

Cabana Installation & Release Notes



Welcome to the Cabana Source Code!

These notes are divided into two sections:

- [Installation Notes](#)
- [Release Information](#)

Cabana requires DevForce

The Cabana application will neither compile nor run without DevForce installed. Fortunately, DevForce is easy to get.

You can download and install a DevForce edition such as DevForce Express from our website for *free* at www.ideablade.com/downloads.html.

Installation of Cabana is nearly painless but there are a few places that can give you momentary trouble. For example, you may need to tweak the connection string to find your copy of the IdeaBlade Tutorial database.

Please read the [Installation Notes](#) carefully !

Please read also the document “CabanaStructure” that describes the files and solution structures. You can get the latest version from our website at <http://www.ideablade.com/Cabana/CabanaStructure.pdf>

Both documents are absolute “musts”.

What Cabana Is and Is Not

Cabana is supposed to be ... well, fun. It's intended to show you some of the things you can do as a DevForce enterprise application developer.

Cabana is not intended to be a sophisticated application

Cabana demonstrates intermediate and advanced techniques in the context of a Composite UI Application Block (CAB) application. We want the application itself to be as simple as possible without obscuring the key points. Our customers build large, sophisticated applications that draw on these techniques.

Cabana source code is not for beginning DevForce application developers

Cabana is a challenging place to start if you have just installed DevForce. Please try the DevForce tutorials first as they establish a foundation for the ideas and suggestions here. Please feel free to explore Cabana for interesting details and for clues about solutions to advanced issues – just keep in mind that it is a tutorial, not a real application.

Install Cabana Code (for DF v.3.5.3.1)

Installation Basics

Cabana installation changes in version 2.3.0.0.

The basic instructions have been moved from this document to the “Getting Started Guide”, available in the download package and from the Cabana website (www.ideablade.com/Cabana).

Pick the Right Download File

There are two versions at this time.

CabanaDotNetCS.zip Dedicated to applications based on the DotNet UI controls only. DevForce Express users should use this package.

CabanaDevExCS.zip Dedicated to applications based on the Developer Express UI controls.

The Cabana in this package is rooted in the Developer Express UI controls but includes a mix of DotNet and DevEx views.

You must have DevForce Professional or DevForce Enterprise editions to use this package.

Unzip download package into the directory of your choice.

Open the “Getting Started Guide” and take it from there.

CAB Source

You don't need the CAB or SCSF source code to use Cabana. You don't need the Guidance Automation Toolkit or Guidance Automation Package either because Cabana doesn't use SCSF recipes¹.

It is often useful to see how CAB code actually works. You won't be able to debug into CAB or SCSF (sad to say) but you can open up the solutions and read the code.

Get them from <http://www.codeplex.com/smartclient>.

Troubleshooting the Install

The “Getting Started Guide” has troubleshooting suggestions. The ones that follow here may also be of value.

Here we cover a few common “gotchas”

1. DevForce version incompatibility
2. Solution seems not to build or run
3. Visual Studio complains during build (phantom messages)

¹ Cabana itself began as the output of the 2006 SCSF application generation recipe. Subsequent refactorings make that inheritance hard to detect. We don't recommend using the original SCSF recipes (they are demonstrations, unsuitable for production code). We would recommend developing your own recipes but that is so difficult that we have abandoned them in favor of Visual Studio templates and wizards – inferior to recipes but not bad and definitely easier to build, maintain, and install.

Cabana Installation & Release Notes

4. Can't find IdeaBladeTutorial database / wrong IdeaBladeTutorial database
5. Can't get the Visual Studio Toolbox to work properly
6. Test projects don't compile or run; missing NUnit.

DevForce Version Compatibility

Your version of DevForce (DF) may not be the same as the version we used to build Cabana (currently 3.5.3.1). If your DF version differs from the Cabana version, you may be ok. If Cabana won't build, you may only to fire up the Object Mapper and regenerate the DataRow classes.

