



Elsinore 2023

SA1: Project Management using Cider and Tatin (Part 1)



Gilgamesh Athoraya, Kai Jaeger, Morten Kromberg

Materials to Download

- From <https://github.com/dyalog-training/2023-SA1>
2023-SA1.pptx

- Also on circulating USB stick:

2023-SA1.pptx

CiderTatin folder (saves downloading & installing)



Dyalog: Making APL more enjoyable



Tatin



Cider



Dyalog: Making APL more enjoyable



Tatin



Cider

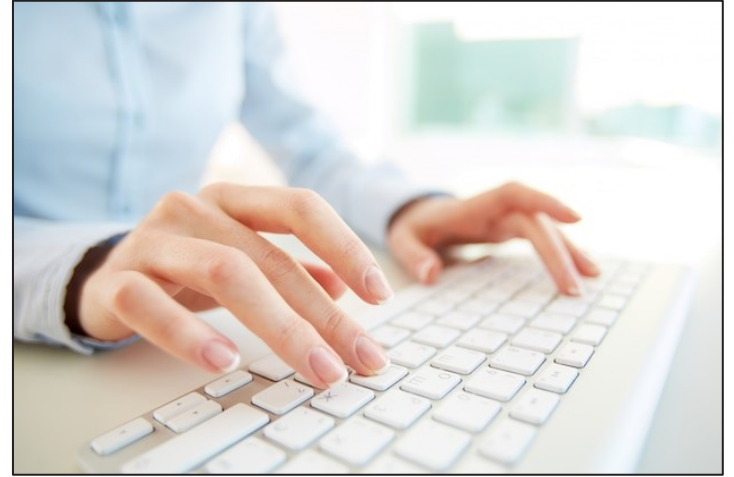


Dyalog: Making APL more enjoyable



Hands On!

- ✦ Unlike many workshops, there *should* be plenty of time for experimentation



Session 1: Introduction

- What is a Project and Why Would you want one?
- Installing, Enabling and Upgrading
 - Tatin Client and Cider
 - One of .NET 6.0-8.0
- What about Link?
- Finding Cider Documentation
- Creating a Project
 - Understanding Project Configuration Options
- Opening a Project
 - (under the covers)

Exercise 0:
Installation!

Exercises 1 & 2:
Create & Open a Project!



Session 2: Dependencies / Packages

- What is a Package?
- **Tatin:** APL Packages
 - Finding Documentation
 - Finding and Installing Packages
 - "Where do they go?"
- Dependencies of Dependencies
- **NuGet:** .NET Packages
 - Finding and Installing
 - "Where do they go?"

Exercise 3:
Add a Tatin Dependency (or two)!

Exercise 4:
Add a NuGet Dependency



12:00-13:00

Session 3: BYO App + Wrap Up

- ◆ Development Dependencies
- ◆ Build Your Own App
- ◆ Recap and Conclusions
- ◆ Link, Cider and Tatin ToDo Lists

SP1 This Afternoon:

Creating your own Packages

Exercise 5:
Build Your Own Application



What is a Project

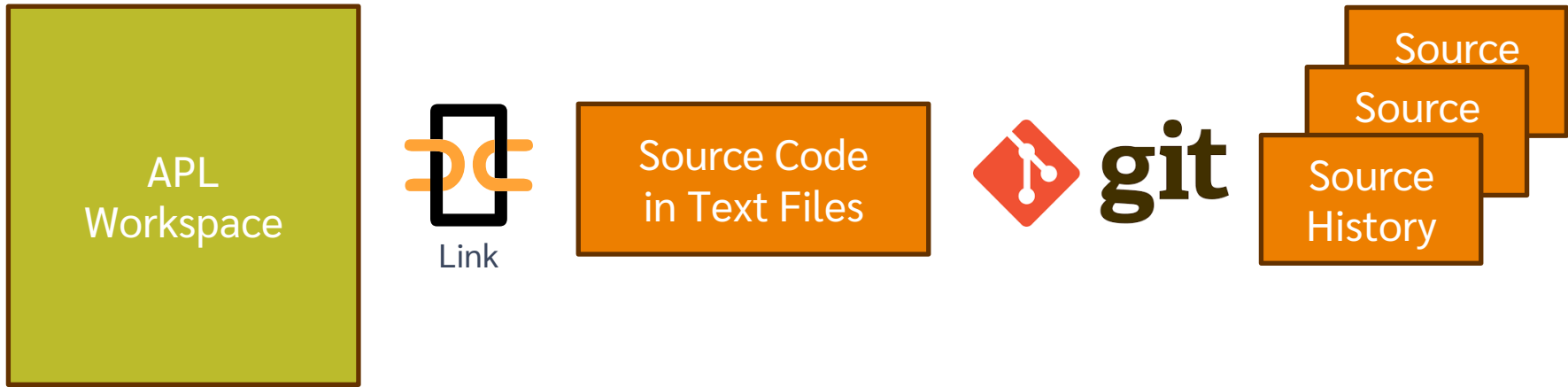
- ◆ And WHY Would I Want One?



First, There Was The Workspace



Then There was Link (and git/svn etc)



What More Could You Want?

- Load other code that we **depend** on
- Run some code on **opening** the project
- Run a **build** function
- Decide **where** to load the code
- Run **tests**
- Set **Link options** to be used when loading the source code
- Set **IO, ML**

```
cider.config
x + - □
File Rediger Vis
{
  CIDER: {
    dependencies: {
      nuget: "nuget-packages"
      tatin: "tatin-packages"
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockproj"
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
Ln 1, Col 1 80% Windows (CRLF) UTF-8
```



Cider is a **Project** Manager

A Project is a linked source folder,
a config file, plus optional dependencies



Tatin is a **Package** Manager

A Package is a project wrapped
up for consumption by others



History

- ◆ Tatin development started in 2020 using Acre
 - ◆ We decided that we needed a "more agnostic" / "less opinionated" project management system
- ◆ Cider was born in 2021
 - ◆ Initially as an internal tool for Tatin development
- ◆ Tatin is now close to v1.0 (v0.98.0)
- ◆ Cider still a prototype (v0.35.0)
 - ◆ Likely to evolve in next year or two
- ◆ Cider is based on Link, which is now at v4.0.11





the journal of the British APL Association



Kenneth E. Iverson
1920-2004

Current issue



Articles in press
Full index

Volumes

- [26](#)
- [25](#)
- [24](#)
- [23](#)
- [22](#)
- [21](#)
- [20](#)
- [19](#)
- [18](#)
- [17](#)
- [16](#)
- [15](#)
- [14](#)
- [13](#)
- [12](#)
- [11](#)
- [10](#)
- [9](#)
- [8](#)
- [7](#)
- [6](#)
- [5](#)
- [4](#)
- [3](#)
- [2](#)
- [1](#)

Search the index Google the archive

- home
- advertise
- archive**
- character-
- mapped code
- contribute
- in press
- team
- subscribe
- about us
- blog
- books
- community
- committee
- consultants
- events
- sponsors
- contact
- fonts
- interpreters
- video

archive/20/1 (July 2003)

Why APL Programmers Don't Use Libraries

by Morten Kromberg (email: mkromberg@insight.dk)

*The next best thing
to knowing something is
knowing where to find it.*

Samuel Johnson

As Stefano Lanzavecchia announced in Vol.19 No.4 (aka "April 2003"), the BAA is starting a Library Project, and I have volunteered to be the Librarian. The goal is to create a repository for freely available code, decently documented – supporting as many APL (and A+, J, K) interpreters as possible.

Here We Go Again

"Here we go again", I hear the old APL dogs groan to themselves. I was tempted to join in the chorus myself. We've been there, done that and – as everyone knows – you can't get APL programmers to use libraries (or, according to popular myth, adhere to standards of any kind). There is no doubt that – if you compare APL developers to most other programming communities – one of the ways in which we are different is in our use of libraries. For the most part, we don't – and in many situations, there are good reasons for this. My task as the librarian – as I see it – is to identify the most important reasons why we make less use of libraries, and – to the extent that it is profitable to us – help develop an APL library culture. I have



Archive articles posted online on request: ask the [archivist](#).

Most Importantly

- ◆ In addition to being valuable educational resources...
- ◆ Packages are critical to keeping APL competitive as a tool for building modern applications
- ◆ Support web protocols and components, data formats, operating system APIs, security requirements, etc
- ◆ There are a few things you don't want to build from the ground up each time



Installing Cider and Tatin

- ◆ If you have v19.0, use]tools.activate

```
]tools.activate all  
cmddir set to: /home/mkrom/.dyalog/dyalog.190U64.files/SessionExtensions/CiderTatin:  
              /home/mkrom/MyUCMDs:/opt/mdyalog/19.0/64/unicode/SALT/spice  
Now restart APL to complete activation.
```

- ◆ Or use the "CiderTatin" folder on the USB stick



Installing Cider and Tatin

- If you have v19.0 use]tools.activate

```
]tools.activate all  
cmddir set to .../com/.dyalog/dyalog.190U64.files/SessionExtensions/CiderTatin:  
.../com/MyUCMDs:/opt/mdyalog/19.0/64/unicode/SALT/spice  
Now restart ... to complete activation.
```

- Or use the "CiderTatin" folder on the USB stick
- Or follow normal Tatin installation instructions
 - ... and then use Tatin to install the package Cider
 - (see the following slides)



Cleanup If Necessary

In the following, replace ...blabla... by whatever is appropriate, e.g.

```
Dyalog APL 18.2 Unicode  
Dyalog APL-64 18.2 Unicode  
Dyalog APL-64 19.0 Unicode
```

And `mkrom` by your user id.

Check for the existence of an existing installation:

```
]settings cmdir  
C:\Users\mkrom\Documents\...blabla...\StartupSession\CiderTatin;  
C:\Users\mkrom\Documents\MyUCMDs;  
C:\Program Files\Dyalog\Dyalog APL-64 19.0 Unicode\SALT\spice
```

Remove existing installation:

```
]settings cmdir "~C:\Users\mkrom\Documents\...blabla...\StartupSession\CiderTatin"  
(repeats old setting)
```

```
3 □DELETE 'C:\Users\mkrom\Documents\...blabla...\StartupSession\CiderTatin'
```



Installing from USB "CiderTatin"

As before, replace ...blabla... by whatever is appropriate, e.g.

```
Dyalog APL 18.2 Unicode  
Dyalog APL-64 18.2 Unicode  
Dyalog APL-64 19.0 Unicode
```

And `mkrom` by your user id.

Copy CiderTatin to:

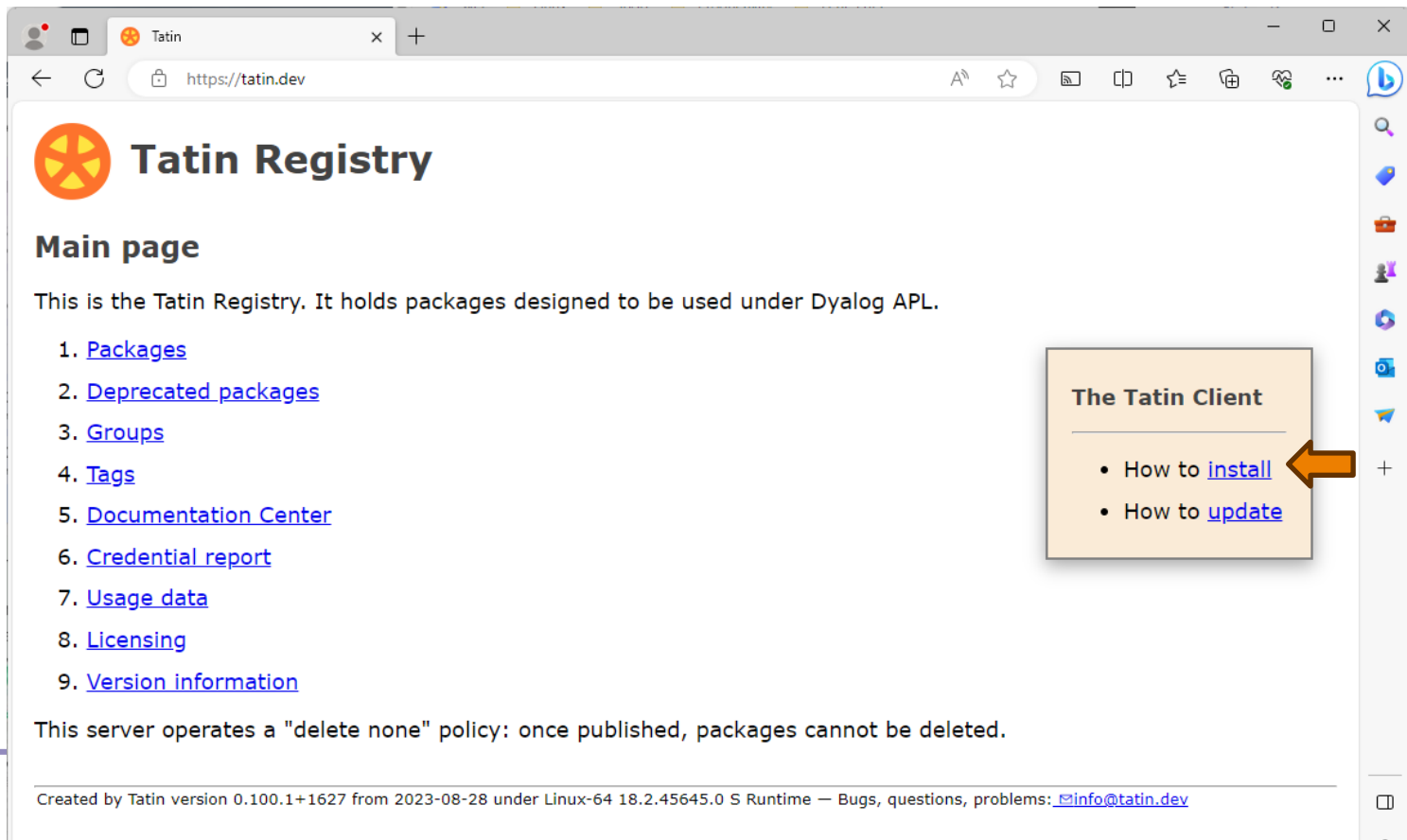
```
C:\Users\mkrom\Documents\...blabla...\SessionExtensions\CiderTatin
```

Tell UCMD system where to find it:

```
]settings cmddir ",C:\Users\mkrom\Documents\...blabla...\SessionExtensions\CiderTatin"  
(repeats old setting)
```



Normal Tatin Installation



The screenshot shows a web browser window displaying the Tatin Registry website. The browser's address bar shows the URL `https://tatin.dev`. The page features the Tatin logo (an orange circle with a white star-like shape) and the title "Tatin Registry". Below the title, the text "Main page" is displayed. A paragraph states: "This is the Tatin Registry. It holds packages designed to be used under Dyalog APL." A list of nine links is provided: 1. [Packages](#), 2. [Deprecated packages](#), 3. [Groups](#), 4. [Tags](#), 5. [Documentation Center](#), 6. [Credential report](#), 7. [Usage data](#), 8. [Licensing](#), and 9. [Version information](#). A note at the bottom states: "This server operates a 'delete none' policy: once published, packages cannot be deleted." A callout box titled "The Tatin Client" is overlaid on the right side of the page, containing two bullet points: "• How to [install](#)" and "• How to [update](#)". An orange arrow points from the "install" link in the callout box to the "install" link in the list on the page. The footer of the page reads: "Created by Tatin version 0.100.1+1627 from 2023-08-28 under Linux-64 18.2.45645.0 S Runtime — Bugs, questions, problems: info@tatin.dev".

Tatin Registry

Main page

This is the Tatin Registry. It holds packages designed to be used under Dyalog APL.

1. [Packages](#)
2. [Deprecated packages](#)
3. [Groups](#)
4. [Tags](#)
5. [Documentation Center](#)
6. [Credential report](#)
7. [Usage data](#)
8. [Licensing](#)
9. [Version information](#)

This server operates a "delete none" policy: once published, packages cannot be deleted.

