Edition Market and Bids Proposal

Secondary market for Artwork Editions and the bidding paradigm used. Last updated 12 June 2020.

Actors

These are the Customers that take part in the marketplace. All actors need to be logged in to participate.

Owner

This is the current owner of the Artwork Edition. They control the placing, withdrawal and acceptance of bids.

Bidder

This is a Customer who wishes to place a bid to purchase a placed Artwork Edition.

Purchaser

This is a Customer who has won the bidding, and is required to pay the price.

System

This is our web-site that automatically or manually processes the bidding system and the subsequent purchases. The reconciling of the bids can be automatic and/or manually controlled; involving the sending of e-messages and payment by the Stripe 3rd party system. A midnight Task is used to control timed event.

Web-Pages

A set of web-pages generally related to specific Actors.

Owner Resale Control

This web-page is reached by a link from the Owners web-page. The Owner of the Artwork Edition initialises the placing of an Asset from their Owners web-page, but controls all aspects of the placing from this web-page. If the Owner has more than one placing, this page will list them all here, each with their own section of the screen.

Market Listings

This web-page is for any visitor to view all Artwork Edition placings as an item list, with each item occupying a section of the screen. Each section will give details of the Asset with information supplied by the Owner regarding price, paradigm and exotic data. Only Customers are allowed to place bids, visitors will be redirected to the log-in/register web-page if they try to place a bid.

Bidding

This web-page navigation-link will appear on the Market listings web-page. It is for those Customers that have placed a bid, to see both how they are progressing and to modify the bid they placed, or withdraw it. If the Customer has placed more than one bid, they will be shown as screen sections. This is also the page used to make the actual payment and/or authorisation.

Management

This Administrator web-page will list all placings in a master-slave display, to allow staff interaction in the bidding processes, if required.

Fast Auction or Slow Market

These are essentially the same except for the time-scales and the method of payment. The development choice is either to choose one to implement, or implement both, and allow the Owner to choose the paradigm to use. The terms Auction and Market may be used interchangeably in this document and elsewhere; the actual paradigm in use is indicated by the 'Fast' or 'Slow' annotations.

Fast Auction

This is a short duration (less than 6 days) process that requires all bidders to authorise the payment of their bids during placement. This allows for a completely automatic process to occur either at the end of the auction or immediately for a sale-price bid. The funds will only be collected when the auction is over. If a bidder is serious then they should not mind a reserve-charge being placed on their credit card. The requirement for this process to be so short is due to the automatic expiry of the credit-card reserve-charge after 7 days.

Authorisation Hold Periods

The length in time an Authorisation-Hold or Reserve-Charge is valid for depends on the card issuer and the card type (credit or debit). This can vary from 24 hours to 30 days, with debit cards generally being the shortest. For credit cards the vast majority support a minimum of 7 days. An authorisation-hold cannot normally be cancelled, but Stripe has an automatic process for cancelling authorisation-holds, but I do not know what they charge for this service.

To extend the validation process, we could save the card details and then re-present the charge if the validation has expired. However, these methods require a more complex set-up as well as consent from the customer; also, extra data is required to be retained is a secure way for each customer: these methods preclude the use of the pre-built Stripe-Checkout. We can of course just mention to bidders that certain debit cards are not suitable for this process and that bids may lapse due to this restriction.

Slow Market

This process can be of any length as no financial action is taken until a purchase is confirmed. The downside is that further communication is required with the successful bidder to ensure payment is made, and if not, then other bidders need to be contacted to fulfil the sale. This could end up with a number of bidders being contacted with requests to pay and some of them losing-out to more prompt payers. I would suggest a 24-hour dwell period between requests to pay to the various bidders, with e-messages to unsuccessful customers when the sale completes, thanking them for their patience.

Recommendations

Initially implement the Fast-Auction paradigm, with caveats in the displayed notes, to help define the process. Secondly implement the Slow-Market paradigm, with e-messaging, inviting customers to pay, as an interim solution. When these are both working well, upgrade the Market-Auction system to where we obtain consent to securely store bidder's card details, and collect a payment automatically at the end of the auction. This will require work on the client to implement the specific user interface as we cannot use Stripe's prebuilt 'Checkout' for this paradigm; combined with server work, to securely store card details. However, as we will need to do similar work for Alipay, we can combine both upgrades into the same work project.