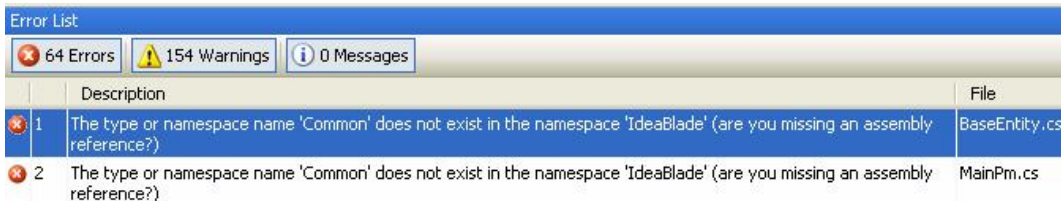
If you are using a different version of DevForce (not recommended), you may need to run the Object Mapper and regenerate entities in the business object project, `Cabana.Model` before you build `Cabanaxxx`.

However, we strongly recommend that you use the latest version of DevForce and Cabana.

Solution Seems Not To Build or Run

There are two solutions: `CabanaLib` and `Cabanaxxx`.

You must build `CabanaLib` first in order to build the projects needed by the application. These projects output their assemblies to the "Lib" directory. If you neglect to build it and build the other solution, you'll get a boat load of errors such as:



	Description	File
1	The type or namespace name 'Common' does not exist in the namespace 'IdeaBlade' (are you missing an assembly reference?)	BaseEntity.cs
2	The type or namespace name 'Common' does not exist in the namespace 'IdeaBlade' (are you missing an assembly reference?)	MainPm.cs

You may be tempted to run `CabanaLib` after you've built it. It does not run.

There is only one solution that runs: `Cabanaxxx` (as in `CabanaDotNetCS.sln`).

You must proceed in the following order:

- ⇒ Build `CabanaLib` *first*.
- ⇒ Build `Cabanaxxx` *second*.
- ⇒ The startup project is called `CabanaShell`; this should start when you press F5

Projects in the ExternalSource folder are unavailable

There is an "ExternalSource" folder in the application solution. All of its projects are unloaded and, thus, "unavailable." That's how it is supposed to be; the file "ExternalSource.ReadMe.txt" in the folder explains why.

Visual Studio complains during build (phantom messages)

There is some misbehavior when you build either Cabana solution that we haven't learned how to cure.

The behavior is alarming but harmless; VS is complaining about nothing.

You can be assured of this by the fact that *the build succeeds* despite the noise.

"Missing components" Warnings

You may see a flurry of warning messages after the build.

Cabana Installation & Release Notes

Of greatest apparent concern are warnings that read as follows:

`warning The referenced component 'Scsf.Infrastructure.Interface' could not be found`

This is an illusion, an artifact of the build process. In fact, all components were found.

Can't Find IdeaBladeTutorial Database

Cabana gets its data from the IdeaBladeTutorial database that ships with DevForce – the one we use for our DevForce tutorials.

You will need the latest version of this database. You should confirm that you have the correct IdeaBladeTutorial database installed and that the connection string(s) shipped with Cabana actually work in your SQL Server.

Use the Appropriate Tutorial Database

The latest Cabana depends upon the IdeaBladeTutorial database that shipped with DevForce version 3.5.2.2. If you installed this version – or a later version – you are in good shape ...

... unless you neglected to replace a previous version of the database.

When DevForce installs, it checks to see if you have already have an IdeaBladeTutorial database in your SQL Server. You won't if you are a first time DevForce installer. You will have one if you've installed DevForce before.

The DevForce install does not remove a pre-existing IdeaBladeTutorial database.

This is intentional. You may have made changes that you want to keep.

But this could mean that you have a tutorial database that is missing tables or data that Cabana requires. Such is certainly the case if your tutorial database came from a DevForce version prior to 3.5.2.2.

You can contact support to get a copy of the IdeaBladeTutorial database.

You will have to do the following:

- ⇒ Launch SQL Server Manager
- ⇒ Detach IdeaBladeTutorial
- ⇒ Locate the database files² and squirrel them away (optional).
- ⇒ Run the tutorial database installer from “Start | IdeaBlade DevForce | Tools | Database Installer”.
- ⇒ Confirm that the database was installed³.

