Sprint 2

GOAL: Automatically evaluate all incoming requests and determine which ones can be approved without the need for a manual review

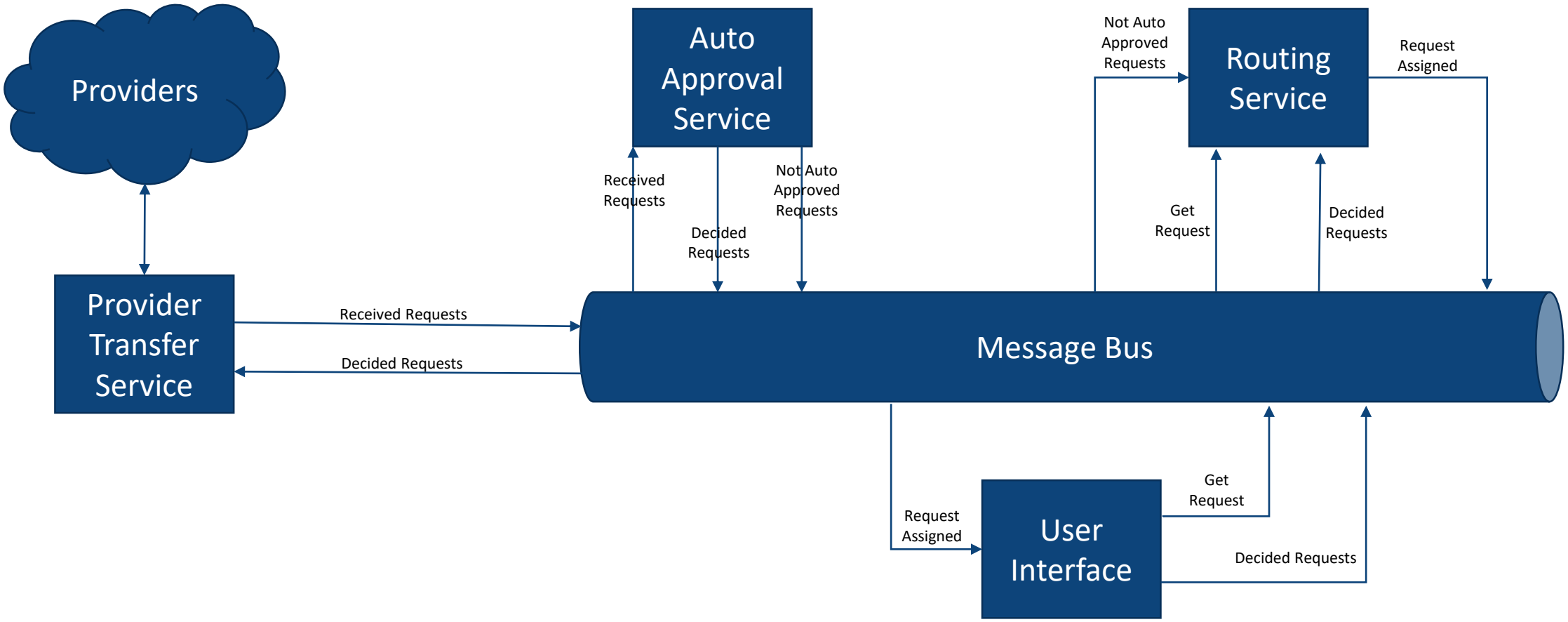
User Story 2-1

As a TAH physician, I need all incoming requests for service to be evaluated for automatic approval so that I can dedicate my time to more complex requests which require my expertise to make a decision.

