Scheduling Standards Consortium

API Requirements

Version 1.0.0

Published on October 02, 2023

Introduction

Problem Statement

Appointment Scheduling is a major point of friction for carriers, brokers, and shippers alike. Complex and nuanced appointment scheduling processes lead to inefficiencies when procuring and managing appointments across the freight industry. The SSC aims to simplify the integration of systems across the fragmented ecosystem that exists today. We anticipate a standard set of communication protocols will enable shippers and carriers to more seamlessly connect with each others' systems, enabling efficient data sharing and greater efficiencies across the supply chain.

Objectives

The primary objectives of the SSC are to (1) define an API standard for sharing scheduling information, (2) eliminate manual processes by automating interactions where possible, (3) implement the standardized interfaces and integrations across core systems, and (4) advocate for the standard across the industry. The SSC aims to partner with shippers, carriers, brokers, and solutions providers to drive towards a standard. The purpose of this document is to start the conversation and to provide an overview of the key workflows that the scheduling API will need to support. The SSC will continue to refine documentation informed by feedback from the industry.

Benefit

Aligning as industry leaders on a set of standards will simplify the integration of systems across the fragmented ecosystem between shippers, carriers, and intermediaries. All such parties will benefit from simpler interfaces and integrations.

Key Theme

Given that existing TMS and Appointment Scheduling solutions, are the source of truth for appointment data and appointment restrictions, the SSC's proposal is to simplify carrier-side appointment logic and nuance, and empower the TMS's and Appointment Scheduling Vendor's to implement necessary rules and validations for appointment booking.

Definitions & Conventions

Key Definitions

- Load: A load is a truck load of goods that is moving between an origin and a destination, with 0-to-many stops in between. It is possible for a load to have multiple pickups and multiple deliveries. Each load has a unique identifier that will be defined by the TMS. This identifier will be referred to as the primary reference number.
- Stop: A stop is a point along a load's journey. For example, the first pickup location, the last delivery location, and any locations in between will be referred to as stops.
- Dock Groups: Dock groups are a predefined set of dock doors that are eligible for a particular load. Dock groups are configured in the TMS or scheduling software and can vary by a number of unique constraints including shipping vs. receiving, dry goods vs. refrigerated goods, unique constraints based on shipper/consignee, and unique constraints by carrier.
- Live Loading: Describes a load in which the freight will be loaded into or unloaded from the carrier's trailer upon their arrival at a location.
- Preloaded: Describes a load in which the freight will already be loaded onto a trailer that is onsite at a location. In this scenario, a carrier will bobtail to the pickup location (arrive without a trailer) or drop an empty trailer before attaching the loaded trailer.
- Drop: Describes a load in which the carrier is expected to unhook the loaded trailer in the yard or at a dock door upon arrival. Carrier may bobtail out or depart with another trailer from the location.

Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119, and RFC 8174 when, and only when, they appear in all capitals, as shown here.

Scope

In Scope

- Core Systems Involved: The first phase of this initiative will primarily focus on defining standards to facilitate appointment scheduling transactions between carriers/brokers and shipper TMS dock scheduling applications. We aim to extend these standards to standalone dock scheduling solutions (i.e. non-TMS scheduling solutions) in the next phase of this initiative.
- Load Types: The first phase of this initiative will focus on over-the-road full truckload. Multi-stop loads (i.e. loads that pickup or dropoff at multiple locations) are also included as in scope.
    - As referenced above, a load is a truckload of goods that is moved between 1 and N locations.
- Equipment Types: This first phase of this initiative will support the 3 primary full truckload equipment types: Van, Reefer, Flatbed.
- Loading Types: Live, Drop, and Preload appointment scenarios will be supported.

Out of Scope

- Intermodal
- Transload
- Rail
- LTL
- Drayage

API Communication Options

Overview

With intersystem communication via API, usually, the expectation is for synchronous responses whereby the client contacts the server and a response is given within a reasonable service-level agreement (SLA), usually sub-second. However, there will be occasions when the response cannot be provided back in realtime. This could be due to manual tasks that must be executed by users of the system which the server supports or because the process to retrieve the correct answer takes additional time. In these scenarios, an asynchronous response option to the client needs to be provided whereby the client receives the response at a later time. Asynchronous communication options include webhooks, callbacks, and polling. With the appointment scheduling specification, the SSC does not want to prescribe a communication method or expectation but to provide a specification that allows for all possible options so the most appropriate one can be used, given the situation. In the following section, we describe four mechanisms that may be implemented to establish communication for data exchange.

Synchronous

Synchronous communication is the RECOMMENDED option out of all API communication options as it provides the client with an immediate response and is easiest to implement. When using synchronous APIs, blocking behavior on the server side is expected since the client will pause waiting for a response. See the “Best Practices” section for more details on implementation recommendations.

Asynchronous

Asynchronous communication is preferred in the appointment scheduling space for systems which require more time to process requests before responding to clients. In contrast to synchronous communication where the client waits for an immediate response from the server before continuing, asynchronous communication enables clients to submit requests and continue their work while the server processes their requests in the background, with the intent to receive responses when they become available. This section explains different implementation options for asynchronous communication.

Webhooks

A webhook is an HTTP-based callback function that allows lightweight, event-driven communication between two applications. To set up a webhook, the client gives a unique URL to the server API and specifies which event it wants to know about. Once the webhook is set up, the client no longer needs to poll the server; the server will automatically send the relevant payload to the client’s webhook URL when the specified event occurs. This is the RECOMMENDED way to communicate asynchronously.

Callbacks

A callback is a mechanism provided by clients to be executed by the server once a specific process is completed. Similar to webhooks, this mechanism is commonly used in APIs that require a later response once a process is invoked. Whereas webhooks are registered once prior to processes being invoked, clients provide

a callback URL to the server with each request where results are expected to be sent. This approach for communication is RECOMMENDED when a one-time registration is not possible.

Polling

Unlike webhooks and callback functions, polling is when the client, after submitting a request for a process to be executed, is responsible for contacting the server repeatedly to see if the process has been executed or not. The subsequent polling requests can be at regular intervals, random intervals, or exponential backoff. This option is NOT RECOMMENDED to be used except in cases where asynchronous responses are required but webhook or callback functions cannot be provided by the TMS. We RECOMMEND establishing reasonable expectations with clients up front regarding polling frequency and duration, and implement appropriate rate limiting mechanisms to guard against misbehavior. See the “Best Practices” section for more details on implementation recommendations.

Security

Enforcing API security involves the following components:

- Identification: The client/caller validates their identity and specifies their intended action or purpose.
- Authentication: The client/caller provides authentication credentials to gain access.
- Authorization: The system determines the granted permissions for the client/caller based on their identity and request.

Identification

Identification establishes the context for who is making a request and what the request is for. This is especially important in the cases where the clients are acting as a broker on behalf of multiple carriers. It is RECOMMENDED that the information be passed as an additional API header field that is supplemental to the API Header as defined by the SSC.

Authentication

The SSC does not define any requirements on specific authentication methods that may be available to support inbound API request authentication. However, it is RECOMMENDED by the SSC to use the authorization header as defined in RFC7235 with the OAuth 2.0 Framework following RFC6749.

API implementers are expected to publicly document the authentication method(s) they support. Clients are expected to comply with the security requirements of a given TMS / appointment scheduling solution in order to use the API.

Example

An example using the Authorization header is provided below.

```
Unset
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6Ikp
vaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQss
w5c
```

Authorization

The authorization of a call should include validating the authentication of the user, ensuring the user is able to act for a specific identifier, and is able to ultimately act on the request details sent in the request. The SSC currently does not require either party to standardize on a specific authorization method, for example, Access Control Lists (ACLs), Role Based Access Control (RBAC), Attribute Based Access Control (ABAC), etc. However we strongly encourage all requests to be validated.

Capability URLs

Capability URLs provide the ability for implementers of webhooks/callback functions to communicate to clients. When following best practices, the URL created by the client will be as secure as the client would want. If a client chooses to use Capability URLs, we RECOMMEND following the best practices provided by W3C. Below are several principles to consider when using capability URLs.

Capability URLs SHOULD Use HTTPS

Using HTTPS does not prevent all exposure of the URL but, generally, it is best practice to use HTTPS as much as possible to prevent accidental exposure of private information in the request.

Capability URLs SHOULD Be Shared Securely

The expectation is for the capability URL to only be shared during the process of registering a webhook or callback URL via the API to prevent leakage. Though sharing URLs through more manual means outside of the webhook or callback processes is NOT RECOMMENDED, the SSC acknowledges that there may be scenarios where URLs must be hardcoded into a system for proper response. In scenarios such as these, it is ultimately up to the client to share their URL information in the most secure way possible.

Capability URLs SHOULD Be Monitored

If any suspicious activity is observed while using a capability URL, it is RECOMMENDED to create a new capability URL and re-register the URL with each TMS. It is up to the client to determine what activity is considered “suspicious.”

Capability URLs SHOULD Be Changed Periodically

The risk of a capability URL being discovered increases with time and repeated usage. The best way to prevent accidental discovery outside of IP or referrer defenses is to expire and change the URL periodically. The frequency of these changes should be discussed with your security team and follow any best practices defined by your organization.

Adding Custom Properties

For the best integration experience for all parties, the SSC encourages the use of the API specification as-is. However, implementers may have specific requirements which may necessitate the addition of custom properties to the specification. To ensure custom fields do not overlap with future potential additions, it is encouraged to follow these practices for any implementer-specific additions.

API Headers

Implementers SHOULD only add custom properties to the API Headers that extends the “context” of the request being placed. Practices should follow standards placed in RFC7231 Section 5. Additional custom properties that can be included in the header is limited to the following:

- Security Authentication Credentials (RFC7231 Section 5.4)
- Who the request is on behalf of. This can be handled in the following:
    - Authentication Credentials
    - Custom API Header
- Who the request is for. This can be handled in the follow ways:
    - Custom API Header

Furthermore, implementers SHOULD NOT add custom properties to the API headers that should be a part of the body of the API specification.

Request and Response Bodies

Similar to RFC6648 Section 3, creators of new properties to be used:

- SHOULD assume that all properties they create might become standardized, public, commonly deployed, or usable across multiple implementations;
- SHOULD employ meaningful property names that they have reason to believe are currently unused;
- SHOULD expect to continue to support any custom attributes wholly and entirely by themselves and for their clients without expectation of support from the SSC;
- SHOULD NOT prefix their property names with "X-" or similar constructs;
- SHOULD NOT expect that the SSC will continue to support custom request header, bodies, or response headers, bodies in future spec revisions;
- SHOULD NOT add custom properties to the body that should be a part of the header.
- MUST NOT remove or rename any parts of the published spec or change the data types of any published specification objects / parameters;
- MUST NOT define additional APIs that serve a similar purpose to an API in this specification. Additional APIs can be provided to help your clients; for example, a search API that has not defined by the SSC can be provided by a TMS to look up a required identifier in the specification if the identifier is not known to the client;
- MUST NOT burden clients with the responsibility of constructing facilities- or customer-specific request bodies, responses, or headers. For example, a client should not need to know before making a request to a TMS system that a particular customer at a particular facility requires a new property to be inserted into the request body, otherwise the request will fail.

In summary, adding a new property to the request body or headers SHOULD be avoided if possible as it forces more customization between the clients. Feel free to contact the SSC for additional field suggestions to add to the specification.

Best Practices

In this section, we will provide a list of best practices that should be considered when building an API that provides synchronous or asynchronous responses.

Customer- or Facility-Dependent API Variations SHOULD Be Avoided

While discouraged, the SSC is aware there may be times when an API implementer may need to add additional custom properties to request headers and/or request and response bodies. However, the SSC strongly discourages any such customizations that vary based on other aspects of the request context, such as which customer or facility the request is for. For example, requiring a field 'foo' when making a request for one customer but requiring a different field 'bar' when making a request to a different customer should be avoided. Similarly, defining supported values for a field that vary based on the customer or facility should also be avoided. Such variations greatly increase the complexity and effort required for a client to successfully integrate with scheduling systems at scale.

Requests with Load and Stop Information

Within the requirements, you may notice that it is required for TMSs / appointment scheduling solutions to provide a way for carriers / brokers to pass information to uniquely identify a load or a stop. The API specification defines these identifiers as follows:

- The load’s primary reference number.
- The load’s purchase order number.
- The load’s BOL number.
- The stop’s location identifier.
- The stop’s stop sequence number.
- The stop’s street address.
- The stop’s region.
- The stop’s locality.
- The stop’s country.
- The stop’s postal code.

However, none of these unique identifiers are required. This is intentional as the SSC does not want to prescribe for a system what information they may or may not need. Ultimately, it will be up to the discretion of the system to clearly define via their own documentation if specific identifiers laid out by the SSC are required or optional for their specific implementation. As a rule of thumb, you should provide additional information about the load or stop if you are uncertain the information you originally planned to provide will adequately identify both.