Connection String

There are connection string specifications in several places, all of them reading as follows:

```
Provider=SQLOLEDB.1;Integrated Security=SSPI;  
Persist Security Info=False;Initial Catalog=IdeaBladeTutorial;  
Data Source=localhost
```

² They are usually the two files named IdeaBladeTutorial_Data.MDF and IdeaBladeTutorial_log.LDF located in C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data

³ This will work fine if it worked when you installed DevForce before. If you had trouble before (e.g., you were using SQL Server Express), you'll need to take the same remedial steps that worked for you before. IdeaBlade customer support can help you.

Cabana Installation & Release Notes

You won't be able to connect (and login) if any aspect of this string is wrong, such as:

- You're not using MS SQL Server (2000 or 2005)
- You have a named instance of MS SQL Server (rather than a default instance)
- You can't use the SQL OLEDB provider
- You can't use integrated security (Windows Authentication)
- The database isn't named "IdeaBladeTutorial"
- You can't refer to the database server as "localhost" (e.g., named pipes not enabled for your server).

Launch the Object Mapper while examining the [Model](#) project to discover if this is a problem for you. If the Object Mapper can't find the database, you'll have to find the connection string that does. The DevForce Installation Guide and Developer Guide offer some useful discovery tips.

Once you have a working connection string, you'll have to put that string in the following places:

- The Object Mapping file for the Model project (via the Object Mapper)
- The Object Mapping file for the Server project (via the Object Mapper)
- The two RdbKeys in the AppHelper.IdeaBlade.ibconfig; remember to rebuild - not just build – this project after you make these changes.

The solution includes an application, `DbConnectionCleaner`, that will guide you through the process of setting up a working connection string.

- ⇒ Launch Visual Studio.
- ⇒ Open the "Setup Folder"
- ⇒ Select the "IdeaBlade.DbConnectionCleaner" project
- ⇒ Right-click the mouse and select "Debug | Start new instance" from the context menu.

When you decide to explore the n-tier deployment options, you'll also have to correct the strings in:

The n-tier deployment of Cabana is not ready yet.

- The connection strings in the `[DataSourceKeyInfo]` table in IdeaBladeTutorial; see the next topic if you don't have such a table.
- The "security" RdbKey in the Server.IIS. IdeaBlade.ibconfig (but not in the "default" key).
- The "security" RdbKey in the Server.WinSvc. IdeaBlade.ibconfig (but not in the "default" key).

You must make these changes manually. `DbConnectionCleaner` will not help you.

CAB Designer Components in Visual Studio Toolbox Fail

So you want to drag a CAB component from the toolbox onto the canvas?

You don't have a Toolbox tab with CAB components? Skip the "problem" section below and go to the Remedy".

Problem

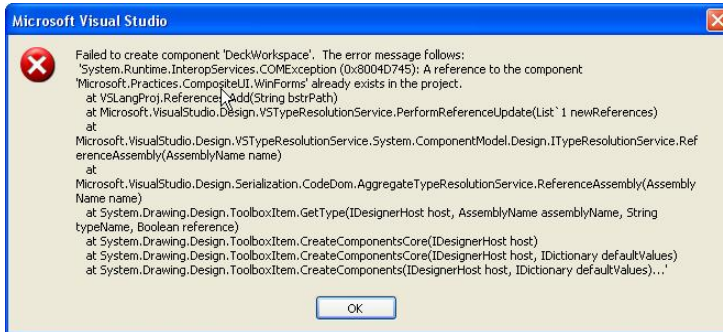
You might encounter two problems.

- The Toolbox tab with your CAB components disappeared.
- Visual Studio throws a fit when you drag a component onto the canvas.

Cabana Installation & Release Notes

No Tab? The likely cause is that Visual Studio can't find the directory where it thinks these components should reside. This has to be the Cabana "Lib" directory. Chances are you erased your Cabana solution or moved it. Go to the remedy section.

You have the tab but you get this:



The likely cause here is that your toolbox tab points to CAB components from assemblies that are different from the ones you are referencing in your application solution.

Maybe you have other CAB applications and your Visual Studio tab components reference the dlls in that solution?

In this example, my UserControl is in a project that references the `Microsoft.Practices.CompositeUI.WinForms` assembly in the application's "Lib" directory but the Visual Studio toolbox component references the `Microsoft.Practices.CompositeUI.WinForms` produced by building the CAB source code⁴.