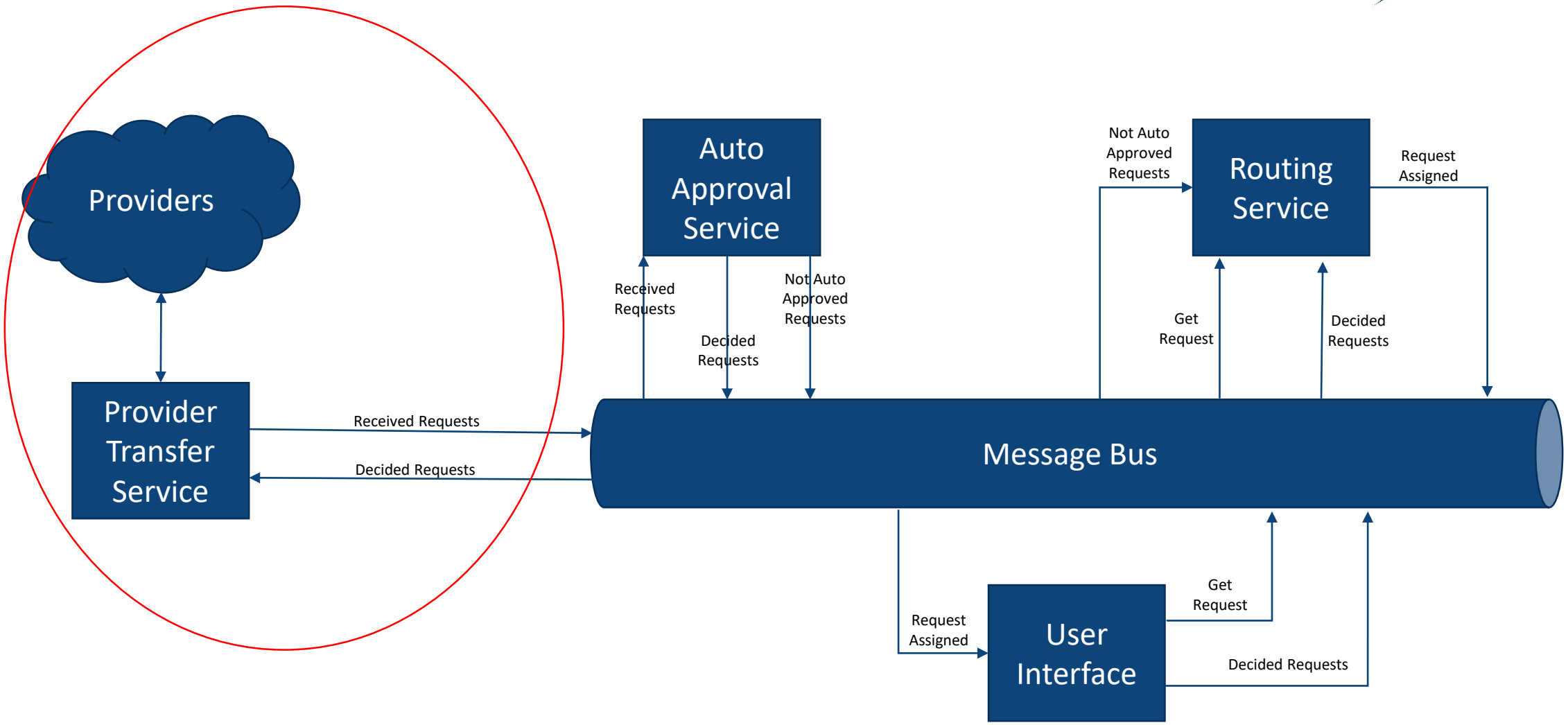
Acceptance Criteria

1. All incoming requests must be evaluated by Auto Approval Service
2. For each request, the Auto Approval Service must publish a result indicating whether or not the request was automatically approved
3. Service must be able to handle up to 10,000 requests per minute

