Order/Invoice Immutability

This discussion paper aims to clarify the reasons as to when *Order*s and *Invoice*s, and their corresponding *Order Item*s and *Invoice Item*s, become immutable.

# Business Problem

The original motivation included these two related issues:

* (EST-1571) discover that the seller was incorrect on an *Order*; needs fixing
* (EST-1602) realise that an *Order* should be discarded (eg is a duplicate scan), but it has already been marked as completed/approved

This led into two sorts of discussions:

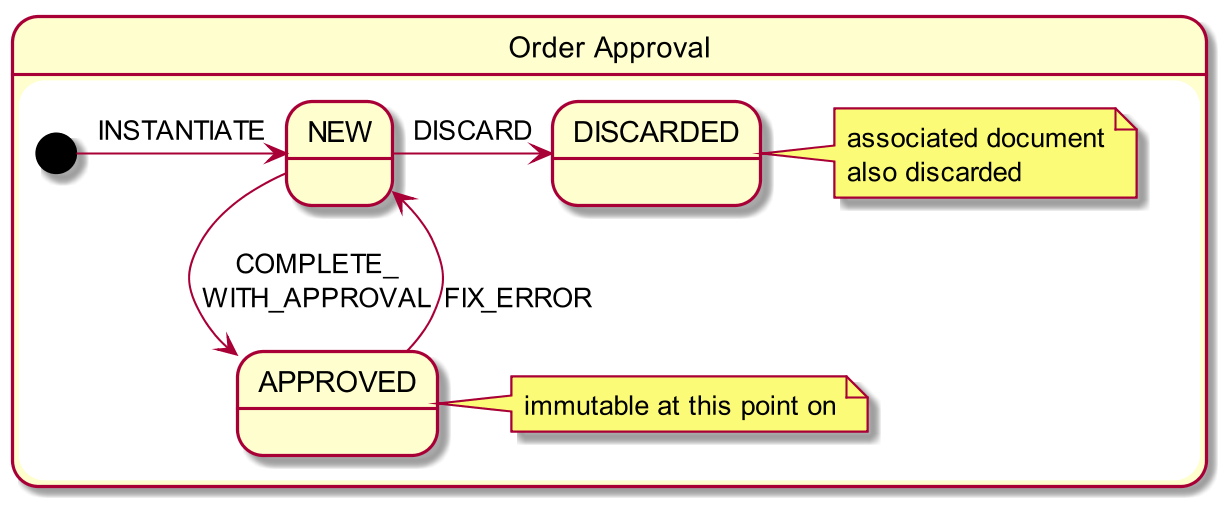
* how to make Estatio more user-friendly by defaulting information where possible when linking *Order*s and *Invoice*s together
* a deeper discussion about what additional constraints there are on whether an *Order* or an *Invoice* and its child items has become immutable.

This paper tackles the **second** of these issues.

# Reasons why data becomes immutable:

## Internal ("soft")

The first main reason why we treate data is immutable is that it has gone through an internal approval cycle. However, this is only a "soft" constraint, in that we do allow *Invoice*s to be rejected, sent back to fix a data issue, and therefore to go round the loop.

We also intend to do the same for *Order*s (though rather than "reject" an *Order*, the terminology is "fix error"). Sending an *Order* back to be fixed also means it can be discarded.

## External ("hard")

The other reason why data becomes be treated as immutable is when it has been reported/sync’d over to another system/organisation external to estatio.

Strictly speaking, we could distinguish between:

* extra-ECP : ie to another system or organisation outside of ECP, eg IBP.
* intra-ECP : ie to another system (eg CODA) that is nevertheless within ECP.

Probably PWC could be thought of as an "internal" interaction.

For extra-ECP interactions, there’s little to discuss: any adjustments that have to be made once the information has been extracted from Estatio will have to be done as reversals/deltas. Credit notes are one "real-world" example of this.

For intra-ECP interactions, we have more options, so should discuss them.

Since the automated software that will do the copy the data from one system to another (Estatio to CODA, say) is stuff that we will write, we *do* at least have the option of a design that would keep Estatio simple/naive, and move the complexity into the "glue".

For example, rather than treating *Invoice Item*s as immutable once reported (meaning that changes must be handled as reversals/deltas) we could if we wanted just post snapshots of the current state; ie always the "absolute" current view. For anyone using Estatio only this is simpler "view of the world".

On the other hand, the downside is that there is "magic" within the software that syncs Estatio to CODA: it would need to convert the snapshots of current state coming from Estatio into reversals/deltas within CODA. There are two consequences of this:

* from a business viewpoint, the information in Estatio vs CODA would look quite different, so would not facilitate conversations between Property Administration vs Accounting.
* from a technical viewpoint, it just moves the complexity from one bit of custom software (Estatio) to another (the glue that sits between Estatio and CODA). Most likely we’d prefer to keep the complexity in Estatio than to have it sitting in the glue.

**To conclude**: although intra-ECP vs extra-ECP interactions are theoretically different, it makes the most sense to treat them the same. (This was the working assumption in previous meetings/discussions anyway, so nothing really changes),

Thus, **whenever data is extracted from Estatio, for whatever purpose, then that data becomes immutable**.

**Important**

To make this explicit: we are proposing that Estatio could make *Invoice Item*s or *Order Item*s immutable **even if** their parent *Invoice*/*Order* has not yet been fully approved.

If the approver finds an issue with an item that requires a change, then the amendment required would be done by reversing out the incorrect items, and reposting an amendment.

## What actually is immutable?

The data that is extracted from Estatio is, basically, the *Order Item*s or *Invoice Item*s, with some information such as supplier copied down from their "header" *Order* or *Invoice*.

Once an item becomes immutable (by way of being copied to some other system), these parent *Order*/*Invoice* objects must also be treated as immutable. For example, we cannot change the seller on an *Order* once one of its items has been reported to PWC.

**Important**

Therefore: if the approver finds an issue with a "header" *Order* or *Invoice* (specifically, the incorrect seller), then this would require reversing the item on the original *Order*/*Invoice*, and creating a new "header" *Order*/*Invoice* for the correct seller with the reposted item.

On the other hand, the *link*s between *Order Item*s and *Invoice Item*s (by which we accrue *Invoice*s against *Order*s) are *never* treated as immutable. Estatio allows these to be adjusted at any time.

# Worked Example

The following applies to both *Order*s and *Invoice*s.

|  |  |
| --- | --- |
| While mutable:    invoices mutable.png | Reporting to an external system, the header object (eg *Invoice*) and its child items both become immutable (shown as shaded): invoices immutable.png |
| To amend immutable (same seller):  invoices amended same seller.png  Note that the "item #2" is immutable (because it is reversing out "item #1") but has *not yet* been reported. (We might use a timestamp rather than a single flag to indicate if reported or not). | invoices amended different seller.pngTo amend an immutable item to change the seller requires the creation of a new invoice referencing the new party: |