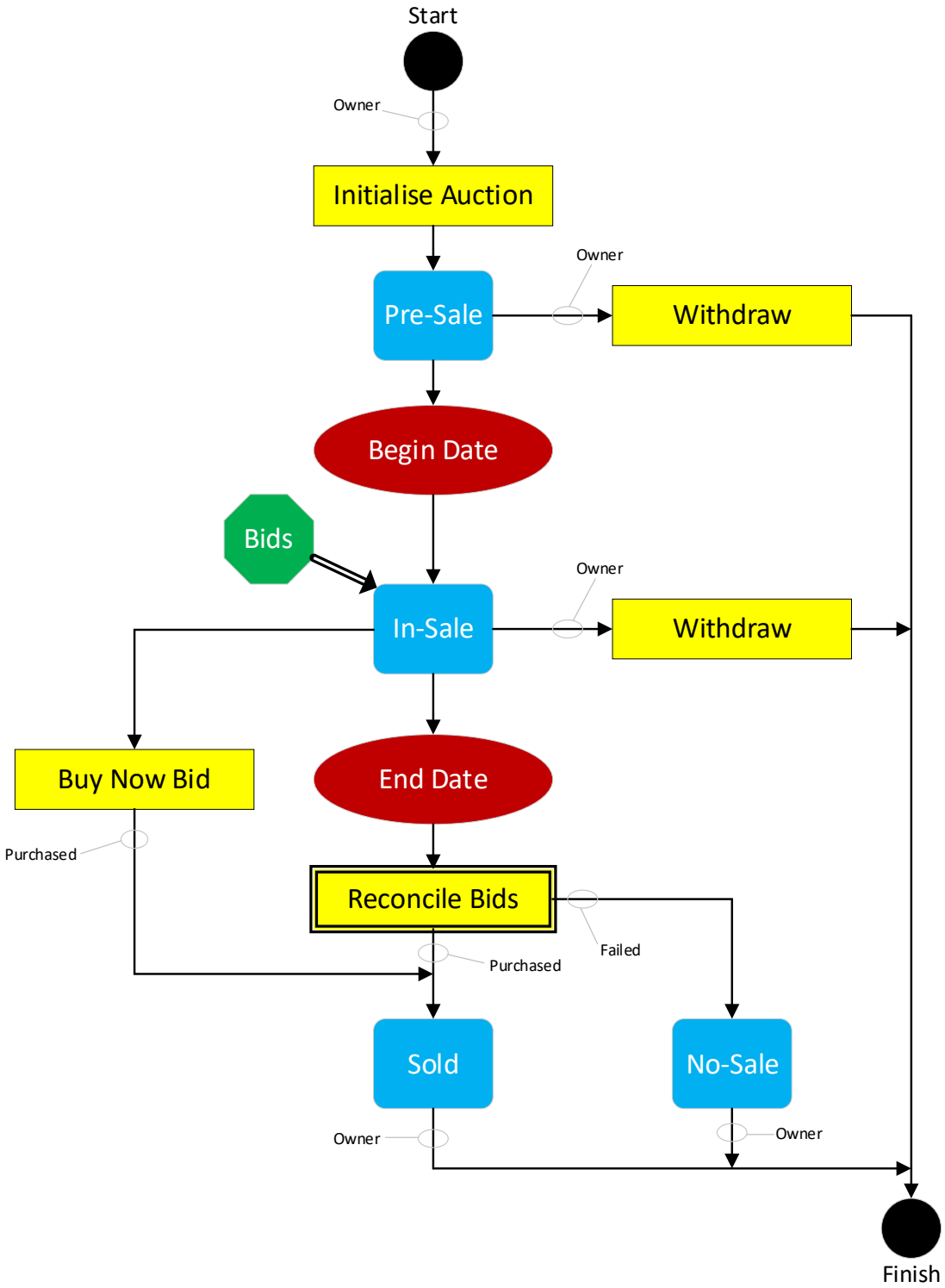
Commission and Currency

Both the seller and buyer have to pay a commission on the sale.
The bids are understood to include the seller's commission but not the buyer's commission, this will be displayed in the purchase sub-form only.
All participants will be informed of the commission rates as appropriate.
The actual commission rates will be determined by the Edition's Artwork records.
The currency used is to be the same as that of the original, as defined in the Edition's Artwork records.

