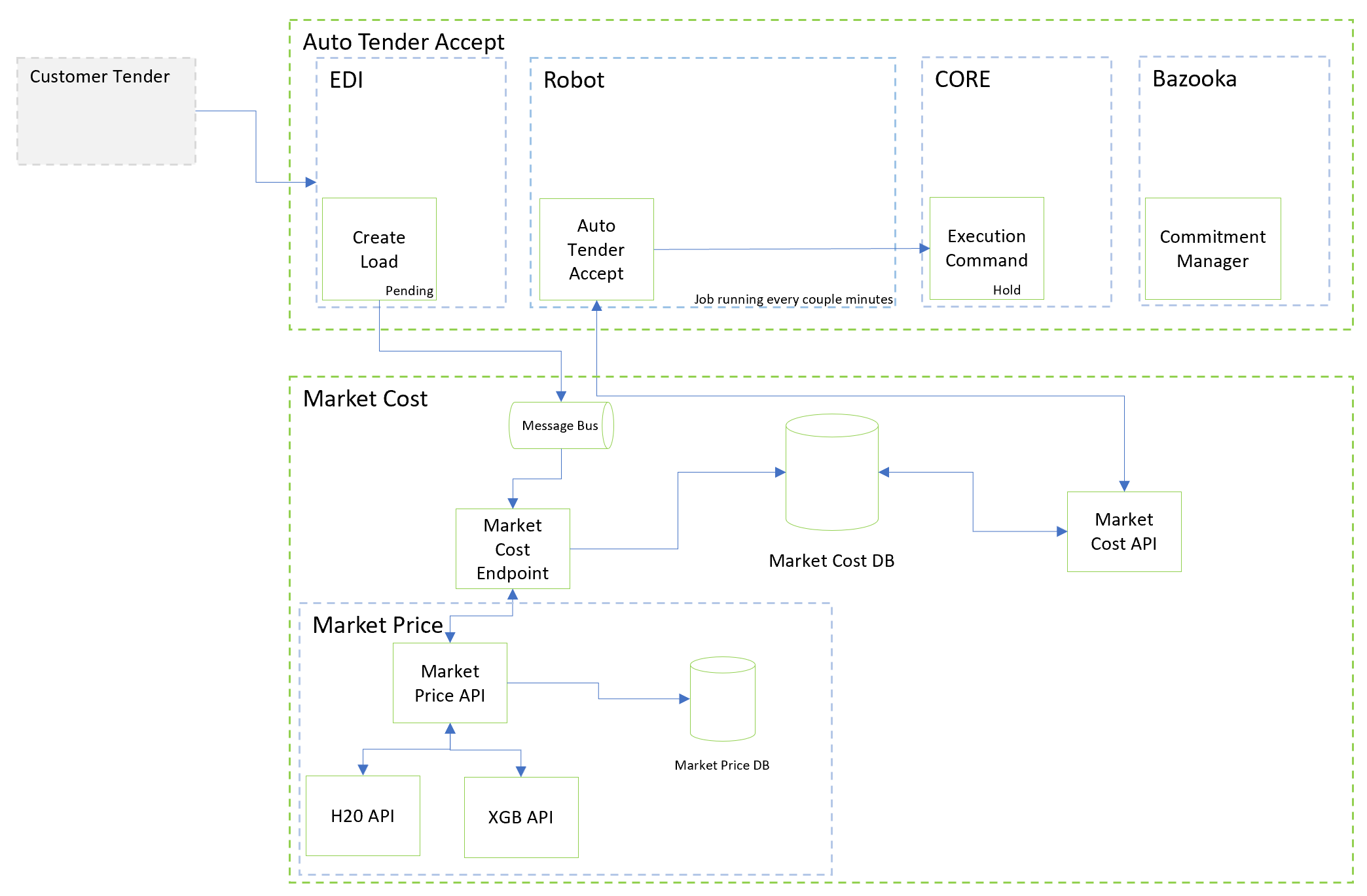
Auto Accept Tender

# Current Workflow

As an organization we want to auto accept tenders that come from customers if the customers rate exceeds the market cost plus a margin along with the rate is under the customer rate quote’s commitment and capacity values. This workflow has been built on top of several other existing initiatives.

1. Customer Tenders through EDI
2. Market Cost/Market Price initiatives
3. Commitment Manager for pending loads

With this implementation we are utilizing a Robot, Market Cost/Price and a Core service (execution command) to identify and progress the Load to a hold state if we decide we want to accept the tender. Below is a diagram of the workflow.