Created by Tatin version 0.100.1+1627 from 2023-08-28 under Linux-64 18.2.45645.0 S Runtime — Bugs, questions, problems: info@tatin.dev

The Tatin Client

- How to [install](#)
- How to [update](#)



Tatin Installing and Updating x +

← ↻ <https://tatin.dev/Assets/docs/InstallingAndUpdatingTheTatinClient.html>

5. How to install in 18.0 and 18.2 & 19.0

Instructions:

1. Download the latest release of the Tatin client from <https://github.com/aplteam/Tatin/releases>
2. Unzip it into the target folder

As a result you should see something like this:

```
... \Dyalog APL[-64] <version> Unicode Files\SessionExtensions\CiderTatin\Tatin
```



Releases · aplteam/Tatin

https://github.com/aplteam/Tatin/releases

Product Solutions Open Source Pricing

Search or jump to... Sign in Sign up

aplteam / Tatin Public

Notifications Fork 6 Star 20

Code Issues 15 Pull requests Discussions Actions Projects Security Insights

Releases Tags Find a release

3 days ago

aplteam

v0.102.2

2f917f0


Compare

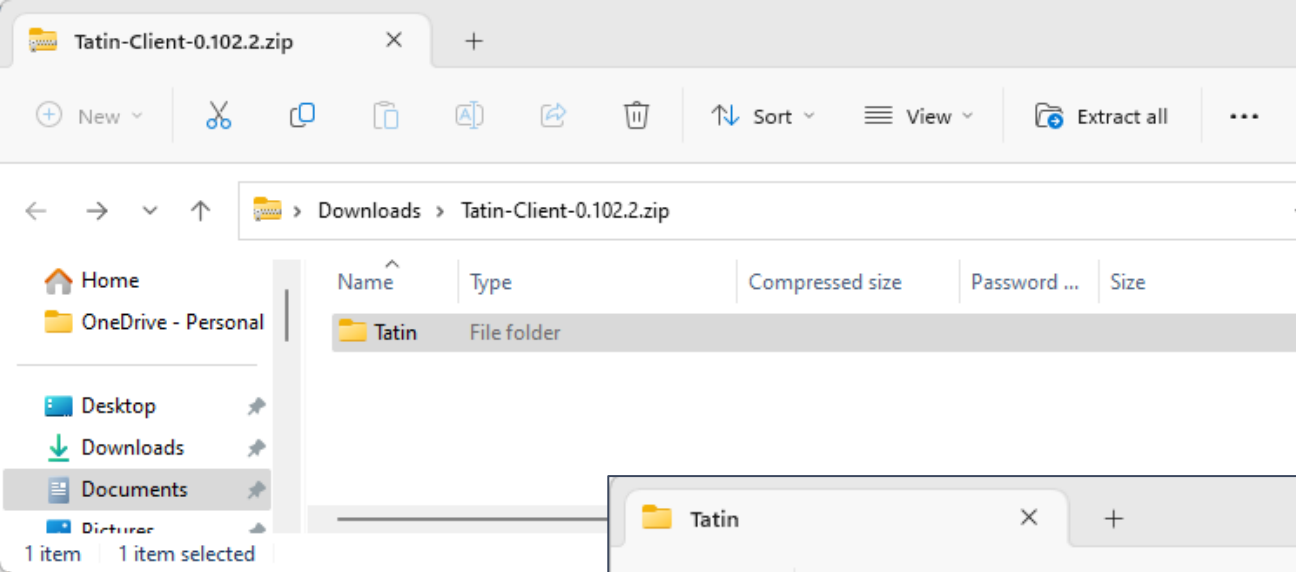
Version 0.102.2 Latest

Documentation corrected regarding the installation.

Assets 6

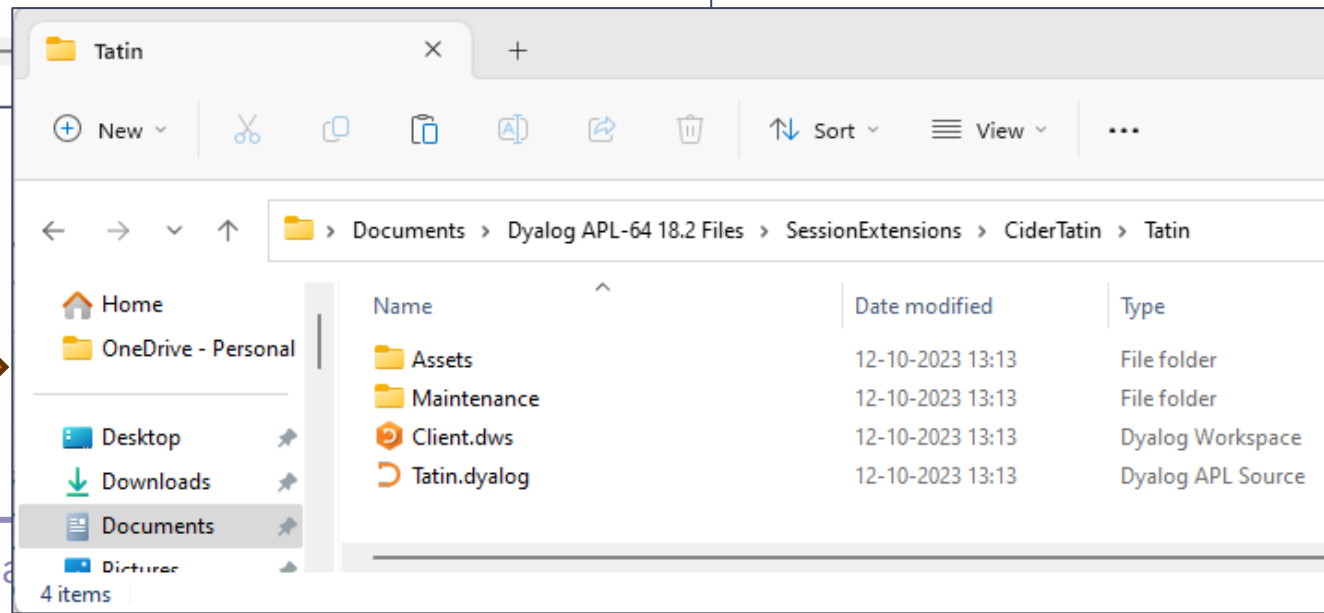
Run.aplf	2.13 KB	3 days ago
Tatin-Client-0.102.2.zip	1.07 MB	3 days ago
Tatin-Documentation-0.102.2.zip	146 KB	3 days ago
Tatin-Server-0.102.2.zip	13.2 MB	3 days ago
Source code (zip)		5 days ago
Source code (tar.gz)		5 days ago





Copy & Paste

Resulting in... →



Tell the UCMD system where to look

- Replace mkrom with your user name (□AN)
- If you are not using 64 Unicode
 - Replace "Dyalog APL-64 Unicode Files" below with
 - Dyalog APL[-64] 18.x [Unicode] Files
- Under Linux or macOS, the folder name will be
 - /home/<□AN>/dyalog.<version>U<bit>.files/SessionExtensions/CiderTatin

```
]settings cmddir ",C:\Users\mkrom\Documents\Dyalog APL-64 18.2 Unicode Files/SessionExtensions/CiderTatin" -permanent  
(displays previous setting)
```

```
]tatin.version  
Tatin 0.102.3+1685 2023-10-13
```



Installing Cider

In the following, replace [Folder] with the name of the folder that Tatin was installed into. Probably something like:

```
C:\Users\mkrom\Documents\Dyalog APL-64 18.2 Unicode Files/SessionExtensions/CiderTatin
```

```
]tatin.installPackages Cider "[Folder]/Cider"
```

```
Sure you want to create and install into [Folder]/Cider ? (Y/n) y
```

```
Installed into blabla/CiderTatin/Cider:
```

```
aplteam-Cider-0.37.4
```



Note

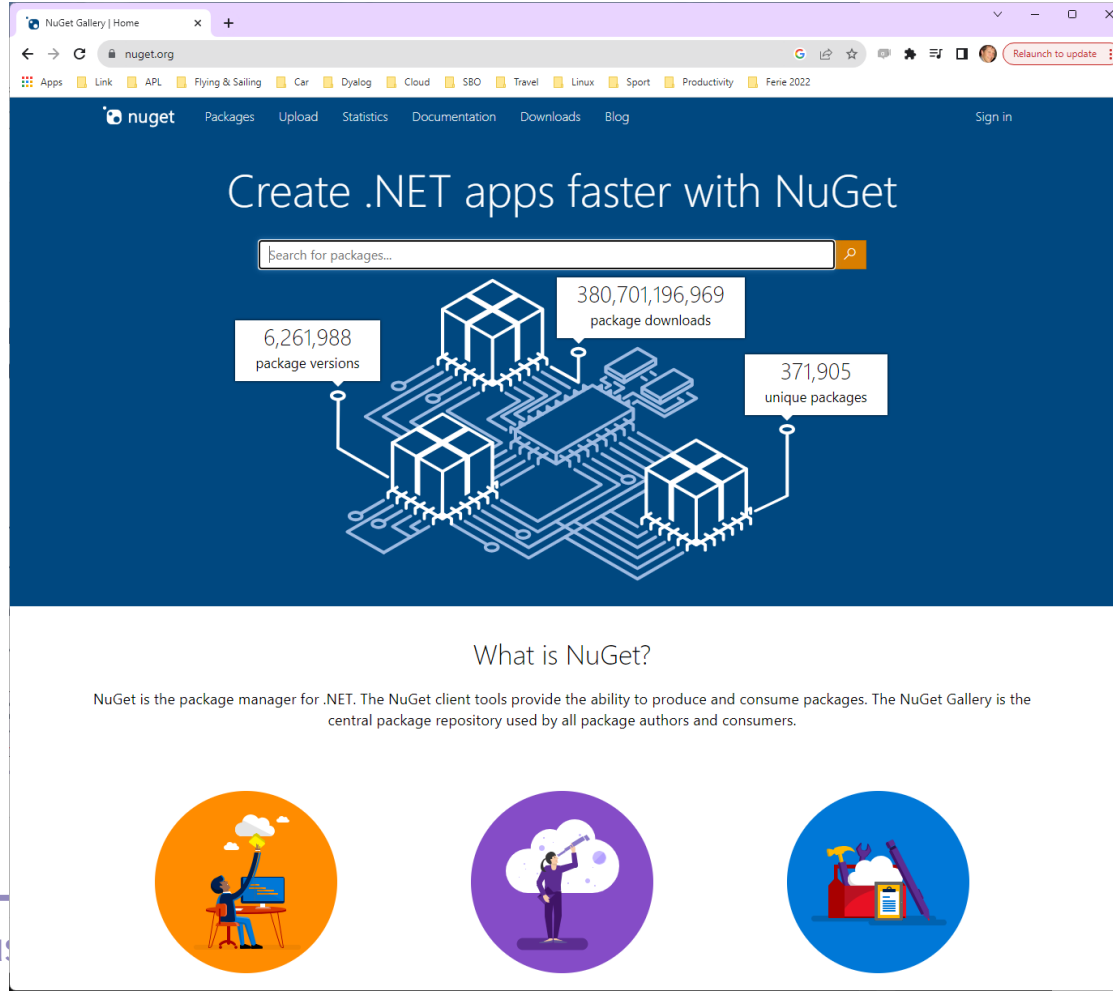
- These instructions make Cider and Tatin available as user commands
- The APIs (`SE.Tatin` and `SE.Cider`) are not available until the first call to a user command.
- You can materialise them with e.g.

```
SE.UCMD 'Cider.Version'
```



NuGet Packages

- NuGet is the .NET package manager
- To use NuGet packages with Dyalog APL, you need .NET 6.0 or later
- (we may add support for 4.0 / "Framework" later)



The screenshot shows the NuGet Gallery website. The header includes the NuGet logo and navigation links: Packages, Upload, Statistics, Documentation, Downloads, and Blog. A search bar is prominently displayed with the text "search for packages...". Below the search bar, three statistics are presented in white boxes with lines pointing to a central graphic of a circuit board with three server racks. The statistics are: 6,261,988 package versions, 380,701,196,969 package downloads, and 371,905 unique packages. Below the statistics, the heading "What is NuGet?" is followed by a paragraph: "NuGet is the package manager for .NET. The NuGet client tools provide the ability to produce and consume packages. The NuGet Gallery is the central package repository used by all package authors and consumers." At the bottom, there are three circular icons: an orange circle with a person at a computer, a purple circle with a person holding a telescope, and a blue circle with a person at a workbench.

NuGet Gallery | Home

nuget.org

Apps Link APL Flying & Sailing Car Dyalog Cloud SBO Travel Linux Sport Productivity Ferie 2022

nuget Packages Upload Statistics Documentation Downloads Blog Sign in

Create .NET apps faster with NuGet

search for packages...




6,261,988 package versions

380,701,196,969 package downloads

371,905 unique packages



What is NuGet?

NuGet is the package manager for .NET. The NuGet client tools provide the ability to produce and consume packages. The NuGet Gallery is the central package repository used by all package authors and consumers.



[Microsoft].NET History

- As .NET celebrates 20 years of existence, Microsoft is moving to the new open source, cross-platform .NET.

Name	Platforms	Version Numbers
Microsoft.NET Framework	Windows	1 2 4.0 (aka "4.8" ...)
".NET Core" 	Windows Linux macOS	3.0 3.1 
".NET"	Windows Linux macOS	5.0 6.0 7.0 8.0

- Dyalog v18.0 added a bridge to .NET 3, to complement the 20 year old bridge to the .NET framework.
- v18.2 was tested with 3.1 ("Core") but works with 5.0 and later
- v19.0 targets 8.0 (Long Term Support version due on Nov 8th 2023)



Installing .NET

- Microsoft Windows is shipped with the DotNet Framework (version 4.8) installed
 - .NET 6-8 need to be installed separately
- Current support for NuGet packages requires .NET 6.0 or later
- Version 18.2 was shipped configured for .NET 3.1, but seems to work fine with 6.0-8.0
- The v19.0 .NET bridge is significantly more mature (stable/complete) than 18.2



Home / Download

.NET Release Candidate (RC) Want to try out the latest RC release? .NET 8.0.0-rc.1 is available. Get .NET RC

Free. Cross-platform. Open source.

Download .NET For Windows

.NET 7.0

Standard Term Support Recommended

.NET SDK x64

Version 7.0.11, released September 12, 2023

[All .NET 7.0 downloads](#) [All .NET versions](#)

.NET 6.0

Long Term Support

.NET SDK x64

Version 6.0.22, released September 12, 2023

[All .NET 6.0 downloads](#)

Feedback



Download .NET 8.0

Not sure what to download? [See recommended downloads for the latest version of .NET.](#)

8.0.0-rc.1

Go-live

Security patch

[Release notes](#) Latest release date September 14, 2023

Build apps - SDK

SDK 8.0.100-rc.1

OS	Installers	Binaries
Linux		Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine
macOS	Arm64 x64	Arm64 x64
Windows	Arm64 x64 x86 winget instructions	Arm64 x64 x86
All	dotnet-install scripts	

Full version

8.0.100-rc.1.23463.5

Visual Studio support

Run apps - Runtime

ASP.NET Core Runtime 8.0.0-rc.1

The ASP.NET Core Runtime enables you to run existing web/server applications. **On Windows, we recommend installing the Hosting Bundle, which includes the .NET Runtime and IIS support.**

Full version

8.0.0-rc.1.23421.29

IIS runtime support (ASP.NET Core Module v2)

18.0.23234.0

OS	Installers	Binaries
Linux		Arm32 Arm32 Alpine Arm64 Arm64 Alpine x64 x64 Alpine

Feedback



Configure APL to use .NET

- ◆ ... if you want to use NuGet packages later ...



