LinearBase

How-To Digitally Model a New Databank Program and Code

Release 0.1

Included

- Model.zip
- Host.zip
- Docker image linearbase/lineardb-amd64:latest available from https://hub.docker.com

Auto-Generated Artifacts

- Databank files
- Databank Plain Old CLR Object (POCO) code
- LinearBase.Host test project
- Profiles supporting Centralised, Decentralized, and Distributed deployments
- Initializers for use with LinearBase.Host.exe

Summary

The purpose of Digitally Modelling from a JavaScript Object Notation (JSON) file is to create a new Databank with no content and add it to a Realm with the minimum of effort. The auto-generated code allows for relational queries and transactional updates across Windows, Android, iOS, Mac, and Linux with Docker-on-Linux for server, simultaneously.

A Realm contains one or more Realm Identity folders defined by a unique RealmId.

Each RealmId folder contains system Databank files that define a transaction boundary and describe each individual Model's schema, behaviour, and interaction with other transaction boundaries in other Realms.

Download the Modelling and Host Programs

Create a folder to store the programs, it can be any location including a flash drive for example 'E:\LinearBase'.

The transformation process writes databank files to a Realm folder. For this example, and to keep all the files in one place, create another folder called Realms under E:\LinearBase resulting in E:\LinearBase\Realms. This folder is not fixed and can be moved to another location, if necessary, post transformation.

Navigate to https://linearbase.co/#download

Modeller Program

Click or tap the 'Download Modeller' button. This will download a 32Mb compressed file called Model.zip to your device's download folder.

Figure 1 Download Digital Modeller Program

Extract the compressed files to the folder created earlier by selecting Medel.zip and choosing the 'Extract All…' option from the context menu.

Unless you change the default options this will create a new folder E:\LinearBase\Model.

There are seven files in the zip archive, once decompressed the files we're interested in are LinearBase.Model.exe and launchOptions.json.

The template.json, weblogicmono.json, address.json, customer.json, and weblogic.json support files help demonstrate the modelling process.

We will run the LinearBase.Model.exe program using the launchOptions.json and command line options at the end of the demonstration.

Host Program

If you haven't already, click or tap the 'Download Host' button. This will download a 31Mb compressed file called 'Host.zip to your device's download folder.

Figure 2 Download Widows Host Program

Extract the compressed files to the folder created earlier by selecting Host.zip and choosing the 'Extract All…' option from the context menu.

Unless you change the default options this will create a new folder E:\LinearBase\Host.

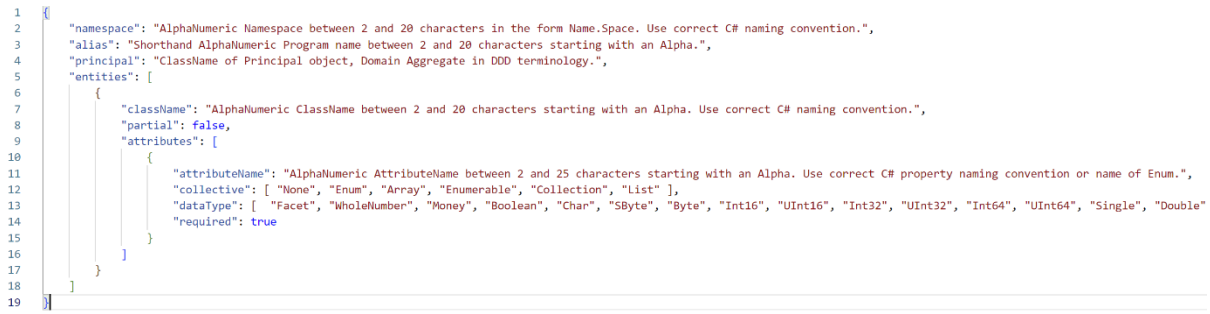
There are three files in the zip archive, once decompressed the file we're interested in is LinearBase.Host.exe. The program requires the other two support files to run.

The LinearBase.Host.exe program is initiated using launchOptions.json configuration file or by command line arguments. The modelling process auto-generates these settings for both options and writes them to the relevant Realm folder upon completion.

The 'Start in Console on Finish' option described later uses the command line method to initiate a standalone 'Centralized' topology instance for use in development.