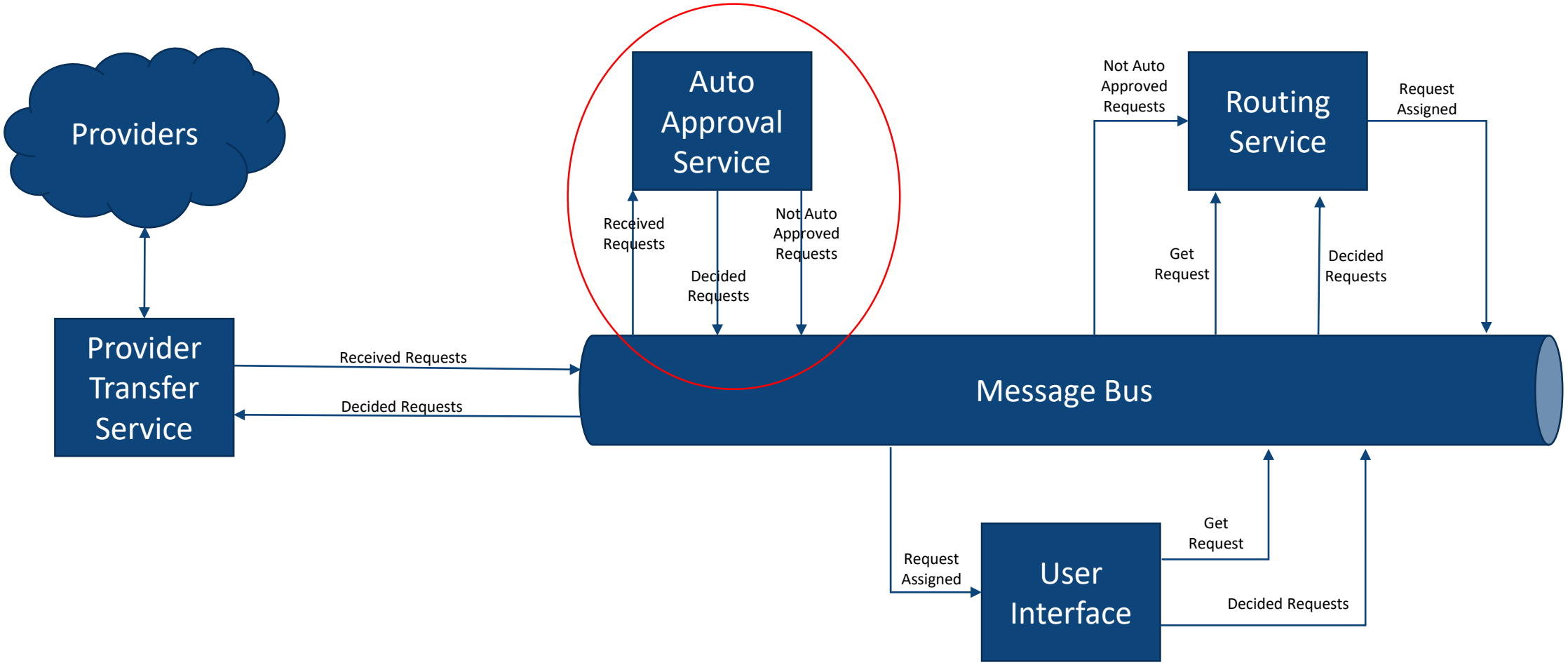
Final Solution



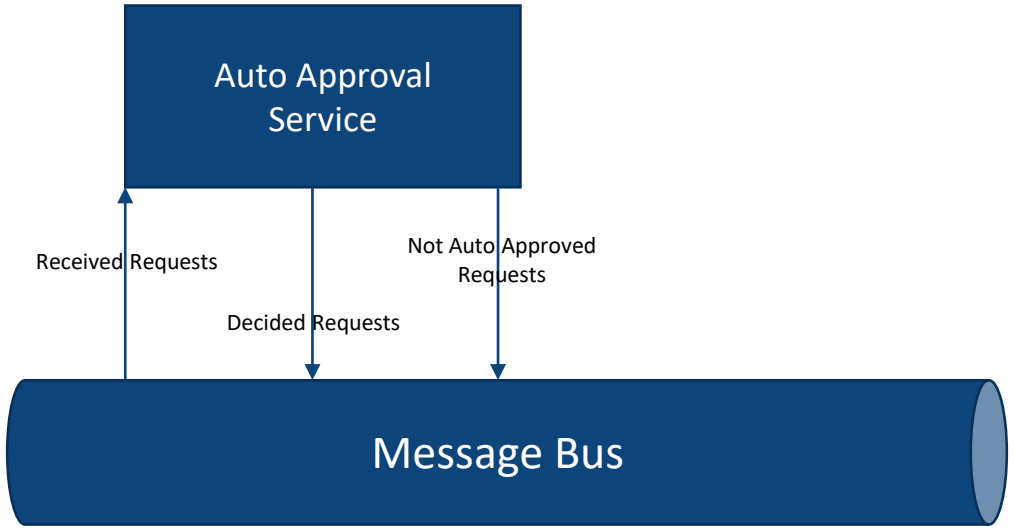
Final Solution



Final Solution



User Story 2-1: Design



- Create our first Consumer
- Simulate logic which evaluates the requests for automatic approval
- Publish result of auto-evaluation to the Message Bus

User Story 2-1: Message Bus Design

- Update Existing Received Requests Event Hub