Pick an Installed .NET Runtime

Pick one
of these



```
Command Prompt
C:\>dotnet --list-runtimes
Microsoft.AspNetCore.App 3.1.32 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 6.0.22 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.AspNetCore.App 8.0.0-rc.1.23421.29 [C:\Program Files\dotnet\shared\Microsoft.AspNetCore.App]
Microsoft.NETCore.App 3.1.32 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 6.0.22 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 8.0.0-rc.1.23419.4 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.WindowsDesktop.App 3.1.32 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 6.0.22 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]
Microsoft.WindowsDesktop.App 8.0.0-rc.1.23420.5 [C:\Program Files\dotnet\shared\Microsoft.WindowsDesktop.App]

C:\>|
```

(Even Microsoft sometimes still calls it "Core" 😊)



DYALOG_NETCORE=1

- Under Windows, APL will use the .NET Framework by default
- Set `DYALOG_NETCORE=1` to switch from Framework to ".NET"
 - Environment Variable, Command Line, Registry or Config File
- Unfortunately, selecting the VERSION of .NET to use requires editing `.json` files in the main Dyalog folder (see next slide)
 - Version 18.2 defaults to .NET Core (3.1)
 - Version 19.0 will default to 8.0
 - "User Meeting Edition" defaults to 6.0 because 8.0 is a Release Candidate



How to switch to using .NET 6.0 with 18.2 on Linux

Forum rules

The FAQ is a read-only forum which is in general updated only by employees of Dyalog Ltd. It replaces the FAQ page which existed under <http://www.dyalog.com>. Rather than rejecting other posts to this forum, such posts will be put in a moderation queue, and moved to a more appropriate forum.

POSTREPLY ↩

Search

1 post • Page 1 of 1

HOW TO SWITCH TO USING .NET 6.0 WITH 18.2 ON LINUX

by [VinceDyalog](#) on Tue Aug 16, 2022 9:40 am

by JohnDJDyalog on Fri Aug 05, 2022 3:56 pm in this thread "SUPPORT FOR MICROSOFT .NET 6.0"
<https://forums.dyalog.com/viewtopic.php?f=20&t=1858>

We will be supporting .NET 6.0 "out of the box" with the next version of Dyalog, 19.0 (I cannot say yet when that will be available).

Until then it should be possible to move to .NET 6.0 by making minor changes to some of Dyalog's configuration files. We cannot formally support this route, but I don't think there will be any problems.

In the dyalog installation you will see files called:

Dyalog.Net.Bridge.deps.json
Dyalog.Net.Bridge.runtimeconfig.json

Each of these files contains version strings which will include the fragment "3.1". Change those "3.1"s to "6.0" and you should be good to go.

The following are the changes I made:

```
Dyalog.Net.Bridge.deps.json: "name": ".NETCoreApp,Version=v6.0",  
Dyalog.Net.Bridge.deps.json: ".NETCoreApp,Version=v6.0": {  
Dyalog.Net.Bridge.runtimeconfig.json: "tfm": "netcoreapp6.0",  
Dyalog.Net.Bridge.runtimeconfig.json: "version": "6.0.0"
```

Best Regards
John Daintree.

POSTREPLY ↩

1 post • Page 1 of 1



Tatin Installing and Up | install .net - Search | dyalog forums - Search | Dyalog Forums • View

https://forums.dyalog.com/viewtopic.php?f=22&t=1863

How to switch to using .NET 6.0 with 18.2 on Linux

Forum rules
The FAQ is a read-only forum which is in general updated only by employees of Dyalog Ltd. Posts that are not in accordance with the forum rules, such as rejecting other posts to this forum, such posts will be put in a moderation queue, and will not be visible to other users.

POSTREPLY | Search this topic... | Search

HOW TO SWITCH TO USING .NET 6.0 WITH 18.2 ON LINUX

by VinceDyalog on Tue Aug 16, 2022 9:40 am

by JohnDJDyalog on Fri Aug 05, 2022 3:56 pm in this thread "SUPPORT FOR .NET 6.0 ON LINUX" <https://forums.dyalog.com/viewtopic.php?f=20&t=1858>

We will be supporting .NET 6.0 "out of the box" with the next version of Dyalog that will be available).

Until then it should be possible to move to .NET 6.0 by making minor changes to the configuration files. We cannot formally support this route, but I don't think that will be a problem.

In the dyalog installation you will see files called:

- Dyalog.Net.Bridge.deps.json
- Dyalog.Net.Bridge.runtimeconfig.json

Each of these files contains version strings which will include the fragment "6.0" and you should be good to go.

The following are the changes I made:

```
Dyalog.Net.Bridge.deps.json: "name": ".NETCoreApp,Version=v6.0",
Dyalog.Net.Bridge.deps.json: ".NETCoreApp,Version=v6.0": {
Dyalog.Net.Bridge.runtimeconfig.json: "tfm": "netcoreapp6.0",
Dyalog.Net.Bridge.runtimeconfig.json: "version": "6.0.0"
```

Best Regards
John Daintree.

POSTREPLY

```
Dyalog.Net.Bridge.deps.json | Dyalog.Net.Bridge.runtimeconfig.json
```

```
{
  "runtimeTarget": {
    "name": ".NETCoreApp,Version=v8.0",
    "signature": ""
  },
  "compilationOptions": {},
  "targets": {
    ".NETCoreApp,Version=v8.0": {
      "Dyalog.Net.Bridge/1.0.0": {
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

```
Dyalog.Net.Bridge.deps.json | Dyalog.Net.Bridge.runtimeconfig.j
```

```
{
  "runtimeOptions": {
    "tfm": "net8.0",
    "framework": {
      "name": "Microsoft.NETCore.App",
      "version": "8.0.0-rc.1.23419.4"
    }
  },
  "configProperties": {
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

Updating Cider & Tatin

```
]Cider.UpdateCider -?
```

Attempts to update the currently running version of Cider

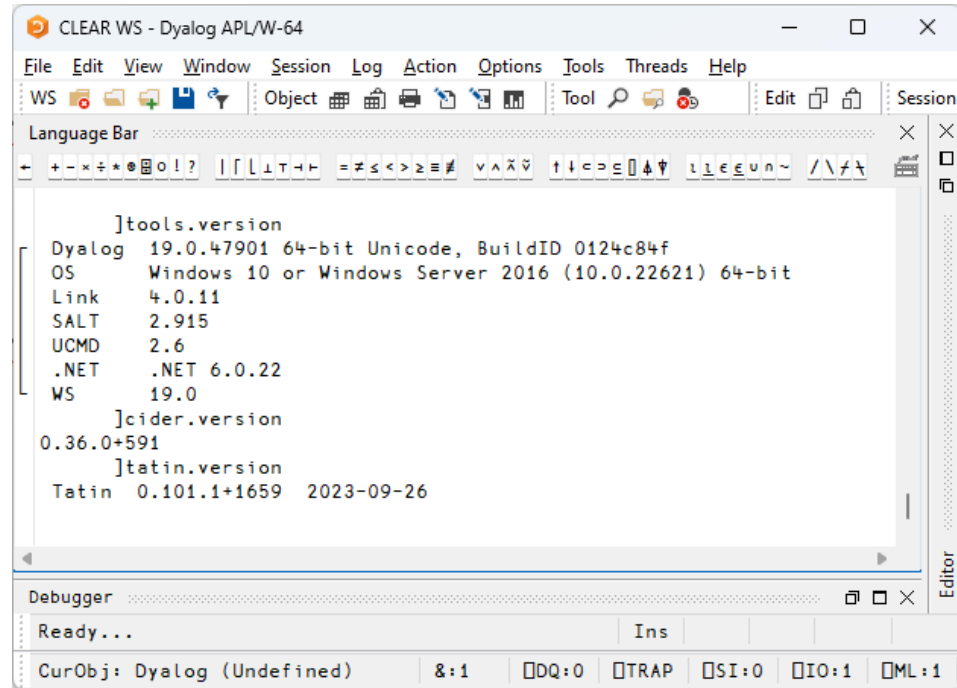
```
]Tatin.UpdateTatin -?
```

Attempts to update the Tatin client and reports the result



Exercise 0

- Install and Verify
 - Tatin
 - Cider
 - .NET
- Set APL up to use .NET



```

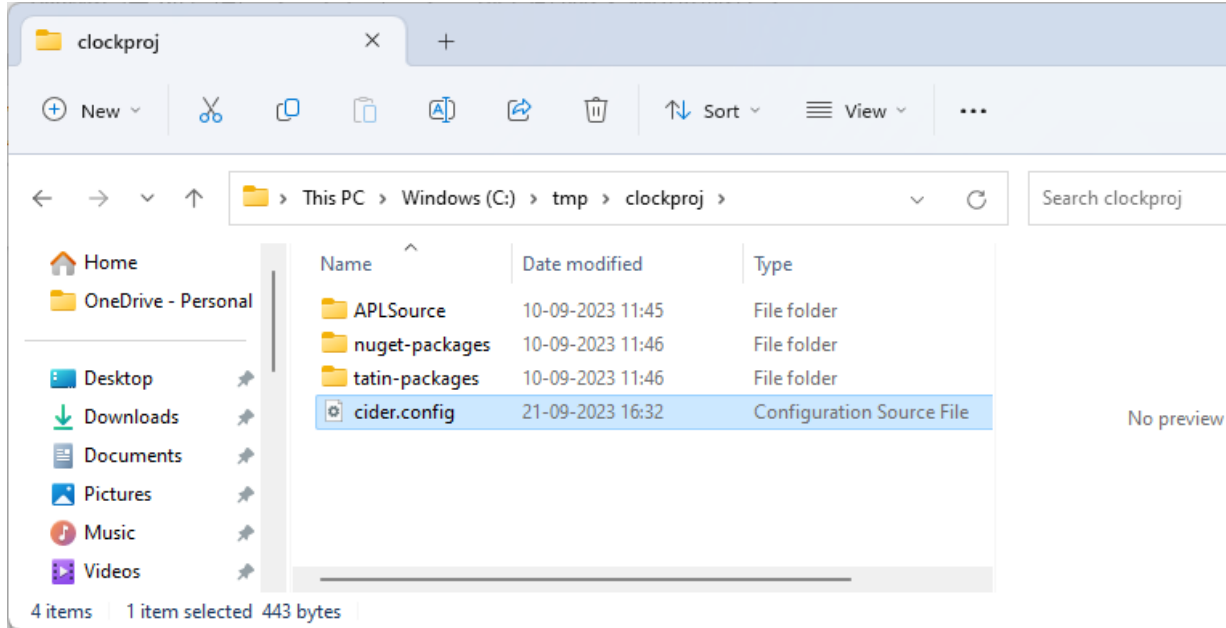
]tools.version
Dyalog 19.0.47901 64-bit Unicode, BuildID 0124c84f
OS Windows 10 or Windows Server 2016 (10.0.22621) 64-bit
Link 4.0.11
SALT 2.915
UCMD 2.6
.NET .NET 6.0.22
WS 19.0
]cider.version
0.36.0+591
]tatin.version
Tatin 0.101.1+1659 2023-09-26

```



Back to Cider...

What *is* a Cider Project?



```
cider.config
File Rediger Vis
{
  CIDER: {
    dependencies: {
      nuget: "nuget-packages",
      tatin: "tatin-packages",
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockproj",
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
Ln 1, Col 1 | 80% | Windows (CRLF) | UTF-8
```

Loading Code

source ("APLSource")

identifies the sub-folder to be loaded
(ONLY this folder will be loaded into the WS)

parent ("#")

the location source will be loaded into

projectSpace (default is project folder name)
name of the space to be created within parent

- These three parameters decide **what** will be loaded, and **where** it will go within the workspace
- You can override the last two using **-parent** and **-projectSpace** modifiers

(default settings in parentheses)

```
cider.config
File Rediger Vis
{
  CIDER: {
    dependencies: {
      nuget: "nuget-packages",
      tatin: "tatin-packages",
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockproj",
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
Ln 1, Col 1 | 80% | Windows (CRLF) | UTF-8
```

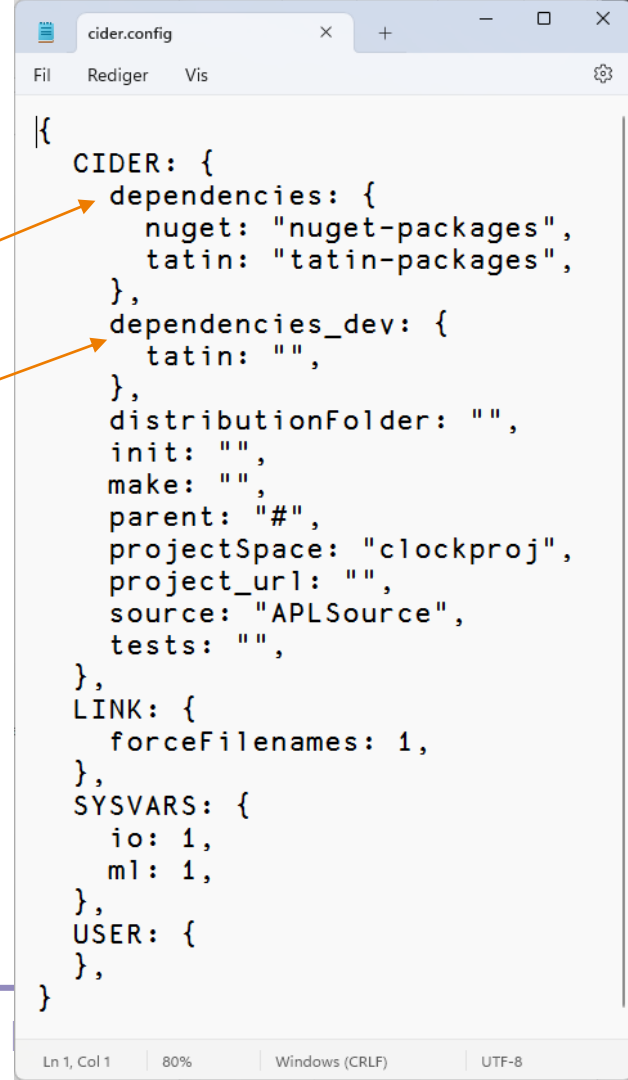
Dependencies

dependencies ("nuget-packages" and "tatin-packages")
names of folders that will contain the dependencies

dependencies_dev ("")
name of a folder that may contain Tatin dependencies used only during development

- All dependencies are loaded when the project is opened
- Each folder name can be followed by **=targetns** if dependencies should be loaded somewhere else than projectSpace (typically done for **dependencies_dev**)

For example `tatin: "dev-packages=devtools"`



```
{
  CIDER: {
    dependencies: {
      nuget: "nuget-packages",
      tatin: "tatin-packages",
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockproj",
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
```

Running Code

`init ("")`

An APL expression to run on project open.

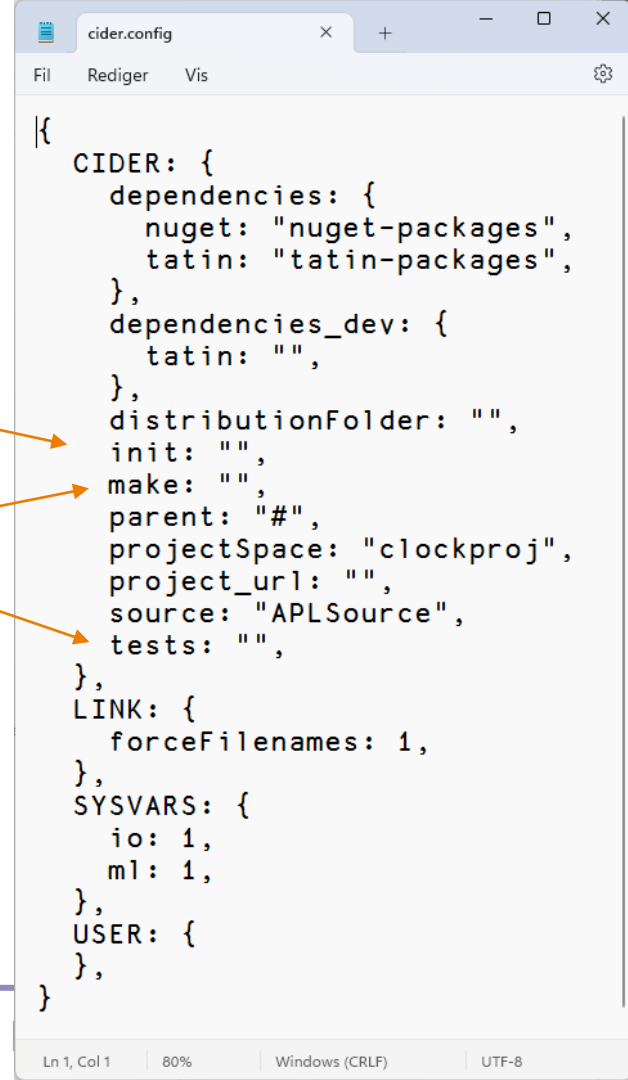
`tests ("")`

An APL expression to run your tests.