What a pain!

Remedy

You must create a toolbox tab with CAB designer components from the same assembly that your project is referencing. That means the toolbox tab must reference the assemblies in the Lib directory of your application.

Do the following:

- ⇒ Add a new tab to the toolbox, e.g. CabanaLib
- ⇒ Right-click in the tab and pick "Choose items ..."
- ⇒ Click the Browse button and navigate to the Lib directory
- ⇒ Pick the `Microsoft.Practices.CompositeUI.WinForms` assembly
- ⇒ In the filter text box, type "Microsoft.Practices.Comp"
- ⇒ Click OK
- ⇒ Click the Browse button again
- ⇒ Pick the `DotNet.CompositeUI.Extensions` assembly
- ⇒ In the filter text box, type "CompositeUIE"
- ⇒ Click OK

You are done unless you are using third party controls. If using third party controls such as DevEx, continue

- ⇒ Click the Browse button again
- ⇒ Pick the `DevEx.CompositeUI.Extensions` assembly

⁴ The assembly in C:\Program Files\Microsoft Composite UI App Block\CSharp\Source\CompositeUI.WinForms\bin\Debug

Cabana Installation & Release Notes

- ⇒ In the filter text box, type “CompositeUIE”
- ⇒ Click OK

Test Projects don't build or run; missing NUnit

Unit testing of Cabana is not ready yet.

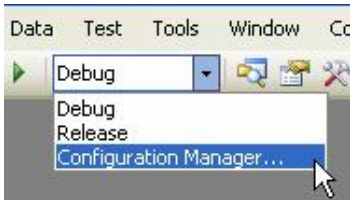
The test projects (all ending in ".Test") rely upon NUnit, the popular and free unit testing tool.

This installation package sets the solution build configuration to exclude these projects. Many of you will not have NUnit installed and these projects did not build successfully for want of the NUnit libraries.

This means you can't test properly until you add NUnit and include the test projects back into the build – a practice we strongly recommend.

You should use unit testing in your application and Cabana offers clues as to how to proceed. Here's how to re-engage unit testing with Cabana.

- ⇒ Shut down Visual Studio.
- ⇒ [Download NUnit v.2.4.1 for .NET 2.0](#) and install it.
- ⇒ Launch Visual Studio.
- ⇒ Open the “Configuration Manager” from the tool strip.



- ⇒ Find the test projects and check their checkboxes so that they are included in future builds.
- ⇒ Rebuild the solution.

Install IdeaBlade CAB Wizards & Templates

Many of you have heard of the Smart Client Software Factory (SCSF) and its ability to automatically generate applications, projects, view/presenter pairs and other useful code.

It does so by means of the Guidance Automation Toolkit (GAT) and “recipes” that ship with SCSF.

Most of the shipped recipes are really just tutorial examples. We wouldn’t use them for production code.

The Patterns and Practices folks understand this full well and encourage us to develop our own recipes to suit our particular application development needs.

Unfortunately, recipe development is difficult (to say the least). We have gone in a different direction to achieve the same effect. We are promoting the use of Visual Studio Templates, supported by wizards.

These are describe in their own “Getting Started Guide”, available in the download package and on the web.

You can learn about Visual Studio 2005 Templates at <http://msdn2.microsoft.com/en-us/library/6db0hwky.aspx>.

Visual Studio Templates in Cabana versions prior to v.2.3.0.0

Earlier version of Cabana introduced bare bones Visual Studio templates for rapidly developing views. These have been supplanted by the “Wizards & Templates” feature introduced in v.2.3.0.0.

The Cabana Project team has not yet tackled the intricacies of GAT recipe development. In the meanwhile, we have converted some code file collections into Visual Studio Templates.

The default VS template capabilities are not nearly as rich as GAT’s so we cannot come close to duplicating the SCSF / GAT experience. But we do have some templates that can speed your development time and, in this section, we show you how to install them.

Release Information

Version 2.3.0.0

This release is built for DevForce version 3.5.3.x. It will not compile for earlier versions because of an important change to the `IdeaBlade.Persistence.Entity` class affecting the implementation of `IDataErrorInfo`.

This is a major release that introduces Cabana Wizards.