Set Reasonable SLAs

It is a good practice to set a reasonable SLA for clients to expect a synchronous response back from a request. SLAs and response times SHOULD be sub-second to no more than a few seconds at maximum. In scenarios where more than a few seconds are required to respond, asynchronous communication options should be considered.

Rate Limiting

There are risks that higher-than-expected traffic will deny other callers access to the server, including a caller issuing too many requests, misconfigured systems, or system bugs. To protect against this, it is RECOMMENDED that a TMS have account-level access control to rate-limit callers.

HTTP Status Codes

IANA keeps an official registry of HTTP status codes that lists the corresponding RFCs that define each status. When handling status codes, servers SHOULD use codes according to the semantics described in the corresponding RFC. Using the status codes correctly is important as it tells the client how to proceed after making a request. Definitions of the 4xx, 5xx status codes are not defined in this specification and are expected to be followed according to their standard RFC definitions.

Webhook/Callback URL Security

In an attempt to make interoperability as easy as possible, the SSC proposes to use capability URLs to better enable authentication when accessing a client system while also allowing the client to control the level of security they provide by following best practices as defined by W3C. See the “Capability URLs” section for more details.

Rescheduling Appointments and Asynchronous Responses

When a reschedule appointment request is rejected, the expectation is that the existing scheduled appointment is maintained. If the communication about the rejection back to the carrier / broker must be returned asynchronously via a Webhook, the SSC recommends that two responses are sent back where one is the failure of the request and the second is the reaffirmation to the carrier / broker that they still have the appointment.

For Your Information

Stale Stop Sequence Numbers

If the TMS supports using a stop sequence number to fetch or schedule appointments, it is possible that the stop sequence number changes, which is considered a load-level event. The appointment API defined by the SSC focuses on appointment change events, not load-level events. Depending on TMS implementation, a carrier who is registered to receive notifications about a stop's appointments may not receive a notification that the stop sequence number has changed.

Identifier Resolution

It is possible that certain identifiers may not be known to the client. This specification supposes both parties have resolved any identifiers before utilizing this API. To help clients resolve IDs, implementers MAY create endpoints that can be used to resolve these IDs for their clients. APIs created like this would be outside of the SSC ownership and should be treated as helpers for their clients. If you believe the API could provide value to the SSC, please contact a member of the SSC to further discuss the use case.

One example is location identifiers. To keep the scheduling API focused on scheduling concerns, the SSC is not addressing fragmentation in resolving addresses and locations. In the case that a TMS requires a locationId to be provided for a scheduling workflow, it is expected that the TMS provides a solution for the clients which will then use the locationId in the scheduling API.

Canceling Appointments

When canceling an existing appointment, scenarios could arise where the appointment that is being canceled has pending requests from the carrier / broker, e.g., carrier / broker submits a reschedule appointment request prior to making a decision to cancel the appointment altogether. In general, it makes sense that canceling an appointment would also result in the cancelation of any pending requests related to that appointment. However, TMSs / appointment scheduling solutions may have unique implementations that may require additional efforts internally or from carriers / brokers to resolve these scenarios. For these reasons, the SSC has chosen not to prescribe expected behavior, but encourages TMSs / appointment scheduling solutions to clearly outline their expectations in their documentation that they provide to carriers / brokers who choose to integrate with their technology.

Requirements

fetchAvailableAppointments

As a carrier / broker, I expect to fetch a list of available appointment times for a particular load on a particular day at a particular stop in a TMS / appointment scheduling solution.

- In a request, the carrier / broker expects to provide unique identifiers for three scenarios.
    - The carrier / broker MUST be able to seek a new list of available appointments with load and stop identifiers. See the “Requests with Load and Stop Information” section under Best Practices for more details.
    - The carrier / broker MAY be able to seek a new list of available appointments with an existing appointment identifier.
    - The carrier / broker MAY be able to poll for the status of an asynchronous request with a request identifier.
- In a request, the carrier / broker MUST be able to request available appointments using either an exact date or date range.
- The carrier / broker MUST be able to pass a callback URL in the request header if the carrier is expecting a callback response.
- In a response, the TMS / appointment scheduling solution can produce results for three scenarios.
    - If the information passed by the carrier / broker fails to find a unique match in the system, the TMS / appointment scheduling solution MUST be able to return failure responses. For this scenario:
        - The response MUST contain a status of failure.
        - The response MAY contain problem detail objects outlining why a particular failure occurred.
    - If the list of available appointments will be returned asynchronously, the TMS / appointment scheduling solution MUST be able to return pending responses. For this scenario:
        - The response MUST contain a pending status and a unique request identifier to assist with asynchronous response review.
    - If a list of available appointments will be returned synchronously or asynchronously via callback or polling, the TMS / appointment scheduling solution MUST be able to return success responses. For this scenario:
        - The response MUST contain a success status, the available appointments (specific appointments and/or appointment windows), and available appointment response types.
            - Available appointment response types SHOULD indicate if they can be scheduled synchronously or asynchronously, if the facility is first come, first served, or if they are preset and require confirmation.
        - The response MAY contain available appointment identifiers for scheduling and rescheduling, dock group and door assignment, and the location identifier.
        - The response MAY contain location address information, including street address, region, locality, country, and postal code.
        - The list of available appointment options SHOULD be restricted by necessary exclusions including, but not limited to, equipment type, commodity, consignee, “must-pickup-on”

and “must-deliver-by” dates, etc. It is important to ensure dock groups are properly configured so that only eligible times are returned.

- If there are no available appointments for the information passed by the carrier / broker, the list of available appointments in the success response SHOULD be empty.
- Asynchronous success responses via webhook MUST be communicated via carrier / broker-provided callback endpoints using the fetch-available-appointments event, defined in the "Webhooks" section