`make ("")`

An APL expression to launch your build process

- Cider **will run** the "init" expression on project load
- Cider **will not** currently run your tests or build processes
- User commands `]cider.make &]cider.runtests` will **display** the settings, but leave it up to you to run things



```
cider.config
Fil Rediger Vis
{
  CIDER: {
    dependencies: {
      nuget: "nuget-packages",
      tatin: "tatin-packages",
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockproj",
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
Ln 1, Col 1 | 80% | Windows (CRLF) | UTF-8
```

Link Options and Sysvars

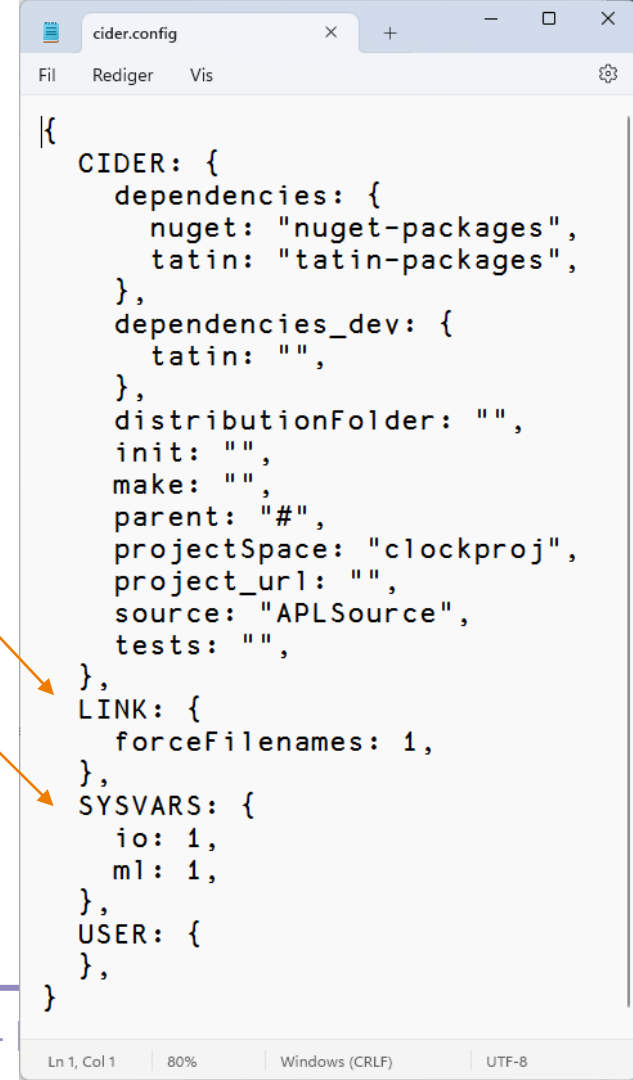
LINK section

Includes any non-default Link options that you want set when your source is linked

SYSVARS section

Allows you to declare values for system variables

- Both of these sections are now unnecessary due to Link enhancements:
 - v3 added support for system variables
 - V4 uses a .linkconfig file to store non-default options within the source folder (in this case, APLSource/.linkconfig)
- However, they arguably still have value as documentation of important aspects of your project configuration



```
cidr.config
Fil Rediger Vis
{
  CIDER: {
    dependencies: {
      nuget: "nuget-packages",
      tatin: "tatin-packages",
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockproj",
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
Ln 1, Col 1 | 80% | Windows (CRLF) | UTF-8
```

"Convenience" options

`distributionFolder` ("")

The target for a build process

`project_url` ("")

A pointer to where the project is hosted, especially if it is on GitHub.

- You can use these in your build functions and documentation.
- When using Cider to develop Tatin packages, atin's "BuildPackage" acts on **distributionFolder** and will include **project_url** in the package description

```
cider.config
x + - □ x
Fil Rediger Vis
{
  CIDER: {
    dependencies: {
      nuget: "nuget-packages",
      tatin: "tatin-packages",
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockproj",
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
Ln 1, Col 1 | 80% | Windows (CRLF) | UTF-8
```

[Global] Cider Configuration

- ◆ Cider also has a **global** configuration file
- ◆ At the moment, this is "work in progress", and the only configuration option stored here is...
 - ◆ **ExecuteAfterprojectOpen:** a string which is executed after opening ALL projects



[Global] Cider Configuration

- Two functions are provided to access the global configuration file:

```
SE.Cider.GetCiderGlobalConfigFilename
C:\Users\mkrom\.cider\config.json

>NGET SE.Cider.GetCiderGlobalConfigFilename
{
  ExecuteAfterProjectOpen: "",
}

gc<SE.Cider.GetCiderGlobalConfigFileContent
≠gc.ExecuteAfterProjectOpen
0
```



Aliases



- Aliases provide a way to avoid typing long folder names over and over again

```
fldr← 'c:\tmp\clockproj' A Long, annoying folder name
]SE.Cider.AddAlias folder 'clocks'
]SE.Cider.GetAliasFileContent ''
clocks c:/tmp/clockproj
]cider.openproject '[clocks]'
```

```
...
Project successfully loaded and established in "#.clockproj"
```

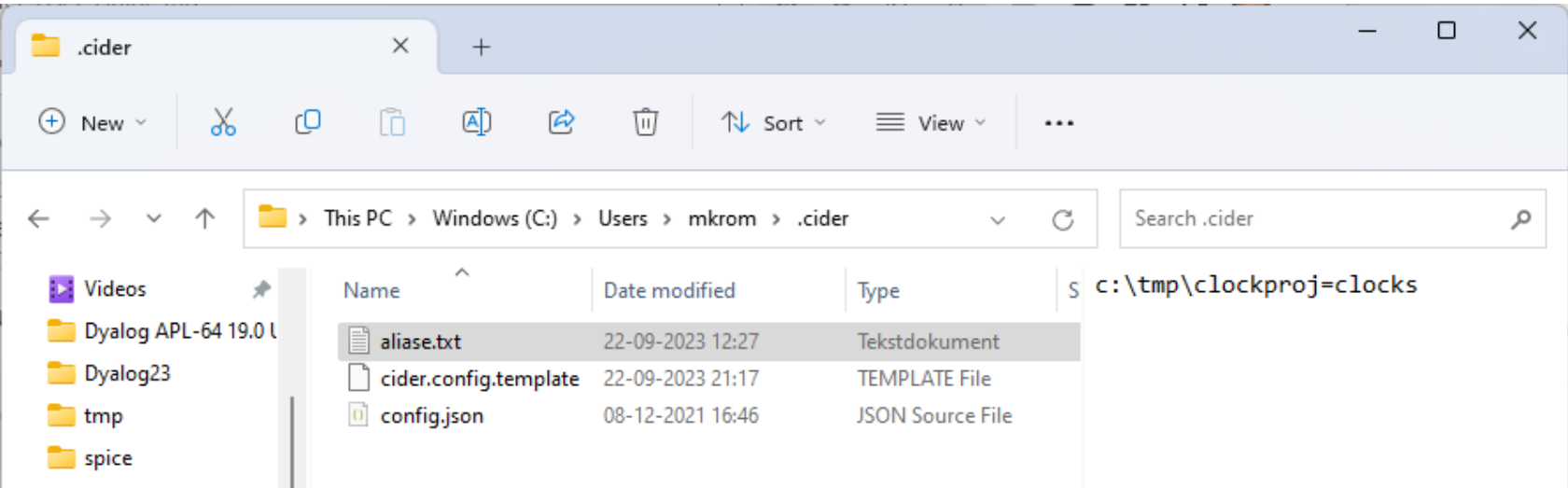
```
NB. ]Cider.ListAliases -edit
```



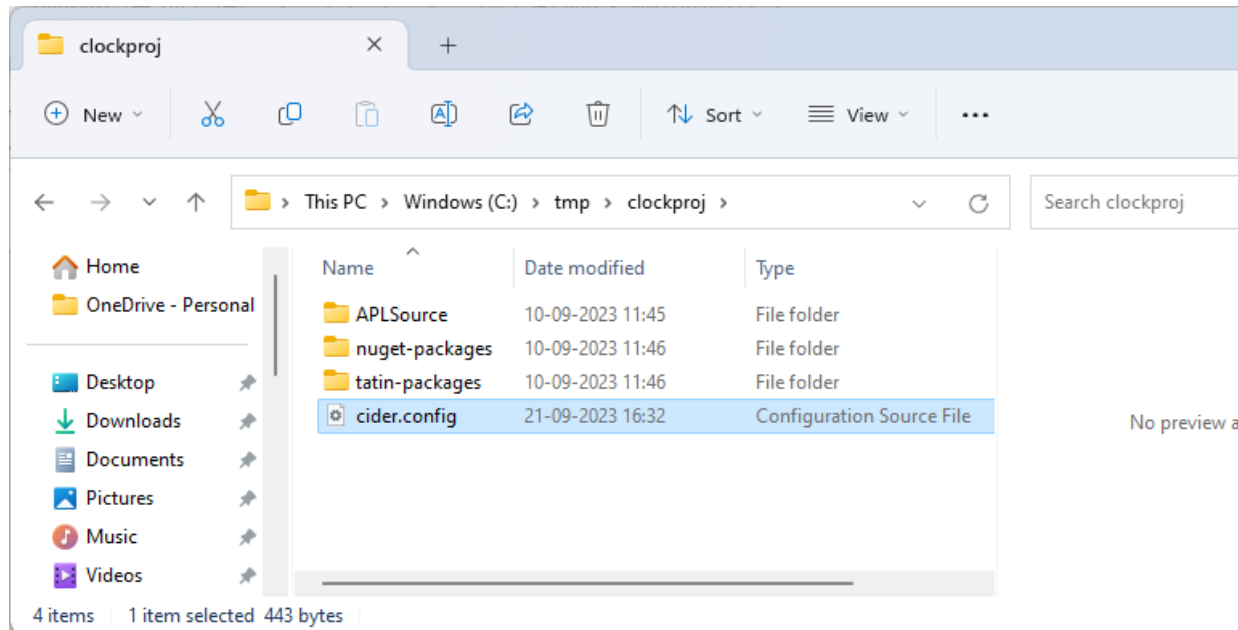
The [HOME]/.cider Folder

Contains:

- Global config file config.json (ExecuteAfterProjectOpen)
- Alias file aliase.txt
- Template config.json file to be used when creating new projects



Back to the Cider Project



A screenshot of an IDE window titled 'OSE.edit.cider_config'. The window shows the contents of the 'cider.config' file, which is a JSON configuration file. The code is as follows:

```
{
  CIDER: {
    dependencies: {
      nuget: "nuget-dependencies",
      tatin: "tatin-dependencies",
    },
    dependencies_dev: {
      tatin: "tatin-dependencies_dev",
    },
    distributionFolder: "Dist",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "anotherproject",
    project_url: "",
    source: "APLSource",
    tests: "",
    version: "",
  },
  LINK: {
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
```

Creating a Project

```
]Cider.CreateProject c:\tmp\anotherproject  
"c:/tmp/anotherproject" does not exist yet - create? (Y/n) y
```

```
Project successfully created; open as well? (Y/n) y
```

```
LINK:watch=both
```

```
Project successfully loaded and established in "#.anotherproject"
```

```
#.anotherproject.[]nl -i10
```

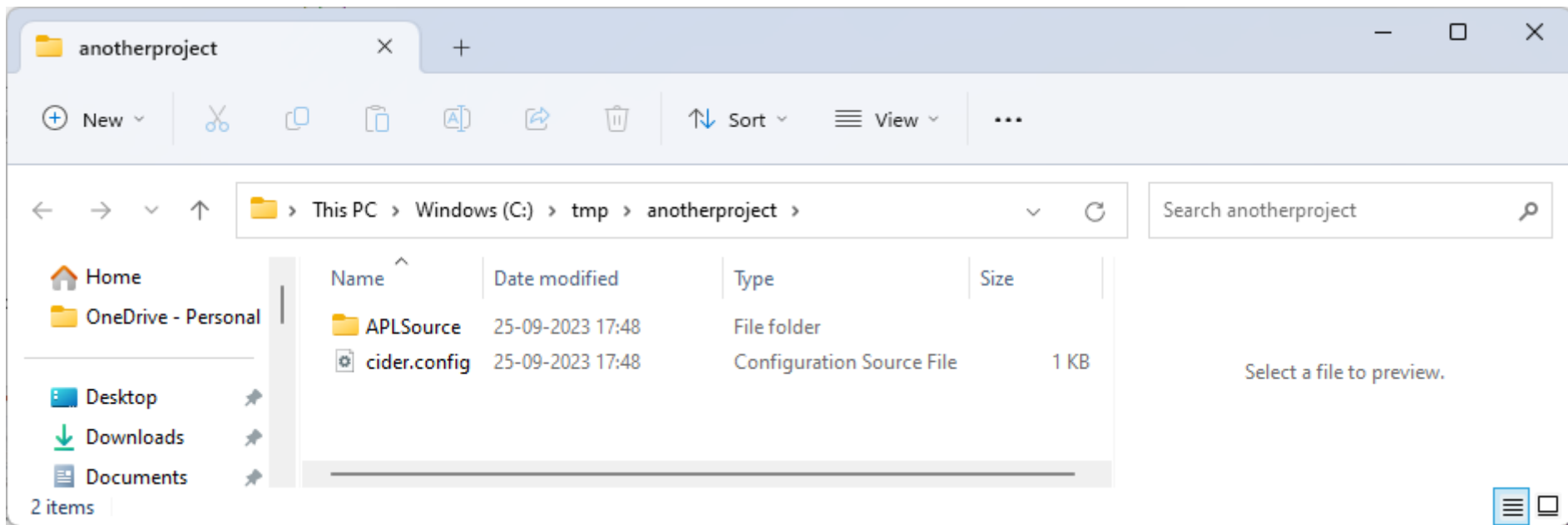
```
CiderConfig
```



```
File Edit Syntax Refactor View  
Search...  
{  
  CIDER: {  
    dependencies: {  
      nuget: "nuget-dependencies",  
      tatin: "tatin-dependencies",  
    },  
    dependencies_dev: {  
      tatin: "tatin-dependencies_dev",  
    },  
    distributionFolder: "Dist",  
    init: "",  
    make: "",  
    parent: "#",  
    projectSpace: "anotherproject",  
    project_url: "",  
    source: "APLSource",  
    tests: "",  
    version: "",  
  },  
  LINK: {  
  },  
  SYSVARS: {  
    io: 1,  
    ml: 1,  
  },  
  USER: {  
  },  
}
```

Nested Array (coloured as JSON)

The New Project Folder



Cider Documentation

```
]Cider.Help  
--- Select document to be viewed: -----  
1. Cider-API-Reference  
2. Cider-User-Guide  
3. Contributing  
  
Select one or more items (q=quit, a=all) : 2
```

A screenshot of a web browser displaying the 'Cider User Guide' table of contents. The browser's address bar shows the file path: C:/Users/mkrom/Documents/Dyalog%20APL-64%2019.0%20U... The page title is 'Cider User Guide'. The table of contents includes:

- [1. What is Cider for?](#)
- [2. The solution](#)
 - [2.1. Requirements](#)
 - [2.1.1. APL Version](#)
 - [2.1.2. Tatin](#)
 - [2.1.3. Git](#)
 - [2.2. Installation](#)
 - [2.2.1. Version 19.0](#)
 - [2.2.2. Version 18.0 and 18.2](#)
 - [2.3. Upgrading Cider](#)
 - [2.4. Configuration](#)
 - [2.4.1. Global configuration](#)
 - [2.4.2. Project configuration](#)
 - [2.5. CreateProject](#)
 - [2.5.1. Configuration parameters](#)
 - [2.5.1.1. CIDER](#)
 - [2.5.1.1.1. distributionFolder](#)
 - [2.5.1.1.2. parent](#)

CLEAR WS - Dyalog APL/W-64

File Edit View Window Session Log Action Options Tools Threads Help

WS Object Tool Edit Session APL385 Unicode 16

Language Bar

```

]cider -?

CIDER:
AddNuGetDependencies  Add one or more NuGet packages as dependencies
AddTatinDependencies  Add one or more Tatin packages as dependencies
CloseProject          Breaks the Link between the workspace and the files on disk
Config                Allow the user to edit Cider's config file
CreateProject         Makes the given folder a project folder
Help                  Offers to put the HTML files on display
ListAliases           List all defined aliases with their folders
ListNuGetDependencies List NuGet packages installed as dependencies
ListOpenProjects      List all currently open projects
ListTatinDependencies List Tatin packages installed as dependencies
Make                  Build a new version of the project
OpenProject           Load all source files into the WS and keep it linked by default
ProjectConfig         Puts the config file of a project on display
RunTests              Executes the project's test suite (if any)
UpdateCider           Tries to update Cider
Version               Returns name, version number and version date as a three-element vector

]                A for general user command help
]grp -?          A for info on the "GRP" group
]grp.cmd -?     A for info on the "Cmd" command of the "GRP" group

```

Debugger

Ready...