Modelling Process

The modelling process accepts a JavaScript Object Notation (JSON) file [in the following format](#)



```
{
    "namespace": "AlphaNumeric Namespace between 2 and 20 characters in the form Name.Space. Use correct C# naming convention.",
    "alias": "Shorthand AlphaNumeric Program name between 2 and 20 characters starting with an Alpha.",
    "principal": "ClassName of Principal object, Domain Aggregate in DDD terminology.",
    "entities": [
        {
            "className": "AlphaNumeric ClassName between 2 and 20 characters starting with an Alpha. Use correct C# naming convention.",
            "partial": false,
            "attributes": [
                {
                    "attributeName": "AlphaNumeric AttributeName between 2 and 25 characters starting with an Alpha. Use correct C# property naming convention or name of Enum.",
                    "collective": [ "None", "Enum", "Array", "Enumerable", "Collection", "List" ],
                    "dataType": [  "Facet", "WholeNumber", "Money", "Boolean", "Char", "SByte", "Byte", "Int16", "UInt16", "Int32", "UInt32", "Int64", "UInt64", "Single", "Double"
                    "required": true
                }
            ]
        }
    ]
}
```

Figure 3 - template.json

template.json

model

The Models to include in the Realm. A Model is analogous to a database or portion of a database and represents a Bounded Context in Domain Driven Design terminology.

namespace

Alpha-Numeric Namespace between 2 and 20 characters in the form Name.Space. Use correct C# naming convention.

alias

Shorthand Alpha-Numeric Program name between 2 and 20 characters starting with an Alpha. Ensure this is unique within the Realm as duplicate aliases will overwrite initialization settings.

principal

ClassName of Principal object, Domain Aggregate in DDD terminology. See className below.

entities

Collection of one or more Entities to include in the Model. An Entity is analogous to a database table.

className

Alpha-Numeric ClassName between 2 and 20 characters starting with an Alpha. Use correct C# naming convention.

partial

Mark the generated class as partial allowing for additional behaviour logic.

attributes

Collection of one or more Attributes to include in the Entity. An Attribute is analogous to a database field in a table.

attributeName

Alpha-Numeric AttributeName between 2 and 25 characters starting with an Alpha. Use correct C# property naming convention or name of Enum.

collective

Expose attribute as one of the following values:

- None – Primitive or reference, i.e. Name.Space.ClassName
- Enum –Name of Enum, only one representation per Entity in this release
- Array – T []
- Enumerable – IEnumerable<T>
- Collection – Collection<T>
- List – List<T>

dataType

One of the following values:

- Facet (String) – Search optimized
- WholeNumber (Int32)
- Money (Decimal)
- Boolean
- Char
- SByte
- Byte
- Int16
- UInt16
- Int32
- UInt32
- Int64
- UInt64
- Single
- Double
- Decimal
- DateTime
- String
- Sentence
- Guid
- Bytes - for byte array specify Array collective

Or, to create an Composite or Aggregate association use a Fully Qualified ClassName reference in the form Name.Space.ClassName.

required

Is the field required, false represents a nullable attribute.

WebLogic Example

Four files contained in Model.zip file reflect the WebLogic Order Processing Database Schema weblogicmono.json, weblogic.json, customer.json, and address.json.

weblogicmono.json

Single file combining Web.Logic (weblogic.json), User.Detail (customer.json), and Post.Office (address.json) namespaces as a single Model.

Model

Modelling is known as the code-first approach and generates an Object Relation schema. In this situation the parent object holds a reference to the child object. Internal processes manage primary and foreign key associations and allocations, therefore there is no need to create primary or foreign keys when modelling. Write the model to reflect the resulting object with its properties, references, and collections.

For this example, we'll create a new Realm using the weblogic.json file then append the address.json and customer.json Models to the new Realm.

Open the E:\LinearBase\Model folder and copy weblogic.json, address.json and customer.json files to the E:\LinearBase\Realms folder.

There are two methods to model a new Databank program and create the associated code: configuration file and command line. Both accept the same values.

Command line takes precedence over launch options.

We'll use the launchOptions.json configuration file option for weblogic.json file and the Command Line option for address.json and customer.json files.

The input values are…

Script_FilePath

The full path and file name of the .json file to model. This setting is required.

In this example it is the file we just moved: E:\ LinearBase\Realms\weblogic.json

Script_RealmName

The name of the Realm to add the Model to. This setting is required.

If the Realm does not exist it is treated as a new Realm and the process will create it. If the Realm already exists the Script_RealmId value below dictates whether to append to an existing Realm or create an additional RealmId and transaction boundary.

