements for handling of validation and verification attributes. [1362]

Silverlight UI controls perform lots of excess .NET validation calls. Most DevForce developers use the more robust/capable DevForce "Verification" validation instead. .NET validation support turned off by default to improve performance (you can turn it on at will).

Added QueriedEntities collection to [EntityServerQueryInterceptor](#). [1374]

IdeaBlade DevForce Release Notes

Now you can examine queried entities on the server **before they are sent to the client**. You could record what was queried for audit purposes (“What returns has a tax preparer tried to view?”). You can write fine-grained authorization logic to prevent transmission to client of forbidden entities ... in case they slip past a protective query filter you added before the query was processed.

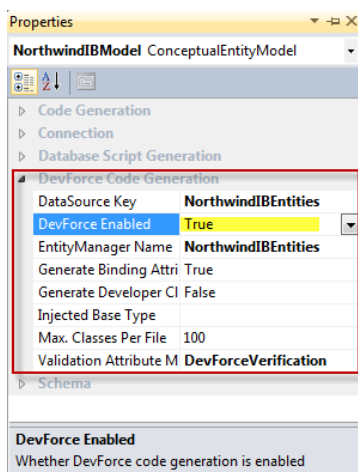
You cannot add or remove entities to the collection (yet). You can modify the entities (e.g., obscure the Social Security Number) before they continue on their way to the client. But be careful, especially if you expect the user to be able to update these entities and save them later!

Support `EntityManager.SaveChanges` from within `EntityManagerQueryInterceptor` and `IEntityLoginManager`. [1375]. Useful for persisting queried-entity-audit-trail as described for [1374] or persisting login activity.

DevForce code generation can be disabled and enabled on a per EDMX basis. [1376]

Entity Framework code generation is governed by the EDMX “Custom Tool” property setting which is “EntityModelCodeGenerator” by default. When DevForce took over code generation, we used to replace that value with “<RemoveTo...> EntityModelCodeGenerator</>”.

Doing so both disabled Entity Framework code generation and left a reminder as to what the value had been to make it easier for you to restore EF code gen if you wanted to. However, that string triggered compilation warnings which annoyed many developers.



Instead, we’ve added a model level tag (visible within the Entity Framework designer’s property window) to control whether to generate DevForce or Entity Framework entity classes from the EDMX.

When you toggle the setting:

DF code generation = true

EDMX custom tool <= null (empty)

Add our .tt and IB.Designer files.

DF code generation = false

EDMX custom tool <= EntityModelCodeGenerator

Removes our .tt and .IB.Designer files.

Observe that, for a given EDMX in a particular project, you may generate either EF classes or DevForce classes but not both. Previously you could physically generate both but it never worked because the two code generators duplicated the class definitions (e.g., two Employee classes, one EF and one DF). This change makes obvious the need to choose one path or the other.

ReferenceStrategy is accessible directly from `NavigationEntityProperty` [1382]

Makes it easier to reach and set `ReferenceStrategy` which you might want to do to prevent lazy loading of `Customer.Orders`.

```
Customer.PropertyMetadata.Orders // NavigationEntityProperty
    .ReferenceStrategy = EntityReferenceStrategy.NoLoad;
```

This setting disables lazy loading of the `Customer.Orders` property only. You can change the setting at runtime at will. Meanwhile, lazy loading remains enabled for the application as a whole.

IdeaBlade DevForce Release Notes

You can still load the orders of a “some customer” on demand by writing

```
Customer.PropertyMetadata.Orders
    .GetEntityReference(someCustomer)
    .Load(MergeStrategy.OverwriteChanges);
```

Removed dependence on `Microsoft.CSharp.dll` in `IdeaBlade.EntityModel.SL` [1385].

The CSharp assembly weighs more than ½ megabyte. If you didn’t need it, your XAP would take longer to download than it should.

The CSharp assembly has little to do with the C# language. In fact, it is only required if you use the dynamic keyword ... which we stopped doing in our Silverlight EntityModel assembly (we still use it in our regular .NET EntityModel assembly).

We reserve the right to depend on the CSharp assembly in future; we would only do so if it provided truly substantial advantage to the product ... and thus to you.

`IHideObjectMembers` in Silverlight and `IdeaBlade.Core` namespaces to hide distracting `Object` class members (`Equals`, `GetHashCode`, `GetType`, `ToString`) from IntelliSense. The members are still available; they just don’t clutter the IntelliSense member listing. [1386]

Added `EntityManager.LinkForAuthentication` method [1392]

All EntityManagers must be “logged in” before they will permit any server operations. You may need to create multiple EntityManagers. You only want to authenticate once.

Use this message to pass the authentication token from one EntityManager to another. You could write:

```
var secondManager = new NorthwindIBEntities();
// pass reference to authentication token in first manager
secondManager.LinkForAuthentication(firstManager);
```

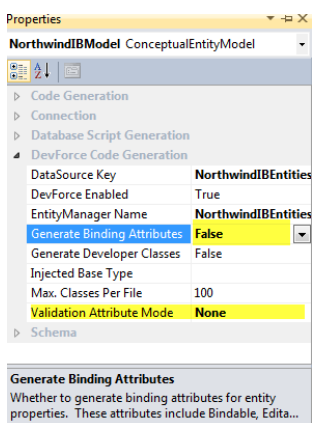
Notes:

- Although an EntityManager is the parameter, the second manager acquires a reference to the authentication token, not to the manager. It can survive the destruction of the first manager.
- Logging out of one manager has an uncertain effect on the other manager. If we logout the first manager, the second manager may be able to continue working with the server if that server is able to re-authenticate the second manager using the authentication token.
- Logging in the first manager again gives it a new authentication token. Now the first manager and the second manager are using different tokens and behaving as if they were two applications even if both are logged in under the same user.
- You cannot call `LinkForAuthentication` on a target manager that is already logged in.
- You cannot call `LinkForAuthentication` with a source manager that is not logged in.
- It follows that when you log out, you should destroy all other managers that were authenticated with it via `LinkForAuthentication`.
- This authentication transfer mechanism is an alternative to the EntityManager semi-copy-constructor overload that takes an existing manager as a parameter.

Can override DevForce T4 generation of individual attributes. [1361]

IdeaBlade DevForce Release Notes

Can suppress code generation of most non-essential attributes. [1401]



Setting “Generate Binding Attributes” to false blocks generation of the “UI Hint” attributes: Bindable, Editable, and Display.

Setting “Validation Attribute Mode” to “None” blocks generation of all validation and verification attributes.

Added Visual Studio templates for 2-tier and n-tier Windows applications. [1377]

Added “ReadMe” files to Silverlight templates [1405]

Learning Resource Updates

010_IntroToDevForce. Tutorial section (with accompanying code sample) added to describe how to add a *Visual Studio Team Test* project to a Visual Studio solution.

030_BaseApps\SilverlightApps\Tutorials\100SLV_Tour. Tutorial section (with accompanying code sample) added to describe how to add a *Silverlight Unit Test Framework* project to a Visual Studio solution.

040_BusObjPersistence_TopicDocumentSnippets solution.

- Added a set of demo methods to illustrate the client- and server-side life cycle (fetching, saving) event handlers and interceptors.
- Began process of refactoring demo methods so they can be more easily subjected to automated test
- Created corresponding test methods for about half of the demo methods in a new Test project. (Work to be continued for next release.)

040_BusObjPersistence\Business Object Persistence.pdf. Updated and enhanced material on the life-cycle event handlers and interceptors.

070_Validation. *Validation Through Verification* topic document extensively reworked to reflect current state of DevForce Verification facilities. *_TopicDocumentSnippets* code solution added which contains all code snippets from the topic document in runnable form.

Edited .SLN files for all solutions to move the appropriate project first in the sequence of projects. This causes it to be selected by Visual Studio as the Startup Project when the solution is first opened, even in the absence of an .SUO file.

How DevForce Initializes New Entity Properties

When you “new” an entity of type T or call `EntityManager.CreateEntity<T>`, DevForce delivers an **initialized** new instance of T. This instance is associated with but not yet attached to the EntityManager.