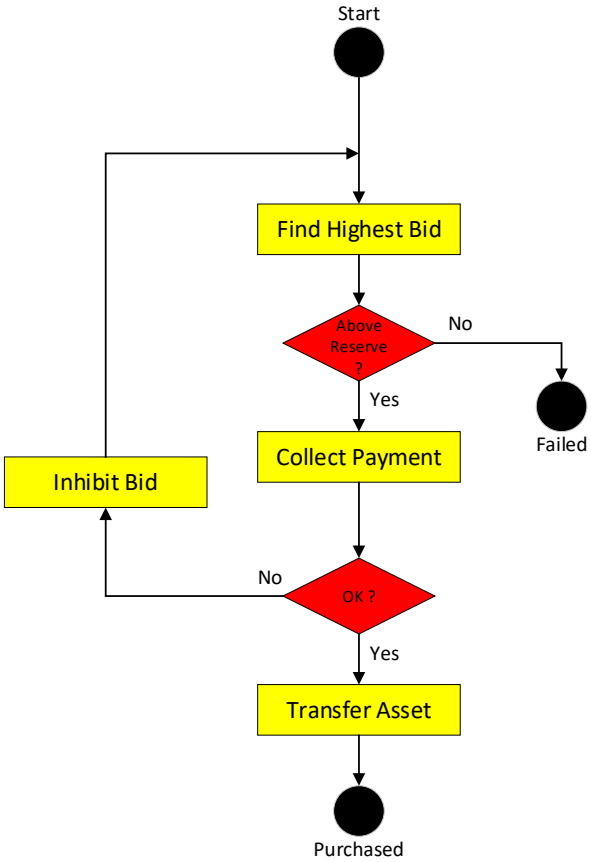
Bidding Paradigm Flowcharts

The purchase process used here is the same as for the initial purchase of Artwork Editions, except that the checking for availability is not required.

Market Auction Flowchart

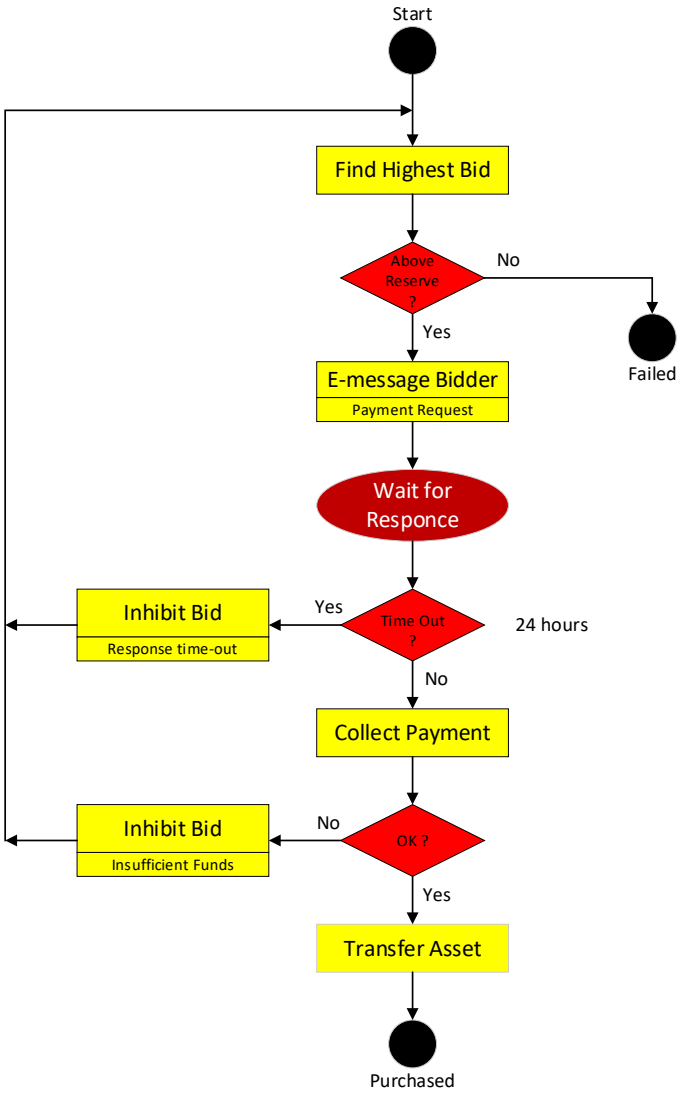


Reconcile Bids Flowcharts



Fast Auction Flowchart

This paradigm uses a two-step payment process of Authorise only and then Collect.