Multiple Realm identities within a Realm is disabled in this release.

Script_Culture

The Culture to specify in the default auto-generated Profiles. Only en-GB and en-US are currently supported. If omitted en-GB is used. This setting is required.

Script_RealmId

RealmId is required only when appending a new Model to an existing Realm. For a new Realm set Script_RealmId to 'null'.

RealmId is the eight-character upper-case alpha-numeric folder name containing the system Databank files. This is auto-generated when creating a new Realm and is located under the Realm folder.

Script_ParsHost

ParsHost is required only when appending a new Model to an existing Realm. For a new Realm set Script_ParsHost to 'null'.

ParsHost is the URL and port number of the Primary Active Realm Server (PARS) instance, for example http://localhost:60001.

An active PARS instance is required to communicate with during the append Model process.

For information on generating an active PARS instance read 'How-To Host a Databank Program'

Script_HostExePath

The full path to the folder containing the LinearBase.Host.exe file. Assuming the files are in the default location specified at the start the path is E:\LinearBase\Host. This setting is optional.

Including this setting is equivalent to selecting the 'Start in new console on finish' option when using the Digital Transformation process.

Script_HostPort

If the Script_HostExePath value is specified the resulting PARS instance can optionally listen on a port ranging from 2000 to 65535 when creating a new Realm. If omitted zero is used meaning don't listen. This setting is optional.

If appending a Model to an existing Realm, this setting is ignored as the port number of PARS instance takes precedence.

Configuration File Option

Configuration file option supplies values to LinearBase.Host.exe when the program is run.

Navigate to E:\LinearBase\Model and open the launchOptions.json file.

```
{
  "Script_FilePath": "",
  "Script_RealmName": "",
  "Script_Culture": "en-GB",
  "Script_RealmId": null,
  "Script_ParsHost": null,
  "Script_HostExePath": null,
  "Script_HostPort": null
}
```

Figure 4 - launchOptions.json file

Replace each field with the relevant values surrounded by double quotes. Escape backslashes, that is replace each single backslash with two backslashes, for example E:\LinearBase****Host becomes E:****LinearBase****Host

```
{
  "Script_FilePath": "E:\\LinearBase\\Realms\\weblogic.json",
  "Script_RealmName": "Logical",
  "Script_Culture": "en-GB",
  "Script_RealmId": null,
  "Script_ParsHost": null,
  "Script_HostExePath": "E:\\LinearBase\\Host",
  "Script_HostPort": "60001"
}
```

Figure 5 - Completed launchOptions.json

Navigate to E:\LinearBase\Model\LinearBase.Model.exe and initiate run by double clicking or double tapping. Optionally select the program, right click, and select 'Open' from the context menu.

The console application opens and reads the launch options.

Any issues identified with the settings or schema integrity will display at the bottom of the console output.

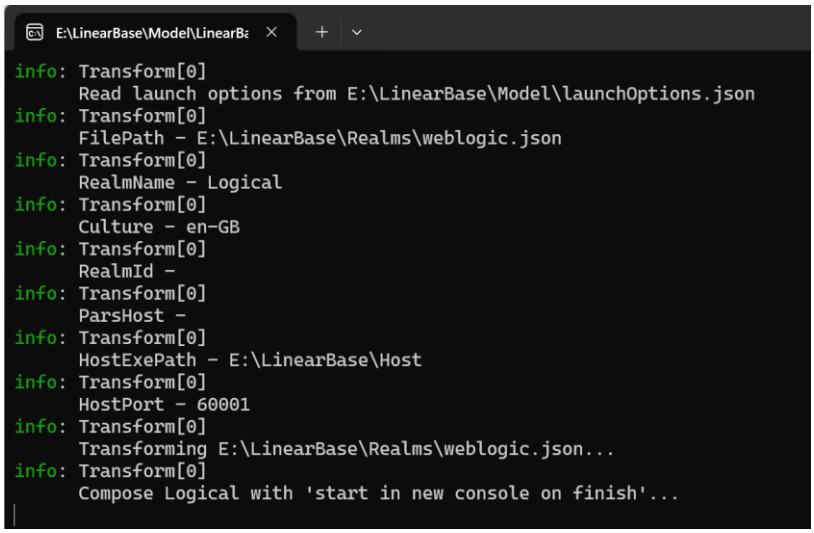


Figure 6 - Modelling process in action

If successful the process will initiate the 'run' command on LinearBase.Host.exe then complete and close automatically.

