Quick Guide

Migrating FakeXrmEasy 1.x to the latest versions

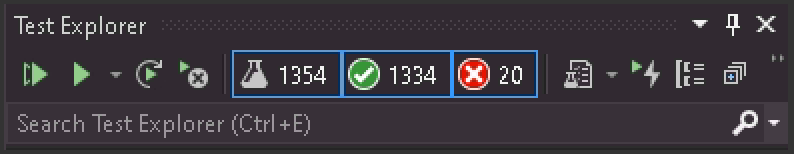
If you had a project that was using FakeXrmEasy (`FXE`) v1.x, this guide will help you migrate your projects to the latest versions in a few steps.

`FXE` allows you to **build, run, test and debug your applications 100%** `In-Memory` , which is extremely `fast` , and more `carbon friendly` .

DYNAMICS VALUE

Benefits

- Drive development of `.net` applications that connect to Dataverse through **automated testing** `at scale`.



- Increased `developer efficiency`: prevent and find `bugs` instantly.
- ↓ CPU cycles than with integration testing > ↓ energy > `carbon friendly`.

DYNAMICS VALUE

Benefits (II)

- `Mitigate` issues `locally` due to how `FXE` mimics Dataverse functionality.
- `Shortens` the developer feedback loop, cause you don't have to deploy in-between changes.
- `Open Source`: source code is public and **fully auditable**.

Background

You could use `FXE v1.x` to dev & test both server-side and client-side applications in either `net452` or `net462` before.

With the latest version of the `DataverseClient`, `1.0.1`, that targets `netcoreapp3.1`, we can use now `netcore` to build **client-side applications**...

...but we must still use the previous `Microsoft.CrmSdk.CoreAssemblies` package for **server-side development**.

Server-side applications

Examples of server side applications are:

- **Plugins**
- **Custom Actions / Apis**
- **Custom Connectors**
- ...

These run inside the Dataverse's app pool / sandbox service once deployed.

DYNAMICS VALUE

Client-side applications

Examples of client-side applications are:

- **Azure Functions**
- **Bespoke Web Api implementations**
- **MVC web applications**
- **Blazor applications**
- **Console apps**
- **Messaging apps**

DYNAMICS VALUE

Choosing your FXE version

Use `v2.1.2` for Server-Side...

... applications that use the `Microsoft.CrmSdk.CoreAssemblies` package.

Use `v3.1.2` for Client-Side...

... applications that use the new `Microsoft.PowerPlatform.Dataverse.Client` `v1.0.1` package.

DYNAMICS VALUE

Step 1a: Refactor to a base class

- Move all references to `XrmFakedContext` and `IOrganizationService` to a base test class
- This will make using the new middleware configuration a whole lot easier.
- Plus, it makes unit tests `smaller`, `clean`, and easier to `read`.

DYNAMICS VALUE

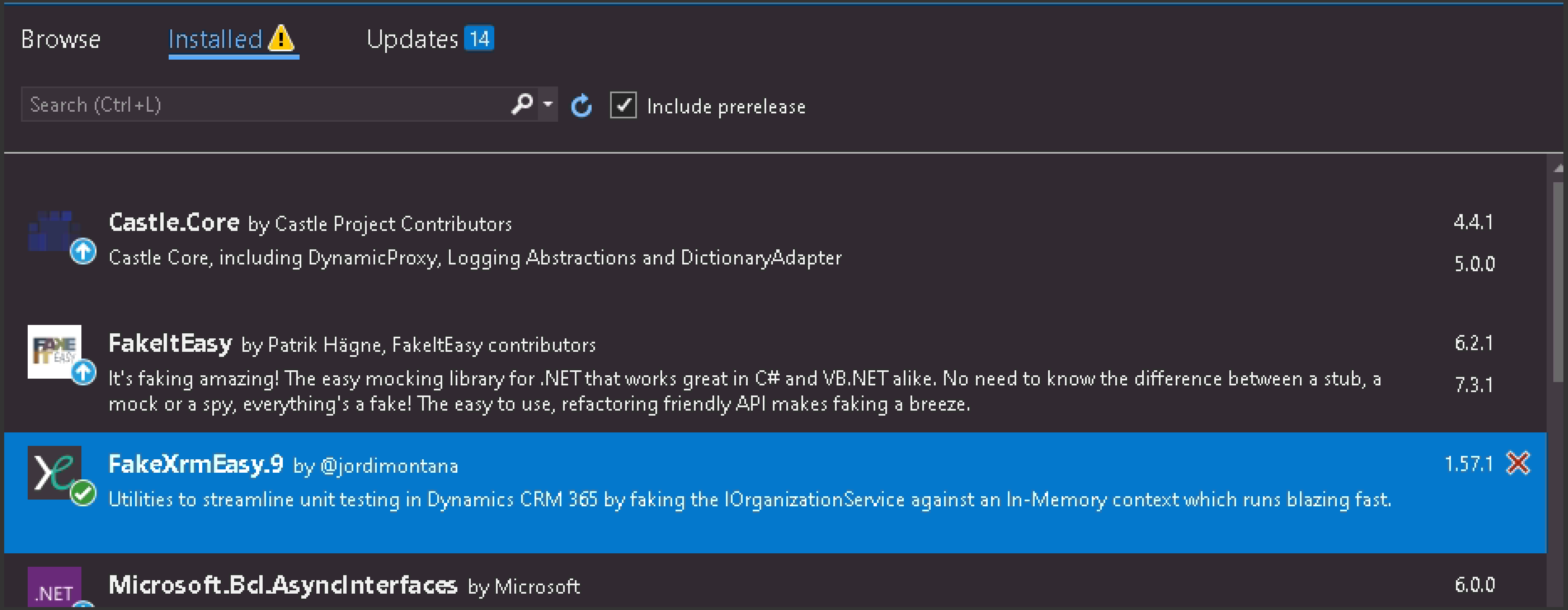
Step 1b: Refactor to a base class

```
public class FakeXrmEasyTestBase
{
    protected readonly IOrganizationService _service;
    protected readonly XrmFakedContext _context;

    public FakeXrmEasyTestBase()
    {
        _context = new XrmFakedContext();
        _service = _context.GetOrganizationService();
    }
}
```