Slow Market Flowchart

This paradigm uses a single step payment process of combined Authorise and Collect.

Market Data Model Tables

Market Table

Field	Type	Key	Details
MarketId	uuid	primary	Market entity object identifier
OwnerId	uuid	foreign	
AssetId	uuid	foreign	
Market_BeginDate	datetime	secondary	start of the auction (00:00)
Market_EndDate	datetime	secondary	finish of the auction (23:59)
Market_State	a32		overall state of auction
Market_SalePrice	n4		The price for immediate sale
Market_ReservePrice	n4		The minimum price for automatic sales
Market_LowestBid	n4		minimum acceptable bid price
Market_ViewFlags	n4		Set of user view flags

View-Flags: In-Auction or not, Privacy setting hide Owner, Privacy settings hide bids, Fast-Slow Market Auction (etc.).

Market Bids Table

Field	Type	Key	Details
MarketBidId	uuid	primary	
MarketId	uuid	foreign	
UserId	a450	foreign	user identity guid
MarketBid_DTS	datetime	secondary	
MarketBid_State	a32		
MarketBid_Ammount	n4	secondary	
MarketBid_Comment	a400		customer's comment on bid
MarketBid_Flags	n4		control flags for bid

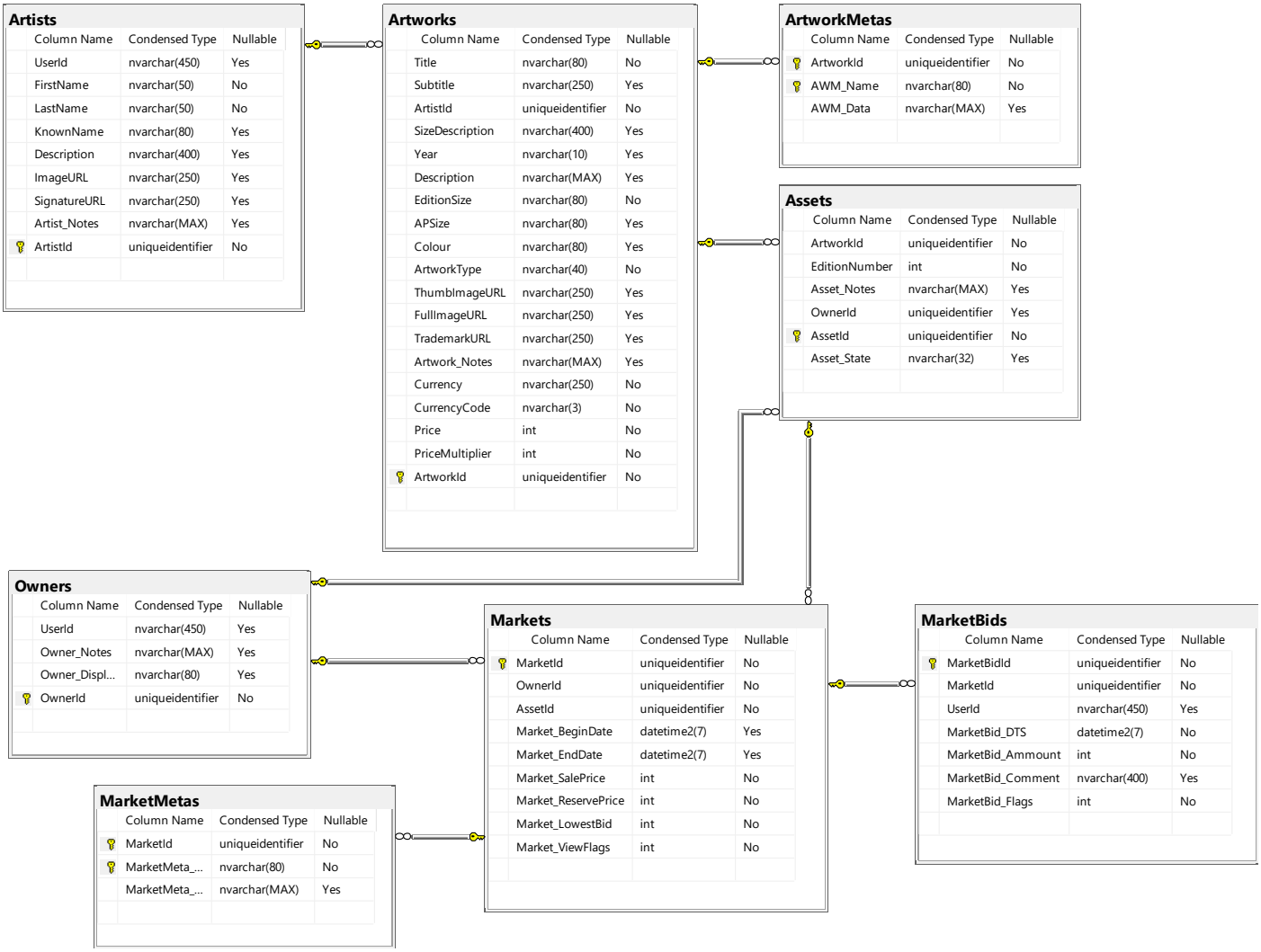
Bid-Flags: Inhibit for reconcile, Inhibit by Owner, Request to pay (etc.).
Bid-State: Init, Invisible, .