A separate console opens and loads an active PARS instance for the new 'Logical' Realm we just created.

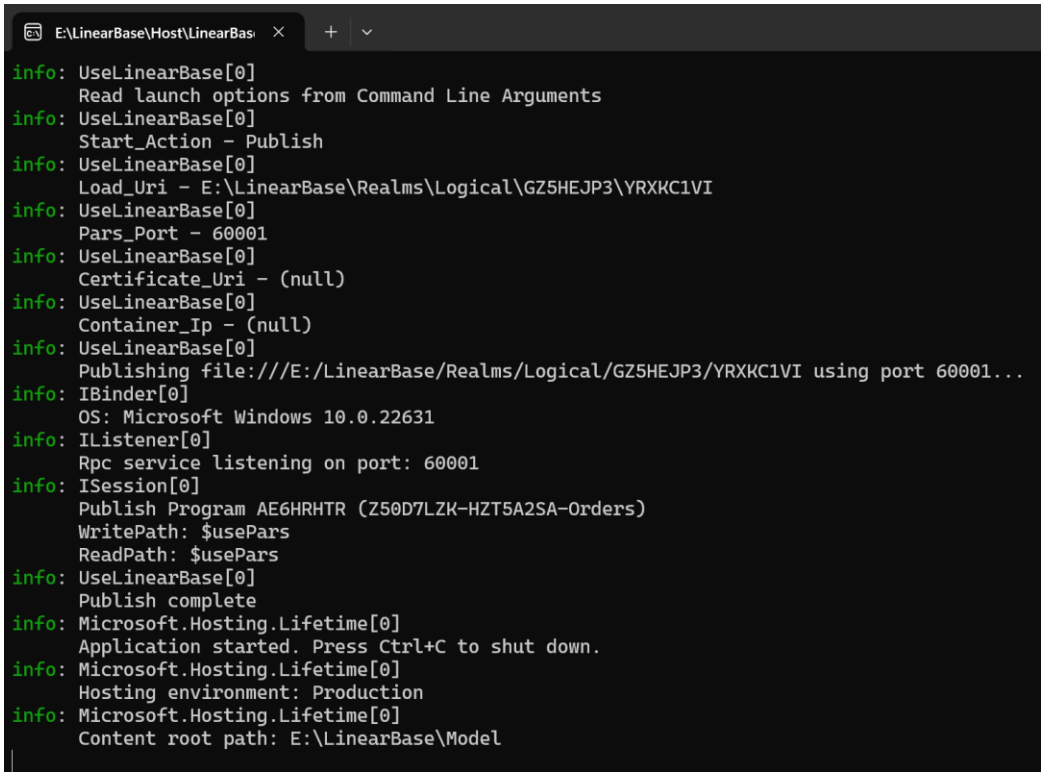


Figure 7 - Primary Active Realm Service (PARS) instance hosted in a console application

DO NOT CLOSE THIS CONSOLE APPLICATION it's hosting our PARS instance on http://localhost:60001 and we need it in this example to append additional Models to the Realm.

The line '***Load_Uri - E:\LinearBase\Realms\Logical\GZ5HEJP3\YRXKC1VI***' identifies this as a Centralized PARS instance as it's loading from a file location not a URL as used in Distributed and Decentralized deployments we'll see later.

Cloudless enabled Apps and programs can now interact with this instance in Centralized, Distributed, and Distributed data topologies simultaneously. For more information on topologies, Profiles, or interacting with an active PARS instance read 'How-To Host a Databank Program'

Next, we'll use the Command Line option to append the address.json Model to the new 'Logical' Realm we just created.

First, we need to identify the auto-generated RealmId containing the system Databank files.

Navigate to E:\LinearBase\Realms\Logical. Here we find three new folders.

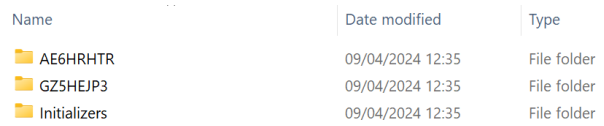


Figure 8 - Logical Realm Databank folders

Folder AE6HRHTR contains the newly Modelled weblogic.json Program and associated code. AE6HRHTR is the new Program's unique auto-generated ProrgramId.