Figure 1: fetchAvailableAppointments with Synchronous Response

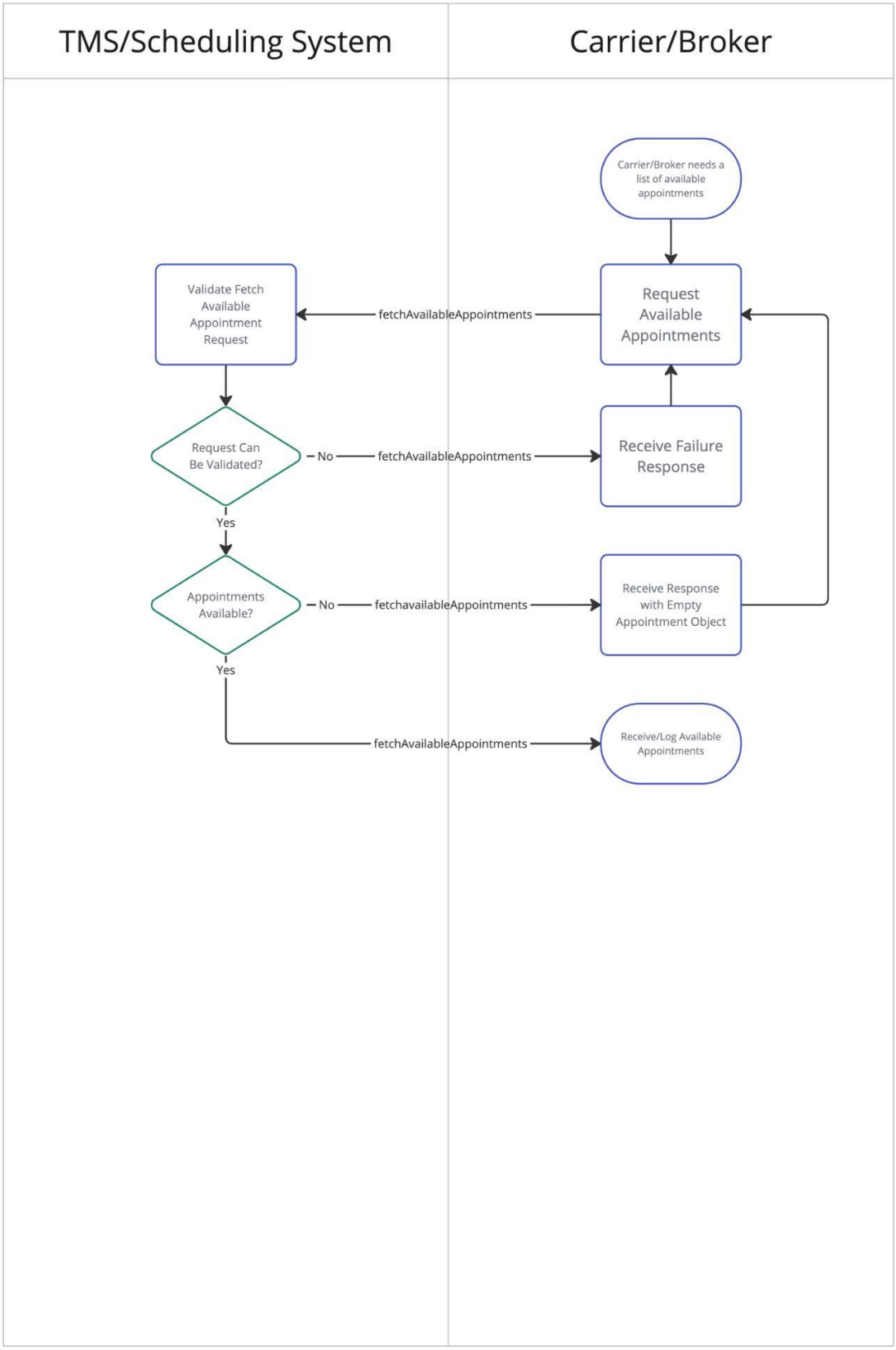


Figure 2: fetchAvailableAppointments with Asynchronous Polling

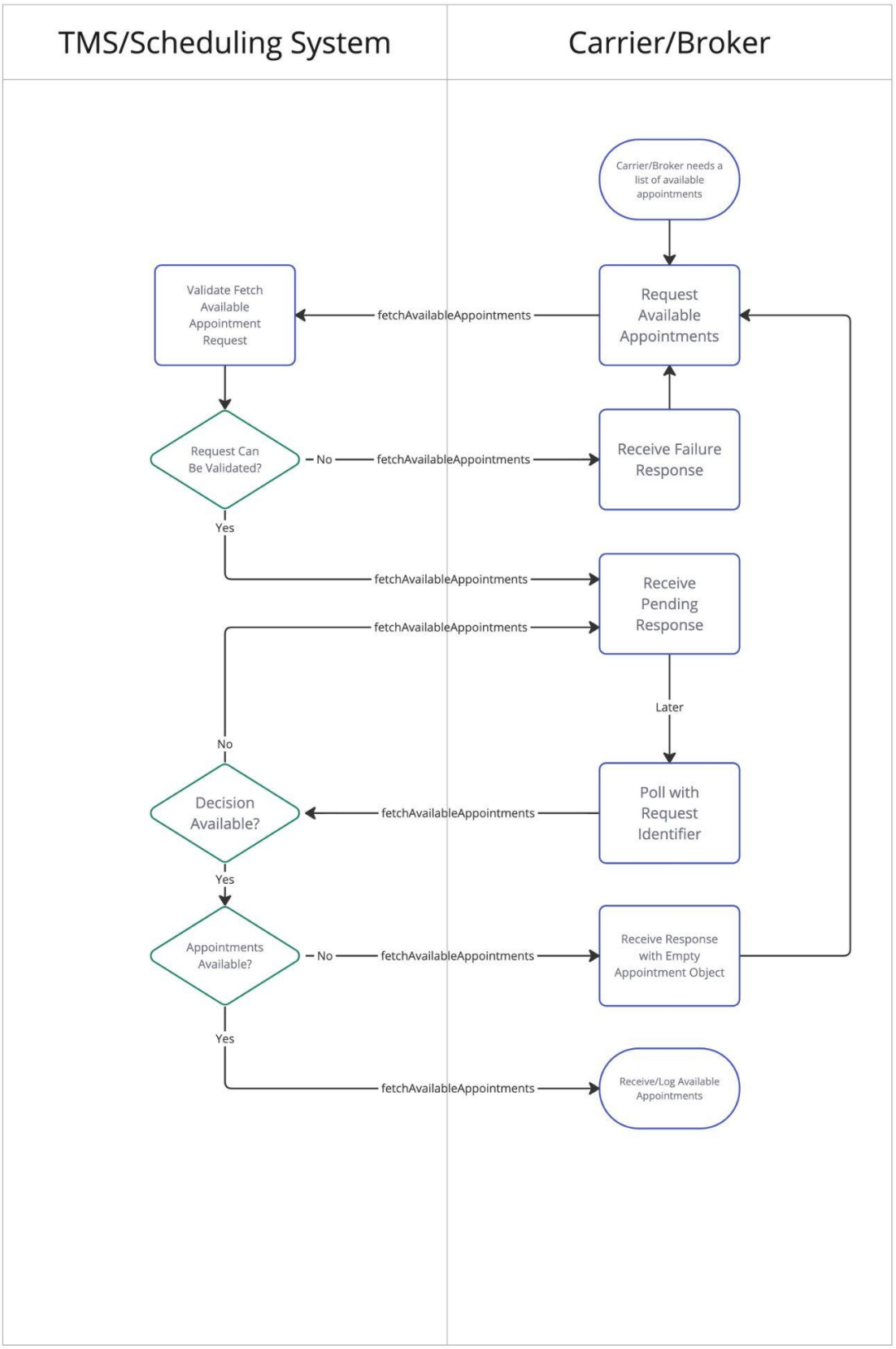
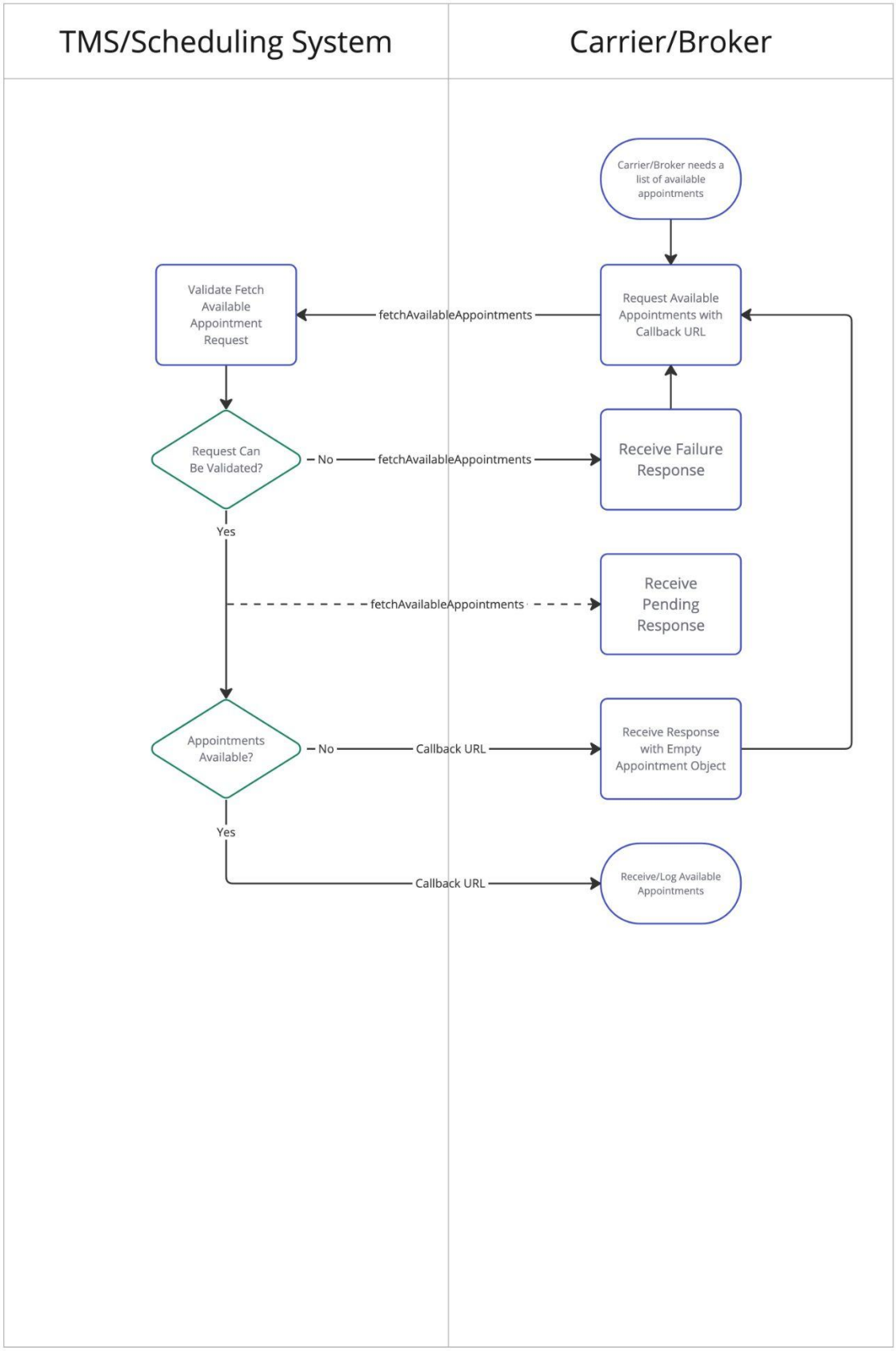


Figure 3: fetchAvailableAppointments with Asynchronous Response via Callback URL or Webhook



scheduleAppointment

As a carrier / broker, I expect to schedule available appointment times for a particular load on a particular day at a particular stop in a TMS / appointment scheduling solution.

- In a request, the carrier / broker MUST be able to schedule an available appointment with load and stop identifiers. See the “Requests with Load and Stop Information” section under Best Practices for more details.
- In a request, the carrier / broker expects to provide preferred appointment information for two scenarios.
    - The carrier / broker SHOULD be able to schedule a preferred available appointment with appointment detail information. In this scenario:
        - The carrier / broker MUST be able to pass arrival window information, including a start date and time and an appointment duration.
        - The carrier / broker MAY be able to pass an available appointment type, dock group, and dock door information.
    - The carrier / broker SHOULD be able to schedule a preferred available appointment with a unique appointment identifier. In this scenario:
        - The carrier / broker MUST be able to pass an available appointment identifier.
        - The carrier / broker MAY be able to pass an available appointment type.
- In a request, the carrier / broker MAY be able to pass reason codes and comments.
- If expecting to receive a response via callback, the carrier / broker SHOULD be able to pass a callback URL in the request header.
- In a response, the TMS / appointment scheduling solution can produce results for three scenarios.
    - If the information passed by the carrier / broker fails to find a unique match in the system or if the scheduled appointment request is rejected synchronously or asynchronously, the TMS / appointment scheduling solution MUST be able to return failure responses. For this scenario:
        - The response MUST contain a status of failure.
        - The response MAY contain problem detail objects outlining why a particular failure occurred.
    - If confirmation or rejection of a scheduled appointment request will be returned asynchronously, the TMS / appointment scheduling solution MUST be able to return pending responses. For this scenario:
        - The response MUST contain a pending status.
    - If confirmation of a scheduled appointment will be returned synchronously or asynchronously via callback or polling, the TMS / appointment scheduling solution MUST be able to return success responses. For this scenario:
        - The response MUST contain a success status, the appointment start date and time, and the appointment duration.
        - The response MAY contain additional appointment detail information including the scheduled appointment identifier, dock group, dock door, available appointment status, appointment confirmation number, and location identifier.
        - The response MAY contain location address information, including street address, region, locality, country, and postal code.

- Asynchronous success responses via webhook MUST be communicated via carrier / broker-provided callback endpoints using the appointment-changed event, defined in the "Webhooks" section.

Figure 4: scheduleAppointment with Synchronous Response for AUTOMATIC Appointment Types

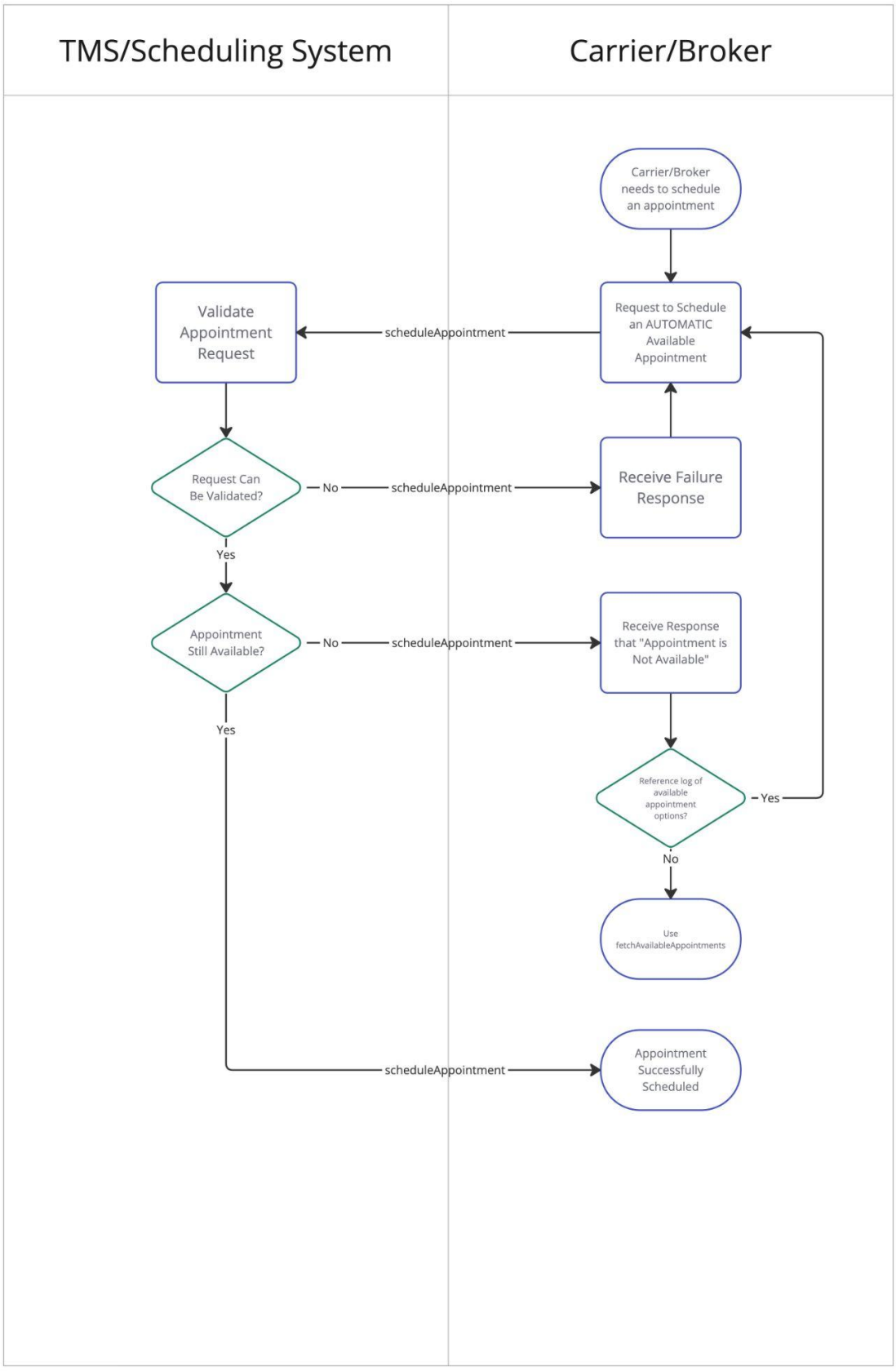


Figure 5: scheduleAppointment with Synchronous Response for CARRIER_CONFIRMATION_REQUIRED Appointment Types