DevForce provides every (non-navigation) `DataEntityProperty` with an initial value. It first looks for a `[DefaultValue]` attribute – which can be specified either via a buddy class or in the EDMX (using the Default Value setting). If the attribute is not found, it then calls `EntityMetadata.DefaultValue` which sets a default value for non-nullable properties based on the data type.

IdeaBlade DevForce Release Notes

Of course any initialization you put in your constructor trumps these defaults.

You can override how DevForce calculates default values. The `EntityMetadata.GetDefaultValue` delegates to the function returned by the static `EntityMetadata.GetDefaultValueFunction` property.

We initialize this property to the static `EntityMetadata.DevForceDefaultValueFunction`. But you can write your own such function and substitute it for ours.

Please note that this is an advanced feature and you are responsible for ensuring the proper behavior of your substitute. You can call our function within yours. Take heed: although this function hasn't changed in ages we reserve the right to change it if we have to. You may prefer to replace it completely.

As this time, the `DevForceDefaultValueFunction` is as follows:

```
///<summary>
/// Returns the default value of a type: usually '0' or null for any data type.
/// Note that this is subtly different from the
/// <see cref="TypeFns.GetDefaultValue"/> method
/// in that it returns Today for a default DateTime.
/// N.B: While you are welcome to call it yourself,
/// this is a "DevForce Internal" function and we may change it in future.
///</summary>
public static object DevForceDefaultValueFunction(Type pType)
{
    if (pType == typeof(string))
    {
        return string.Empty;
    }
    if (pType == typeof(int))
    {
        return 0;
    }
    if (pType == typeof(long))
    {
        return 0L;
    }
    if (pType == typeof(short))
    {
        return (short) 0;
    }
    if (pType == typeof(double))
    {
        return 0.0;
    }
    if (pType == typeof(float))
    {
        return 0f;
    }
    if (pType == typeof(decimal))
    {
        return 0M;
    }
    if (pType == typeof(bool))
    {
        return false;
    }
    if (pType == typeof(DateTime))
    {
        return DateTime.Today;
    }
    if (pType == typeof(DateTimeOffset))
    {
        return new DateTimeOffset(DateTime.Today);
    }
}
```

IdeaBlade DevForce Release Notes

```
}  
if (pType == typeof(Guid))  
{  
    return Guid.Empty;  
}  
if (pType == typeof(byte))  
{  
    return (byte) 0;  
}  
return null;  
}
```

If you wanted a different default for DateTime values, you could write your own function like so:

```
///<summary>  
/// Returns the default value of a type in the manner of  
/// <see cref="EntityMetadata.DevForceGetDefaultValueFunction"/>  
/// except that the DateTime default is DateTime.Now  
/// with DateTimeKind.Unspecified.  
///</summary>  
public static object MyDefaultValueFunction(Type pType)  
{  
    if (pType == typeof(DateTime))  
    {  
        return DateTime.SpecifyKind(DateTime.Now, DateTimeKind.Unspecified);  
    }  
    return EntityMetadata.DevForceGetDefaultValueFunction(pType);  
}
```

You would initialize the `EntityMetadata` class before creating new entities as follows:

```
EntityMetadata.DefaultValueFunction = MyDefaultValueFunction;
```

Known Issues

Trigger links do not work properly for cross-object validation. For example, a cross-object verifier defined in the Order class to ensure that the OrderDate does not precede the related sales rep/(Employee)'s HireDate will execute upon changes to Order.OrderDate, but not upon changes to Employee.HireDate (as it should also do). For now you will have to duplicate the verifier definition in the Employee class. We intend to fix this problem by the next release.

Version 6.0.1

Summary

Version 6 inaugurates a new era at IdeaBlade, the **DevForce 2010** era.

As such, it deserves its own Release Notes. The Release Notes for older versions remain available as a separate document and are worthy of reference as you make the transition to DevForce 2010.

DevForce 2010 coincides with the equally new .NET 4 releases from Microsoft. While DevForce 2010 preserves substantial conceptual continuity with DevForce 2009, the new product represents both a dramatic step forward and a decisive break. We have much to talk about.

DevForce 2010 only supports .NET 4.0 and Silverlight 4.0. You must use Visual Studio 2010.

Visual Studio 2010 supports desktop .NET versions 2, 3.5, and 4. It supports Silverlight versions 2, 3 and 4. This can be very confusing, especially when you are migrating existing applications.

When you move to DevForce 2010 be sure that all of your desktop and web projects target .NET 4.0 and all Silverlight projects target Silverlight 4.0 as discussed below

Although .NET permits a .NET 4.0 assembly to call a .NET 3.5 assembly, it does not permit the converse, and mixing .NET versions is almost always a bad idea.

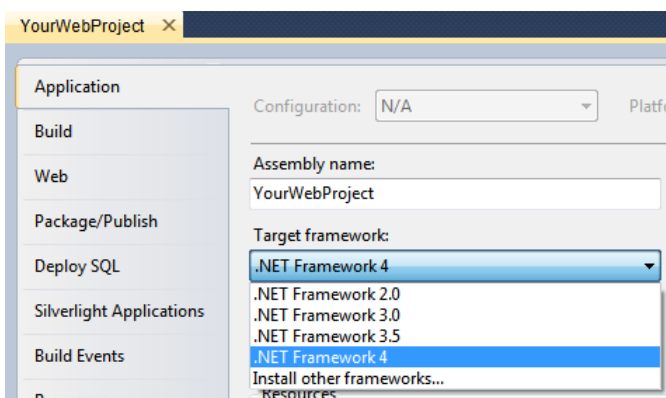
Setting the Target Version of .NET Assemblies

To control the version of .NET with which a project will be compiled into an assembly, select the project node in the Visual Studio Solution Explorer and press [Alt] + [Enter]. Alternatively, you can right-click and select *Properties* from the context menu..

From that point, the path to the setting depends upon whether your Desktop or Web project is in C# or VB.

C# Desktop and Web Projects:

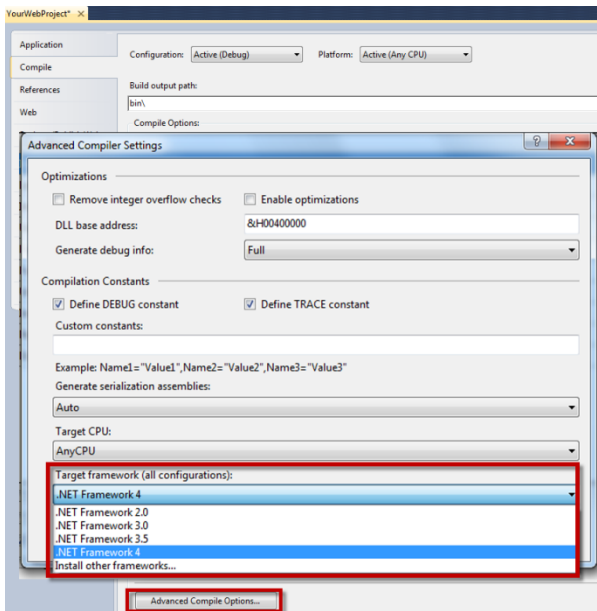
On the *Application* tab, select the *.NET Framework 4* in the *Target framework* ComboBox.



Visual Basic Desktop and Web Projects:

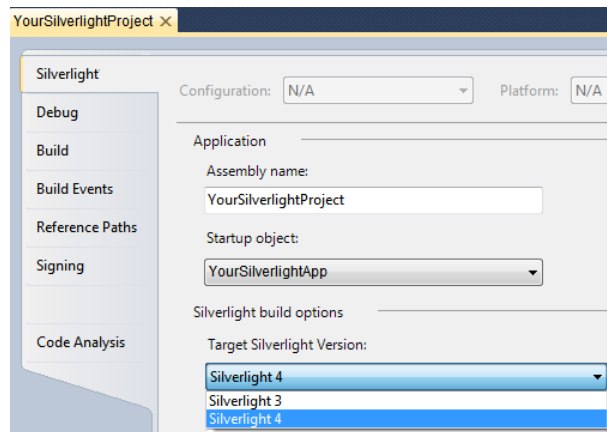
On the *Compile* tab, click the [Advanced Compile Options] button at the bottom. In the resulting *Advanced Compiler Settings* dialog, pick the target version of .NET in the *Target framework* ComboBox

IdeaBlade DevForce Release Notes



C# and Visual Basic Silverlight Projects:

Choosing the target Silverlight version is the same for both C# and Visual Basic projects. On the *Silverlight* tab, select *Silverlight 4* in the *Target Silverlight Version* combobox.