CurObj: lSIGN(Undefined) &:1 □DQ:0 □TRAP □SI:0 □IO:1 □ML:1

Editor



CLEAR WS - Dyalog APL/W-64

File Edit View Window Session Log Action Options Tools Threads Help

WS Object Tool Edit Session APL385 Unicode 16

Language Bar

]CIDER.CreateProject -??

]CIDER.CreateProject

Requires a path to a folder ("source") that is about to become a project.

You might also specify a namespace. If no namespace is provided then the name of the namespace is derived from the path. If the namespace does not yet exist it will be created. The namespace will be LINKed to "source", and a namespace CiderConfig, holding the configuration data, will be injected into the namespace.

For a root project (the whole of #, NOT recommended) you must specify # as 2nd argument.

- * Creates a file "cider.config" in that folder
- * Lets the user edit that file and makes sure that all mandatory settings are specified correctly
- * In case an alias is specified the alias is saved
- * Finally it attempts to open the new project

If no path is specified it acts on the current directory, but in that case the user is prompted for confirmation to avoid mishaps.

-acceptConfig: By default a file cider.config is created, and an error is thrown in case it already exists. You can use -acceptConfig to force CreateProject to accept an already existing config file.

-noEdit: With -noEdit you can prevent the user from being asked to edit the config file.

-alias: In case you are going to work on the new project frequently you may specify -alias=name

-batch After a project has been created successfully, the user will be asked whether she wants to open the project as well. You can enforce that without the user being interrogated by setting the -batch flag. Mainly useful for test cases and possibly an automated build process.

-ignoreUserExec Suppress execution of a user function defined in Cider's config file on this occasion

Note that the alias is not case sensitive

Debugger Ready... Ins

CurObj: LSIGN (Undefined) &:1 DDQ:0 PTRAP ISI:0 IO:1 OML:1

Editor



Exercise 1

If you get a FILE NAME ERROR, delete
C:\Users\<>AN>\.cider\cider.config.template

- Create a Project
- Give it an Alias
- Create a function to tell you how many functions you have in your project.
- Configure the project to run that function on Open
- Run]Cider.Help and take a brief look at the documents



Opening a Project

```
]CIDER.OpenProject /folder/name
```

```
or ]CIDER.OpenProject [alias]
```

```
or ns←[]SE.Cider.CreateOpenParms ''  
  ns.folder←'/tmp/clockproj'  
  ns.projectSpace←'cp'
```

```
[]SE.Cider.OpenProject ns
```

```
...blabla...
```

```
Project successfully loaded and established in "#.cp"
```



Switches (from]OpenProject -??)

-parent	The project is loaded into Cider.(parent.projectSpace) unless this is (temporarily) overwritten by setting the -parent= and/or the -projectSpace= option(s).
-projectSpace	The project is loaded into Cider.(parent.projectSpace) unless this is (temporarily) overwritten by setting the -projectSpace= and/or the -parent= option(s).
-import	By default the namespace is linked to folder. By specifying the -import flag this can be avoided: the code is then loaded into the workspace with the Link.Import method, meaning that changes are not tracked. With -import the status of neither Git nor any installed Tatin packages is checked.
-alias=	In case you are going to work on a project frequently you may specify -alias=name for quicker access. (In order to remove an alias use]Cider.ListAliases -edit)
-noPkgLoad	By default the Tatin packages from the installation folder(s) defined in the config will be loaded. If you don't want this specify -noPkgLoad
-watch	Defaults to "both" (if .NET is available) but may be "ns" or "dir" instead. * "ns" means that changes in the workspace are saved on disk * "dir" means that any changes on disk are brought into the WS * "both" means that any changes in either "ns" or "dir" are reflected accordingly However, currently "both" works only under Windows. Note that the flag does NOT change the config file on disk Refer to the Link documentation for details.
-verbose	If this is specified then Cider reports every single step it carries out.
-ignoreUserExec	Suppress execution of a user function defined in Cider's config file once
-batch	Does not print to the session and prevents user interaction. In a situation were user must decide what to do an error is thrown.



What Happens on Project Open?

1. Create a namespace according to within the specified parent space
2. Set `IO` and `ML` according to `ciderconfig` `SYSVARS` section
3. Use `Link` to load all the source code from the folder named by `source`
4. Load Tatin and/or NuGet packages (more about this soon)
5. Inject `CiderConfig` namespace containing the config file settings
6. `CiderConfig.HOME` is set to the path the project was loaded from
7. Execute `CiderConfig.init` if it is non-empty
8. Execute `ExecuteAfterprojectOpen` (from `GlobalConfig`) if non-empty and `ignoreUserExec` is not 1
9. Display the contents of `ToDo` variable, if it exists
10. *If* the project folder is managed by Git, display the result of "git status"



Exercise 2

- ◆)CLEAR and open your project

1. Using the User Command

2. Using

```
ns←[]SE.Cider.CreateOpenParms ''
ns.folder←'blah'
ns.someprop←'somevalue'
[]SE.Cider.OpenProject ns
```

- ◆ Create a ToDo variable which reminds you to add HttpCommand as a dependency

- ◆ Use]Link.Add to save yourproj.ToDo to file

- ◆ NB: Unlike functions and operators, Link does NOT save new variables by default

- ◆ Close and reopen your project



Session 2: Dependencies / Packages

- What is a Package?
- **Tatin:** APL Packages
 - Finding Documentation
 - Finding and Installing Packages
 - "Where do they go?"
- Dependencies of Dependencies
- **NuGet:** .NET Packages
 - Finding and Installing
 - "Where do they go?"

Exercise 3:
Add a Tatin Dependency (or two)!

Exercise 4:
Add a NuGet Dependency



So... What is a Package?



(From Longman Dictionary of Contemporary English)



A Project is...

Source Code +

- ◆ Dependencies (packages) loaded from a package manager
- ◆ Environment configuration
- ◆ Development tools and processes
- ◆ Can be opened and "set up" by a Project Manager (Cider)

A Package is...

A "build" of a project...

- ◆ In a standard format
- ◆ Can be **found, downloaded** and **installed** by a "Package Manager"
- ◆ Cider supports the development of Tatin Packages
- ◆ Cider can load Tatin + NuGet Packages



Tatin



Package manager for Dyalog APL

A tasty way to package APLs

48 Packages

NuGet



Package manager for .NET

Related to "Chocolatey"

~~371,905~~ 374,154 Packages



```
]z←tatin.listPackages
{α,≠ω}⊔{(-1+ωι'-')↑ω}¨3↓z[;1]
aplteam 42
davin 4
dyalog 2
```

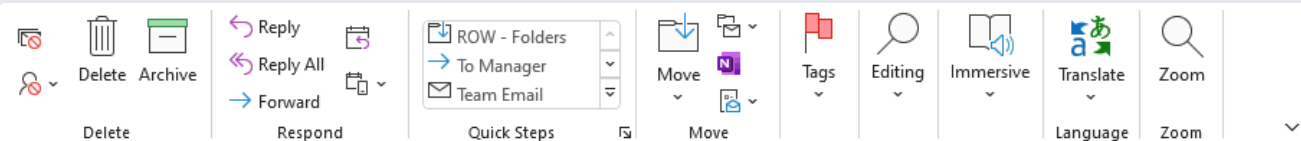
```
¨2↑z
dyalog-HttpCommand 1
dyalog-Jarvis 1
```



Introducing Tatin

- ◆ Original design by Kai Jaeger and Gilgamesh Athoraya
- ◆ Developed by Kai (first lines of code written in 2020)
- ◆ Funded by Dyalog
- ◆ Input from various people at Dyalog
- ◆ Logo by Adam Brudzewsky
- ◆ Many thanks to Davin Church, the first real user of the system other than Kai himself
- ◆ Paul Mansour is not to blame for the current design of Tatin, but has been an important inspiration

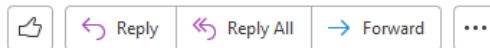


File Message Help

RE: pakkman - the name



Morten Kromberg <mkrom@dyalog.com>
To 'Kai Jaeger'



on 15-07-2020 13:12

You replied to this message on 15-07-2020 14:32.

Andy points out that "tart" has unfortunate connotations in English.

How about "Tatin", which is a nice kind of tart. In the readme, we explain the origins of the name. We can use "aplt.get", similar to "apt-get" for the names of things that get stuff from it?

It is nice to use names that are not acronyms.

-----Original Message-----

From: Kai Jaeger <kai@aplteam.com>
Sent: 15. juli 2020 11:00
To: Morten Kromberg <mkrom@dyalog.com>
Subject: Re: pakkman - the name

And it can be pronounced apple-tart, right?

Much better than FDAPM!

On Wed, 15 Jul 2020 at 08:19, Morten Kromberg <mkrom@dyalog.com> wrote:

>

> So, while shaving, the name "apltart" popped into my head. An APL tart is (obviously) an attractive way to package APLs.

>



- RE
- M
- Info
- Main page
- Contents
- Current events
- Random article
- About Wikipedia
- Contact us
- Donate
- Contribute
- Help
- Learn to edit
- Community portal
- Recent changes
- Upload file
- Tools
- What links here
- Related changes
- Special pages
- Permanent link
- Page information



WIKIPEDIA
The Free Encyclopedia

Not logged in [Talk](#) [Contributions](#) [Create account](#) [Log in](#)

Article [Talk](#)

Read [Edit](#) [View history](#)

Search Wikipedia

Tarte Tatin

From Wikipedia, the free encyclopedia

The **tarte Tatin** (French pronunciation: [taʁt taˈtɑ̃]), named after the Tatin sisters who invented it and served it in their hotel as its signature dish, is a [pastry](#) in which the fruit (usually apples) is [caramelized](#) in butter and sugar before the tart is baked. It originated in [France](#) but has spread to other countries over the years.

- Contents** [\[hide\]](#)
- 1 [History](#)
 - 2 [Variations](#)
 - 3 [See also](#)
 - 4 [References](#)
 - 5 [External links](#)

History [\[edit\]](#)

The tarte Tatin was created accidentally at the Hôtel Tatin in [Lamotte-Beuvron, Loir-et-Cher](#), 169 km (105 mi) south of [Paris](#), in the 1880s.^[1] The hotel was run by two sisters, Stéphanie and Caroline Tatin.^[2] There are conflicting stories concerning the tart's origin, but the most common is that Stéphanie Tatin, who did most of the

Tarte Tatin



Type	Tart
Place of origin	France
Region or state	Centre-Val de Loire
Created by	Tatin sisters
Main ingredients	Apples or other fruits

[Cookbook: Tarte Tatin](#)
 [Media: Tarte Tatin](#)

Finding Packages – www.tatin.dev



Tatin Registry

List of packages

Package name	Description	Major Versions	Project URL	OS	UC	Tags
aplteam-ADOC	Automated generation od documentation	1	github.com	Lin, Mac, Win	Yes	documentation
aplteam-APLGit2	Git interface from Dyalog APL via Git Bash	1	github.com	Lin, Mac, Win	Yes	apl-git-interface
aplteam-APLGUI	Collection of GUI utilities	1	github.com	Win		gui-tools,gui
aplteam-APLProcess	Start an APL process from within Dyalog APL	1	github.com	Lin, Mac, Win		process
aplteam-APLTreeUtils2	General utilities required by most members of the APLTree library	1	github.com	Lin, Mac, Win		tools,utilities
aplteam-Cider	A project manager for Dyalog APL that cooperates with Tatin	1	github.com	Lin, Mac, Win	Yes	project-management
aplteam-CodeBrowser	Tool useful for code reviews	1	github.com	Lin, Mac, Win	Yes	code-browsing,code-reviews
aplteam-CodeCoverage	Monitors which parts of an application got actually executed	1	github.com	Lin, Mac, Win		code-coverage,test-framework,unit-tests
aplteam-CommTools	Communication tools for interactions in the session: YesOrNo and Select	1	github.com	Lin, Mac, Win		communication-tools,yes-or-no,select-tool
aplteam-Compare	Allows comparing and merging objects in the workspace with a file or a file with another file	3	github.com	Lin, Mac, Win		comparison-tool,merge-tool
aplteam-CompareFiles	Cover for comparison utilities	1	github.com	Lin, Mac, Win	Yes	file-comparison,comparison-utilities

Finding Packages



Tatin Registry

List of packages


zip

Package name	Description	Major Versions	Project URL	OS	UC	Tags
aplteam-DotNetZip	Zippping and unzipping with .NET Core on all major platforms	2	github.com	Win		zip-tools
aplteam-SevenZip	Zip files with the Open Source tool 7Zip	1	github.com	Win		zip-tools
aplteam-ZipArchive	Zippping and unzipping with .NET on Windows and zip/unzip on other platforms	2	github.com	Lin, Mac, Win		zip-tools

Created by Tatin version 0.100.1+1627 from 2023-08-28 under Linux-64 18.2.45645.0 S Runtime — Bugs, questions, problems: info@tatin.dev



Finding Packages

 **Tatin Registry**

List of packages

Package name	Description	Major Versions	Project URL	OS	UC	Tags
aplteam-APLProcess	Start an APL process from within Dyalog APL	1	github.com	Lin, Mac, Win		process
aplteam-Execute	Start a process from within APL	1	github.com	Win		process,run-applications

We **already** have enough packages to (sometimes) make it difficult to decide which one to use (and dyalog-APLProcess yet to come 😊)



Package Details



All versions of "aplteam-CodeCoverage-0"

Package ID	Publishing date	Config	Dependencies
<code>aplteam-CodeCoverage-0.10.3</code>	2023-06-23 18:46:15	Show	None
<code>aplteam-CodeCoverage-0.10.2</code>	2023-04-09 14:32:41	Show	None
<code>aplteam-CodeCoverage-0.10.0</code>	2023-03-23 18:27:55	Show	Show
<code>aplteam-CodeCoverage-0.9.4</code>	2022-12-12 17:27:32	Show	None
<code>aplteam-CodeCoverage-0.9.3</code>	2022-10-06 13:43:27	Show	None
<code>aplteam-CodeCoverage-0.9.2</code>	2022-09-19 06:54:57	Show	None
<code>aplteam-CodeCoverage-0.9.1</code>	2021-10-29 08:33:42	Show	None
<code>aplteam-CodeCoverage-0.9.0</code>	2021-10-07 17:28:11	Show	None
<code>aplteam-CodeCoverage-0.7.3</code>	2021-07-26 09:34:03	Show	None
<code>aplteam-CodeCoverage-0.7.2</code>	2021-03-04 20:33:32	Show	None