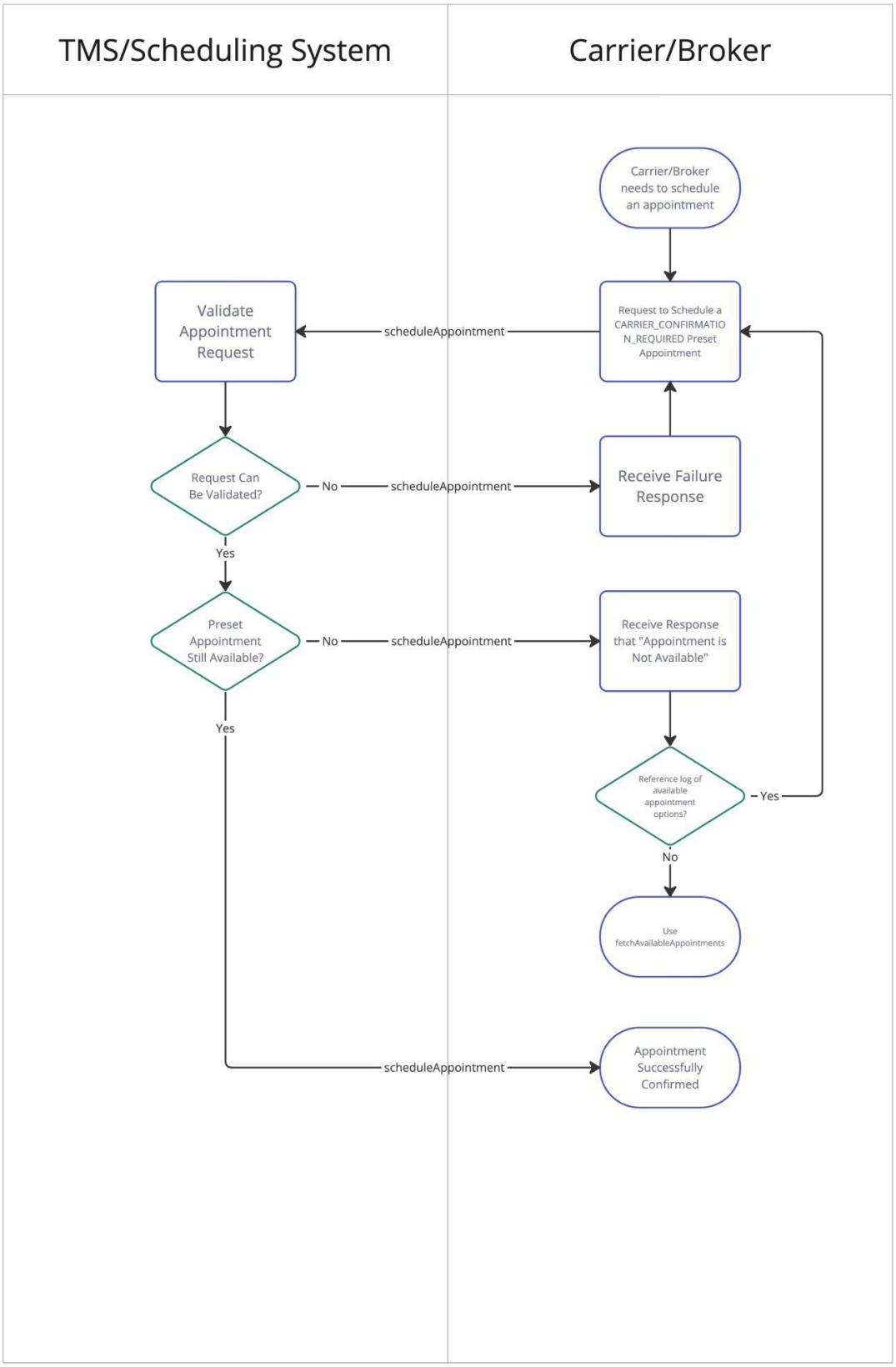


Figure 6: scheduleAppointment with Asynchronous Response for DEFERRED Appointment Types via Polling

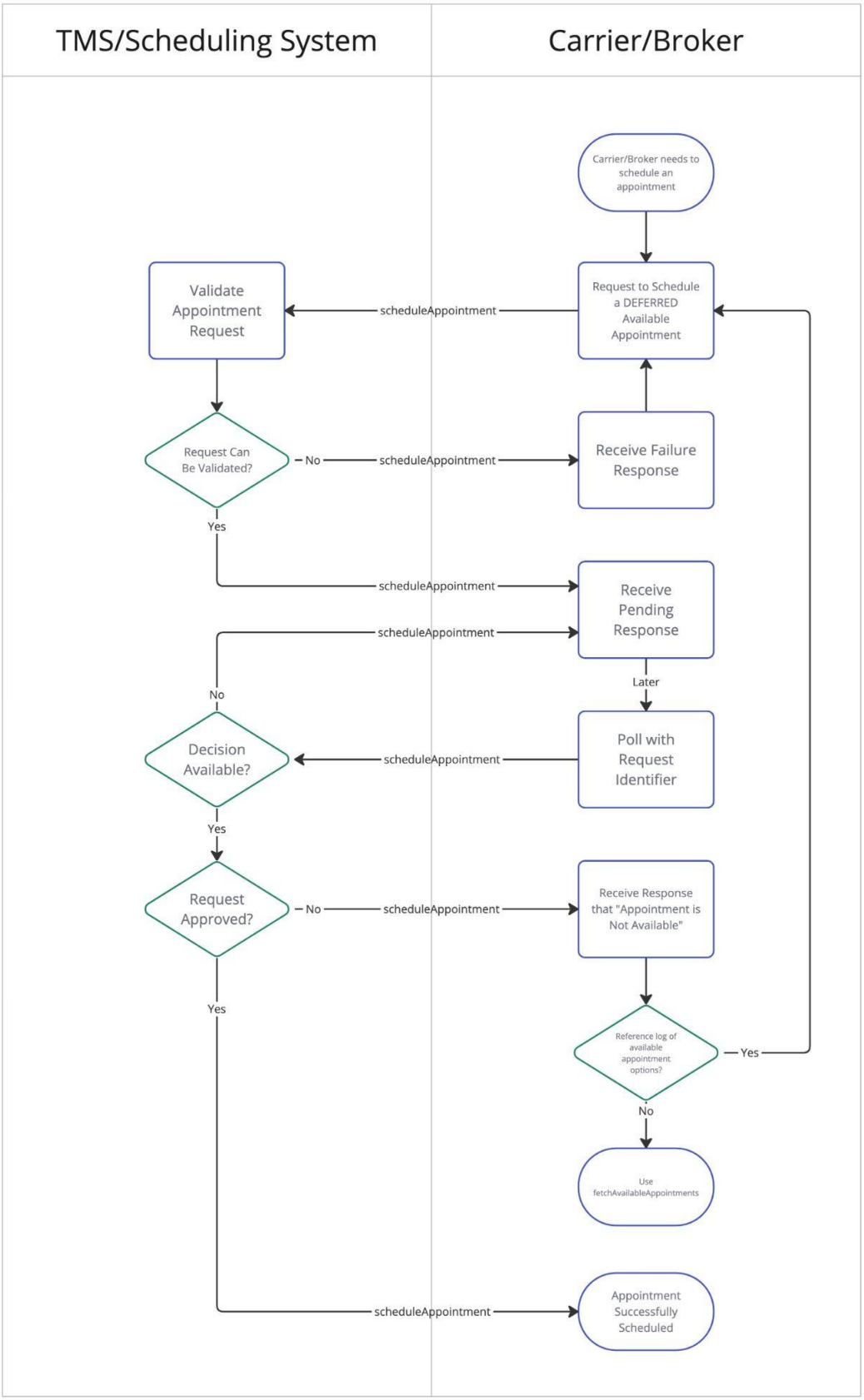
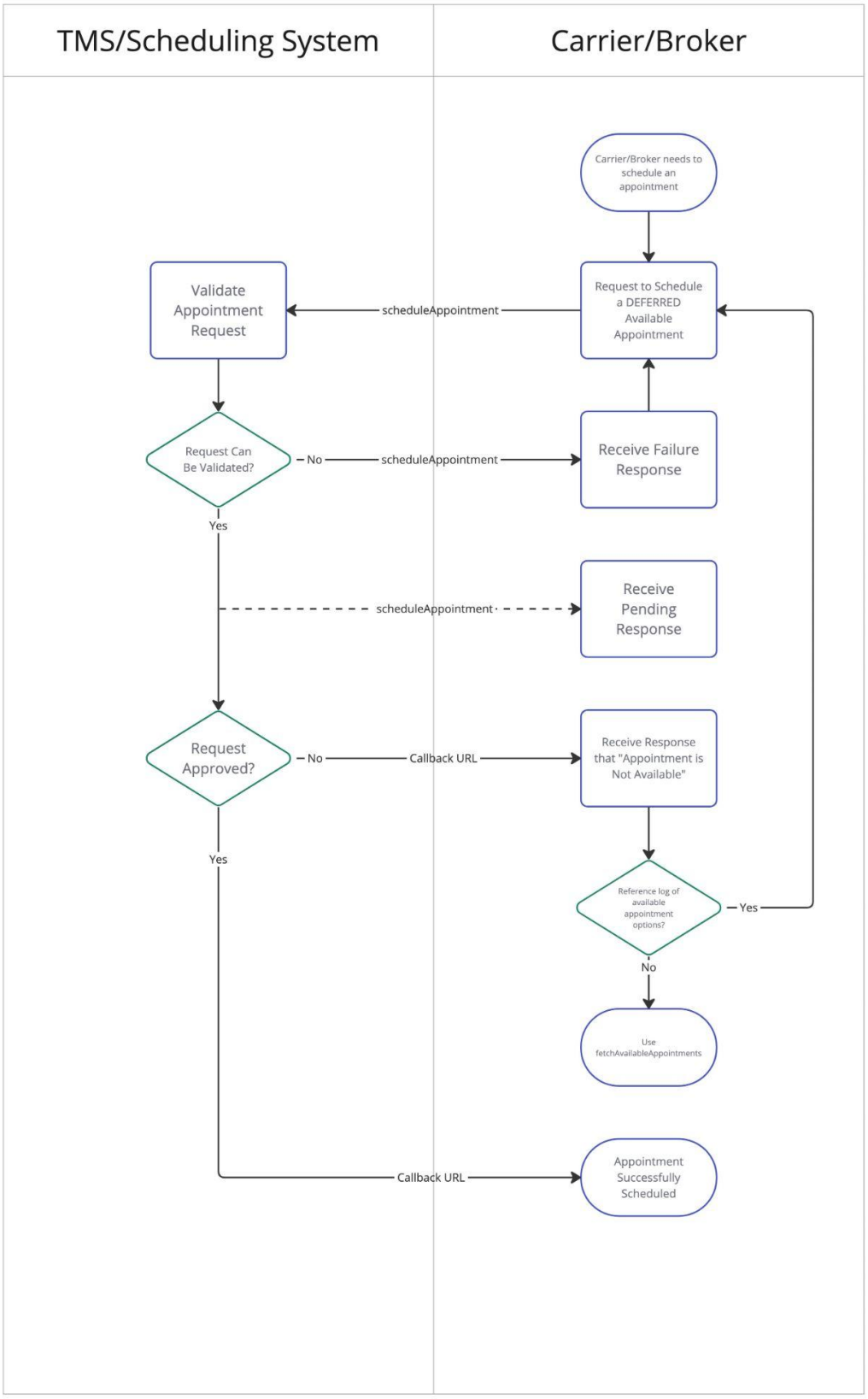


Figure 7: scheduleAppointment with Asynchronous Response for DEFERRED Appointment Types via Callback URL or Webhook



rescheduleAppointment

As a carrier / broker, I expect to reschedule available appointment times for a particular load on a particular day at a particular stop in a TMS / appointment scheduling solution.

- In a request, the carrier / broker expects to provide unique identifiers for two scenarios.
    - The carrier / broker MUST be able to reschedule an appointment with load and stop identifiers. See the “Requests with Load and Stop Information” section under Best Practices for more details.
    - The carrier / broker MAY be able to reschedule an appointment with a unique appointment identifier for their existing appointment.
- In a request, the carrier / broker expects to provide preferred appointment information for two scenarios.
    - The carrier / broker SHOULD be able to reschedule an appointment with new available appointment detail information. In this scenario:
        - The carrier / broker MUST be able to pass arrival window information, including a start date and time and an appointment duration.
        - The carrier / broker MAY be able to pass an available appointment type, dock group, and dock door information.
    - The carrier / broker SHOULD be able to reschedule an appointment with a unique appointment identifier for a new available appointment. In this scenario:
        - The carrier / broker MUST be able to pass a new available appointment identifier.
        - The carrier / broker MAY be able to pass an available appointment type.
- In a request, the carrier / broker MAY be able to pass reason codes and comments.
- If expecting to receive a response via callback, the carrier / broker SHOULD be able to pass a callback URL in the request header.
- In a response, the TMS / appointment scheduling solution can produce results for three scenarios.
    - If the information passed by the carrier / broker fails to find a unique match in the system or if the rescheduled appointment request is rejected synchronously or asynchronously, the TMS / appointment scheduling solution MUST be able to return failure responses. For this scenario:
        - The response MUST contain a status of failure.
        - The response MAY contain problem detail objects outlining why a particular failure occurred.
        - If the reschedule appointment request fails, the original appointment information for the carrier / broker should remain.
    - If confirmation or rejection of a rescheduled appointment request will be returned asynchronously, the TMS / appointment scheduling solution MUST be able to return pending responses. For this scenario:
        - The response MUST contain a pending status.
        - If the reschedule appointment request is pending, the original appointment information for the carrier / broker should remain until a confirmation or rejection in the TMS / appointment scheduling solution occurs.
    - If confirmation of a rescheduled appointment will be returned synchronously or asynchronously via callback or polling, the TMS / appointment scheduling solution MUST be able to return success responses. For this scenario:

- The response MUST contain a success status, the appointment start date and time, and the appointment duration.
- The response MAY contain additional appointment detail information including the rescheduled appointment identifier, dock group, dock door, available appointment status, appointment confirmation number, and location identifier.
- The response MAY contain location address information, including street address, region, locality, country, and postal code.
- Asynchronous success responses via webhook MUST be communicated via carrier / broker-provided callback endpoints using the appointment-changed event, defined in the "Webhooks" section.
- If the reschedule appointment request is confirmed, the original appointment information for the carrier / broker MUST be updated in the TMS / appointment scheduling solution. This MAY result in an updated appointment confirmation number.
- If the reschedule appointment request is rejected, the original appointment information for the carrier / broker should remain.