Mixing Application Versions on the Same Machine

Your DevForce 2010 solution must be built with Visual Studio 2010 and target .NET 4 and Silverlight 4.

You can still develop existing application solutions based on prior versions of DevForce (DevForce 2009, even DevForce 2005) on the same machine. Visual Studio 2008 runs fine side-by-side with Visual Studio 2010. You can also develop DevForce 2009 and DevForce 2010 applications in VS 2010. Your primary risk is confusing yourself.

Breaking changes

DevForce 2010 is a watershed release that breaks decisively from DevForce 2009.

IdeaBlade DevForce Release Notes

Major changes appear across the board. The most important derive from the product's reliance on the new Microsoft .NET technologies released in the spring of 2010:

- Visual Studio 2010
- .NET 4
- Silverlight 4
- Entity Framework v.4

DevForce 2010 **does not work** with:

- Earlier versions of Visual Studio
- Earlier versions of .NET or Silverlight
- Entity Framework v.3.5

DevForce 2010 supports Silverlight, WPF, Windows Forms, ASP WebForms, and ASP MVC clients.

This release of DevForce 2010 targets primarily the WPF and Silverlight developer.

DevForce 2009 Windows Forms Binding Managers are not shipped with DevForce 2010 although they are available and we continue to maintain them for DevForce 2009 and DevForce 2005.

You can still program to the BindingManager APIs with DevForce 2010. But the BindingManager Visual Designers have not been ported to Visual Studio 2010; you'll have to revert to VS 2008 to use them.

Please contact IdeaBlade if you intend to develop Windows Forms applications in DevForce 2010 or if you want more specifics on maintaining existing DevForce Windows Forms application to VS 2010.

Breaking Change Overview

Other DevForce 2010 documentation will cover features and practices in detail, largely from the perspective of someone approaching DevForce for new development. Here we cover changes that are pertinent to developers of existing DevForce applications who are considering moving to DevForce 2010.

We've organized those changes into broad categories as follows

- Entity Framework
- Entity Data Model Wizard and Designer
- Entities and the Entity Manager
- Validation
- Business Object Server
- Configuration

Such sweeping change might be alarming and discouraging. In practice, we believe you will find that

- The conceptual similarities between DevForce 2009 and DevForce 2010 are strong;
- conversion is mostly mechanical;
- based on our experience, conversion of a large application (~100,000 lines) should take about a week; a battery of automated tests is strongly advised;
- DevForce 2010 is more capable, more performant, and easier to use than any prior DevForce version; and
- converting to the new Microsoft technologies (e.g. Entity Framework) will be similarly beneficial.

Your mileage may vary, as always. On to the changes themselves...

Entity Framework

Microsoft overhauled the Entity Framework in version 4. By all accounts it is substantial improvement upon what we believe was a good Version 1.

IdeaBlade DevForce Release Notes

EF v.4 continues the core concepts of EF v.1 ... but the changes are radical and led us to re-design and re-write most of DevForce's interaction with the Entity Framework.

EF 4 supports a POCO ("plain old CLR object") alternative to entities derived from Entity. Thanks to POCO we can move data directly between DevForce entities and the core EF framework without transitioning through EF-generated classes. We anticipate some performance improvement simply by eliminating the transition.

We no longer need to generate EF entity classes – classes that always felt redundant and in the way. We've disabled EF class generation by the Entity Data Model (EDM) designer. (You can restore EF's code generation if you wish.)

EF v.1 hid Foreign Keys (FKs) from you. You knew that every Order carried a CustomerId to link it back to the parent customer. But CustomerId could not appear in your model, nor could you mention it in a query. You were obliged to speak in terms of Association objects. Whatever the theoretical basis, the practical consequence was to make many operations that ought to have been simple very difficult.

EF v.4 concedes that we know about FKs and now supports "FK Associations" in addition to the original, hidden-FK approach that they now call "Independent Associations". EF4 may support the old way (of "Independent Associations"), but we do not: we insist that you construct your model with the FKs exposed. See the "Entity Model Designer" discussion below.

Unfortunately, there is not yet a way to convert an EF v.1 EDM file (the XML representation known as "EDMX") into the new EF v.4 format. The safest bet today is to regenerate your Entity Data Model from the database ... to start as if from scratch ... and then reapply your customizations (e.g., re-naming, inheritance, custom associations, defining queries, etc.).

This conversion – imposed by the Entity Framework, not by DevForce – will likely be the most time-consuming and error-prone aspect of a migration to .NET 4 and DevForce 2010.

Entity Data Model Wizard and Designer

You add an Entity Data Model to a project using the "ADO Entity Data Model" template. The template kicks off the "EDM Wizard" which guides you over several screens through the process of picking a database and deciding how the wizard should generate an Entity Data Model from the database schema.

The EDM Wizard

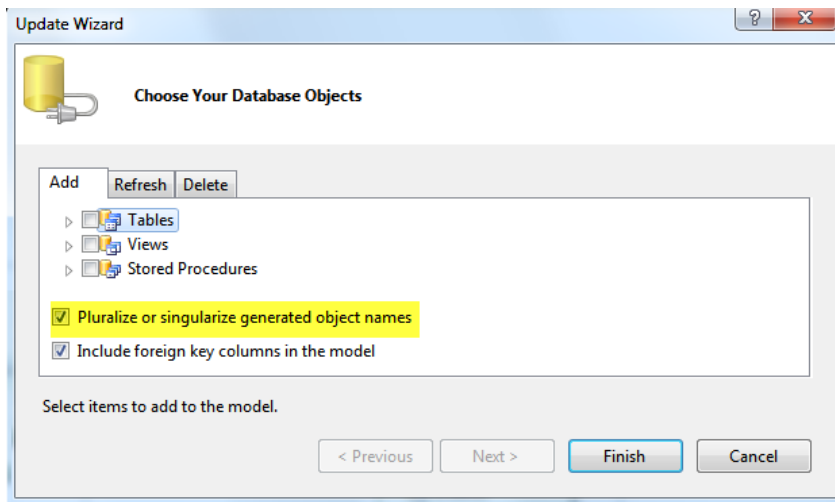
The new EF4 EDM Wizard is much like the original, but there are a few differences that are of particular importance to the DevForce developer.

IdeaBlade DevForce Release Notes

EF 1's EDM Wizard made no attempt to translate table and field names into reasonable entity and entity property names. If the table was called "Orders", EF generated an entity named "Orders"; most folks want singular entity names, "Order" rather than "Orders".

The DevForce Object Mapper gave you the option to apply DevForce "pluralization" rules to translate database names into names appropriate for entities.

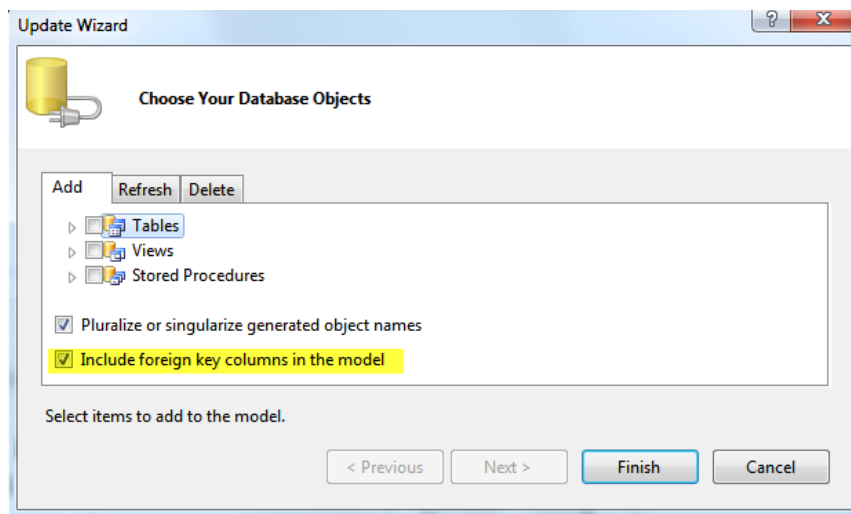
EF 4's EDM Wizard now offers its own pluralization option as seen here:



It's not perfect – neither was ours – but it is good enough that we no longer supply our pluralization tool. When you need more complex and configurable translations from database table and field names, search the web for third-party tools: they're out there.