Details of <aplteam-CodeCoverage-0.10.0>

```
{
  api: "CodeCoverage",
  assets: "",
  date: 20230323.182755,
  description: "Monitors which parts of an application got actually executed",
  documentation: "",
  files: "",
  group: "aplteam"
  io: 1,
  license: "MIT",
  lx: "",
  maintainer: "kai@aplteam.com",
  minimumApiVersion: "18.0",
  ml: 1,
  name: "CodeCoverage",
  os_lin: 1,
  os_mac: 1,
  os_win: 1,
  project_url: "https://github.com/aplteam/CodeCoverage",
  source: "APLSource/CodeCoverage.aplc",
  tags: "code-coverage,test-framework,unit-tests",
  userCommandScript: "",
  version: "0.10.0+55",
}
```



Tatin Registry

Dependencies of "aplteam-CodeCoverage-0.10.0"

Dependencies

[aplteam-APLTreeUtils2-1.1.3](#)

[aplteam-Tester2-3.3.1](#)

[aplteam-CommTools-1.3.0](#)



]Tatin.ListPackages

```
]Tatin.ListPackages -group=dyalog
```

```
Registry: https://tatin.dev
```

Group & Name	# major versions
-----	-----
dyalog-HttpCommand	1
dyalog-Jarvis	1

```
]Tatin.ListPackages -tag=crypto
```

```
Registry: https://tatin.dev
```

Group & Name	# major versions
-----	-----
aplteam-HashPasswords	1

```
]tatin.listtags
tags from https://tatin.dev
-----
apl-git-interface
build
calculations
chm
code-browsing
code-coverage
code-reviews
command-generation
communication-tools
comparison-tool
comparison-utilities
components
config-files
converter
copy
cryptography
date
dates
...
...
utilities
validation
webservice
windows-event-log
windows-registry
wincsp-interface
write
yes-or-no
zip-tools
```



Adding a Tatin Package

- Example: I use `HttpCommand` in just about every new project
- To add it to our Cider project:

```
]Cider.AddTatinDependencies HttpCommand  
1 Tatin dependency added:  
  dyalog-HttpCommand-5.2.0
```

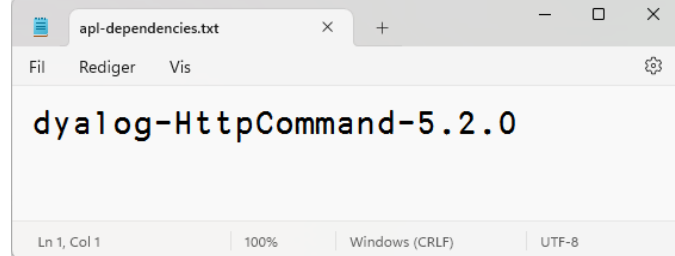
- Since we did not specify a version, we get the latest.
- Cider & Tatin create a reference to the loaded package within our project space:

```
D08.HttpCommand.Get 'www.dyalog.com'  
[rc: 0 | msg: | HTTP Status: 200 "OK" | #Data: 22580]
```

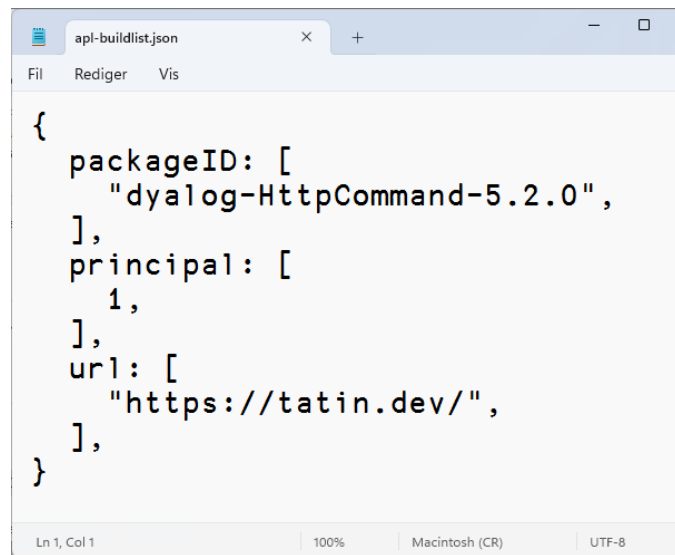


Installing Using Tatin

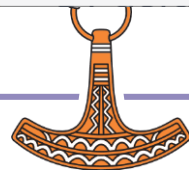
- `Tatin.InstallPackages` "installs" the package into a folder
 - (`Cider.AddTatinDependencies` calls it for you)
- Installed packages are registered in `apl-dependencies.txt` and `apl-buildlist.json`
- Version numbers are major.minor.patch
- If you do not specify a complete version number, Tatin fills in the blanks:
 - No version gives you the latest version
 - Latest patch if you specify major.minor
 - Latest minor version if you specify major



```
apl-dependencies.txt
Fil Rediger Vis
dyalog-HttpCommand-5.2.0
Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8
```

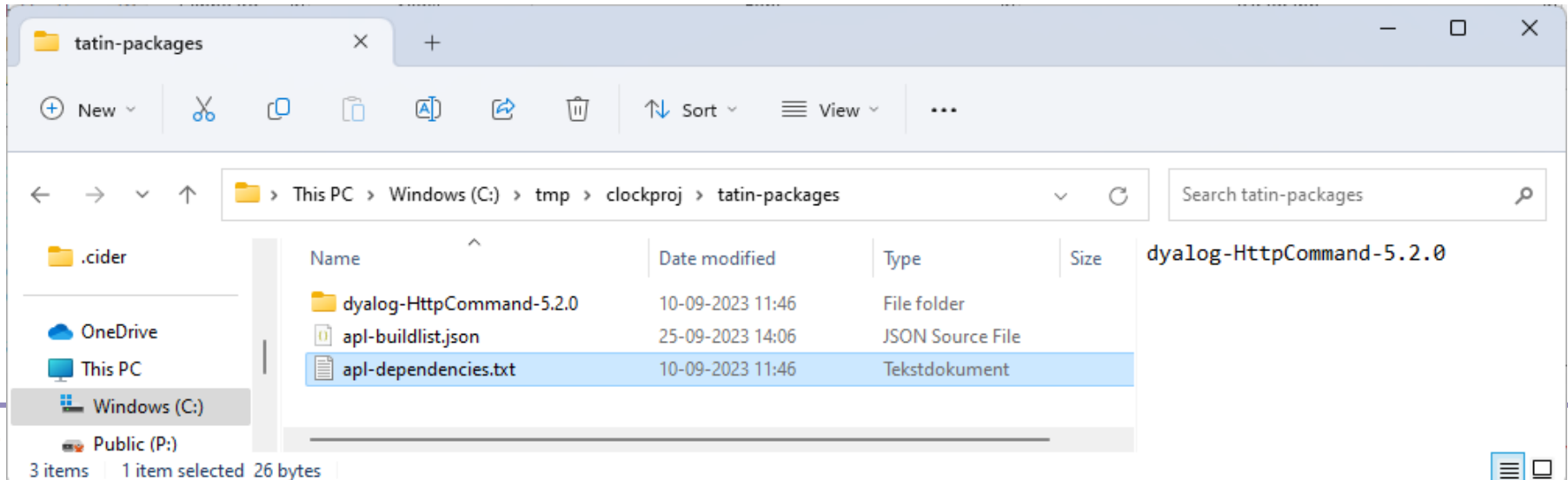


```
apl-buildlist.json
Fil Rediger Vis
{
  packageID: [
    "dyalog-HttpCommand-5.2.0",
  ],
  principal: [
    1,
  ],
  url: [
    "https://tatin.dev/",
  ],
}
Ln 1, Col 1 | 100% | Macintosh (CR) | UTF-8
```



Cider with Tatin

- By default, Cider directs Tatin to put dependencies in "tatin-packages"
- The packages themselves are in sub-folders



Cider - Tatin Collaboration

After:

```
]cider.openproject c:\tmp\clockproj  
Project successfully loaded and established in "#.clockproj"
```

The following are all equivalent:

```
]Cider.AddTatinDependencies HttpCommand  
]Tatin.InstallPackages dyalog-HttpCommand-5.2.0  
]Tatin.InstallPackages dyalog-HttpCommand c:\tmp\clockproj\tatin-packages
```

- Cider knows how to call Tatin to add packages
- Tatin knows where to find the installation folder of an open Cider project
 - (but you can explicitly specify the target folder if you want)
- Tatin will install the latest version of a uniquely identified package



Why are Cider & Tatin Separate?

- ◆ Cider and Tatin **are aware** of each other:
 - ◆ `]tatin.installpackages` will install to the currently open Cider project by default
 - ◆ `]tatin.buildpackage/publishpackage` will use Cider's `distributionPackage` setting as a default
 - ◆ Cider can have Tatin packages as dependencies
- ◆ Cider **requires** Tatin to load and run (it is a Tatin package)
 - ◆ Cider can **ALSO** manage NuGet dependencies
 - ◆ Possibly other dependency types to come
- ◆ Tatin **requires** Cider during development of Tatin itself
- ◆ However: At runtime, Cider and Tatin **do not require** each other



LoadPackages or LoadDependencies?

- ◆ Tatin.**LoadDependencies** loads the set of **installed** "dependencies" into a namespace
 - ◆ ... according to `apl-dependencies.txt` and `buildlist.json`
- ◆ When you open a project, Cider calls **LoadDependencies** for you
 - ◆ It will resolve any sub-dependencies, only loading each package once (more about this later)
- ◆ You can use **LoadPackages** to interactively load a package into a running APL session
 - ◆ This is only intended for interactive experimentation



What Happens on Project Open?

1. Create a namespace according to within the specified parent space
2. Set `IO` and `ML` according to `ciderconfig` `SYSVARS` section
3. Use `Link` to load all the source code from the folder named by `source`
4. Load Tatin and/or NuGet packages **(more about this soon)**
5. Inject `CiderConfig` namespace containing the config file settings
6. `CiderConfig.HOME` is set to the path the project was loaded from
7. Execute `CiderConfig.init` if it is non-empty
8. Execute `ExecuteAfterprojectOpen` (from `GlobalConfig`) if non-empty and `ignoreUserExec` is not set
9. Display the contents of `ToDo` variable, if it exists
10. *If* the project folder is managed by Git, display the result of "git status"



Loading Dependencies

We previously installed a dependency: `dyalog-HttpCommand-5.2.0`:

```
]Cider.OpenProject /tmp/anotherproject  
Project successfully loaded and established in "#.clockproj"
```

`LoadDependencies` has created a reference to the `HttpCommand` class inside our `projectSpace`, so we can easily reference it:

```
#.anotherproject.HttpCommand.Get 'https://www.dyalog.com'  
[rc: 0 | msg: | HTTP Status: 200 "OK" | #Data: 22580]
```

But this is an illusion: in fact, `#.anotherproject.HttpCommand` is a reference to a space Tatin uses to store **ALL** loaded packages:

```
#.anotherproject.HttpCommand  
#._tatin.dyalog_HttpCommand_5_2_0.HttpCommand
```



Loading Dependencies, Continued

The "host" namespace of a loaded package contains a namespace `TatinVars`, which contains information about the package (mostly for use by the package code). Despite the name, it contains functions:

```
#.anotherproject.HttpCommand.##.TatinVars.[]NL ~3
ASSETS CONFIG DEPENDENCIES GetFullPath2AssetsFolder HOME ID URI

#.anotherproject.HttpCommand.##.TatinVars.(ID URI)
dialog-HttpCommand-5.2.0+1 https://tatin.dev/
```

When Cider opens a project which it can see *is* a Tatin package, it injects a `TatinVars` space so that you can refer to this information both during development and when the package is loaded as a dependency.



Dependencies of Dependencies

Great fleas have little fleas upon their backs to bite 'em,
And little fleas have lesser fleas, and so *ad infinitum*.

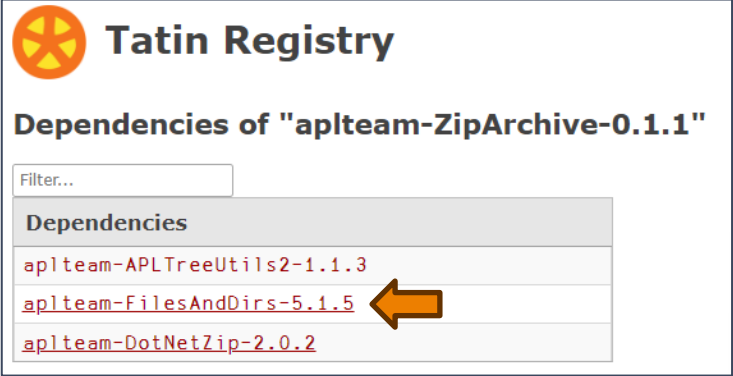
Both Tatin and NuGet will automatically load such dependencies

Augustus De Morgan was a British mathematician and logician. He formulated De Morgan's laws and introduced the term mathematical induction, making its idea rigorous.



Dependencies of Dependencies

- A package that you install can have dependencies of its own
- Tatin will automatically install them for you, and load them into the workspace
- Only the "Primary" dependency will be made available as a reference in YOUR namespace
 - The dependency namespace will have references to sub-dependencies, of course



Tatin Registry

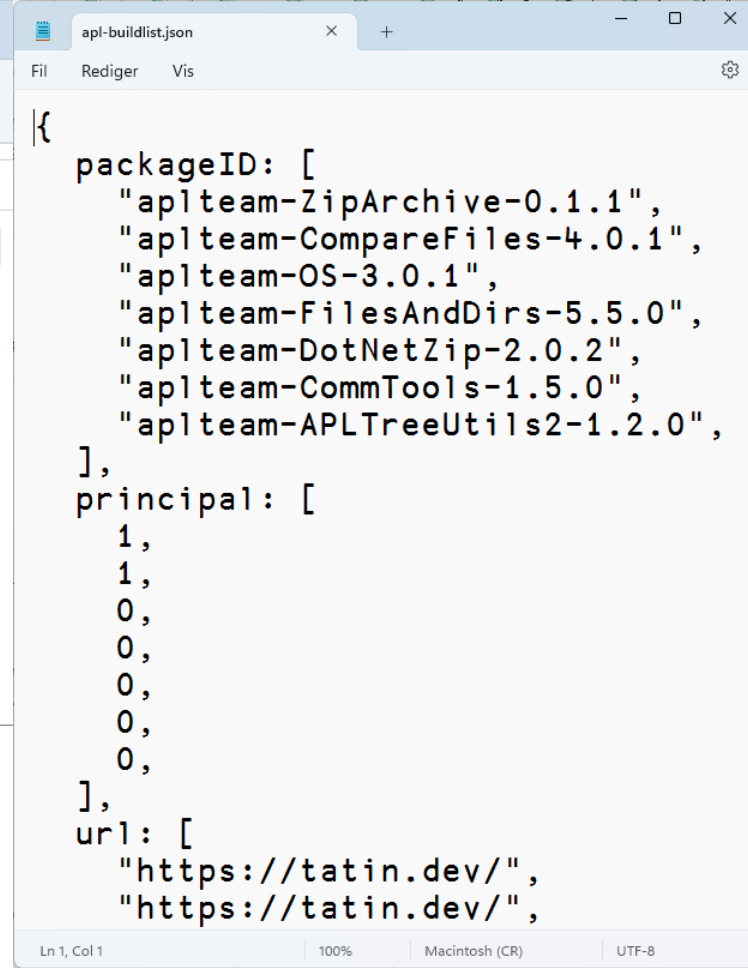
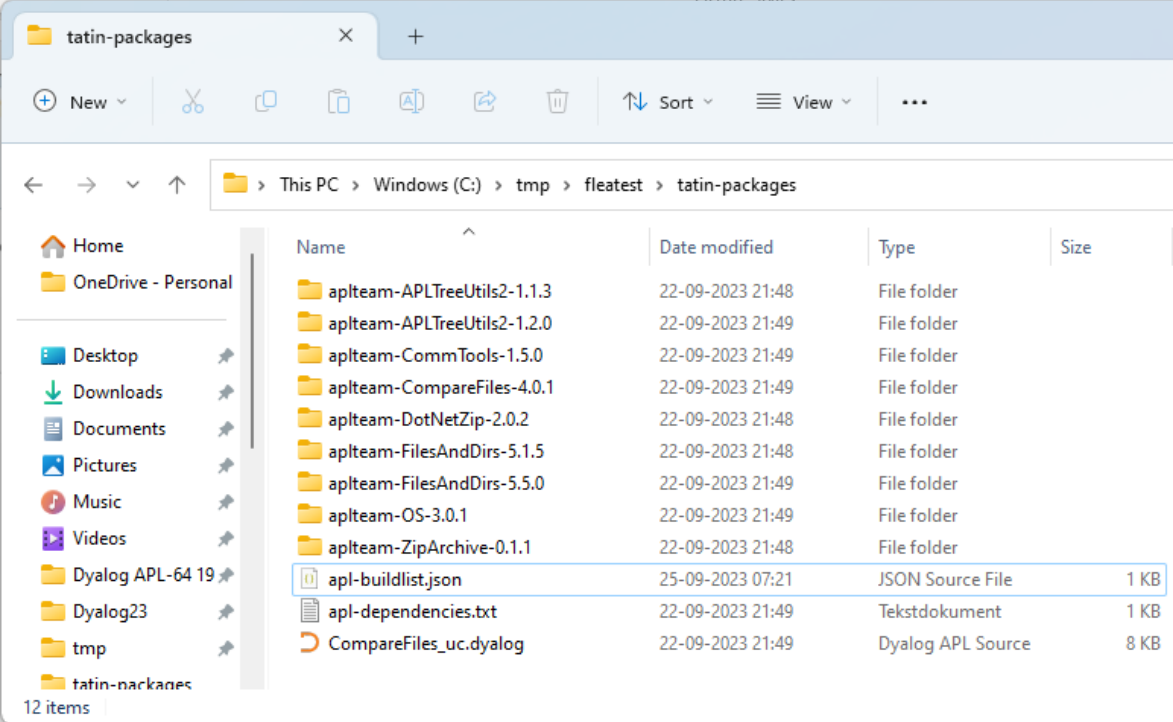
Dependencies of "aplteam-ZipArchive-0.1.1"

Filter..