There has been some refactoring but the most significant changes have been to the names and locations of many classes and assemblies – changes in support of the Wizards.

Cabana installation changes profoundly. The new “Getting Started Guide” explains installation anew and also covers how to use the Wizards to build your own Cabana-style application.

Change Details

Forthcoming. Please check the website for the most recent Release Notes (www.ideablade.com/Cabana).

Cabana Installation & Release Notes

Version 2.2.0.0

This release is built for DevForce version 3.5.2.x. It will compile and run on DF 3.5.1.1 but you must have and install the updated IdeaBladeTutorial database distributed with DF 3.5.2.3.

This is a major release with new patterns and a massive refactoring. It is, in effect, a code reset.

Please read the document “**CabanaStructure**” that describes the files and solution structures.

This document is an absolute must. You can get the latest version from our website at <http://www.ideablade.com/Cabana/CabanaStructure.pdf>.

Documentation remains rough. Improving on it is our highest development priority and you should see new material in the next month or two.

“**CabanaReference**” is a draft of the architecture with numerous class diagrams: <http://www.ideablade.com/Cabana/CabanaReference.pdf>.

ObjectBuilder Version Nightmare

Previous Cabana versions relied upon the Microsoft.Practices.ObjectBuilder assembly produced from source code we all can download. This was version 1.0.**51205.0**.

When they built Enterprise Library version 3.0 using ObjectBuilder, they decided that the ObjectBuilder assembly should be strongly named and signed with their certificate. They didn’t change the code but they did produce a new build with version number 1.0.**51206.0**.

This creates a significant hardship for CAB application developers. Our application can only refer to one version of the ObjectBuilder – either 1.0.**51205.0** or 1.0.**51206.0**.

If we want to use Enterprise Library v.3, we have to reference the 1.0.**51206.0** ObjectBuilder. We can still look at the source code – it’s the same as for 1.0.**51205.0**; but we can’t debug into it because we don’t have the pdb files and the specific source code files used to build 1.0.**51206.0**. Moreover, it is easy to have dangling references to 1.0.**51205.0**.

Separation of “our” code from “your” code

One of the most important developments, beginning in this release, is the separation of the “application” representing what you might build from the source code that third parties – mostly us at IdeaBlade – will maintain.

This separation is evident in the delivery of two solutions, **CabanaLib** and **CabanaXXXCS**.

The build implications of this separation are covered in the Installation guide above.

The goal is to make it easier for you to discard Cabana while leveraging the Cabana infrastructure. The distribution of classes between these two solutions is covered thoroughly in the **CabanaStructure** document.

DotNet and DevExpress Versions

Cabana, the application proper, now comes in two flavors for the DotNet and Developer Express UI control suites: **CabanaDotNetCS** and **CabanaDEVEXCS**.

We are working on a version for Infragistics controls.

The DotNet version uses nothing but DotNet controls.

Cabana Installation & Release Notes

The DevEx version uses a combination of DotNet and Developer Express controls. The DevEx Cabana sports a different look and an extra "Admin" module that demonstrates a nested grid (aka, drill down grid).

We intend it to be 100% DevEx controls some day but, for now, we've at least shown the way.

The DevEx CAB wrappers date from 2006. There was a new version of these wrappers just before this release; we'll incorporate them in the next Cabana version.

The DevEx version requires at least the trial version of the Developer Express WinForms control suite (that's what I use). You can get it from <http://www.devexpress.com/>.

This CabanaDevEx version was built with DevEx suite 7.1. DevEx just moved on to 7.2. Unfortunately, you can't get the trial version of 7.1 from DevEx and neither Cabana or DevForce 3.2.2.2 will work with 7.2 yet.

The third party control vendors are always doing this to us. We'll catch up next release. Meanwhile, if you don't have DevEx and you want to try Cabana, contact IdeaBlade support and ask them to send you a copy of the DevEx 7.1 trial.

The Admin Module

While the Admin module is nominally concerned with maintaining user permissions, please be aware that it is just a demo and that this version of Cabana does not make use of the "security" tables that are the target of the module nor can this version of Cabana modify those tables.

The Admin module is also interesting for its use of "business object proxy" classes – classes that stand-in for persistable business objects. We'll discuss the "why" and "how" of this technique in future documentation.