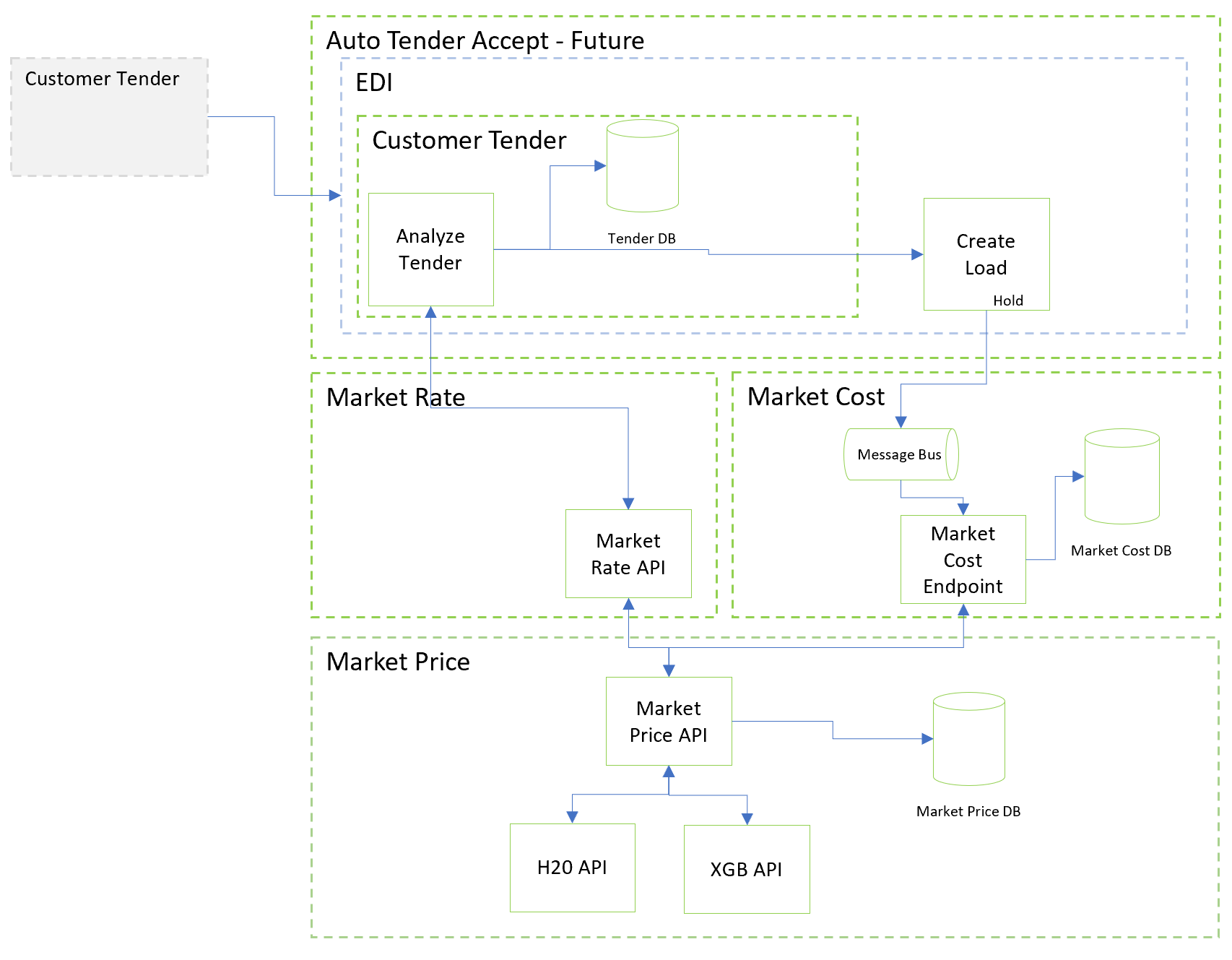
# Concerns

Though we are good to go live with this current implementation for all customers, there are long term concerns with this implementation.

1. We are creating loads for every customer tender regardless if we actually accept that tender. The initial phase is to create and put the load in a pending state.
2. We call Market Cost when a load is created. Since we already incur this expense today, rolling all customers on the robot won’t impact the flow negatively. Furthermore, we don’t really need to call the market cost API, but since it is just querying our database at that point and not requesting a new price there isn’t an added cost.
3. We are adding further calls to our Core service, execution command. As this service is overall impacting our systems negatively, adding more load to it is not a great long-term solution.
4. We are utilizing Commitment Manager to help reps decide if they still want to accept the tender. However, at this point they are analyzing a load, not the tender. We have noticed that many are manually cancelled.
5. We have no way to identify and analyze the customer tender. Once a load is created we really don’t have a clear picture of the tender and why we have accepted it.
6. If we want to continue adding components to the workflow, for example they want to reject the tender we should not add that component within the current scope of the workflow. We would essentially be auto cancelling loads that didn’t need to be created.
7. Market Price is an expensive service. At times a call to it may take 2-5 seconds. This service isn’t sustainable for volume growth and will hit a point of contention as our throughput increases.

# Future Workflow

The future workflow should be to identify the tender and apply rules before we create the load. This will offer flexibility to not only analyze the tender in all states but also build a smarter workflow with managing load creation. On average we cancel 350k loads a year. Minimizing this value will not only help within this workflow but also help our overall Enterprise performance. Below is a diagram for the future state.



We must also re-invest in the Market Price Data Science APIs. They are currently single-threaded which doesn’t allow for us to scale within a productionized environment. Below are metrics currently around our Pricing models. We already are at a point of contention. Adding volume, throughput will not only make these services less performant but also cause potential production issues to both this auto tendering initiative and for overall Market Cost initiatives related to other workflows.