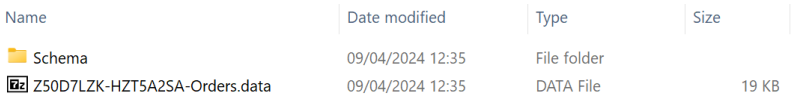


Figure 9 - weblogic.json Program Databank & Code

Folder GZ5HEJP3 contains the system Databank files, it's this Id we need to append additional Models to the new RealmId.

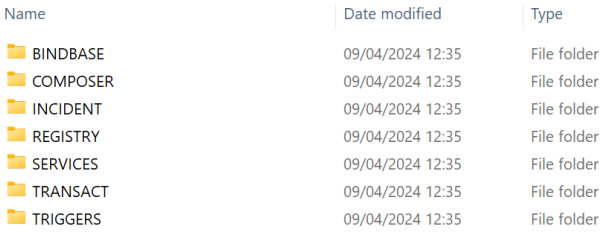


Figure 10 - Realm System Databanks

Command Line Option

Command line, as the name suggests, injects input values to the LinearBase.Model.exe using a Windows command dialogue. To open a command window, press the **WinKey** the type '**CMD'** followed by the return key.

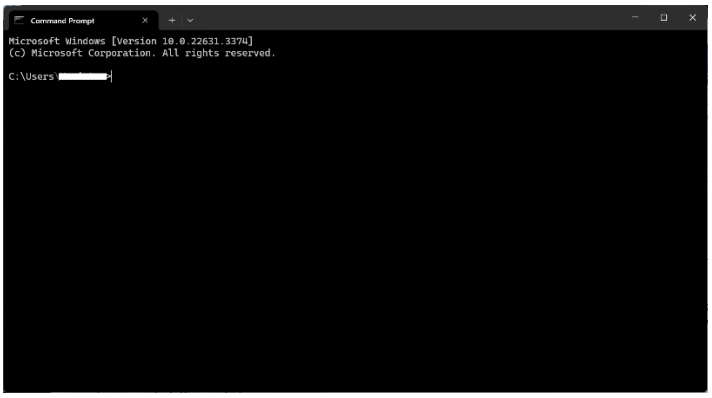


Figure 11 - Command Window

Assuming the files are in the default location specified at the start, change directory to the Model folder by typing '***cd E:\LinearBase\Model***' + return followed by '***E:***' + return to switch to the correct drive.

Each input argument has an associated shortcut of a dash followed by a letter:

- -f = Script_FilePath
- -n = Script_RealmName
- -c = Script_Culture
- -i = Script_RealmId
- -p = Script_ParsHost
- -e = Script_HostExePath
- -h = Script_HostPort

Initiate the address.json modelling process by typing the following and pressing return:

LinearBase.Model.exe -f=E:\LinearBase\Realms\address.json -n=Logical -c=en-GB -i=GZ5HEJP3 -p=http://localhost:60001 -e=E:\LinearBase\Host -h=0

This says use **LinearBase.Host.exe** to transform **E:\LinearBase\Realms\address.json** on Realm **Logical** configured for **en-GB** culture targeting RealmId **GZ5HEJP3** via PARS instance **http://localhost:60001** and start the new Model as a Distributed instance.