Dependencies
aplteam-APLTreeUtils2-1.1.3
aplteam-FilesAndDirs-5.1.5 ←
aplteam-DotNetZip-2.0.2

←





- Notice FilesAndDirs-5.1.5 is not deleted
- Also notice CompareFiles_uc.dyalog



Where Do Dependencies Go?

```
]Cider.OpenProject C:\tmp\fleatest  
Project successfully loaded and established in "#.fleatest"
```

```
)cs fleatest  
#.fleatest
```

```
  [NL -9  
CiderConfig CompareFiles ZipArchive
```

Our dependencies

CompareFiles

```
#.tatin.aplteam_CompareFiles_4_0_1.API
```

```
  ;#.tatin.[nl -9  
aplteam_APLTreeUtils2_1_2_0  
aplteam_CommTools_1_5_0  
aplteam_CompareFiles_4_0_1  
aplteam_DotNetZip_2_0_2  
aplteam_FilesAndDirs_5_5_0  
aplteam_OS_3_0_1  
aplteam_ZipArchive_0_1_1
```

```
  #.tatin.aplteam_CompareFiles_4_0_1.[NL -9  
API APLTreeUtils2 Admin CommTools ComparisonTools FilesAndDirs TatinVars
```



"Minimal Version Selection" (MVS)

- A project can have two or more dependencies that in turn depend on the same package
- Tatin will use **MVS** to select a single version which is loaded
- In this case, version 5.5.0 of FilesAndDirs is the **minimal version** that satisfies all "consumers"
- Each consumer has declared the minimum version that it can accept:
 - ZipArchive wants at least 5.1.5
 - Comparefiles wants at least 5.5.0

Tatin Registry

Dependencies of "aplteam-ZipArchive-0.1.1"

Filter..

Dependencies
aplteam-APLTreeUtils2-1.1.3
aplteam-FilesAndDirs-5.1.5
aplteam-DotNetZip-2.0.2

Dependencies of "aplteam-CompareFiles-4.0.1"


Filter..

Dependencies
aplteam-APLTreeUtils2-1.2.0
aplteam-FilesAndDirs-5.5.0
aplteam-CommTools-1.5.0



```
]Cider.CreateProject /tmp/fleatest
]Cider.AddTatinDependencies aplteam-ZipArchive-0.1.1
aplteam-ZipArchive-0.1.1

)clear
]Cider.OpenProject /tmp/fleatest
fleatest.ZipArchive.##.FilesAndDirs
#._tatin.aplteam_FilesAndDirs_5_1_5.FilesAndDirs
```



Tatin Registry

Dependencies of "aplteam-ZipArchive-0.1.1"

Dependencies
aplteam-APLTreeUtils2-1.1.3
aplteam-FilesAndDirs-5.1.5
aplteam-DotNetZip-2.0.2



```
]Cider.CreateProject /tmp/fleatest
]Cider.AddTatinDependencies aplteam-ZipArchive-0.1.1
aplteam-ZipArchive-0.1.1

)clear
]Cider.OpenProject /tmp/fleatest
fleatest.ZipArchive.##.FilesAndDirs
#._tatin.aplteam_FilesAndDirs_5_1_5.FilesAndDirs

]Cider.AddTatinDependencies aplteam-CompareFiles-4.0.1
aplteam-CompareFiles-4.0.1

)clear
]Cider.OpenProject /tmp/fleatest
fleatest.ZipArchive.##.FilesAndDirs
#._tatin.aplteam_FilesAndDirs_5_5_0.API
```





Dependencies of "aplteam-CompareFiles-4.0.1"

Dependencies

aplteam-APLTreeUtils2-1.2.0

aplteam-FilesAndDirs-5.5.0

aplteam-CommTools-1.5.0

```
createProject /tmp/fleatest
addTatinDependencies aplteam-ZipArchive-0.1.1
archive-0.1.1
```

```
]Cider.OpenProject /tmp/fleatest
fleatest.ZipArchive.##.FilesAndDirs
#._tatin.aplteam_FilesAndDirs_5_1_5.FilesAndDirs
```

```
]Cider.AddTatinDependencies aplteam-CompareFiles-4.0.1
aplteam-CompareFiles-4.0.1
```

```
)clear
]Cider.OpenProject /tmp/fleatest
fleatest.ZipArchive.##.FilesAndDirs
#._tatin.aplteam_FilesAndDirs_5_5_0.API
```



"Minimal Version Selection" (MVS)

Pro:

- ◆ MVS gives reproducible builds:
- ◆ New versions of FilesAndDirs have no effect
- ◆ ZipArchive can upgrade to 5.5.0 w/no change to FilesAndDirs

Con:

- ◆ People struggle to understand MVS
- ◆ Adding a new dependency can change the version of a sub-dependency that gets loaded

Q:

- ◆ Do APL applications need MVS? Discuss...

Tatin Registry

Dependencies of "aplteam-ZipArchive-0.1.1"

Filter...

Dependencies
aplteam-APLTreeUtils2-1.1.3
aplteam-FilesAndDirs-5.1.5 ←
aplteam-DotNetZip-2.0.2

Dependencies of "aplteam-CompareFiles-4.0.1"

Filter...

Dependencies
aplteam-APLTreeUtils2-1.2.0
aplteam-FilesAndDirs-5.5.0 ←
aplteam-CommTools-1.5.0



Exercise 3

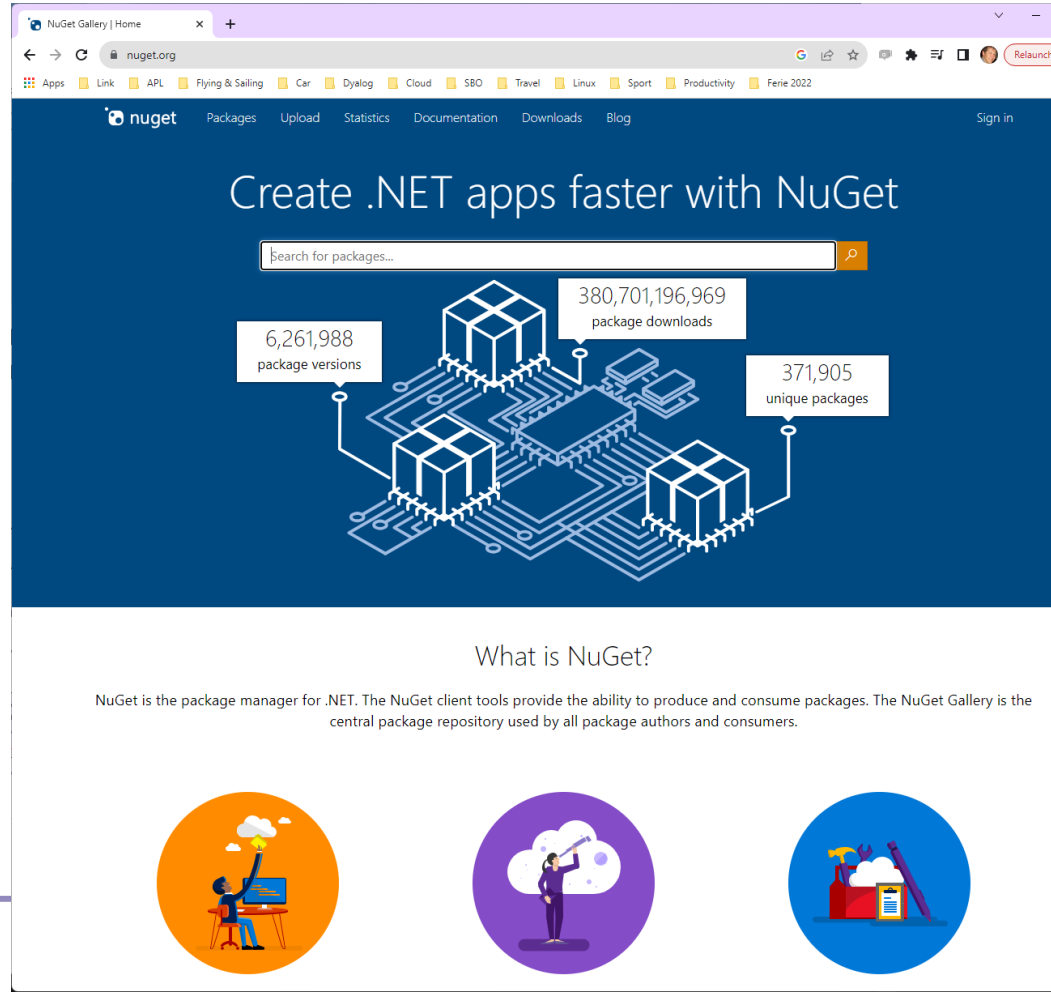
- ◆ Open your project and re-read the ToDo
 - ◆ Add `HttpCommand` as a dependency
- ◆ Write an application function which does something with `HttpCommand`
- ◆ If you have time, write a simple test for your application. If not, write a function which outputs "All tests were successful".
- ◆ Update the "tests" config parameter, and verify the result of

```
]Cider.RunTests
```



NuGet Packages

- NuGet is the .NET package manager



The screenshot shows the NuGet Gallery homepage in a web browser. The browser's address bar displays 'nuget.org'. The page features a dark blue header with the 'nuget' logo and navigation links for 'Packages', 'Upload', 'Statistics', 'Documentation', 'Downloads', and 'Blog'. A search bar is prominently displayed with the placeholder text 'Search for packages...'. Below the search bar, a central graphic illustrates the NuGet ecosystem with three callout boxes: '6,261,988 package versions', '380,701,196,969 package downloads', and '371,905 unique packages'. The graphic itself consists of a central circuit board with several 3D cube-like package icons connected to it. Below the graphic, the text 'What is NuGet?' is followed by a paragraph explaining that NuGet is the package manager for .NET, and the NuGet Gallery is the central repository used by all package authors and consumers. At the bottom of the page, there are three circular icons: a person at a computer, a person with a telescope, and a person with tools.

Finding NuGet Packages (HARD!!!)

The screenshot shows a web browser window with the address bar at `nuget.org/packages?q=fun`. The page header includes the NuGet logo, navigation links for Packages, Upload, Statistics, Documentation, Downloads, and Blog, and a Sign in button. A search bar contains the text 'fun'. Below the search bar, the page displays '11,706 packages returned for fun' and a 'Sort by' dropdown menu set to 'Relevance'. On the left, there are filter sections for Frameworks (listing .NET, .NET Core, .NET Standard, and .NET Framework) and Package type (listing All types, Dependency, .NET tool, and Template). The main content area lists two packages:

- Microsoft.NET.Sdk.Functions** by Microsoft nugetazurefunctions
85,667,952 total downloads | last updated 5 months ago | Latest version: 4.2.0
Build SDK for Azure Functions
- Microsoft.Azure.Functions.Analyzers** by azure-sdk Microsoft
33,538,913 total downloads | last updated 21/05/2021 | Latest version: 1.0.0
Azure Functions analyzers
This package provides development time code analysis for C# Azure Functions.

NuGet Support in Dyalog APL

- In v19.0, the namespace `⊞SE.Dyalog.NuGet` contains tools for installing and using NuGet packages
- For examples of how to use NuGet outside Cider, see the Tests folder at <https://github.com/Dyalog/nuget/tree/main/Tests>
- Cider uses this namespace to add support for NuGet packages, similar to that for Tatin packages

NuGet support currently requires .NET 6.0, 7.0 or 8.0

Support for "Framework" packages MAY follow



Adding a NuGet Package

- Example: NuGet contains a very simple package called "Clock".
- We can add it to our Cider project (by default, we get the latest version):

```
]Cider.AddNuGetDependencies Clock  
Clock 1.0.3
```

- A reference to a namespace hosting the .NET package is created:

```
#.clockproj.Clock.UtcNow.(Hour Minute)  
14 43
```

- In fact, the namespace is empty except for `using`:

```
using clockproj.Clock; using  
,c:/tmp/clockproj/nuget-packages/published/Clock.dll  
,c:/tmp/clockproj/nuget-packages/published/nuget-packages.dll
```

NuGet support currently requires .NET 6.0, 7.0 or 8.0

Support for "Framework" packages MAY follow



dotnet command-line tool

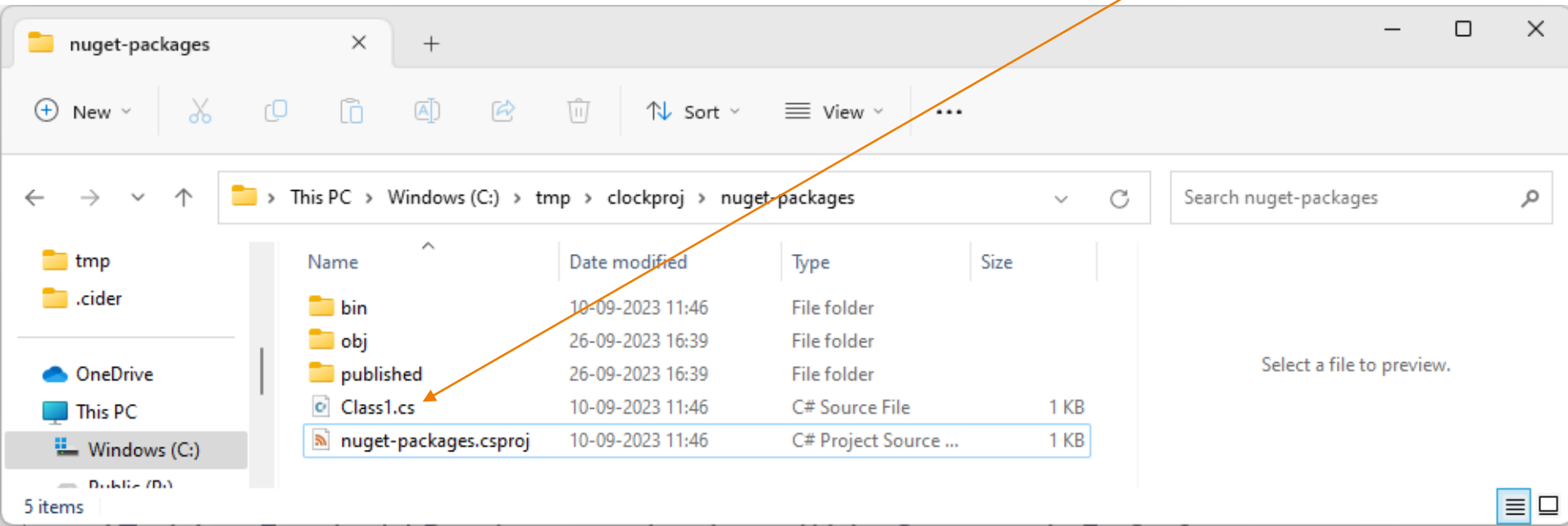
- ◆ Under Windows, Linux and macOS, .NET provides a "dotnet" command which:
 - ◆ Creates .NET projects that we use to define and manage dependencies (complete with a C# class that we never use)
 - ◆ Adds Dependencies
 - ◆ "Publishes" collections of DLLs that implement packages
- ◆ Dyalog's NuGet support depends heavily on this
 - ◆ We just set `□USING` to point to the published DLLs
 - ◆ The alternative is to try to replicate poorly documented .NET behaviours



