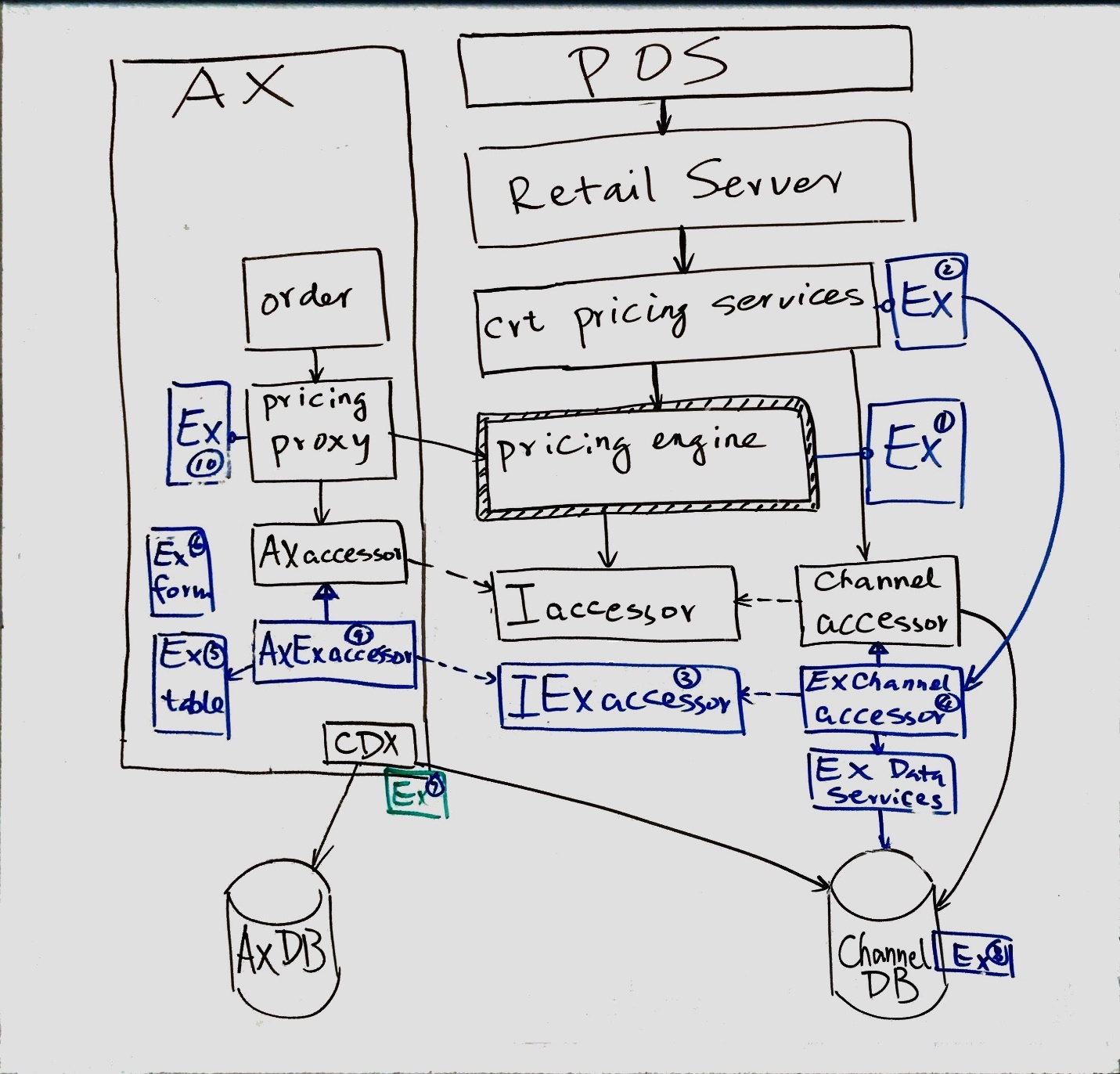
Pricing Sample Tests

# Overview

While creating a new discount type is hard, testing can be challenging as well in a different way. We can deploy it via LCS for testing. Unfortunately, it is not practical because LCS deployment can take a long time and it is all manual.

# Pricing engine and extensibility in one picture



# Sample tests

For now, we can do black box component testing with mock pricing data accessor that reads pricing and discount related data from XML files.

## Test foundation (DLL only)

1. **Mock pricing data accessor** that reads pricing and discount related data from XML files
   1. Discount data for all of out-of-box discount types.
   2. Item and category data
2. **Transaction setup and data helper**
   1. Indicate which discounts to use for the transaction.
   2. Set various discount configurations for test.
   3. Data helper that reads discount data from XML files, to avoid hardcoded values.
3. Transaction discount data consistency **verification**. For example, if we see two discounts in one sales line, can they compound?
4. **Tracing**
   1. Print out pricing and discount data in a sales transaction.
   2. Print out other discount calculation related info.

## Sample tests (source code)

1. Mock pricing data accessor extension with XML.
2. Tests for the new discount type
3. Tests for concurrency control model – pricing zone
4. Tests for concurrency control model – compete with priority and compound across

## Location of test foundation mock data

RetailSdk\Documents\SampleExtensionsInstructions\PricingAndDiscounts\FoundationData

# How to test a new discount type?

We will use amount cap as an example.

## Review thy discount

It’s paramount for us to understand the new discount type and review all the details.

## Test with only one new discount

As an example with amount cap: AmountCapTests.cs

1. 3 basic discount methods: offer price, amount off and percentage off. We need to test cap and no cap for all discount methods.
2. Rounding
   1. with multiple products or with one product with multiple sales lines.
   2. Configuration: split or no split for rounding (PriceContext.HoldTogetherForDiscountRounding)
3. Return
4. Mix of sales and return in one transaction.

## Test for concurrency model of compete within priority and compound across

Now we’re onto how it mixes up with other discount types.

First make sure you understand [concurrency control model of compete within priority and compound across](https://blogs.msdn.microsoft.com/retaillife/2017/01/08/retail-discount-concurrency-control-compete-within-priority-and-compound-across/).

1. Compound across priority (with and without threshold discounts)
2. Compete within priority. (For IDiscountPostBestDeal, it’s after best deal discounts.)
3. Mix in discount base amount adjustment
4. Mix in compound behavior of off original price (PriceContext.DiscountCompoundBehavior)

CompeteModelDiscountBaseAmountTests.cs

CompeteModelOffOriginalPriceDiscountBaseAmountTests.cs

CompeteModelTests.cs

## Test for best price and compound within priority and no compound across

Again, first understand [concurrency control model of best price and compound within priority and no compound across](https://blogs.msdn.microsoft.com/retaillife/2017/01/07/retail-discount-concurrency-control-pricing-zone/).

This is similar to the other concurrency control model.

## Lastly, to repeat, review the code and add tests when appropriate

# References

[Blog discussions about retail discounts](https://blogs.msdn.microsoft.com/retaillife/2017/01/07/retail-discount-concurrency-control-pricing-zone/)

.