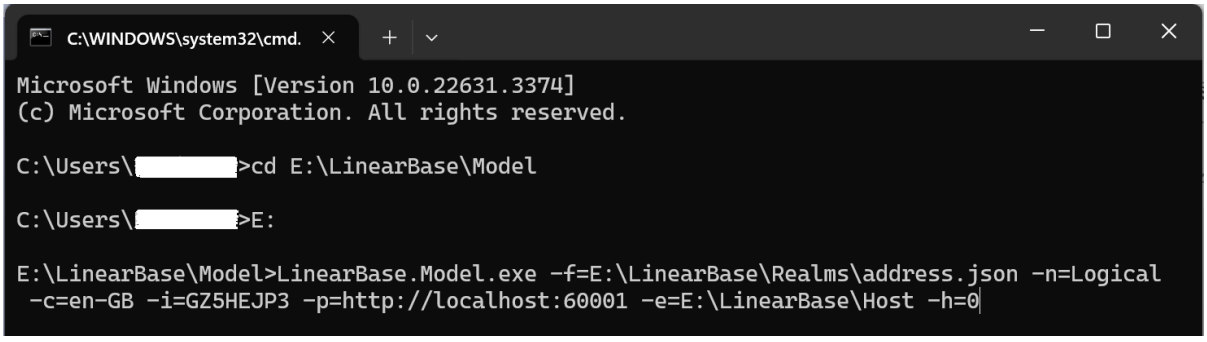


Figure 12 - Command Line Option

Press 'return' to run the command line instruction…

```
E:\LinearBase\Model>LinearBase.Model.exe -f=E:\LinearBase\Realms\address.json -n=Logical
 -c=en-GB -i=GZ5HEJP3 -p=http://localhost:60001 -e=E:\LinearBase\Host -h=0
info: Transform[0]
      Read launch options from Command Line Arguments
info: Transform[0]
      FilePath - E:\LinearBase\Realms\address.json
info: Transform[0]
      RealmName - Logical
info: Transform[0]
      Culture - en-GB
info: Transform[0]
      RealmId - GZ5HEJP3
info: Transform[0]
      ParsHost - http://localhost:60001
info: Transform[0]
      HostExePath - E:\LinearBase\Host
info: Transform[0]
      HostPort - 0
info: Transform[0]
      Transforming E:\LinearBase\Realms\address.json...
info: Transform[0]
      Compose Logical with 'start in new console on finish'...
info: Transform[0]
      Transform complete.
      Press any key to close...

E:\LinearBase\Model>
```

Figure 13 - Command Line Option outcome

On successful completion a separate console application opens similar to…

```
E:\LinearBase\Host\LinearBas
      Read launch options from Command Line Arguments
info: UseLinearBase[0]
      Start_Action - Publish
info: UseLinearBase[0]
      Load_Uri - http://localhost/GZ5HEJP3/UENLVXMZ
info: UseLinearBase[0]
      Pars_Port - 60001
info: UseLinearBase[0]
      Certificate_Uri - (null)
info: UseLinearBase[0]
      Container_Ip - (null)
info: UseLinearBase[0]
      Publishing http://localhost/GZ5HEJP3/UENLVXMZ using port 60001...
info: IBinder[0]
      OS: Microsoft Windows 10.0.22631
warn: IListener[0]
      Rpc service not listening! Port in use: 60001
info: ISession[0]
      Publish Program UZDCSVUV (U4S3OGYA-BM62426F-Postal)
      WritePath: $usePars
      ReadPath: $usePars
info: UseLinearBase[0]
      Publish complete
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: E:\LinearBase\Model
```

Figure 14 - address.json Distributed instance

The significance of this instance is it's using the Distributed Profile to initiate the configuration as can be seen by the line '***Load_Uri - http://localhost/GZ5HEJP3/UENLVXMZ***'. This reads ProfileId **ENLVXMZ** from the active PARS instance describing where the relevant resources are located and how to initialize them.

To append the customer.json Model change address.json to customer.json and re-run the command:

LinearBase.Model.exe -f=E:\LinearBase\Realms\customer.json -n=Logical -c=en-GB -i=GZ5HEJP3 -p=http://localhost:60001 -e=E:\LinearBase\Host -h=0

Inspect Source Files

Assuming successful transformation, navigate to the path chosen in the Compose input dialogue, for example E:/LinearBase/Realms/Logical.

The create and append transformation examples have created five new folders

- AE6HRHTR – Orders (WebLogic) Databank folder
- GZ5HEJP3 – System Databank folder
- Initializers – Initialization instruction folder
- NDBSAS52– Consumer (Customer) Databank folder
- UZDCSVUV – Postal (Address) Databank folder

It is important Realm Databank folders remain under the relevant Realm folder, in this case 'Logical'. It is equally important only a single version of the Realm exist at any one time although it can be moved anywhere. Moving a Realm requires a PARS instance restart to reflect the new location.

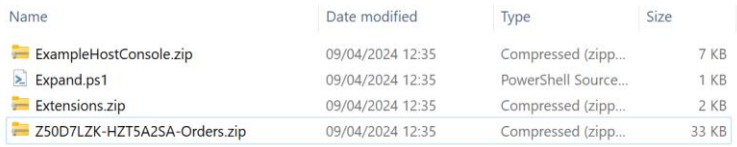
Expand the Orders Databank folder E:\LinearBase\Realms\Adventure\LFWMKUZD.

The process created a 'Schema' folder and a .data Databank file containing compressed, format-neutral binary files.

Expand the E:\LinearBase\Realms\Logical\AE6HRHTR\Schema folder.