DYNAMICS VALUE

Step 2: Uninstall FakeXrmEasy v1.x

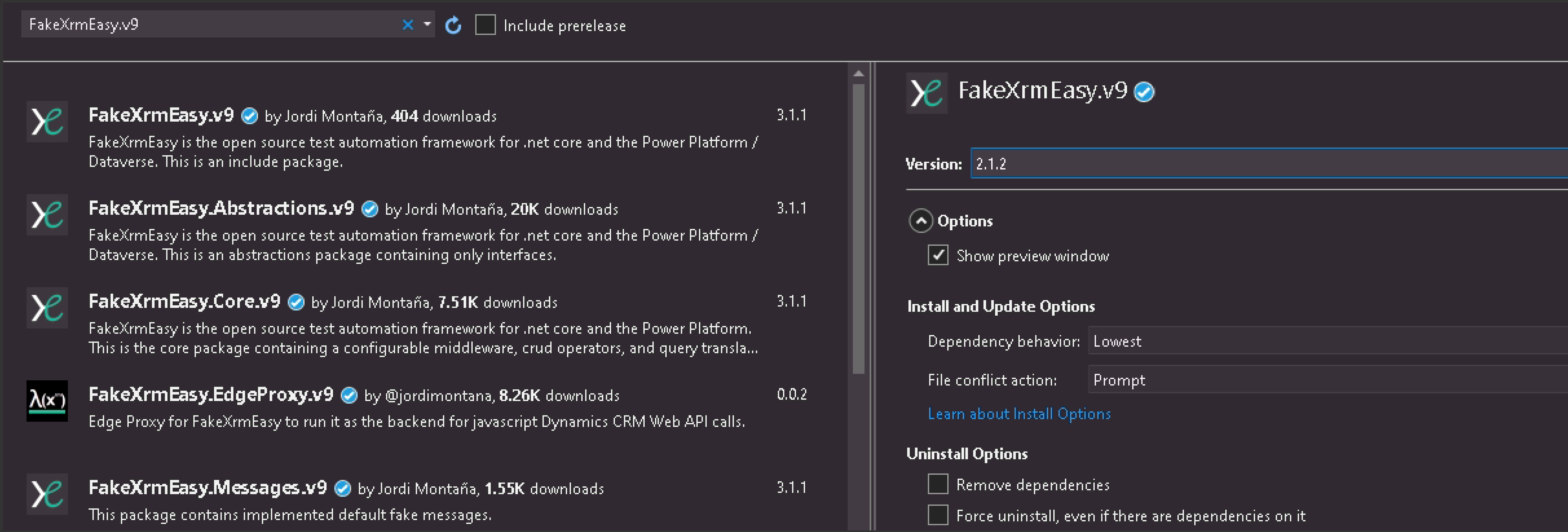


Browse | Installed ⚠ | Updates 14

Search (Ctrl+L) ☑ Include prerelease

Castle.Core by Castle Project Contributors Castle Core, including DynamicProxy, Logging Abstractions and DictionaryAdapter	4.4.1 5.0.0
FakeItEasy by Patrik Hägne, FakeItEasy contributors It's faking amazing! The easy mocking library for .NET that works great in C# and VB.NET alike. No need to know the difference between a stub, a mock or a spy, everything's a fake! The easy to use, refactoring friendly API makes faking a breeze.	6.2.1 7.3.1
FakeXrmEasy.9 by @jordimontana Utilities to streamline unit testing in Dynamics CRM 365 by faking the IOrganizationService against an In-Memory context which runs blazing fast.	1.57.1 ✗
Microsoft.Bcl.AsyncInterfaces by Microsoft	6.0.0