Market Metadata Table

Field	Type	Key	Details
MarketId	uuid	primary	combined primary key
MarketMeta_Name	a80	primary	combined primary key
MarketMeta_Data	a(Max)		

Possible entries: Owner display name, Owner reason for sale, Edition bought date, Edition special notes (etc.).

Market Data Tables



Artists

Column Name	Condensed Type	Nullable
UserId	nvarchar(450)	Yes
FirstName	nvarchar(50)	No
LastName	nvarchar(50)	No
KnownName	nvarchar(80)	Yes
Description	nvarchar(400)	Yes
ImageURL	nvarchar(250)	Yes
SignatureURL	nvarchar(250)	Yes
Artist_Notes	nvarchar(MAX)	Yes
ArtistId	uniqueidentifier	No

Artworks

Column Name	Condensed Type	Nullable
Title	nvarchar(80)	No
Subtitle	nvarchar(250)	Yes
ArtistId	uniqueidentifier	No
SizeDescription	nvarchar(400)	Yes
Year	nvarchar(10)	Yes
Description	nvarchar(MAX)	Yes
EditionSize	nvarchar(80)	No
APSize	nvarchar(80)	Yes
Colour	nvarchar(80)	Yes
ArtworkType	nvarchar(40)	No
ThumbImageURL	nvarchar(250)	Yes
FullImageURL	nvarchar(250)	Yes
TrademarkURL	nvarchar(250)	Yes
Artwork_Notes	nvarchar(MAX)	Yes
Currency	nvarchar(250)	No
CurrencyCode	nvarchar(3)	No
Price	int	No
PriceMultiplier	int	No
ArtworkId	uniqueidentifier	No

ArtworkMetas

Column Name	Condensed Type	Nullable
ArtworkId	uniqueidentifier	No
AWM_Name	nvarchar(80)	No
AWM_Data	nvarchar(MAX)	Yes

Assets

Column Name	Condensed Type	Nullable
ArtworkId	uniqueidentifier	No
EditionNumber	int	No
Asset_Notes	nvarchar(MAX)	Yes
OwnerId	uniqueidentifier	Yes
AssetId	uniqueidentifier	No
Asset_State	nvarchar(32)	Yes

Owners

Column Name	Condensed Type	Nullable
UserId	nvarchar(450)	Yes
Owner_Notes	nvarchar(MAX)	Yes
Owner_Displ...	nvarchar(80)	Yes
OwnerId	uniqueidentifier	No

Markets

Column Name	Condensed Type	Nullable
MarketId	uniqueidentifier	No
OwnerId	uniqueidentifier	No
AssetId	uniqueidentifier	No
Market_BeginDate	datetime2(7)	Yes
Market_EndDate	datetime2(7)	Yes
Market_SalePrice	int	No
Market_ReservePrice	int	No
Market_LowestBid	int	No
Market_ViewFlags	int	No