Figure 8: rescheduleAppointment with Synchronous Response for AUTOMATIC Appointment Types

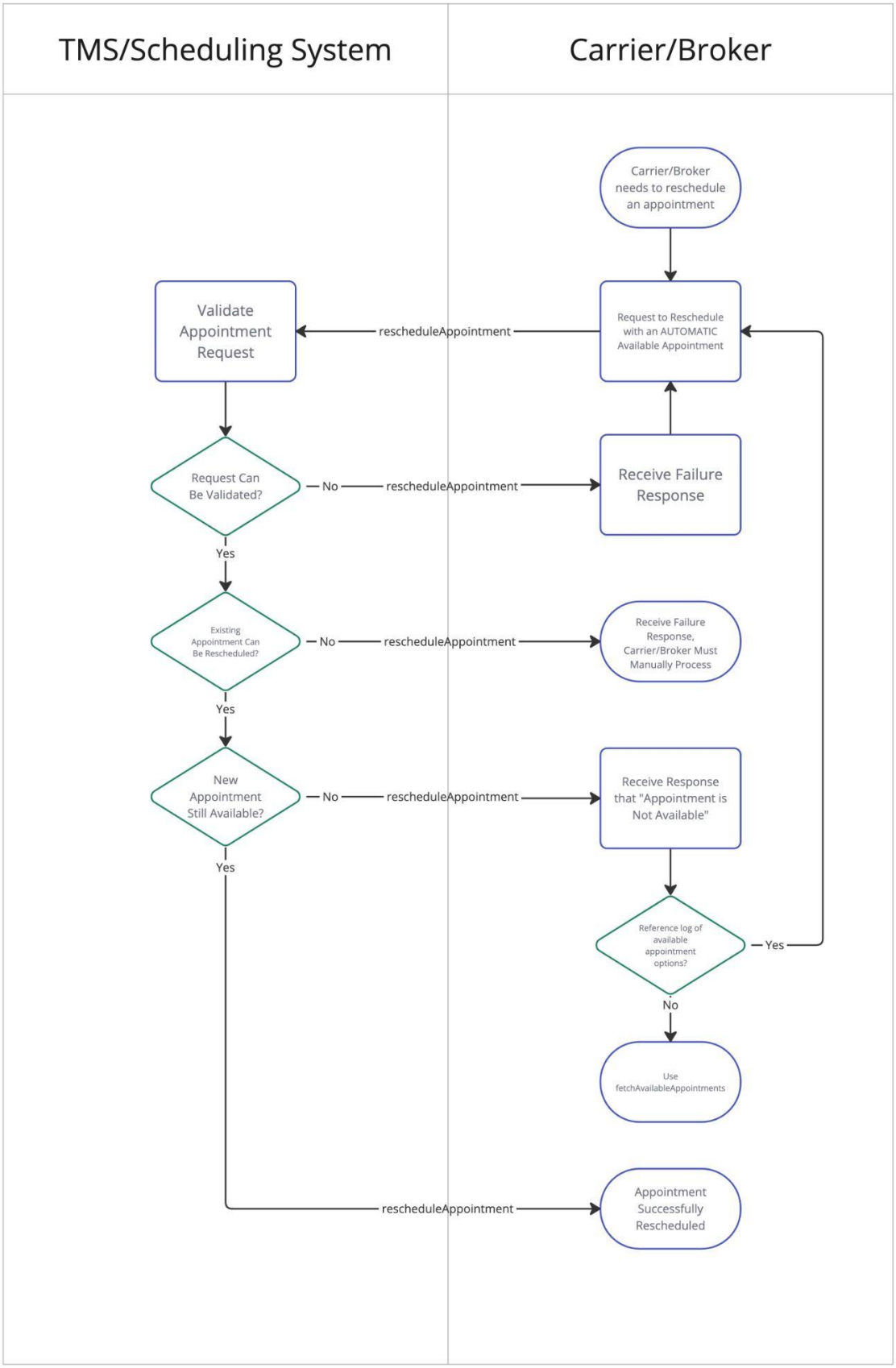


Figure 9: rescheduleAppointment with Asynchronous Response for DEFERRED Appointment Types via Polling

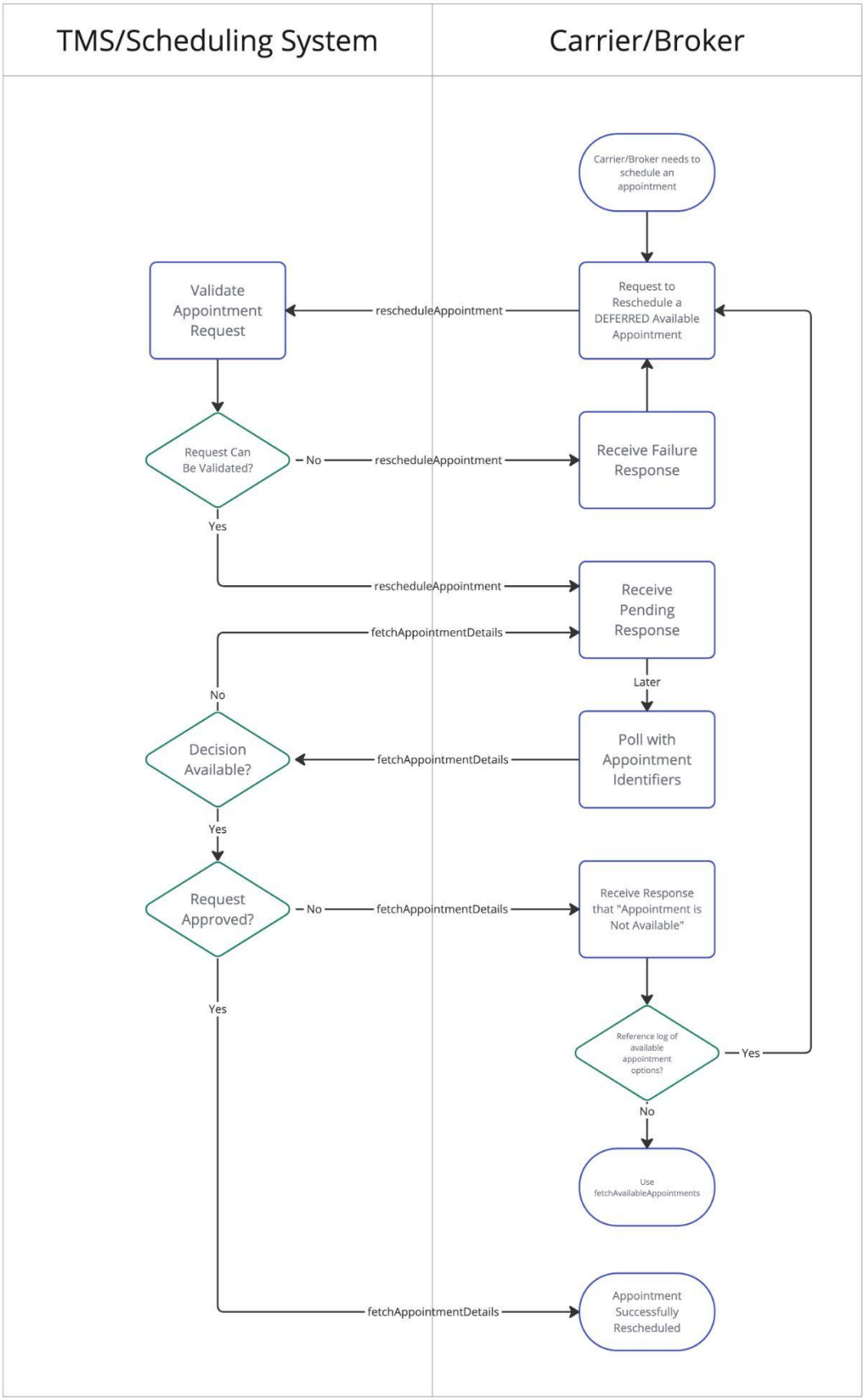
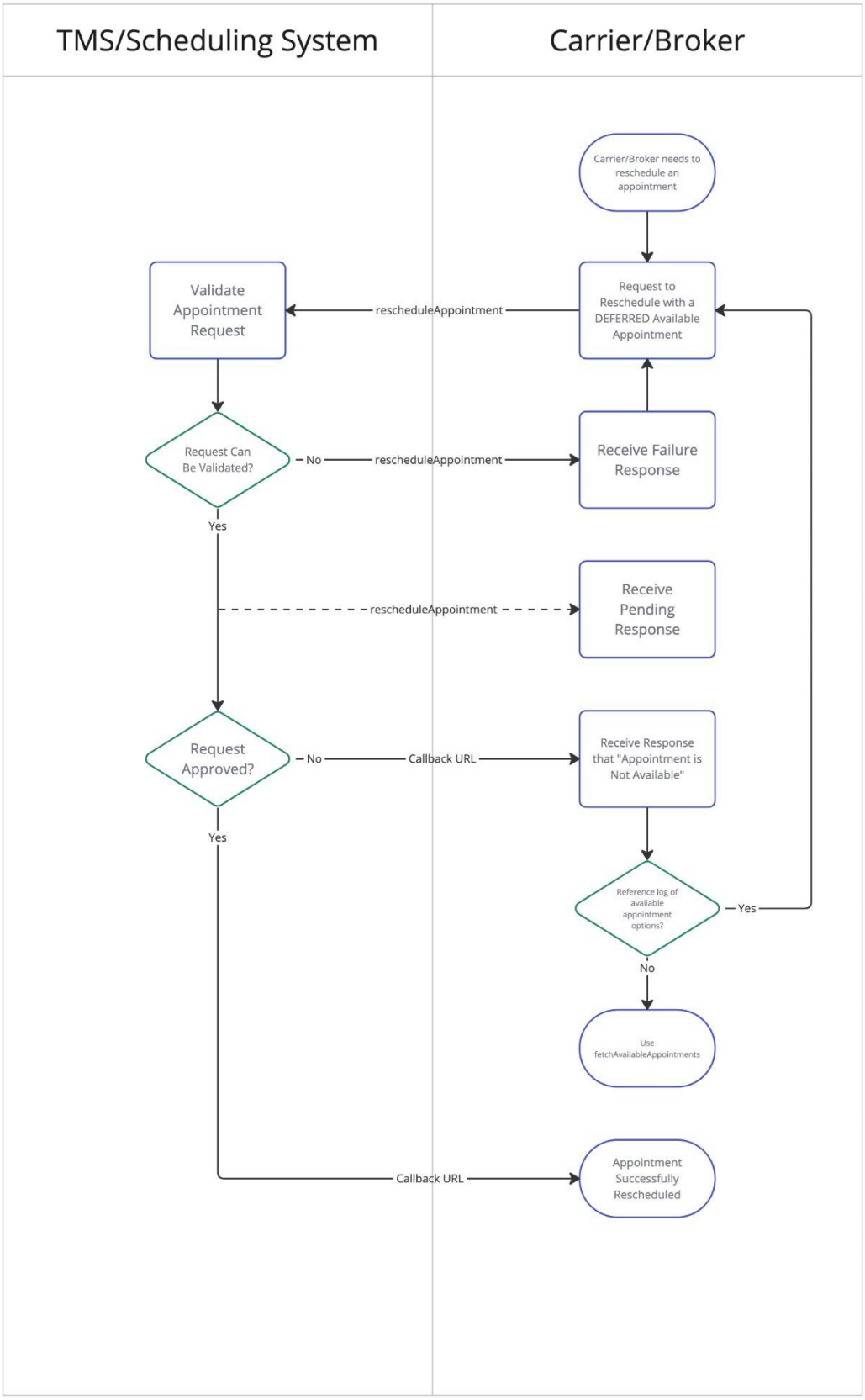


Figure 10: rescheduleAppointment with Asynchronous Response for DEFERRED Appointment Types via Callback URL or Webhook