The DotNet version does not have the "Admin" module found in the DevEx version as the module's primary purpose is to demonstrate drill-down grids ... a feature not available in DotNet's grid controls.

Templates

There are new view templates reflecting our ability to get rid of superfluous class files in this release.

There are some very cool templates on the way for

- building a new application from scratch
- adding a module
- adding a page to a module

You should be able to punch out a new application in about half an hour. We're polishing those now and will release them very soon.

Documentation

Documentation is the gravest weakness of CAB and of Cabana.

We've added some to this release such as the "CabanaReference" document, available at www.ideablade.com/Cabana/CabanaReference.pdf, describes the architecture and has many class diagrams. It's still a work in progress.

We're working on some "how to" scenarios describing both the "how" and the "why" of common development tasks.

We expect to produce some videos in the next few months.

Cabana Installation & Release Notes

Miscellaneous

The source code is dotted with commented out alternatives. For example,

- You can turn off the stopwatch logging by using the `DummyStopwatchLogger` service; see `FoundationModule.FoundationServices`.
- Change visibility of a tab in `Infrastructure.Foundation.DemoDotNetTabVisibility` in the Miscellaneous folder.
- It seems the easiest way to change a page controller's EntityManager is by overriding `GetDefaultEntityManager` as illustrated in `Cabana.UI.SalesOrder.CustomerSearchPageController`.

You'll have to hunt for the others.

Cabana Installation & Release Notes

Version 1.0.4.0

This is the last Cabana release for DevForce version 3.2.1.x. The next releases will require DevForce version 3.3.0.x.

I heavily refactored the `PageController` for this release because I noticed that

- `EmployeePageController` and `CustomerPageController` were very similar.
- Both were hard to follow.
- The tab building logic was similar across the tabs and across the pages.

These conditions cry out for refactoring ... and I heard the cry:

- An abstract base class for page controllers called `EntitySummaryDetailPageController`.
- Instances of it such as `EmployeePageController` and `CustomerPageController`.
- `TabViewBuilder`, another abstract base class that follows the Builder design pattern. Each `TabViewBuilder` instance creates and holds a `TabPage` view and presenter.
- `TabViewBuilder` subclasses implement the specifics of a `TabPage` for a particular purpose.
 - The `GeneralTabViewBuilder` is used to build a `TabPage` that provides more detail about the current entity in the main `BindingSource`. The “General” tab is a typical example.
 - Tab-specific subclasses are defined as **private nested classes** of their parent Controller classes. Inside `EmployeePageController`, for example, are `OrderMasterDetailViewBuilder` and `EmployeeCustomersviewBuilder`.

These are private because (a) no client of the Controller class needs to know about them, (b) they do not have any demonstrable re-usability elsewhere, and (c) keeping them in the Controller class file makes them easier to examine and understand. They could easily be pulled out and made `internal` or `public` if the need arose.

- `TabViewBuilderList`, a class which holds `TabViewBuilder` instances and facilitates calling one of the `TabViewBuilder` “build phase” methods across all to its contained instances.

This expedient cleaned up the Controller code considerably and made adding a `TabPage` as simple as adding its `TabViewBuilder` to the Controller’s `TabViewBuilderList`.

I added the first two Visual Studio Item Templates in this release:

Item Template	Description
<code>ControlView</code>	A C# template producing a View / Presenter pair in which the developer uses the <code>ControlBindingManager</code> UI Designer to populate the SmartPart view with loose controls and bind them to properties of objects in a <code>BindingSource</code> .
<code>GridView</code>	A C# template producing a View / Presenter pair in which the developer uses the <code>DotNetDataGridView</code> UI Designer to add columns to a DotNet <code>DataGridView</code> and bind them to properties of objects in a <code>BindingSource</code> .

The source for these templates is in the zip files in the `ItemTemplates` directory in the root of the Cabana distribution zip.

Version 1.0.3.0

This is the first version to be deployed as an n-tier application from our website. There are only very minor changes.

Cabana Installation & Release Notes

SplashForm

I've added a `SplashForm` so that Cabana displays something to amuse the user while CAB builds the pages.