MarketBids

Column Name	Condensed Type	Nullable
MarketBidId	uniqueidentifier	No
MarketId	uniqueidentifier	No
UserId	nvarchar(450)	Yes
MarketBid_DTS	datetime2(7)	No
MarketBid_Ammount	int	No
MarketBid_Comment	nvarchar(400)	Yes
MarketBid_Flags	int	No

MarketMetas

Column Name	Condensed Type	Nullable
MarketId	uniqueidentifier	No
MarketMeta_...	nvarchar(80)	No
MarketMeta_...	nvarchar(MAX)	Yes

Entity State Machines

Entity State Machines are defined by ‘States’, ‘Actions’ and control ‘Flags’. They are inherently hierarchical and can often be decomposed into sub-states and micro-actions; controlled by the flags. They are often depicted by flowchart diagrams for the overall Main-Path presentation, but not for the complete description of them.

Market Auction

Market States

Pre-Sale: ready for auction start.
In-Sale: available for bids.
Off-Sale: temporary auction halt.
Reconcile: determine winning bid(s).
No-Sale: failed to sell.
Sold: Edition has been purchased.
Withdrawn: Owner has changed its mind.
Completed: Auction all done and finished.

Market Actions

Initialise: by Owner
Inhibit: by Owner
Withdraw: by Owner, cancel all bids.
Begin: by timed Task, start accepting bids.
End: by timed Task, finish accepting bids.
Reconcile: determine winning bid(s).
Buy-Now: a bid meets sale-price.

Market Flags

FastNSlow: Fast-Slow Market Auction (Fast not Slow)
Visible: Appears in the market listing.
PrivacyXX: Privacy Settings

Market Auction Bids

Bid States

Initial: on creation
Purchase: created Purchase record.
Authorised: bidder has had a hold put on their card.
Withdrawn: by Bidder
Request-to-Pay: winning bid for ‘Slow’ auctions
Cancelled: losing bid or failure
Winner: funds have been collected from winning bidder.

Bid Actions

Initialise: by Bidder
Withdraw: by Bidder
Cancel: by reconciler

Bid Flags

PrivacyXX: Privacy Settings
InhibitXX: bid control

Purchases

Purchase States

Initial: on creation
On-Hold: Authorised payment at Hold
In-Process: Authorised payment, updating data-tables
Capture: Asset assigned, awaiting payment
Completed: Purchase finished successfully
Failed: problem with purchase
Cancelled: buyer changed its mind, or failed auction-bid

Purchase Actions

Purchase Flags

Stripe-PrePend: short string for Stripe API Purchase-Id. 'E' = Limited Edition, 'B' = Bid Auth & Hold, 'M' = market auction purchase, 'C' collect payment.

Examples of Use of States and Flags

Market-Index cs Model

```
MarketList = await _context.Markets.Include(m => m.Asset).Include(m => m.Owner)
    .Include(m => m.Asset.Artwork).Include(m => m.MarketBids)
    .Where(m => m.Market_BeginDate < DateTime.Now && m.Market_EndDate > DateTime.Now
            && m.Market_ViewFlags.HasFlag(Market.ViewFlag.Visible))
     .Where(m => ! new[]{"Completed","Sold","No-Sale"}.Contains(m.Market_State))
    .ToListAsync();
```

Market-Details cs-html page

```
<h4>Auction Bids</h4>
@if (new[]{"Completed","Sold","No-Sale"}.Contains(Model.Market.Market_State))
{
    <p>
        This Market Auction is finished.
    </p>
}
else
{
    <p>
        <a asp-page="./BidCreate" asp-route-id="@Model.Market.MarketId">Make a Bid</a> |
    </p>
}
```

Auction Services – Midnight Reconcile function

```
    //# check auction state
     if (oMarket.Market_State == "Sold" || oMarket.Market_State == "No-Sale")
    { // auction over, auto complete after dwell-period
```

Stripe CLI Commands

Saved here to relieve Stripe CLI typing, in the Command Prompt windows.

```
cd ../../stripe

stripe listen --forward-to https://localhost:44315/API/StripeWebHook

stripe logs tail
```