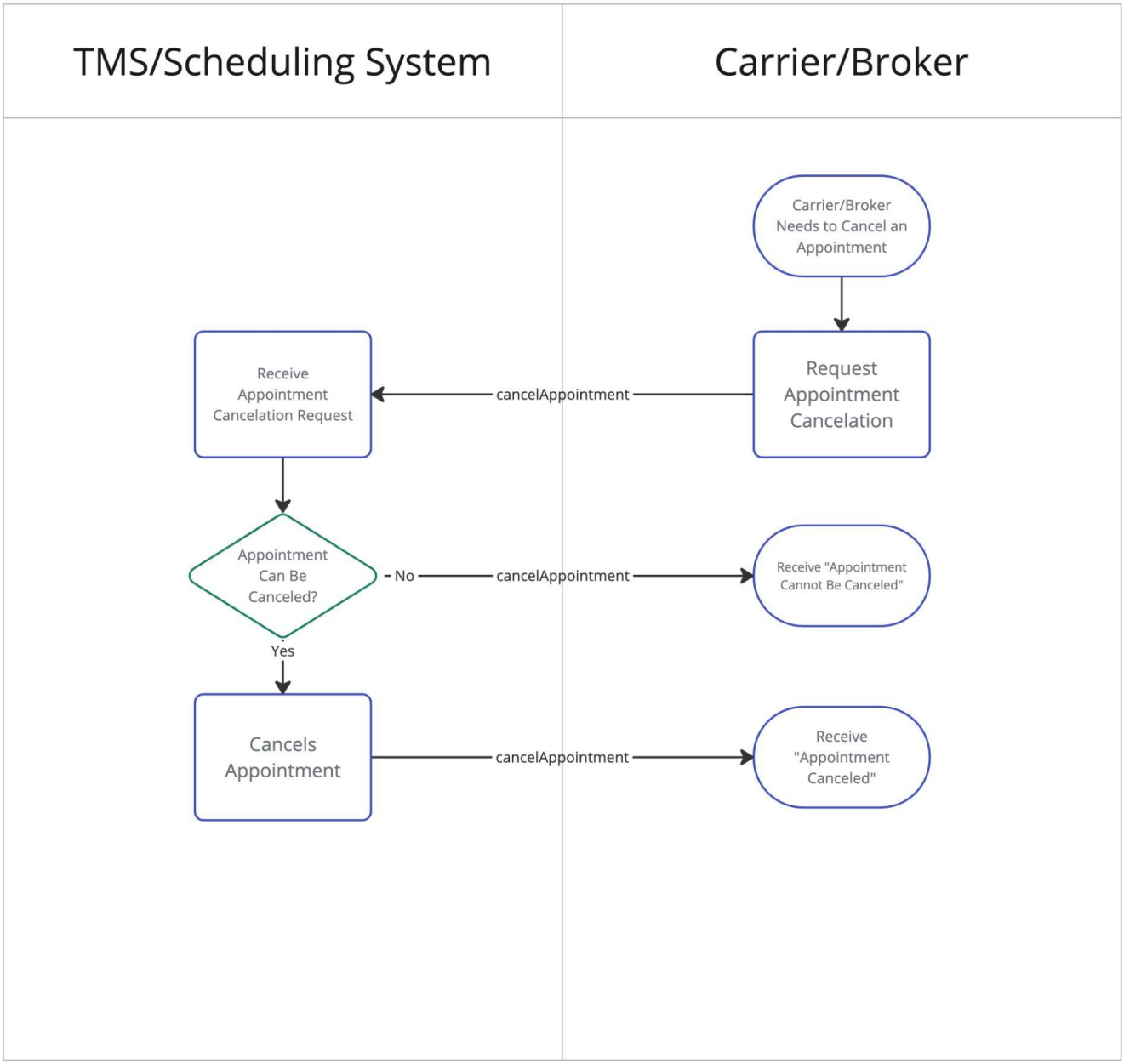


cancelAppointment

As a carrier / broker, I expect to be able to cancel an existing appointment in a TMS / appointment scheduling solution.

- In a request, the carrier / broker expects to provide unique identifiers for two scenarios.
    - The carrier / broker MUST be able to cancel an existing appointment with load and stop identifiers. See the “Requests with Load and Stop Information” section under Best Practices for more details.
    - The carrier / broker MAY be able to cancel an existing appointment with an existing appointment identifier.
- In a request, the carrier / broker MAY be able to pass reason codes and comments.
- In a response, the TMS / appointment scheduling solution can produce results for two scenarios.
    - If the information passed by the carrier / broker fails to find a unique match in the system or if the cancel appointment request is rejected, the TMS / appointment scheduling solution MUST be able to return failure responses. For this scenario:
        - The response MUST contain a status of failure.
        - The response MAY contain problem detail objects outlining why a particular failure occurred.
    - If the cancel appointment request is confirmed, the TMS / appointment scheduling solution MUST be able to return success responses. For this scenario:
        - The response MUST contain a success status.

Figure 11: cancelAppointment with Synchronous Response

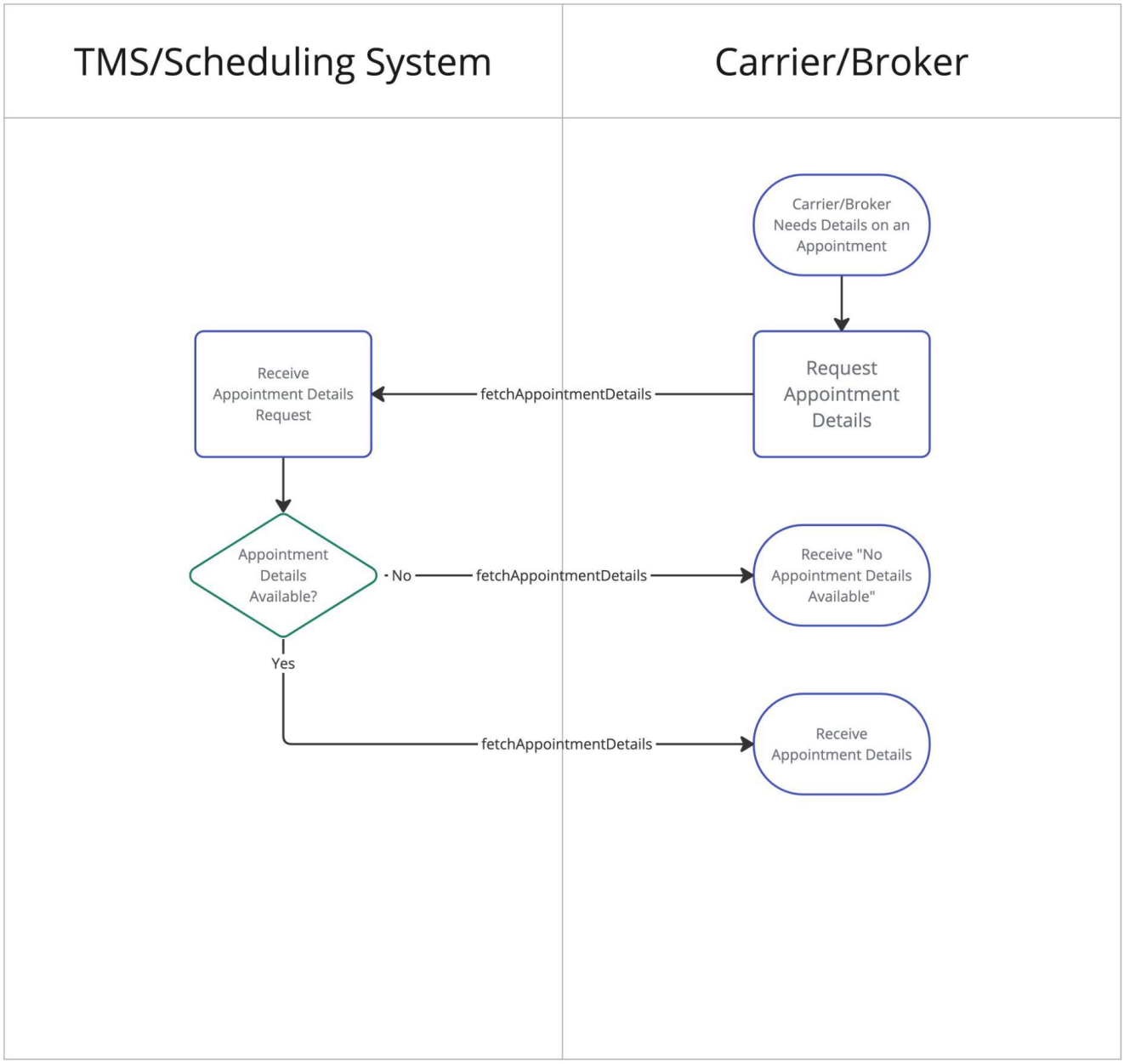


fetchAppointmentDetails

As a carrier / broker, I expect to fetch the existing appointment details for a particular load at a particular stop in a TMS / appointment scheduling solution.

- In a request, the carrier / broker expects to provide unique identifiers for two scenarios.
    - The carrier / broker MUST be able to fetch existing appointment details with load and stop identifiers. See the “Requests with Load and Stop Information” section under Best Practices for more details.
    - The carrier / broker MAY be able to fetch existing appointment details with an existing appointment identifier.
- In a response, the TMS / appointment scheduling solution can produce results for two scenarios.
    - If the information passed by the carrier / broker fails to find a unique match in the system, the TMS / appointment scheduling solution MUST be able to return failure responses. For this scenario:
        - The response MUST contain a status of failure.
        - The response MAY contain problem detail objects outlining why a particular failure occurred.
    - If the fetch appointment detail request is confirmed, the TMS / appointment scheduling solution MUST be able to return success responses. For this scenario:
        - The response MUST contain a success status.
        - The response SHOULD contain one of six appointment statuses. For these statuses:
            - If no appointment has been scheduled for a particular load at a particular stop, the TMS / appointment scheduling solution SHOULD return a status of “not scheduled.”
            - If an appointment has been scheduled for a particular load at a particular stop, the TMS / appointment scheduling solution SHOULD return a status of “confirmed.”
            - If an appointment has been preset for a particular load at a particular stop, but the carrier / broker needs to confirm the appointment or reschedule to a new appointment time, the TMS / appointment scheduling solution SHOULD return a status of “unconfirmed.”
            - If a schedule or reschedule appointment request has been submitted and is under an asynchronous review, the TMS / appointment scheduling solution SHOULD return a status of “pending.”
            - If an appointment was scheduled previously but has been rescinded, the TMS / appointment scheduling solution SHOULD return a status of “canceled.”
        - The response MAY contain additional appointment detail information including the scheduled appointment identifier, the appointment start date and time, dock group, dock door, appointment confirmation number, and location identifier.
        - The response MAY contain location address information, including street address, region, locality, country, and postal code.
        - The response MAY contain the most recent reason code and comment for the appointment.

Figure 12: fetchAppointmentDetails with Synchronous Response



Webhooks

Subscriptions

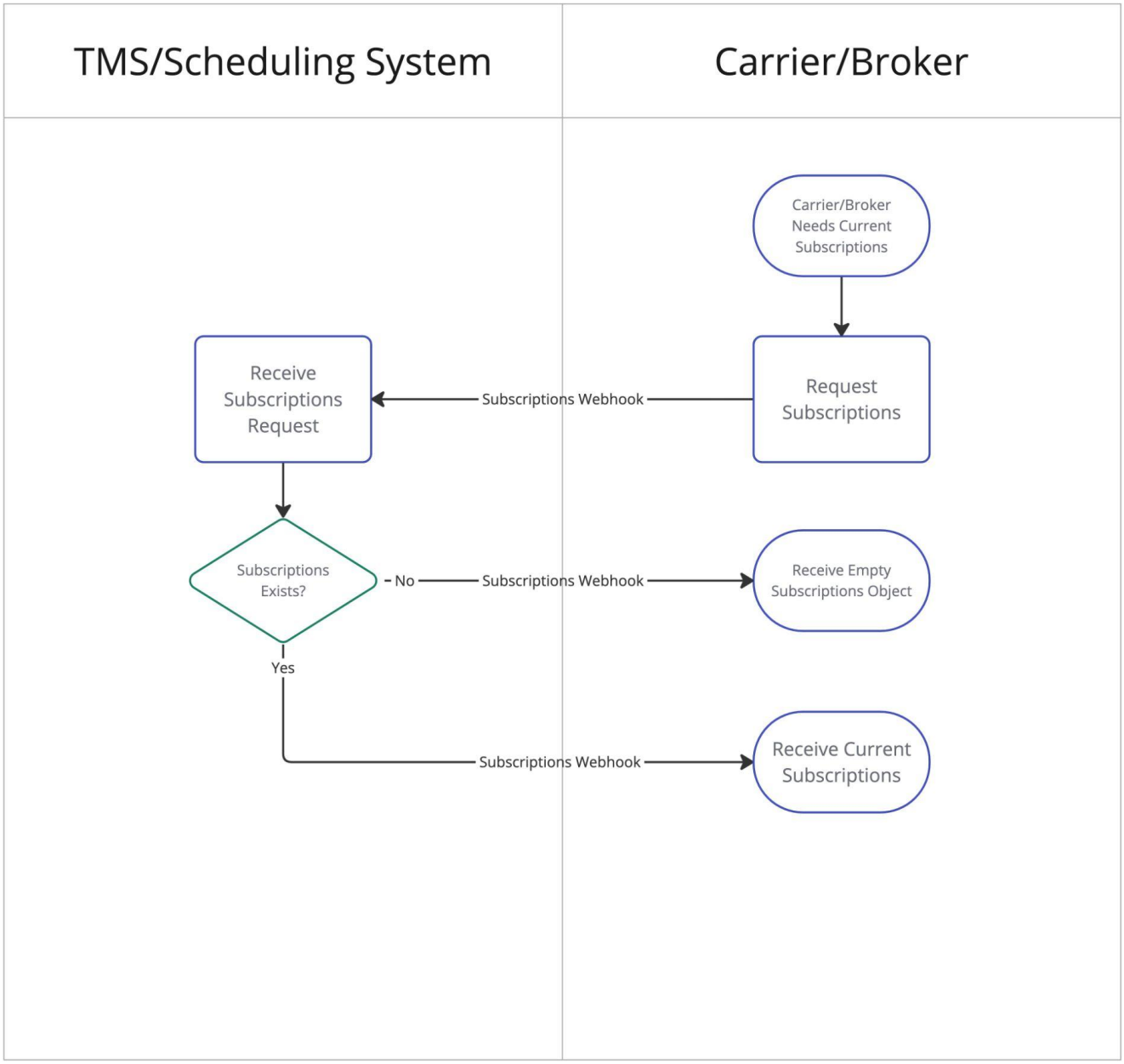
As a carrier / broker seeking to receive asynchronous information via webhooks, I expect to control webhook event subscriptions for appointment management in a TMS / appointment scheduling solution.