I found that this was especially important on “normal” machines accessing our app over the web. It takes an uncomfortably long time for CAB to do whatever it does. Maybe there is some latency on the IIS side too.

I threw this in as a total hack (see `ShellApplication`). I made no attempt to reach for resources in a decoupled way; I just jammed them into the form.

I also rudely abort the thread that displays the splash; this is crude and generally a no-no. But it's only a splash page so I hope I'm forgiven.

Performance and Span Queries with `GetManagedChildren()`

I generalized the approach to span queries that was introduced in Version 1.0.2.0.

I added overloads to the DevForce `GetManagedChildren` method in the `IdeaBlade.Cab.EntityModel.CommonEntity` class so that one can add spans (or otherwise tweak) the method with which DevForce implements a child relation property such as `Employee.Orders`.

See the revised `GetEmployeeOrders` method in `EmployeePageController` for an example.

Version 1.0.2.0

This is the third public release. The visible changes are small but Cabana has taken an important step forward: It is now deployable as a `ClickOnce` application in both 2-tier and n-tier modes.

Performance and Span Queries

N-tier deployment revealed a common performance issue: too many round-trips to the server when fetching data for grids. This was not apparent in development where all tiers are on the same box. It should up in 15-30 second delays in n-tier.

Spans are the usual remedy and they worked again here. See the `GetEmployeeOrders` method in `EmployeePageController` for an example.

QueuedCall

I generalized the `QueuedCall` class in `IdeaBlade.Cab.Library.Utility` so that it can represent any command that needs to be delayed during view configuration.

It's current only use is to postpone the data binding commands issued to presenters by their views. This is a necessity when the view issues the command before the view's `BindingSource` has been made ready. DevForce data binding handles this well but examples of raw .NET data binding don't work unless the actual `BindingSource` is known and prepared. If the presenter were to data bind when asked, it would fail.

I saw this as merely one among many possible cases in which the view would ask the presenter to perform a function that it was not ready to perform. Rather than blow up or do the wrong thing, the presenter can queue the command and try it again later when the conditions are ripe.

`QueuedCall` takes a subroutine to call (the command), its argument(s), and a test method that indicates if the requisite conditions for calling the command have been met. Both methods are delegates – typically instance delegates – so they can have access to variables and methods of the presenter instance in which they are defined.

SplashForm

I've added a `SplashForm` so that Cabana displays something to amuse the user while CAB builds the pages.

Cabana Installation & Release Notes

I found that this was especially important on “normal” machines accessing our app over the web. It takes an uncomfortably long time for CAB to do whatever it does. Maybe there is some latency on the IIS side too.

I threw this in as a total hack (see ShellApplication). I made no attempt to reach for resources in a decoupled way; I just jammed them into the form.

I also rudely abort the thread that displays the splash; this is crude and generally a no-no. But it’s only a splash page so I hope I’m forgiven.

Version 1.0.1.0

This is the second public release.

Infrastructure Project Re-organization

Infrastructure projects have been re-organized as part of a project to

- improve the clarity of what they do
- completely separate (a) the source files provided by Smart Client Software Factory (SCSF), (b) the IdeaBlade files, and (c) your files. The “your files” in this case is the Cabana sample application files.

The goal is that you could completely replace just the Cabana files with your own material and take advantage of the code “assets” provided by SCSF and IdeaBlade.

SCSF will change its code over time. We will certainly change ours. You don’t want to be able to see the source code we provide but you will have a devil of a time keeping up if you make changes in SCSF or IdeaBlade code.

Try to avoid that and be sure to track the changes you feel you have to make. Please let us know about those changes so that everyone can understand and benefit from your contributions.