The Transform process created files

- ExampleHostConsole.zip – the demonstration project
- Expand.ps1 – PowerShell script to unzip the Program files and a demonstration project
- Extensions.zip – Generic host code
- Z50D7LZK-HZT5A2SA-Orders.zip – C# Program poco files unique to this transformation

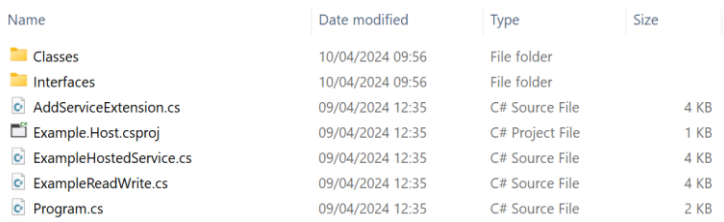


Name	Date modified	Type	Size
ExampleHostConsole.zip	09/04/2024 12:35	Compressed (zipp...	7 KB
Expand.ps1	09/04/2024 12:35	PowerShell Source...	1 KB
Extensions.zip	09/04/2024 12:35	Compressed (zipp...	2 KB
Z50D7LZK-HZT5A2SA-Orders.zip	09/04/2024 12:35	Compressed (zipp...	33 KB

Figure 15 Schema files

Select Expand.ps1, right click and choose 'Run with PowerShell'. This generates another folder named ExampleHostConsole containing the zip file contents.

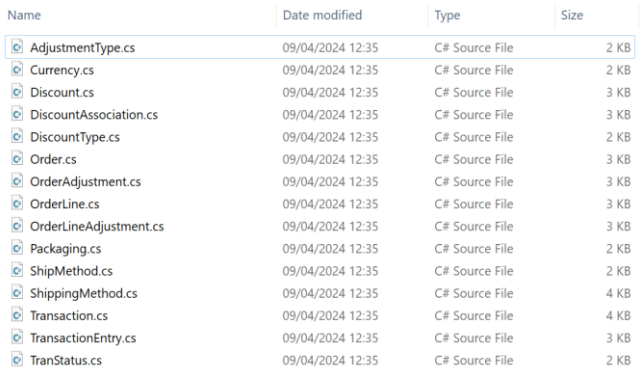
Expand the E:\LinearBase\Realms\ Logical\AE6HRHTR \Schema\ExampleHostConsole folder to inspect unzipped files…



Name	Date modified	Type	Size
Classes	10/04/2024 09:56	File folder	
Interfaces	10/04/2024 09:56	File folder	
AddServiceExtension.cs	09/04/2024 12:35	C# Source File	4 KB
Example.Host.csproj	09/04/2024 12:35	C# Project File	1 KB
ExampleHostedService.cs	09/04/2024 12:35	C# Source File	4 KB
ExampleReadWrite.cs	09/04/2024 12:35	C# Source File	4 KB
Program.cs	09/04/2024 12:35	C# Source File	2 KB

Figure 16 ExampleHostConsole Project

Expand the Classes folder to view auto-generated poco files corresponding to the Table and Field selections made earlier. The example generates Web.Logic.Order classes and interfaces.



Name	Date modified	Type	Size
AdjustmentType.cs	09/04/2024 12:35	C# Source File	2 KB
Currency.cs	09/04/2024 12:35	C# Source File	2 KB
Discount.cs	09/04/2024 12:35	C# Source File	3 KB
DiscountAssociation.cs	09/04/2024 12:35	C# Source File	3 KB
DiscountType.cs	09/04/2024 12:35	C# Source File	2 KB
Order.cs	09/04/2024 12:35	C# Source File	3 KB
OrderAdjustment.cs	09/04/2024 12:35	C# Source File	3 KB
OrderLine.cs	09/04/2024 12:35	C# Source File	3 KB
OrderLineAdjustment.cs	09/04/2024 12:35	C# Source File	3 KB
Packaging.cs	09/04/2024 12:35	C# Source File	2 KB
ShipMethod.cs	09/04/2024 12:35	C# Source File	2 KB
ShippingMethod.cs	09/04/2024 12:35	C# Source File	4 KB
Transaction.cs	09/04/2024 12:35	C# Source File	4 KB
TransactionEntry.cs	09/04/2024 12:35	C# Source File	3 KB
TranStatus.cs	09/04/2024 12:35	C# Source File	2 KB

Figure 17 Classes

Profiles control everything in LinearBase. More on this in the next article ‘How-To Host a Databank Program’.