NuGet Packages – Under the Covers

- By default, NuGet dependencies go in the nuget-packages folder:

C# Stub created by dotnet tool



Same Same – But Different

Tatin

```
apl-buildlist.json
{
  packageID: [
    "dyalog-HttpCommand-5.2.0",
  ],
  principal: [
    1,
  ],
  url: [
    "https://tatin.dev/",
  ],
}
```

#.projectSpace.HttpCommand

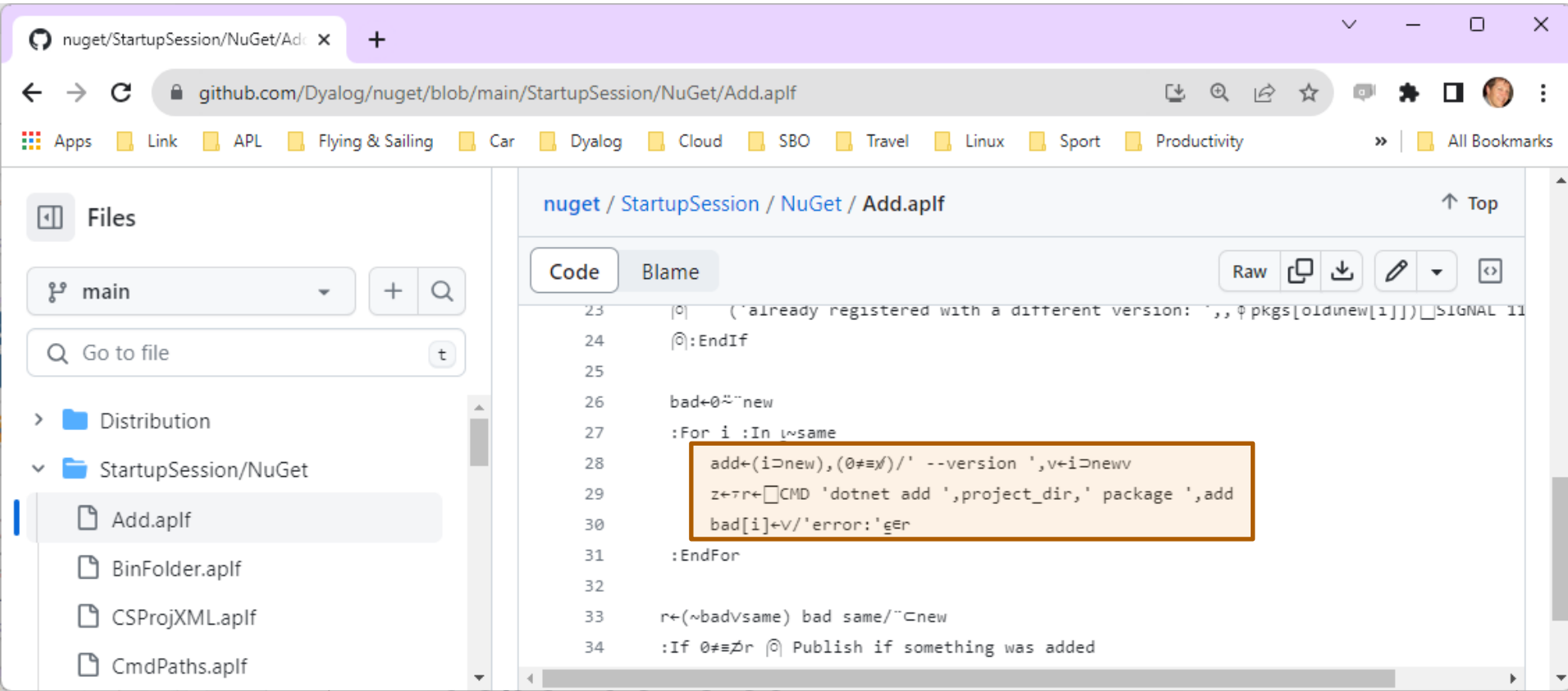
NuGet

```
nuget-packages.csproj
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <RootNamespace>nuget_packages</RootNamespace>
    <ImplicitUsings>enable</ImplicitUsings>
    <Nullable>enable</Nullable>
    <LangVersion>Latest</LangVersion>
  </PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Clock" Version="1.0.3" />
  </ItemGroup>
</Project>
```

#.projectSpace.Clock



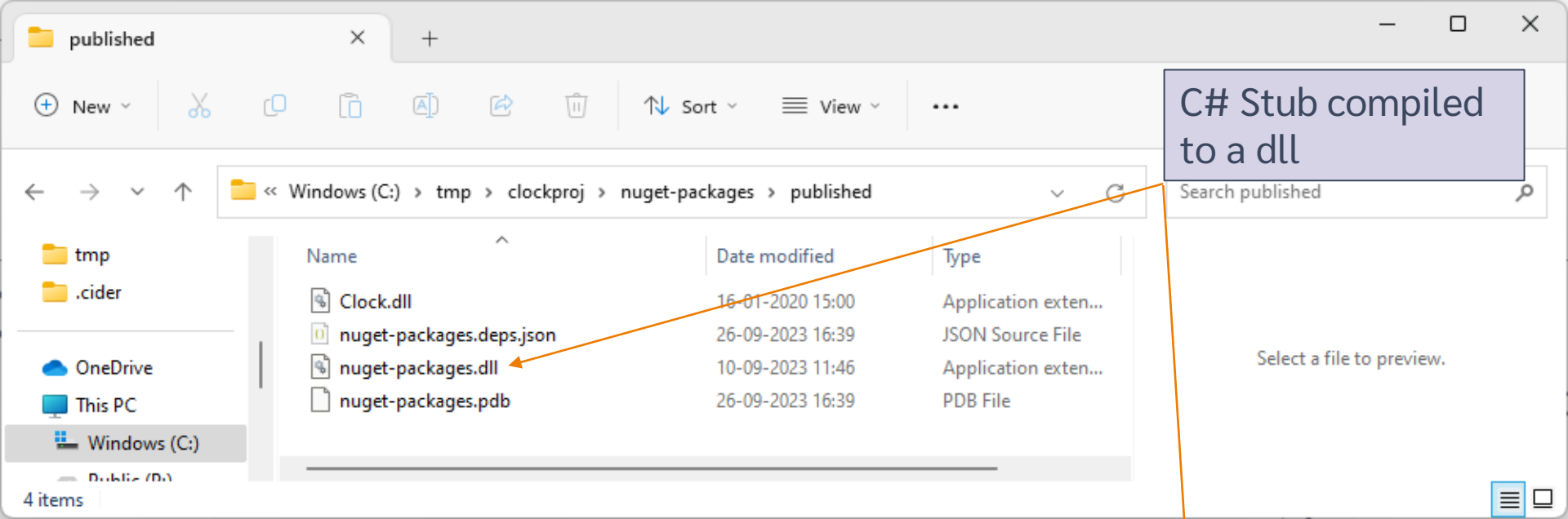
Example of calling dotnet tool



The screenshot shows a web browser displaying a GitHub repository page for the file `nuget/StartupSession/NuGet/Add.aplf`. The browser's address bar shows the URL `github.com/Dyalog/nuget/blob/main/StartupSession/NuGet/Add.aplf`. The page content shows the file's code in a light theme. The code is a PowerShell script with the following visible lines:

```
23 |0| ('already registered with a different version: ', $pkgs[0:Idnew[1]]) SIGNAL 11
24 |0|:EndIf
25
26 bad+0~`new
27 :For i :In $same
28     add+($idnew),($0#≠$)/' --version ',v+idnewv
29     z+7r+[]CMD 'dotnet add ',project_dir,' package ',add
30     bad[i]+v/'error: '$err
31 :EndFor
32
33 r+($~badv$same) bad same/'`Cnew
34 :If 0#≠$r |0| Publish if something was added
```

Line 28 is highlighted with a red box. The code in this line is: `add+($idnew),($0#≠$)/' --version ',v+idnewv`. The file explorer on the left shows the directory structure: `main` (selected), `Distribution`, and `StartupSession/NuGet` (expanded), with `Add.aplf` selected.



```
;.clockproj.Clock. USING  
,/tmp/clockproj/nuget-packages/published/Clock.dll  
,/tmp/clockproj/nuget-packages/published/nuget-packages.dll
```

```
#.clockproj.Clock.UtcNow  
26-09-2023 15:24:34 +00:00
```



Dependencies - Reporting

```
]Cider.OpenProject /tmp/clockproj  
... established in #.clockproj
```

```
]Cider.ListTatinDependencies  
Source          Package-ID          Principal URL  
-----  
tatin-packages/ dyalog-HttpCommand-5.2.0 1      https://tatin.dev/
```

```
]Cider.ListNugetDependencies  
Clock 1.0.3
```

```
clockproj.HttpCommand.Get 'www.dyalog.com'  
[rc: 0 | msg: | HTTP Status: 200 "OK" | #Data: 22580]
```

```
clockproj.Clock.UtcNow  
26-09-2023 15:26:37 +00:00
```



Files

main



Go to file



- > Distribution
- > StartupSession
- ▼ Tests

- Init.aplf
- Read.aplf
- ReadColumns.aplf
- test.aplf
- test_clock.aplf
- test_kafka.aplf
- test_mailkit.aplf
- test_parquet.aplf
- test_selenium.aplf
- userdata1.parquet

nuget / Tests / test_clock.aplf



🔥 mkromberg Initial Commit of working code

85ce7fe · 3 weeks ago 🕒 History

4 lines (4 loc) · 147 Bytes

Code Blame

Raw



```
1 test_clock project_dir; □ USING
2 □ SE.NuGet.Add project_dir 'Clock/1.0.3'
3 □ USING+ □ SE.NuGet.Using project_dir
4 'The time is: ', ⚡ Clock.UtcNow
```

Example of a NuGet Test Case



Exercise 4

- ◆ Add a NuGet dependency
 - ◆ Morten suggests Parquet files – see the NuGet Tests <https://github.com/Dyalog/nuget/tree/main/Tests>
 - ◆ Or just go for Clock 😊
- ◆ Create a project which uses both HttpClientCommand and your NuGet dependency



12:00-13:00 (ish?)

Session 3: BYO App + Wrap Up

- ◆ Development Dependencies
- ◆ Build Your Own App
- ◆ Recap and Conclusions
- ◆ Link, Cider and Tatin ToDo Lists

SP1 This Afternoon:

Creating your own Packages

Exercise 5:

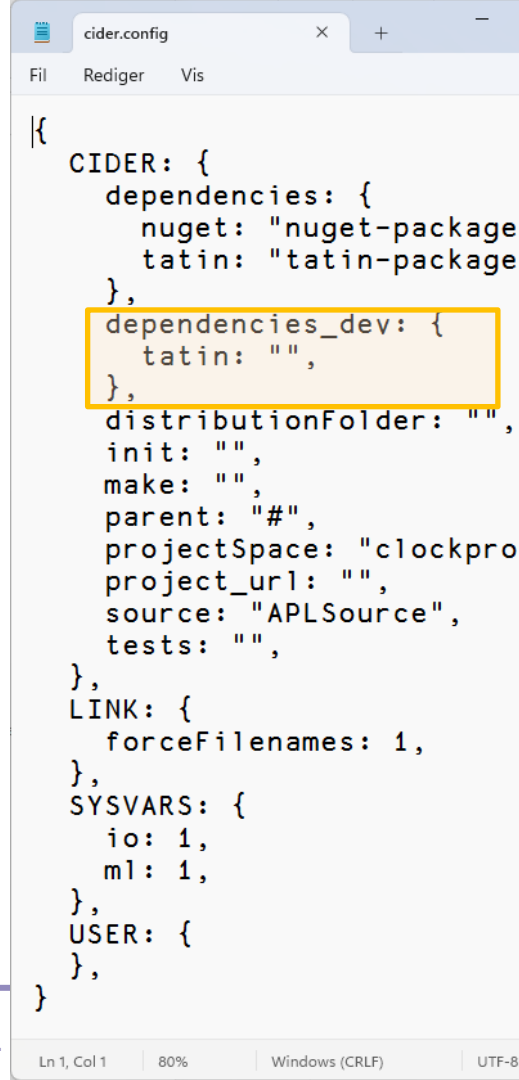
Build Your Own Application



Development Dependencies

- Packages that you need during development, but not runtime
 - At the moment (October 2023), only Tatin dependencies are supported
- To add development dependencies, you must edit `dependencies_dev` in `cider.config`
- You must **also** name the folder explicitly when adding dependencies. For example, if you set `dependencies_dev` to `'tatin-dev-packages'`:

```
]Cider.AddTatinDependencies Tester2 /tmp/fleatest/tatin-dev-packages
```



```
cider.config
Fil Rediger Vis
{
  CIDER: {
    dependencies: {
      nuget: "nuget-package",
      tatin: "tatin-package",
    },
    dependencies_dev: {
      tatin: "",
    },
    distributionFolder: "",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "clockpro",
    project_url: "",
    source: "APLSource",
    tests: "",
  },
  LINK: {
    forceFileNames: 1,
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
Ln 1, Col 1 | 80% | Windows (CRLF) | UTF-8
```

Development Dependencies

- OpenProject loads both sets of dependencies
- Your runtime application should only load the "normal" dependencies
- The separation means that development dependencies will not influence MVS for your runtime dependencies



```
File Edit Syntax Refactor View
Search...
{
  CIDER: {
    dependencies: {
      nuget: "nuget-dependencies",
      tatin: "tatin-dependencies",
    },
    dependencies_dev: {
      tatin: "tatin-dependencies_dev",
    },
    distributionFolder: "Dist",
    init: "",
    make: "",
    parent: "#",
    projectSpace: "anotherproject",
    project_url: "",
    source: "APLSource",
    tests: "",
    version: "",
  },
  LINK: {
  },
  SYSVARS: {
    io: 1,
    ml: 1,
  },
  USER: {
  },
}
}
Nested Array ( coloured as JSON )
```

Cider and Tatin "To Do" lists

- ◆ Review of Names & Messages
 - ◆ Dyalog to help with Documentation
- ◆ Shell-callable API for installation
- ◆ Ability to manage Local / Intermediate package stores within an organisation
- ◆ Is MVS the right choice?
- ◆ Do we need "recommended" packages.
 - ◆ Recommended by whom?
- ◆ Ability to import part of a package (e.g. dfns cmpx)?
- ◆ Actually running tests and builds for you



Link Road Map

Link v4.0 Highlights

- ◆ Configuration Files (incl "Global" config)
- ◆ Link single Class or Namespace file
- ◆ Create/Export/Import default to current namespace if none supplied
- ◆ Support for character vectors, matrices and vec-of-vecs in simple text files
- ◆ Link now being used by APL interpreter to load user code at startup

Link 5 & 6

- ◆ Crawler which will periodically compare workspace to source folders
 - ◆ Postponed from 3.0 to 4.0
 - ◆ Postponed from 4.0 to 5.0
- ◆ Create a proper API
 - ◆ (likely to get bumped to v6)



Exercise 5

- Build Your Own Application
 - As a Cider project
 - Must use at least one dependency