Get All Subscriptions

As a carrier / broker, I expect to fetch a list of current webhook event subscriptions for appointment management in a TMS / appointment scheduling solution.

- In a response, the TMS / appointment scheduling solution SHOULD produce a list of current webhook event subscriptions.
    - For each subscription, the response MAY contain a subscription identifier, event types, and a callback URL.
    - The event types SHOULD include one of two types. For these types:
        - If the carrier / broker is subscribed to receive updates when an appointment is changed, the TMS / appointment scheduling solution SHOULD return a type of “appointment-changed.”
        - If the carrier / broker is subscribed to receive fetched lists of available appointments, the TMS / appointment scheduling solution SHOULD return a type of “fetch-available-appointments.”

Figure 13: Get All Subscriptions via Webhook

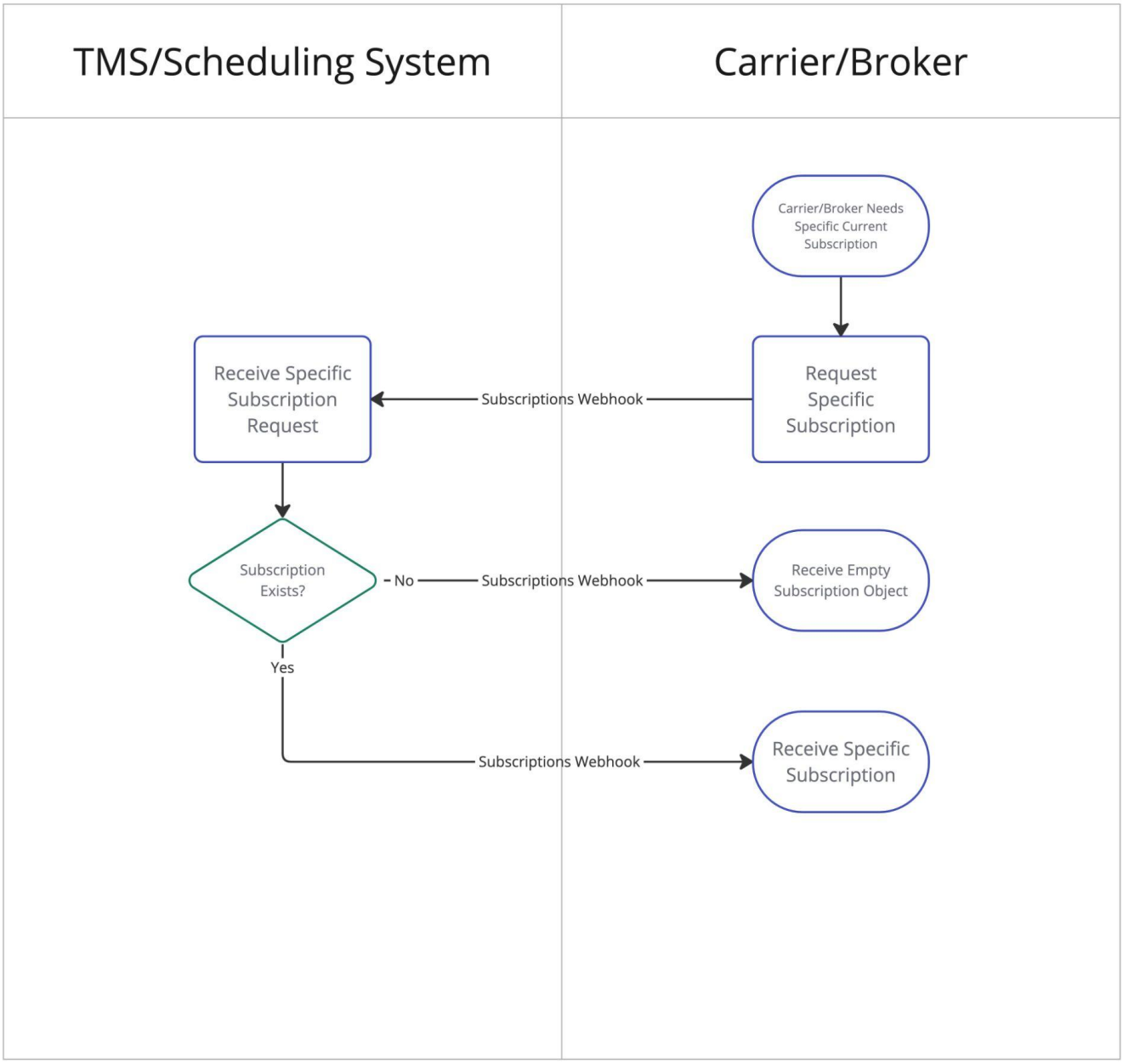


Get Specific Subscriptions

As a carrier / broker, I expect to fetch a specific current webhook event subscription for appointment management in a TMS / appointment scheduling solution.

- In a response, the TMS / appointment scheduling solution MAY produce a specific current webhook event subscription.
    - The response MAY contain a subscription identifier, event types, and a callback URL.
    - The event types SHOULD include one of two types. For these types:
        - If the carrier / broker is subscribed to receive updates when an appointment is changed, the TMS / appointment scheduling solution SHOULD return a type of “appointment-changed.”
        - If the carrier / broker is subscribed to receive fetched lists of available appointments, the TMS / appointment scheduling solution SHOULD return a type of “fetch-available-appointments.”

Figure 14: Get Specific Subscription via Webhook

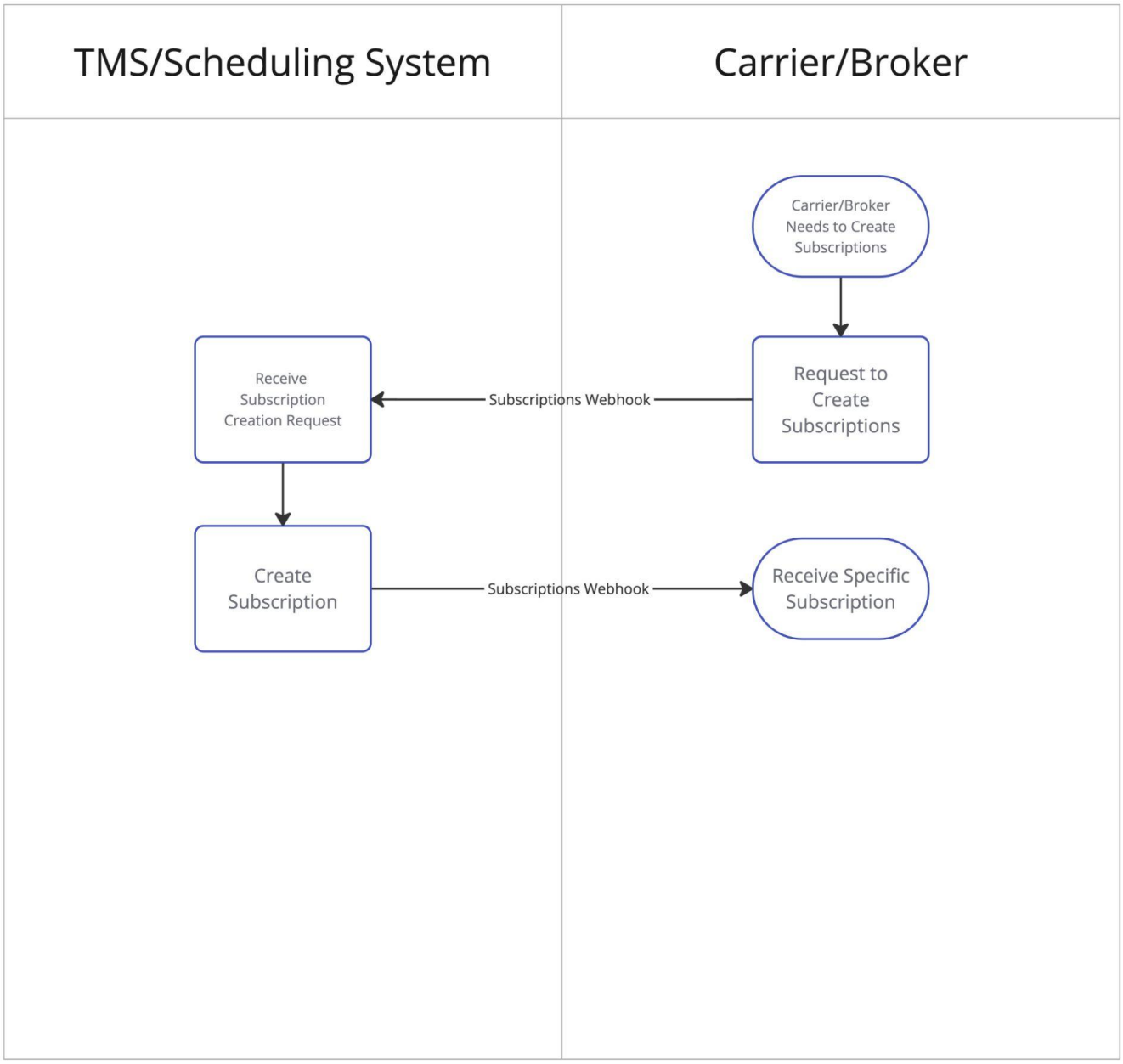


Create Subscriptions

As a carrier / broker, I expect to subscribe to webhook events for appointment management in a TMS / appointment scheduling solution.

- In a request, the carrier / broker MAY be able to subscribe to webhook events with event types and a callback URL.
- In a response, the TMS / appointment scheduling solution MAY confirm the new webhook event subscriptions.
    - For each subscription, the response MAY contain a subscription identifier, event types, and a callback URL.
    - The event types SHOULD include one of two types. For these types:
        - If the carrier / broker is subscribed to receive updates when an appointment is changed, the TMS / appointment scheduling solution SHOULD return a type of “appointment-changed.”
        - If the carrier / broker is subscribed to receive fetched lists of available appointments, the TMS / appointment scheduling solution SHOULD return a type of “fetch-available-appointments.”