Your model must relate entities through "FK Associations" as we noted above.

When you generate your EDM from a database using the EF 4 "EDM Wizard", you must check the box that reads **"Include foreign key columns"**, as shown in the following screen shot.



When the wizard finishes, you see a diagram of your model on the EDM design surface, and specialized design windows, the most important of which is the "Model Browser". The design surface and these windows comprise the "ADO.NET Entity Model Designer", a.k.a. the "EDM Designer". It is the default editing tool when you open an EDMX file in Visual Studio.

The EDM Designer

EF supports models that have both “FK Associations” and the original “Independent Associations”. The EDM Designer makes it easy to switch between the two on an individual association basis as explained in this [article](#).

DevForce 2010 only supports “FK Associations”. Generate them that way the first time and don’t switch any of the associations to “Independent Associations”.

DevForce will refuse to generate a model with “Independent Associations”.

EF and DevForce always supported “[Complex Types](#)”; the original EDM Designer, however, did not understand Complex Types and made life so difficult for the developer trying to use them that most people avoided the feature. The EF4 designer manages Complex Types well ... and we expect to see such types become a standard (and welcome) modeling practice.

Many more features await you in the new EF4 EDM Designer. The new EF has introduced some breaking changes, as well. We won’t cover any of that: please see the Microsoft documentation.

In earlier editions of DevForce, your next step would have been to launch the DevForce Object Mapper from the Visual Studio Tools Menus so that you could configure and refine your Entity Data Model for use with DevForce. But there are changes afoot...

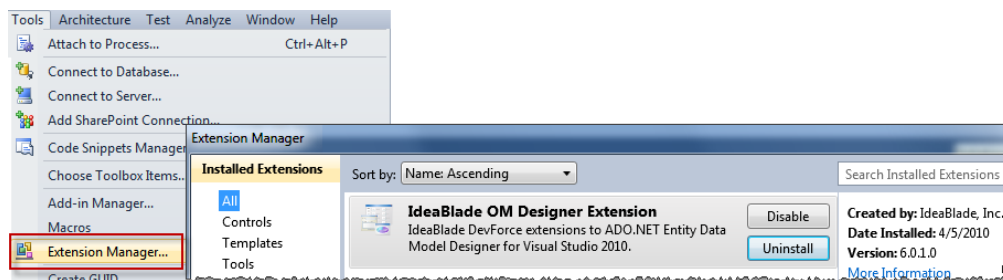
The DevForce EDM Designer Extensions

Look at the Tools menu of a clean Visual Studio 2010 after installing DevForce 2010.

The DevForce Object Mapper is gone!

Most of its capabilities live on ... but they are no longer implemented in a separate Object Mapping tool. They’ve been integrated into the EF 4 EDM Designer by means of the VS 2010 extensibility mechanism.

Open the Extension Manager under the Tools menu and you’ll see the IdeaBlade OM Designer Extension:



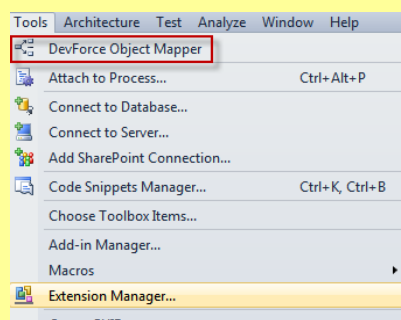
The modal Object Mapper wizard is gone. You need no longer wonder “should I make changes in the EDM Designer or the OM?” With DevForce 2010, you specify your DevForce code generation options within the EDM Designer’s “Model Browser”.

If you install **both** DevForce 2009 and DevForce 2010 in VS 2010, you will see the DF 2009 Object Mapper “AddIn” in the Tools menu.

Please be careful.

The AddIn OM is for EF v.1 models and DF 2009 code generation only. The Designer Extension is for EF v.4 models and DF 2010 applications only.

Mixing them up may *appear* to work ... but it won’t.



IdeaBlade DevForce Release Notes

We won't venture far into the details of DevForce 2010's extensions of the EDM Designer here, as they are covered in separate documentation. Our present purpose is to contrast the former Visual Studio Object Mapper AddIn to the new Designer extensions.

Open an EDMX file. Click any unoccupied spot on the EDM Designer design surface and reveal the property sheet for the model as a whole ("Alt-Enter" usually does it). Click the icon at the top left that arranges the properties in categories.

You will see something like the image on the left. The images on the right are from the former Object Mapper.

DevForce 2010 in the EDM Designer

The screenshot shows the Properties window for a 'NorthwindIBModel' of type 'ConceptualEntityModel'. The 'DevForce Code Generation' category is highlighted in yellow. The properties listed are:

Property	Value
DataSource Key	NorthwindIB
Entity Manager Name	NorthwindEntityManager
Generate Binding Attr	True
Generate Developer C	False
Generate Validation A	True
Generate Verification	True
Injected base type	
Max. Classes Per File	100

Other visible categories include Code Generation, Connection, Database Script Generation, and Schema.

DevForce 2009 Object Mapper

The screenshot shows three settings windows from the DevForce 2009 Object Mapper:

- Project Settings:**
 - Domain Model Project: DomainModel
 - ☒ Create Silverlight Domain Model Project
 - Silverlight Project: (empty)
- Domain Model Settings:**
 - Namespace: DomainModel
 - Entity Manager Name: DomainModelEntityManager
- Save-Related Settings:**
 - ☒ Generate code after save
 - ☒ Create developer partial class files
 - ☐ Update model project's app.config
 - .NET Language: C#
 - ☐ Copy Entity Model artifacts
- Entity Model Settings:**
 - File Name: C:\Users\Ward\Documents\Visual Studio 2010\Projects\
 - Data Source Key Name: Default
 - Namespace: Domain Model
 - Container Name: DomainModel.NorthwindEntities
 - Buttons: Injected Base Types..., Name Pluralizer...
- Default Settings:**
 - Verification Interceptors: Both
 - ☒ Generate Binding Attributes
 - ☒ Generate Validation Attributes
 - Radio buttons: DevForce (selected), .NET

Notice on the property sheet the (faint) "DevForce Code Generation" category, highlighted in yellow. DevForce modeling and code generation specifications appear as properties of this category.

The property values correspond roughly to the details you supplied at either the Project or EDMX model levels in the old Object Mapper. For example, the DataSource key is identical although the default value is no longer "Default". The EntityManager name is the same with a different default. Attribute generation settings are similar.

DevForce 2010 no longer offers to pluralize (EF takes over this function). By default DF2010 will not generate partial classes; you can change this default if you wish. DF 2010 won't generate a Silverlight project ... but it will find one if it exists and create a link to the generated code in your Silverlight project.

IdeaBlade DevForce Release Notes

DevForce always generates code after a save; it infers the .NET language to use from the project's own language choice.

DevForce no longer attempts to maintain your *app.config* file. We'll soon see that configuration is much easier in DevForce 2010. You don't need an *app.config* at all during early development and, when you finally do, you won't feel the need to synchronize automatically with configuration in another project.

In DevForce 2010 you can insert a custom base type between our DevForce Entity root class and your generated class types; such a "base type" is purely of your own devising and does not map to a database table. Use it to create behaviors common to all of the entities in this model.

DF 2010 no longer supports insertion of intermediate abstract classes.