  - Create a Shared Access Policy for consumer
  - Create a Consumer Group for the Auto Approval Service
- New Event Hubs
  - Publish Auto-Approved Requests
  - Publish Not Auto-Approved Requests
  - Shared Access Policies for publisher/sender

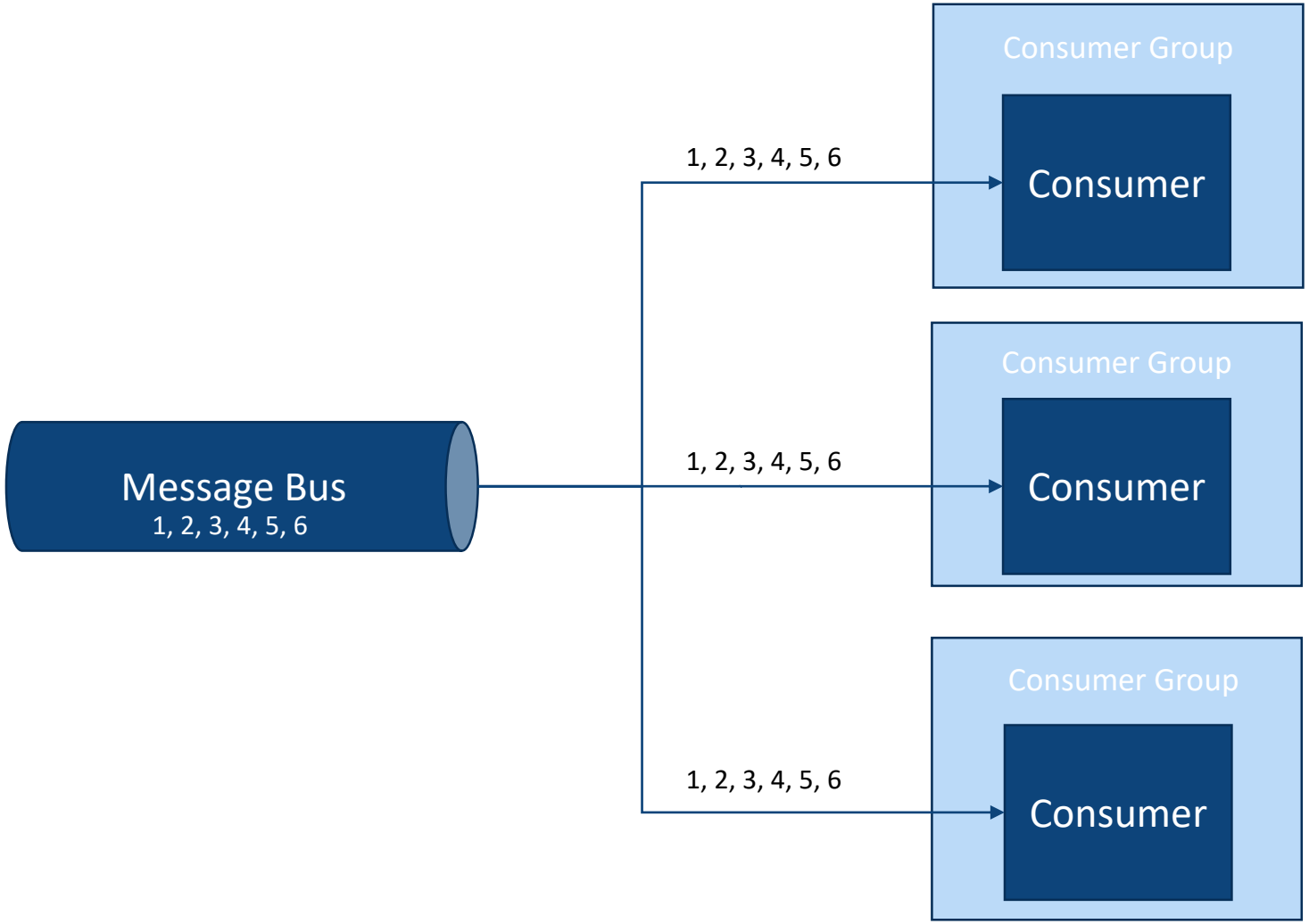
***Note: This isn't necessarily how you would organize events/hubs in a true production system*