Cabana Installation & Release Notes

Here is the new arrangement of the Infrastructure Visual Studio Solution Folder:

Folder	Project	Description
Infrastructure.IdeaBlade	IdeaBlade.Cab.EntityModel	IdeaBlade assets in support of business object models. Includes BrokenRules.
Infrastructure.IdeaBlade	IdeaBlade.Cab.Library	CAB-oriented UI support including base views and presenters, BindingSourceService , and GridBuilderBase
Infrastructure.Scsf	Scsf.Infrastructure.Interface	SCSF interface assets that were originally generated into the Infrastructure.Interface project. This subset of those assets would be unlikely to change from one SCSF app to the next.
Infrastructure.Scsf	Scsf.Infrastructure.Library	All SCSF library assets that were originally generated into the Infrastructure.Library project. They would be unlikely to change from one SCSF app to the next.
Infrastructure	Infrastructure.Foundation	Your application specific assets including resources, services, ListConverters , and application-specific base GridBuilder and view classes. They are particular to Cabana now; you will change this for your application.
Infrastructure	Infrastructure.Interface	Your application Constants. Everything else was relegated to Scsf.Infrastructure.Interface .
Infrastructure	Infrastructure.Layout	The layout of the main “card” in the Shell’s DeckWorkspace. The primary components in Cabana are the OutlookBarWorkspace on the left and the page DeckWorkspace on the right.
Infrastructure	Infrastructure.OutlookBarWorkspace	The OutlookBarWorkspace from Matias Woloski. See below.
Infrastructure	Shell	The application’s “Shell” form and start-up project.

OutlookBarWorkspace

The former mysterious “CompositeUIExtensions.WinForms” project is now the more obvious “Infrastructure.OutlookBarWorkspace” (although it retains this original name for its Namespace).

The project is no longer flying under the “IdeaBlade” banner. We want to make absolutely clear that this is a distinctive and highly valued contribution from Matias Woloski who deserves more overt credit than we gave him initially. There is a PDF in this project that references the original source and describes the changes we made.

The IdeaBlade assemblies no longer have a dependence upon this assembly; Cabana does, as seen in the [Infrastructure.Foundation](#) module.

GridBuilder Changes

The [GridBuilder](#) collection of classes have been revised to make it easier to accommodate similar versions of the same kind of [GridBuilder](#). For example, you can now tweak the [GridBuilder](#) prototype instance of the [OrderGridBuilder](#) to include or exclude the “Customer Name” column.

This was “useful” on the Customer page which does not need to repeat the Order’s Customer name in the grid (it’s always the same ... the Customer shown in the summary above).

Of course it’s a bad idea to clutter a class with knobs and switches. One might have decided to create a separate [GridBuilder](#) class for this purpose, the [CustomerOrderGridBuilder](#). So we created one and showed how it could inherit from [OrderGridBuilder](#) and modify the columns after [OrderGridBuilder](#) was done.

Cabana Installation & Release Notes

`CustomerPageController.AddServices()` shows you how to use either `GridBuilder` approach.

ControlViewPresenter Changes

`ControlViewPresenterBase` now has the ability to delay the action of a View's method call until certain pre-conditions are met.

For example, some of the data bindings depend upon the readiness of the `BindingSource`. If the View called these presenter methods too early – before the `BindingSource` was ready – the bindings were dropped at runtime.

This meant that the View had to be aware of the Presenter's initialization order, at least to the extent of calling some presenter methods in the `OnLoad`. Such awareness is a bad code smell and we were able to sanitize it by enabling the Presenter to delay the binding until the `BindingSource` was ready. The View can call the method whenever it likes and the Presenter will get the timing right.

See `ControlViewPresenterBase.DelayUntilBindingSourceReady`.

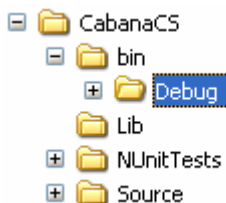
The mechanism is general purpose and, thus, adaptable to other initialization order problems we may encounter in future.

Version 1.0.0.0

This is the first public release so there are no announced changes between this and pre-release versions of Cabana.

Many of the features and behavior are the same as in **Funhouse**, the Cabana's predecessor reference application.

One important difference is that all assemblies are compiled into the "bin/Debug" directory off of the application root directory as in this illustration:



Funhouse Features missing in Cabana

Cabana has new features that were not in Funhouse and would have been difficult to add to Funhouse. Navigation is one example, the expanding and collapsing "Alerts" panel for Broken Rule display is another.

On the other hand, many Funhouse features have not been ported yet to Cabana. Among the most important:

- Save
- Delete and undo
- Order managed list (the filtering of the Order grid by order date).
- Employee filter by name or Id
- Login

These features will cross over soon.