For example, given the EF mapped inheritance hierarchy {Apple – Produce – Entity}, in DF 2009, you could insert an abstract "Fruit" class to go with your "BaseEntity" such that the CLR class hierarchy became {Apple – Fruit – Produce – BaseEntity – Entity}.

That won't work in DF 2010. You can insert "BaseEntity" but not "Fruit" because Entity Framework v.4 rejects it. Entity Framework v.4 insists that all hierarchies consist exclusively of mapped CLR classes; we are allowed to have our own base class.

DevForce 2009 generated all classes into a single file. That file could get big ... sometimes too big for Visual Studio to handle. DevForce 2010, by contrast, can generate the classes into multiple files. By default, it generates a maximum of 100 business classes per file; but you can set this number to the value you desire.

Other DevForce Customizations

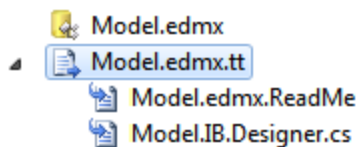
As you move around the EDM Designer, looking at conceptual entities and their "attributes", keep your eye on their associated property sheets. Each has its own "DevForce Code Generation" category where you can specify values that guide DevForce code generation. You'll learn more about them in a separate document.

T4 Code Generation

When you save your Entity Data Model (EDM), your eye may catch a flurry of activity in the Visual Studio Solution Explorer. Perhaps you noticed that ...

.. the DomainModel.ibedmx is gone!

In its place is a node with the same root name as the EDMX and a "tt" file. Expanded it might look like this:



DevForce still records your modeling and code generation specifications in the EDMX, embedded as custom attributes next to the Entity Framework's own EDM material.

The "tt" file is a "[T4¹ Template](#)" used for generating DevForce entity classes. When you save the EDM, the Visual Studio T4 integration automatically runs a custom tool, the "TextTemplatingFileGenerator". You can run it yourself anytime; right-click the "tt" file name and select "Run Custom Tool" from the context menu.

DevForce 2009 also generated entity class files using T4 templates; they were hidden from you. Now, in DevForce 2010, not only can you see the template: you can *customize* it. And you can customize it programmatically using your existing programming skills, without knowing a thing about T4 or having to travel through XML-hell.

¹ T4 stands for "Text Template Transformation Toolkit"

IdeaBlade DevForce Release Notes

You can learn about this from other documentation when you are ready; no need to jump into template customization until you really need to do so. But who can resist an appetizer?

Suppose you want to add your custom “Foo” attribute to every property of every entity. Simple. Subclass the DevForce templating class, `DomainModelTemplate`, and override the property-writing method.

```
public class MyTemplate : DomainModelTemplate {  
    public MyTemplate(  
        Microsoft.VisualStudio.TextTemplating.TextTransformation textTransformation)  
        : base(textTransformation) {}  
  
    protected override void WriteEntityDataPropertyAttributes(EdmPropertyWrapper property)  
    {  
        WriteAttribute("Foo(" + Quote(property.Name) + ")");  
        base.WriteEntityDataPropertyAttributes(property);  
    }  
}
```

This works equally for C# or VB. Check it out!

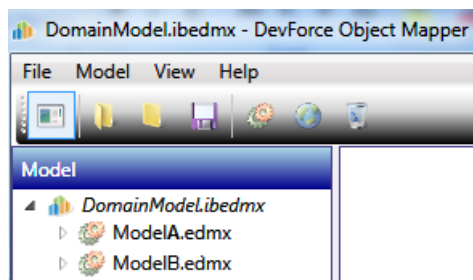
Entity Framework Code Generation Disabled

EF 4 also uses T4 to generate its classes. The EDMX file, like the “tt” file, has a “Custom Tool” property. Before you installed DevForce, Visual Studio would have run the custom tool identified by that property’s value and used it to generate Entity Framework classes. But a DevForce application has no need for Entity Framework classes, so it prevents their generation by “commenting out” the EDMX’s custom tool property.

You can restore EF class generation if you wish – there are still some valid reasons to have EF classes even if DevForce won’t use them. But if you do reactivate the generation of EF classes, please be sure to generate them into their own project. Do not generate them into the project that holds your DevForce entity classes. The EF class names can conflict with the DevForce class names!

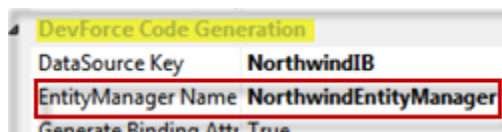
Modeling Multiple Data Sources

The DF 2009 Object Mapper could combine multiple Entity Data Models into a single DevForce entity model:



You can still do that in DF 2010, but the mechanics are different.

In DF2010, you specify an EntityManager class name in the property sheet for your Entity Data Model:



All Entity Data Models in the same project that have the same EntityManager name will have their code generated into the same EntityManager class. That single class will contain the query properties (e.g., “entityManager.Customers” and “entityManager.Orders”) to retrieve the entities originating in all of the different Entity Data Models.

Modeling Web and WCF Services

DevForce 2010 no longer generates a model from a Web or WCF service. We discovered that few used this feature in DevForce 2009.

Web Services used to be difficult and tedious to write. The tooling has since improved and the advent of [OData](#) promises to become a new standard for exposing and consuming data to and from external parties.

We also introduced DevForce POCO objects in DevForce 2009, making it much easier to build a distributed model for any data source, including Web, WCF SOAP, WCF REST and OData services. We've extended and improved POCO support in DF 2010. Expect more framework facilities and guidance from us for modeling non-relational data sources in future DevForce 2010 releases.

Stored Procedures

In DevForce 2009 you could invoke a query backed by the EF representation of a stored procedure. We did not support Create, Update, and Delete operations backed by stored procedures.

DevForce 2010 has no support for any kind of stored procedure. We expect to repair these deficiencies by Fall 2010. Meanwhile, you can write Server Methods that talk directly to EF, or use ADO.Net, to invoke the stored procedures; you then can pour their results into entities that you return to the client.

Entities and the Entity Manager

An entity in DevForce 2010 feels and operates much like an entity in DevForce 2009. The Entity Manager that holds entities and “manages” them feels much the same as well.

But much has changed under-the-hood. DevForce 2010 has been re-engineered to reflect another year of DevForce experience in production environments, and to leverage improvements and new facilities released by Microsoft. It has been specifically re-designed

- (a) to take advantage of new capabilities in .NET 4 and Entity Framework v.4; and
- (b) to align with parallel concepts in WCF RIA Services.

Entities

Entity validation is the major change that will touch many DevForce developers. Validation is covered in a separate section below.

Most developers won't notice the other changes to entities. You're an advanced developer if you run into these:

- The protected members `GetRawValue` and `SetRawValue`, which could be used to drill into entity properties using metadata, have been renamed to `GetValueRaw` and `SetValueRaw`, respectively.
- A custom field initializer that you added to your entity (e.g., `private int foo = 42;`) may not always be set for every new instance. Our client-side “entity materialization” logic can by-pass field initializations. To ensure that “foo” is initialized, either set it lazily or mark “foo” as a WCF [DataMember](#).
- Open a generated entity class, scroll to its “EntityProperty Definitions” region, and you might notice a few changes. The DevForce 2009 static fields have been pushed down into a nested “PropertyMetaData” class inside each entity.

DF 2009	<pre>public static readonly IbEm.DataEntityProperty<Customer, Guid> IdEntityProperty = new IbEm.DataEntityProperty<Customer, Guid>("Id", false, true, IbEm.ConcurrencyStrategy.None, false, IbEm.VerificationSetterOptions.Both);</pre>
------------	--

```
DF 2010 public partial class PropertyMetadata {  
    public static readonly IbEm.DataEntityProperty<Customer, System.Guid>  
        Id =  
        new IbEm.DataEntityProperty<Customer, System.Guid>(  
            "Id", false, true, IbEm.ConcurrencyStrategy.None,  
            false);  
}
```

In DF2010, we encapsulate the metadata inside an inner class rather than have a bunch of property definition fields hanging off the entity itself. It's much cleaner.

The change is reflected in the property definition field name. This will break your code wherever you used the property definition object to refer to the Customer's Id property in a strongly typed way.