User Story 2-1: Message Bus Design

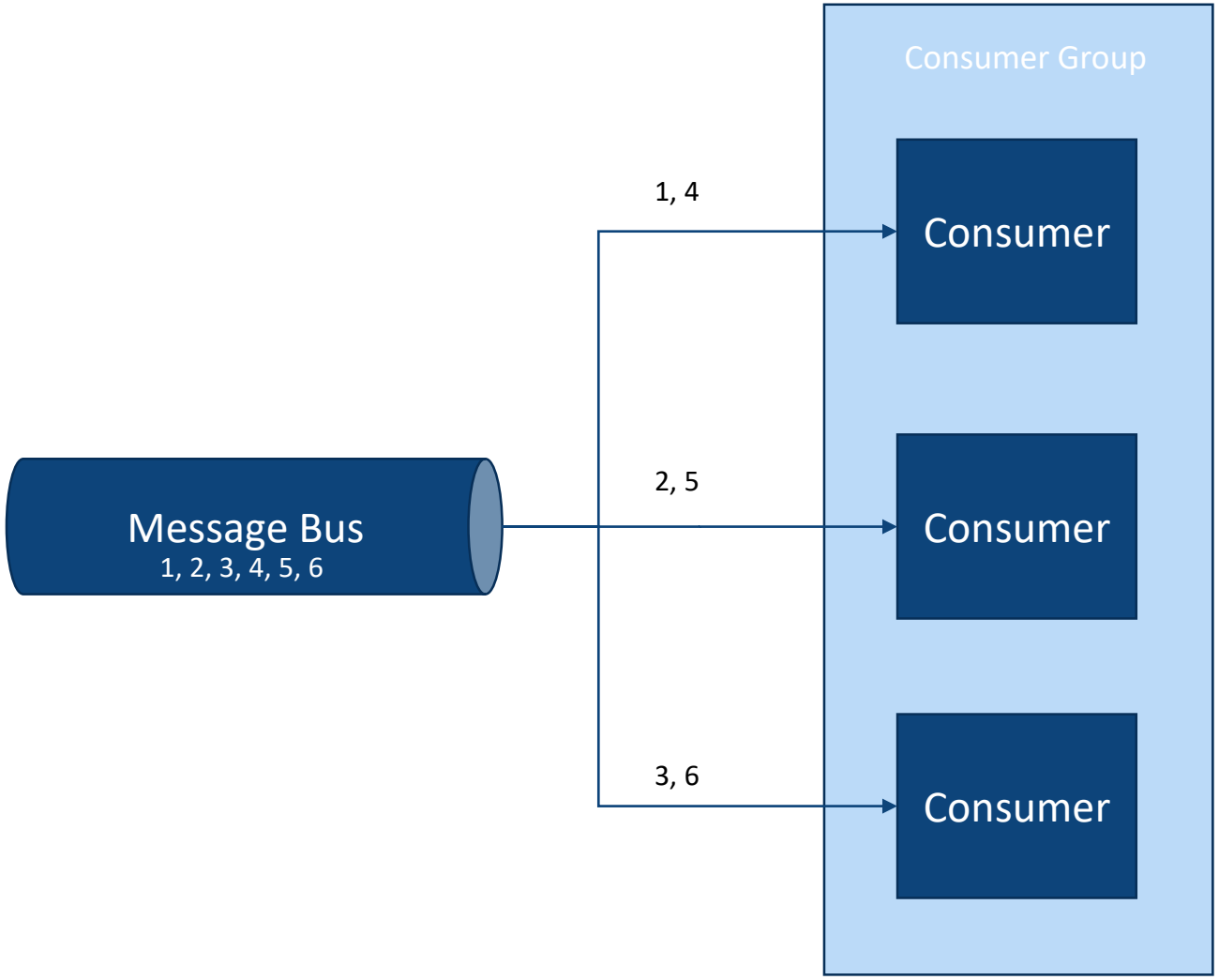
- Update Existing Received Requests Event Hub
  - Create a Shared Access Policy for consumer
  - **Create a Consumer Group for the Auto Approval Service**
- New Event Hubs
  - Publish Auto-Approved Requests
  - Publish Not Auto-Approved Requests
  - Shared Access Policies for publisher/sender