Step 3: Install FakeXrmEasy v2.x/v3.x



FakeXrmEasy.v9 × Include prerelease

FakeXrmEasy.v9 by Jordi Montaña, **404** downloads FakeXrmEasy is the open source test automation framework for .net core and the Power Platform / Dataverse. This is an include package.	3.1.1
FakeXrmEasy.Abstractions.v9 by Jordi Montaña, **20K** downloads FakeXrmEasy is the open source test automation framework for .net core and the Power Platform / Dataverse. This is an abstractions package containing only interfaces.	3.1.1
FakeXrmEasy.Core.v9 by Jordi Montaña, **7.51K** downloads FakeXrmEasy is the open source test automation framework for .net core and the Power Platform. This is the core package containing a configurable middleware, crud operators, and query transla...	3.1.1
FakeXrmEasy.EdgeProxy.v9 by @jordimontana, **8.26K** downloads Edge Proxy for FakeXrmEasy to run it as the backend for javascript Dynamics CRM Web API calls.	0.0.2
FakeXrmEasy.Messages.v9 by Jordi Montaña, **1.55K** downloads This package contains implemented default fake messages.	3.1.1

FakeXrmEasy.v9

Version: 2.1.2

Options

[✓] Show preview window

Install and Update Options

Dependency behavior: Lowest

File conflict action: Prompt

Learn about Install Options

Uninstall Options

[ ] Remove dependencies

[ ] Force uninstall, even if there are dependencies on it

DYNAMICS VALUE

Step 4 (Optional): Update namespaces

You might get build errors because some pre-existing methods have been rewritten as extension methods into dedicated namespaces.

For example, if you're unit testing plugins, include these namespaces:

```
using FakeXrmEasy.Abstractions.Plugins;
using FakeXrmEasy.Plugins;
```

DYNAMICS VALUE

Step 5: Setup middleware config

Now, we'll update the base test class that we refactored in Step 1 to use the brand new, fully configurable, `middleware`.

- Add the necessary usings:

```
using FakeXrmEasy.Abstractions;
using FakeXrmEasy.Abstractions.Enums;
using FakeXrmEasy.FakeMessageExecutors;
using FakeXrmEasy.Middleware;
using FakeXrmEasy.Middleware.Crud;
using FakeXrmEasy.Middleware.Messages;
```

Step 5: Setup middleware config (II)

Replace the `XrmFakedContext` reference by a new interface `IXrmFakedContext` :

```
protected readonly IXrmFakedContext _context;
```

In the next slide we'll define the new middleware and we'll choose a relevant license:

Step 5: Setup middleware config (III)

```csharp
public class FakeXrmEasyTestBase
{
    protected readonly IOrganizationService _service;
    protected readonly IXrmFakedContext _context;

    public FakeXrmEasyTestBase()
    {
        _context = MiddlewareBuilder
                        .New()

                        .AddCrud()
                        .AddFakeMessageExecutors(Assembly.GetAssembly(typeof(AddListMembersListRequestExecutor)))

                        .UseCrud()
                        .UseMessages()
                        .SetLicense(FakeXrmEasyLicense.RPL_1_5)
                        .Build();

        _service = _context.GetOrganizationService();
    }
}
```

Step 6: Re-run existing unit tests

After doing all the previous steps above now we're in a position to run again all the preexisting tests which should still `pass`.

This is because the `IOrganizationService` interface and most of the `FXE 1.x` API is forward compatible with `v2.x` and `v3.x` versions and the latest versions of the `ServiceClient` in the `Microsoft.PowerPlatform.Dataverse.Client` package, which also implements the IOrganizationService interface.

DYNAMICS VALUE

Congrats!

You should have successfully migrated across the latest version!

`Remember`: `FXE` version 2 or later uses a `sustainable OSS` licensing model which is free of charge for lots of scenarios but requires a commercial license for `propietary code and commercial use`.

You're still free to use it for `evaluation purposes` in a commercial context (i.e. PoC).

More info about [pricing here.]()

DYNAMICS VALUE

Thank You!

If you liked this material please give us a 👍 and share it with your peers!

- https://dynamicsvalue.github.io/fake-xrm-easy-docs/quickstart/migrating-from-1x/
- https://dynamicsvalue.com

Let's connect if you have any questions / doubts: info@dynamicsvalue.com

DYNAMICS VALUE