Where you once wrote "`Customer.IdEntityProperty`", now you will write "`Customer.PropertyMetadata.Id`".

The compiler will help you convert by complaining bitterly!

- You can control whether a navigation property, such as `Order.OrderDetails`, "thinks" its data – the related entities – are already in cache. There's an "IsLoaded" flag for each navigation property of each entity instance. If you set it to true, DevForce will only look to the local cache when it exercises that particular navigation property on that particular entity.

It's a new feature and a subtle behavioral change that *might* break your code ... although few of you will be affected and most of you (we certainly hope!) will be grateful.

When you create a new Order, it almost certainly does not have OrderDetails in the database. You're probably going to create those too. If you call "OrderDetails" on a new Order, it should return an empty collection until you add line items. And if you add line items, it follows that those items must already be in the cache. You should never have to go to the database to get the OrderDetails of a new Order.

Unfortunately, DevForce 2009 didn't know this. It always made at least one trip to the database for OrderDetails even though that trip never returned a result.

In DevForce 2010, we set the "IsLoaded" flag to true for all navigation properties of a new entity. DevForce will think the related entities, if they exist at all, are already in cache; it won't make a trip to the server. Of course you can set the flag to *false* if you like. And you may find this flag useful in other contexts.

The "IsLoaded" property is a new member of the `EntityReferenceBase` class. You get an instance of `EntityReferenceBase` by calling the "GetEntityReference" method of the `NavigationEntityProperty` inside the entity's navigation property.

Confused? Here's a line of code that may help:

```
Order  
    .PropertyMetadata.OrderDetails // NavigationEntityProperty for OrderDetails.  
    .GetEntityReference(anOrder)    // EntityReferenceBase for a particular order.  
    .IsLoaded = true;               // Insist that anOrder's details are in cache.
```

Entity Manager

The `EntityManager` class API changed significantly, especially in its treatment of asynchronous operations such as async query.

Asynchronous Queries

We learned a good deal over the last year about what it's actually like to work with asynchronous queries. Few developers have written asynchronous code in the past. Windows and ASP developers tend to count on a fast,

reliable connection to the server; the server is in “line of sight”, as they say. You might have a little asynchronous programming experience if you write AJAX.

In Silverlight, you cannot talk to the server synchronously. Only asynchronous server calls are allowed. So you learn and you adapt. You get used to executing queries that return immediately ... without results. The results ... or exception ... show up later ... when the EntityManager invokes your callback. You figure out how to “amuse” the user until the data finally arrives.

Saving data poses a similar challenge. Your “request to save” returns immediately; you won’t know if the data were actually valid and saved until you hear back from the server. What should the user be allowed to do until we know what the server said? It’s tricky.

On the positive side of the ledger, your application is more responsive now that you aren’t freezing the UI while waiting for the server. Two-tier applications stand to benefit as well. UI-stalling can be a problem even with a good connection, especially when contending for resources with other users. If only asynchronous programming were easier!

We think we’ve made it easier in DevForce 2010. Let’s explain with a simple query lesson.

A DevForce Synchronous Query

Imagine a WPF or Silverlight UI screen with a DataGrid that shows customers, wired up as follows:

```
// List of customers to display
ObservableCollection<Customer> CustomerList =
    new ObservableCollection<Customer>();

_DataGrid.ItemsSource = CustomerList;
```

Here’s our query to fetch all customers:

```
var query = mgr.Customers; // returns an EntityQuery<Customer>
```

You could execute the query synchronously by writing:

```
var results = query.ToList(); // assume it always succeeds
results.ForEach(CustomerList.Add);
```

The “ToList()” LINQ extension method tells DevForce to process the query and return the results ... right now. The method blocks until the Customer entities return. The UI will freeze while you wait. When the server returns, the thread continues, the next line executes, and the customers are added to the CustomerList. The DataGrid listens to the items being added to the ObservableCollection<Customer> and updates itself. The user sees the list and the keyboard unlocks so they can party on the grid.

The “ForEach” extension method in the code sample spares you the trouble of writing a “for each” loop to add each customer in the results to the CustomerList. It’s a nice idiom and we include it in the IdeaBlade.Linq library. Remember to include IdeaBlade.Linq in your “using” statements.

A DevForce 2009 Asynchronous Query

In Silverlight, “query.ToList()” throws an exception because all server operations must be asynchronous in Silverlight. You need to invoke the query asynchronously, perhaps like this.

```
query.ExecuteAsync(
    x => x.Result.ForEach(CustomerList.Add), // Query never fails, right?
    null
);
```

This time the “ExecuteAsync” call returns immediately. The UI remains alive and, if you were binding the CustomerList in a DataGrid, the user could wander around in the empty grid.

IdeaBlade DevForce Release Notes

When the server returns, DevForce invokes your callback ... which is the lambda expression in our example. “x” in the callback has a *Result* property returning *IEnumerable<Customer>*. The balance of the lambda callback is the same as in our synchronous query example.

Suddenly ... perhaps to the user’s surprise ...the grid magically fills with the retrieved customers.

This syntax is almost the same in both DevForce 2009 and DevForce 2010. Almost ...

In DevForce 2009:

- *ExecuteAsync* returns void;
- the “x” in the lambda is **EntityFetchedEventArgs<Customer>** ; and
- the x exposes a “Result” (singular) property for query results.

A DevForce 2010 Asynchronous Query

Here’s the same query again in DevForce 2010:

```
query.ExecuteAsync(  
    x => x.Results.ForEach(CustomerList.Add) // Query never fails, right?  
);
```

In DevForce 2010:

- the “x” in the lambda is an **EntityQueryOperation<Customer>** (... to be discussed);
- the x exposes a “Results” (plural) property for query results ;
- the last parameter (ObjectState) is optional and typically omitted; and
- *ExecuteAsync* returns an **EntityQueryOperation<Customer>** ... the same one passed to the callback.

The presence of the return value can’t harm existing code. The change to the callback parameter could break your code and the change to the name of the property returning customer (Result → Results) almost certainly will. Your async query code won’t compile until you make some repairs.

What’s up with that?

We renamed the result property (Result → Results) to be consistent with a similar property in WCF RIA Service. That alone is not a good enough reason to break your code. But since we were breaking anyway to provide new usability features ... we aligned the property names while we were at it.

Our real motivation came from the realization that not everyone likes callbacks. Microsoft tooling isn’t fond of them either. Many people prefer an event-based syntax such as this:

```
var op = query.ExecuteAsync();  
op.Completed +=  
    (sender, args) => args.Results.ForEach(CustomerList.Add);
```

Some folks don’t like lambdas either. We could write it this way instead:

```
var op = query.ExecuteAsync();  
op.Completed += GotCustomers;  
...  
private void GotCustomers(  
    object sender,  
    EntityQueriedEventArgs<Customer> args)  
{  
    args.Results.ForEach(CustomerList.Add);  
}
```

Notice that *EntityQueriedEventArgs* now returns a *Results* (plural) property.

The “op” value returned from `ExecuteAsync` is a strongly-typed, `EntityQueryOperation<Customer>`. The operation object encapsulates aspects of the query operation such as

- the results of the query (“Results”);
- whether there were errors processing the query (and what they are);
- whether the query resulted in a trip to the server (“WasFetched”) [New!];
- a “Cancel” option and a “Cancelled” property indicating if the operation was cancelled;
- what entities were added to the cache or were changed as a result of the query (“ChangedEntities”); and
- which `QueryStrategy` was actually used (which may differ from the one that was requested in the query) – reported by the `ResolvedQueryStrategy` property. [New!]

The Other Asynchronous Operations

The other asynchronous operations – Connect, Login, Save, Server Method Invocation – all follow the pattern established by the asynchronous query. Each returns its own flavor of “operation” object, all of which inherit from `BaseOperation`.

Best Practices for an Asynchronous User Experience

Good patterns and practices for asynchronous UI implementations are beginning to emerge. IdeaBlade is one source of information on these patterns.