***Note: This isn't necessarily how you would organize events/hubs in a true production system*

User Story 2-1: Consumer Groups



User Story 2-1: Consumer Groups



User Story 2-1: Auto Approval Service Design

- Choosing Azure Function
  - Azure's event-driven, serverless compute option
  - Automatically scales – up to 200 parallel instances
  - We'll discuss other hosting options in later sprint

- Possible triggers
  - HTTP call
  - Timer
  - **Message arrived / event occurred**
  - Database record created, updated, or deleted
  - BLOB created, updated, or deleted

- Most often used in consumption mode – only charged for use
  - Based on number of executions, execution time, and memory used
  - Generally inexpensive for lightweight, infrequently used functionality

Task 2-1: Update ReceivedRequests Event Hub

- Develop in Azure Portal
- Create Listener Shared Access Policy for Consumers
- Create Consumer Group for Auto Approval Service

Task 2-2: Create Hubs With SAPs For Two New Events

- Develop in Azure Portal
- Create *DecidedRequests* and *NotAutoApprovedRequests* hubs
- Create Sender Shared Access Policies for Publishers

Task 2-3: Create Auto Approval Service

- Visual Studio
- C# Azure function with event hub trigger
- Configure connection to *ReceivedRequests* hub
  - SAP listener connection string
  - Consumer group
- Call *AutoEvaluator* (in Business Logic project) to determine approval
- Update status & call publisher for approvals
- Call publisher for non-approvals

Task 2-4: Create Event Publisher

- Visual Studio
- C# class
- Called from our Auto Approval Service function
- Configure connection to *DecidedRequests* and *NotAutoApprovedRequests* hubs
  - SAP sender connection strings
- Create methods for publishing to hubs

Task 2-5: Deploy Auto Approval Service

- Publish from Visual Studio to Azure
- Publish
- **MAKE SURE TO GET THE APPSETTINGS**
- Ensure messages are publishing to *DecidedRequests* and *NotAutoApprovedRequests* hubs

Sprint 2: Retrospective

- We now have an automated process to reduce our clinical staff's workload and enable TAH to process more requests
- Taking advantage of scaling and asynchronicity
  - Function can scale up without risk of missing or re-processing events even if it goes down for a period of time
  - Publishing messages with little concern for who will consume them or how many consumers there will be
- Enabling real-time processing without adding strain to the system
  - Live view of all active requests
  - Returning decisions to providers as soon as they are made