One important rule: ***Encapsulate all references to the `EntityManager` in a helper class.***

Some call it a “Data Service Class”; some call it a “Repository.” Whatever you call it, do it. It simplifies the UI. Your UI code should delegate the asynchronous machinery to this helper class where you hide the details.

Congratulations if you already follow this practice because your conversion to the DevForce 2010 `EntityManager` will be much easier; you’ll only have to change the helper ... not patch up queries and saves scattered throughout the UI.

Miscellaneous Changes

EntityQueriedEvent

The DevForce 2009 `EntityManager.EntityFetched` event is now the `EntityManager.Queried` event. The `EntityFetchedEventArgs` are now the `EntityQueriedEventArgs`.

This is a breaking change that goes beyond renaming. `EntityFetched` used to be raised only if the query resulted in a trip to the server. `EntityQueried` is always raised after a successful query. You’ll have to look at the related `EntityQueryOperation.WasFetched` to determine if there was a server visit.

FindEntities

`EntityManager.FindEntities` now properly returns an `IEnumerable` rather than `IEnumerable<Object>`, as it did (and should not have) in DevForce 2009. It’s a breaking change. The repair is easy; you can use the LINQ “`Cast<T>()`” extension method to cast an `IEnumerable` to an `IEnumerable<Object>` if you really need the latter.

EntityCacheState.Restore

`EntityCacheState` had a “Load” method in DevForce 2009; in DevForce 2010, the `CacheStateManager` and the `SerializationFns` have “Restore” methods to do the same thing. We renamed “Load” to “Restore” to achieve consistency across the three components.

EntityChanging / EntityChanged

The `EntityManager` has two new events, `EntityChanging` and `EntityChanged`, which it raises when **any** entity in its cache is changed. Both events indicate the entity involved and the action involved (e.g., added).

EntityGroup

An `EntityGroup` maintains information about a single type of entity. An `EntityGroup` belongs to an `EntityManager`; there is one of them for each type of entity tracked by its `EntityManager`.

The statement ...

```
manager.GetEntityGroup(typeof(Customer));
```

... would return the `Customer EntityGroup` from the `EntityManager` instance.

The `EntityGroup` also has `EntityChanging` and `EntityChanged` events which are specific to the entities in that `EntityGroup`.

Other new `EntityGroup` properties enable or suppress entity behaviors such as the following:

- Notification of listeners when an entity of this type is modified (`ChangeNotificationEnabled`).
- Tracking of changes to entities of this type (`ChangeTrackingEnabled`).
- Invocation of the DevForce property interceptors for entities of this type (`PropertyInterceptionEnabled`).

Cascading Delete

You already know that “Delete” on the client means “schedule for delete”; the data in the backing store is deleted when you save.

When you delete a parent entity, what happens to the child entities in the entity cache? For example, if you delete an `Order`, what happens to the line item entities?

If your model specifies cascading delete ... of if cascading delete is implied by database constraints on the foreign key (FKs) of the child table ... DevForce will schedule the children (the dependent entities) for deletion at the same time that it “deletes” the parent (the “principal” entity).

In this scenario, we don’t touch the FKs in the children, the `OrderId` in the child line item entities of our example. There is no reason to break the bond between parent and child. They remain connected in their “delete” state.

If we should **not** cascade parent deletes to their children, we try to sever the connection between parent and child. We’ll set the FK of the child to null and the child is marked “Modified” ... if the FK is nullable.

If the child FK (`OrderId` in our example) is not nullable, we are not sure what to do so we do nothing. You’ll have to step in, if you want to change the FK. You might want to handle this by listening to the `EntityChanged` event on the parent `EntityGroup` (as just described) and responding when the change is a “delete.”

UserBase – the new default IPrincipal

DevForce 2009 introduced support for ASP Security. `UserBase` is a class representing the authenticated user in ASP Security; it implements `IIdentity` and `IPrincipal`. We implemented a version of `UserBase` for DevForce developers.

In DevForce 2010 we’ve moved it to the `EntityModel` assembly where it is more readily accessible to server and client code.

Setting the Concurrency Value

Usually you don’t touch the value of the optimistic concurrency property. But you could in DF 2009 via

```
IConcurrencyStrategy.SetNewConcurrencyValue(  
    EntityProperty property, Object entity).
```

The signature has changed in DevForce 2010 to

```
IConcurrencyStrategy.SetNewConcurrencyValue(  
    DataEntityProperty property, Object entity)
```

The first parameter is now `DataEntityProperty` instead of `EntityProperty`.

Validation

Verifier Options Type

The most significant change in the Verification subsystem was the addition of a new *VerifierOptions* type. An instance of the type is available as a property on the *VerifierEngine* class via *DefaultVerifierOptions*; and also on the *Verifier*, *VerifierArgs*, and *VerifierResults* classes as *VerifierOptions*.

The *ExecutionMode* property was moved into this new class. It also contains a property, *VerifierErrorContinuationMode*, which is the older *VerifierOnErrorMode* renamed. (All properties returning *OnErrorMode* have been renamed to *ErrorContinuationMode*.)

All *VerifierOptions* properties can be inherited from a parent class and, by default, do so. The system defaults for *VerifierOptions* are now as follows:

Property	Value
<i>ShouldExitOnBeforeSetError</i>	false
<i>ErrorNotificationMode</i>	<i>VerifierErrorNotificationMode.Notify</i>
<i>TreatWarningsAsErrors</i>	false
<i>ExecutionModes</i>	<i>VerifierExecutionModes.InstanceAndOnAfterSetTriggers</i>
<i>ErrorContinuationMode</i>	<i>VerifierErrorContinuationMode.Continue</i>

The breaking change here is that the behavior of the old *PropertyVerifiers* was the equivalent of what could now be obtained by setting

```
ErrorNotificationMode = VerifierErrorNotificationMode.ThrowException
```

and

```
ExecutionMode = VerifierExecutionModes.InstanceAndOnBeforeSetTriggers.
```

All properties on *VerifierOptions*, if of type bool?, may be set to null. If enums, they may be set to an enum value of 'Inherit' to indicate inheritance.

VerifierEngine is parent to any *Verifiers* contained within it.

Verifier is parent to any *VerifierResults* resulting from its execution.

This means that if *VerifierEngine.DefaultVerifierOptions* is changed, then by default, every *Verifier* and *VerifierResult* will inherit this change.

IdeaBlade.Core.ComponentModel.INotifyDataErrorInfo

IdeaBlade.Core.ComponentModel.INotifyDataErrorInfo has been implemented in the WPF version of DF 2010 with the same semantics as the similarly named class in the Silverlight CLR version of *System.ComponentModel*.

This interface makes it possible for you to configure your entities to collect validation errors rather than throw exceptions (or to do both, if you wish).

Entity Class Validation Properties

The DevForce base *Entity* class has two new virtual methods:

IdeaBlade DevForce Release Notes

```
protected internal virtual bool ValidatePropertyBeforeSet(  
    EntityProperty property, object newValue)  
  
protected internal virtual void ValidatePropertyAfterSet(  
    EntityProperty property, object newValue)
```

When something sets a DevForce entity property, the property setter pipeline calls these methods. You can override them in specific business classes (Customer) or in your own base entity class from which all of your entities derive.² You almost always want to call the base implementations of these methods in your overrides. The base “Before” method calls the `VerifierEngine.ExecuteBeforeSet` for the specified property; the base “After” method calls the `VerifierEngine.ExecuteAfterSet`. However, you could call the base implementation selectively based on conditions at the time; if you were not to call the base implementation, but simply to return a true, validation would be skipped altogether.

If your “Before” method returns *false*, DevForce will not set the entity property’s backing store. It won’t throw an exception either; rather, the set will be ignored silently.

Miscellaneous Changes

Verification is now turned off during `NullEntity` construction.

Name changes:

DevForce 2009	DevForce 2010
<code>VerifierResult.Description</code>	<code>VerifierResult.Message</code>
<code>VerifierResultCollection.AreOk</code>	<code>VerifierResultCollection.Ok</code>

In the `VerifierResultException` class:

- Renamed `VerifierResultCollection` property to `VerifierResults`
- Removed the `VerifierResult` property. (The old `VerifierResult` is the same as the new `verifierResults.Errors.FirstOrDefault()`.)

All methods that take `PropertyDescriptor`s as arguments have been removed.

`EntityProperty.VerificationSetterOptions` has been removed and the `VerificationSetterOptions` class has been renamed `VerifierSetterOptions`. This API may be removed in a later release.

There is now a `VerifierSetterOptions` property available on the `EntityMemberMetadata` class that is available via `anEntityProperty.MemberMetadata.VerifierSetterOptions`. This property can be set directly or via attributes on the desired entity property or metadata type “buddy” property.

Preset and *Postset* suffixes have been replaced by “BeforeSet” and “AfterSet” to adopt the convention used by property interceptors.

All *GetVerifiers* overloads return an `IList<Verifier>` instead of a `VerifierCollection`.

² See the previous section of this document, “The DevForce EDM Designer Extensions” regarding injection of your own base class. You will also find more information in the topic document “010_DevForce Entity Data Model Designer Enhancements”, in the section “Entity-Model-Level DevForce-Specific Properties”.

The *IdeaBlade.Core.ComponentModel.DataAnnotations.MetadataTypeAttribute* has been implemented in the Silverlight version of DF 2010 with the same semantics as the similarly named class in the desktop CLR version of *System.ComponentModel.DataAnnotations*.

Business Object Server

The DevForce Business Object Server (BOS) is the “middle tier” component that mediates between distributed clients and host services -- chief among these being the Entity Framework and the Database Server. The BOS is a stateless, load-balanceable hub for data access, security, validation, and remote service invocations.

The BOS is manifested on the host as the *EntityServer* assembly.

Much has evolved under the hood and the configuration and deployment of the BOS has improved remarkably, meriting its own section. But the BOS API's have not changed much. Here are the prominent changes:

- A single BOS deployment can support both Silverlight and non-Silverlight clients. You no longer have to deploy the BOS twice in two separate directories (one for Silverlight, the other non-Silverlight) as you were obliged to do in DevForce 2009.

You still can mandate one or the other exclusively in configuration if you wish; and the BOS-for-Silverlight-only license is appropriately restricted.

- In DevForce 2009 you implemented *IEntityServerFetching* on the server to intercept, modify, and potentially reject queries before they ran. You implemented *IEntityServerFetched* to perform arbitrary tasks on the server after the query completed successfully.

These functions and opportunities have been consolidated in the *EntityServerQueryInterceptor* in DevForce 2010. The interfaces are no more, and code that implements them will not compile. You'll have to move that logic into your sub-class of *EntityServerQueryInterceptor*.

The *EntityServerQueryInterceptor* implements interception points over a significant portion of the query pipeline. It includes several “template” methods that you override to introduce your custom interception logic.

You'll do so for the same reasons that you wrote to those interfaces in the past; but we think you'll find it easier in DevForce 2010 to arrange for your custom behavior to engage at the proper moments. Here are some of the members:

- *AuthorizeQuery*
- *FilterQuery*
- *ExecuteQuery*
- *AuthorizeQueryResult*
- *DefaultAuthorization*
- *ClientCanQuery*
- *ShouldAuthorizeQueryResult*
- *OnError*

The details are covered in the Business Object Persistence and Security topic documents.

- DevForce 2009 featured interfaces for intercepting server-side save operations, the *IEntityServerSaving* and *IEntityServerSaved*. You could write logic to inspect the entities before they are saved, remove items, add items, or possibly reject the save.

As with their query-oriented cousins, these interface have been removed and we expect you to transfer your interception logic to your custom sub-class of *EntityServersSaveInterceptor*. Some of the members:

- *AuthorizeSave*
- *ValidateSave*
- *ExecuteSave*
- *AuthorizeSaveResult*
- *DefaultAuthorization*
- *ClientCanSave*
- *OnError*

The default `ValidateSave` method performs “instance verification” (aka, object validation) on all entities before save. If any verification fails, DevForce throws an `EntityServerException` with a `PersistenceFailure` type of ‘Validation’.

Previously, server validation was off by default; you had to engage it in your `IEntityServerSaving` implementation. Your application may behave differently now that server-side validation is turned on by default.

Details are covered in separate documentation.

- Renamed `EntitySaveAdapter` to `EntityServerPocoSaveAdapter` (will be called from within `EntityServerSaveInterceptor`)
- The `AdoHelper` class has been dropped. You have easy access to connection strings on the server and can spin up your own raw ADO code as you wish. Opening a server-side EntityFramework context is no great chore either – although you might have to generate the EF classes from the Entity Data Model manually as we turn EF code generation off by default in DevForce 2010.

Configuration and Deployment

Multiple BOS directories, multiple *app.configs*, *web.configs*, *svc* files, WCF custom bindings, dev/test/stage/prod environments, angle-bracket hell ... configuring and deploying an n-tier application can be a nightmare. You shouldn’t have to mess with this junk just to get started. We agree.

Configuration is vastly simplified in DevForce 2010, especially for early development when all you need is a 2-tier desktop application or Visual Studio’s built-in “Cassini” web server for Silverlight. Here are some of the changes we’ve made to make your life easier:

- *EdmKeys* are optional; DevForce infers them from the EF connection strings maintained by the EntityFramework EDM Wizard in the entity model project.
- *EdmKeys* no longer have their own “probe assembly” or “options” specifications
- *RdbKeys* or *WsKeys* (long obsolete) can no longer be specified.
- DevForce 2010 now uses the Managed Extensibility Framework (MEF) to discover and instantiate your custom classes (e.g., your implementations of `ILoginManager` and `EntityServerQueryInterceptor`). DevForce tells MEF to probe assemblies in the *bin* directory (desktop/asp) or the *.xap* (Silverlight) by default.
- A happy consequence of these changes: you usually do not need an *app.config* in a Silverlight start-up project!
- If you do need an *app.config* – perhaps to add additional, top-level assembly probing paths or to specify the remote server (when that time comes) – you can embed it in the start-up assembly as you do today or mark it as “Content”. Marking it as “Content” makes it a loose file in the *.xap* ... accessible for modification without recompilation.
- DevForce 2010 ships with a new validating schema (*xsd*) for `IdeaBlade.configuration` which means you get validation and IntelliSense when you edit IdeaBlade sections of the config files in Visual Studio.
- You no longer need *svc* files for the `EntityService` and `EntityServer` services if you provide a *Global.asax* with this one line in the `Application_Start` method:

```
System.Web.Hosting.HostingEnvironment.RegisterVirtualPathProvider(  
    new IdeaBlade.EntityModel.web.ServiceVirtualPathProvider());
```

DevForce generates them on the fly.

- If you choose to specify explicit *.svc* files, you’ll have to upgrade them as follows (breaking change).

EntityServer.svc

```
2009 <@ServiceHost language="C#"
      Service="IdeaBlade.EntityModel.Server.EntityServer"
      Factory="IdeaBlade.EntityModel.Server.ServerHostFactory"
%>
```

IdeaBlade DevForce Release Notes

2010	<pre><%@ServiceHost language="C#" Service="IdeaBlade.EntityModel.Server.EntityServer" Factory="IdeaBlade.EntityModel.Server.EntityServerHostFactory" %></pre>
EntityService.svc	
2009	<pre><%@ServiceHost language="C#" Service="IdeaBlade.EntityModel.Server.RemoteEntityService" %></pre>
2010	<pre><%@ServiceHost language="C#" Service="IdeaBlade.EntityModel.Server.RemoteEntityService" Factory="IdeaBlade.EntityModel.Server.EntityServiceHostFactory" %></pre>

- The default *web.config* from the new DevForce “Silverlight Application” template is “only” 32 lines. The `<serviceModel/>` section is no longer required. You likely want to revise your *web.config* using this one as a model.
- Use sub-typed *ServiceHostEvents* and *ServiceProxyEvents* to customize WCF configuration in code.
- We removed the *DiscoverableMemberAttribute*.
- We removed the *DiscoverableTypeMode.ServerService* enum value which is no longer needed.

Enjoy your new DevForce 2010!