Figure 15: Create Subscriptions via Webhook

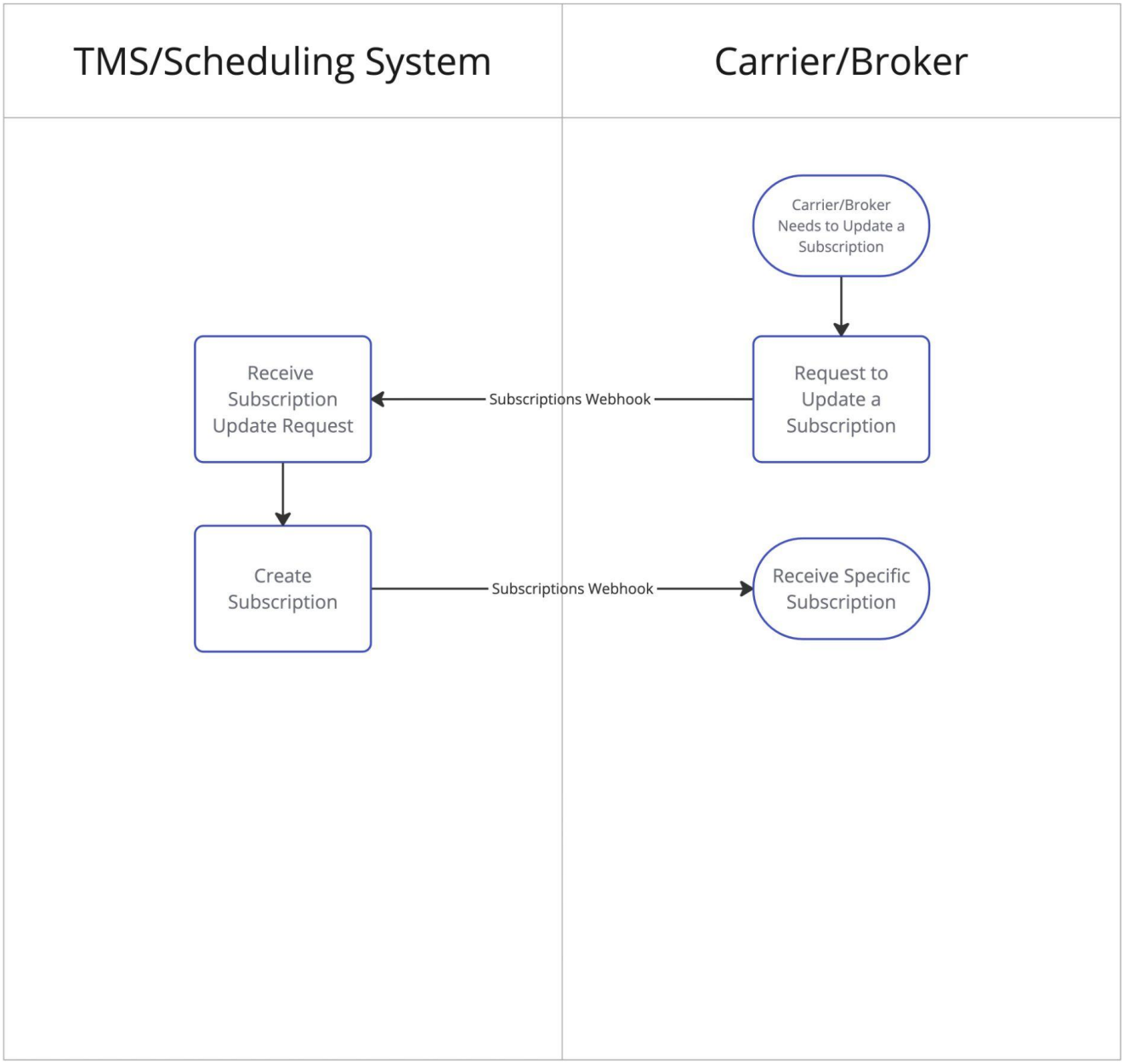


Update Subscriptions

As a carrier / broker, I expect to update subscribed webhook events for appointment management in a TMS / appointment scheduling solution.

- In a request, the carrier / broker MAY be able to update subscribed webhook events with a subscription identifier, event types and a callback URL.
- In a response, the TMS / appointment scheduling solution MAY confirm the updated webhook event subscription.
    - The response MAY contain a subscription identifier, event types, and a callback URL.
    - The event types SHOULD include one of two types. For these types:
        - If the carrier / broker is subscribed to receive updates when an appointment is changed, the TMS / appointment scheduling solution SHOULD return a type of “appointment-changed.”
        - If the carrier / broker is subscribed to receive fetched lists of available appointments, the TMS / appointment scheduling solution SHOULD return a type of “fetch-available-appointments.”

Figure 16: Update Subscriptions via Webhook

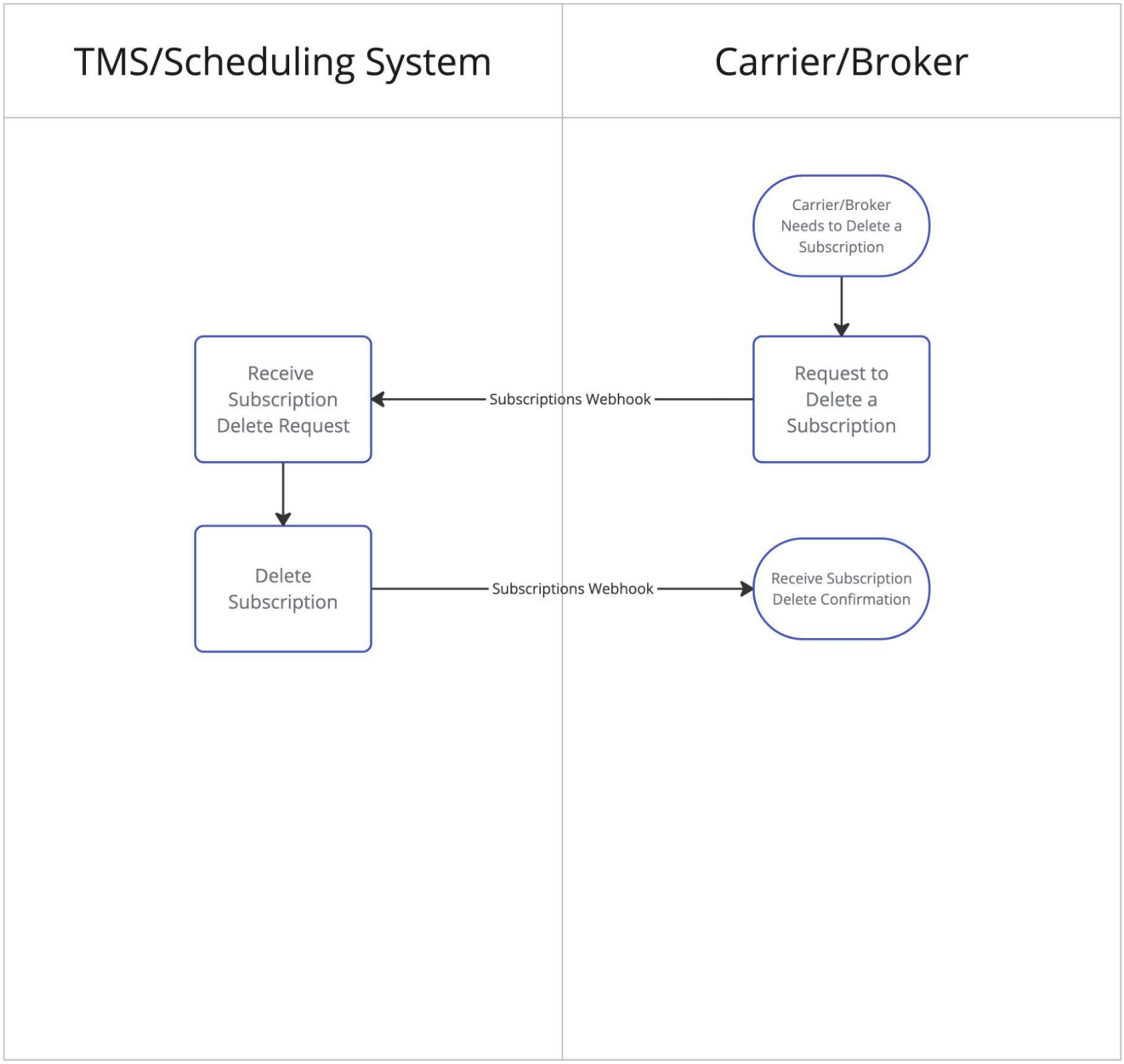


Delete Subscriptions

As a carrier / broker, I expect to delete subscribed webhook events for appointment management in a TMS / appointment scheduling solution.

- In a response, the TMS / appointment scheduling solution MAY confirm the deleted webhook event subscription.

Figure 17: Delete Subscriptions via Webhook



Events

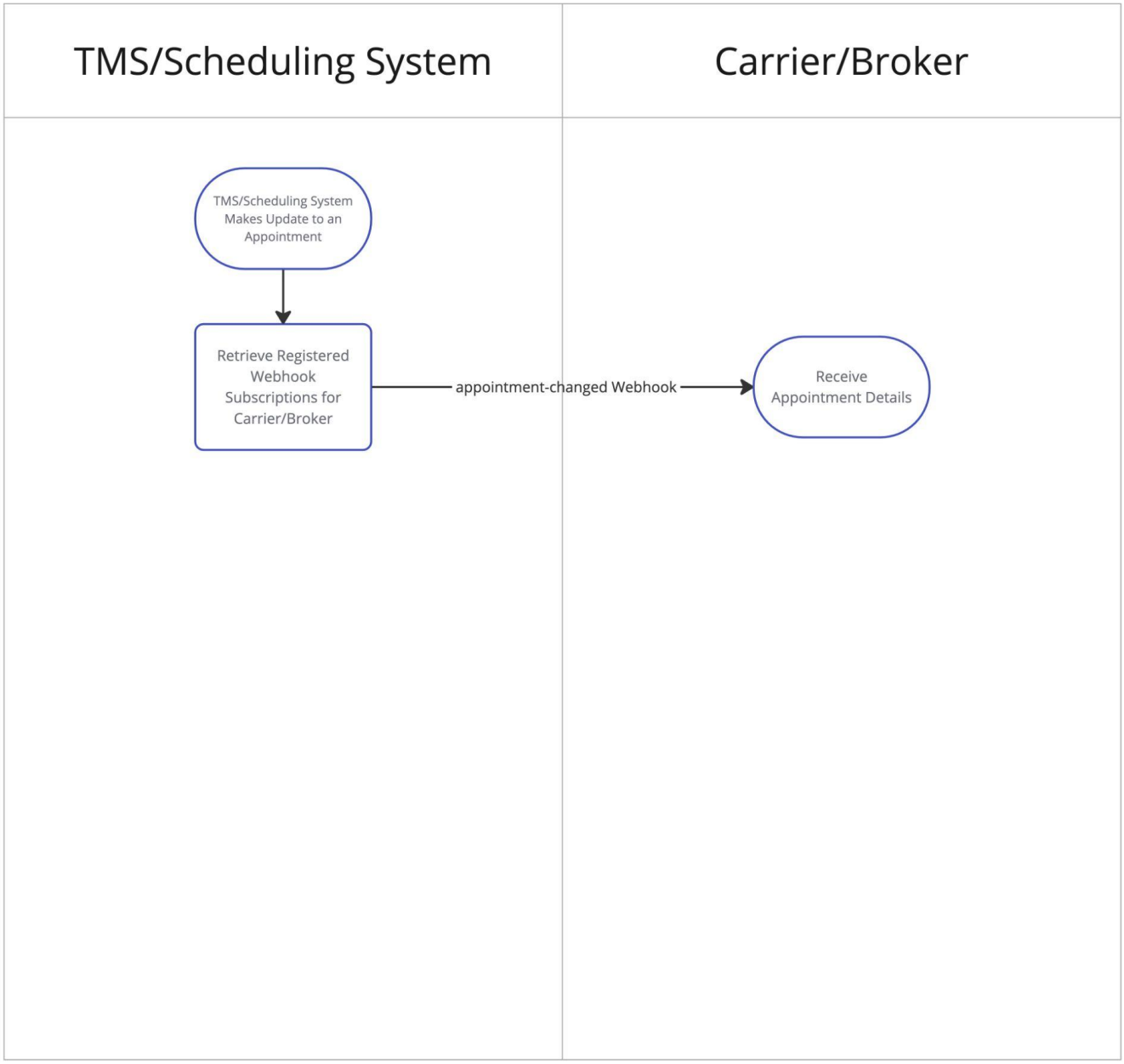
As a TMS / appointment scheduling solution seeking to send asynchronous information via webhooks, I expect to respond to carriers / brokers who have subscribed to webhook events in my system.

Appointment-Changed

As a TMS / appointment scheduling solution, I expect to send asynchronous information about changes to existing appointments to carriers / brokers who are subscribed to the “appointment-changed” webhook event.

- In a request, the TMS / appointment scheduling solution SHOULD be able to provide load and stop identifiers for the current appointment. See the “Requests with Load and Stop Information” section under Best Practices for more details.
- In a request, the TMS / appointment scheduling solution SHOULD provide additional information about the current appointment.
    - The TMS / appointment scheduling solution MUST be able to pass appointment status and arrival window information, including a start date and time and an appointment duration.
    - The TMS / appointment scheduling solution MAY be able to pass an appointment identifier, dock group, and dock door information.
- In a request, the TMS / appointment scheduling solution MAY provide information about the previous appointment.
    - The TMS / appointment scheduling solution MAY be able to pass appointment status, arrival window information, an appointment identifier, dock group, and dock door information.
- In a request, the TMS / appointment scheduling solution MAY be able to pass reason codes and comments.
- In a request, the TMS / appointment scheduling solution MUST provide the time in which the change to the appointment occurred.
- In a response, the carrier / broker MAY confirm the appointment-changed webhook event information was received.

Figure 18: Receive Appointment-Changed Events via Webhook

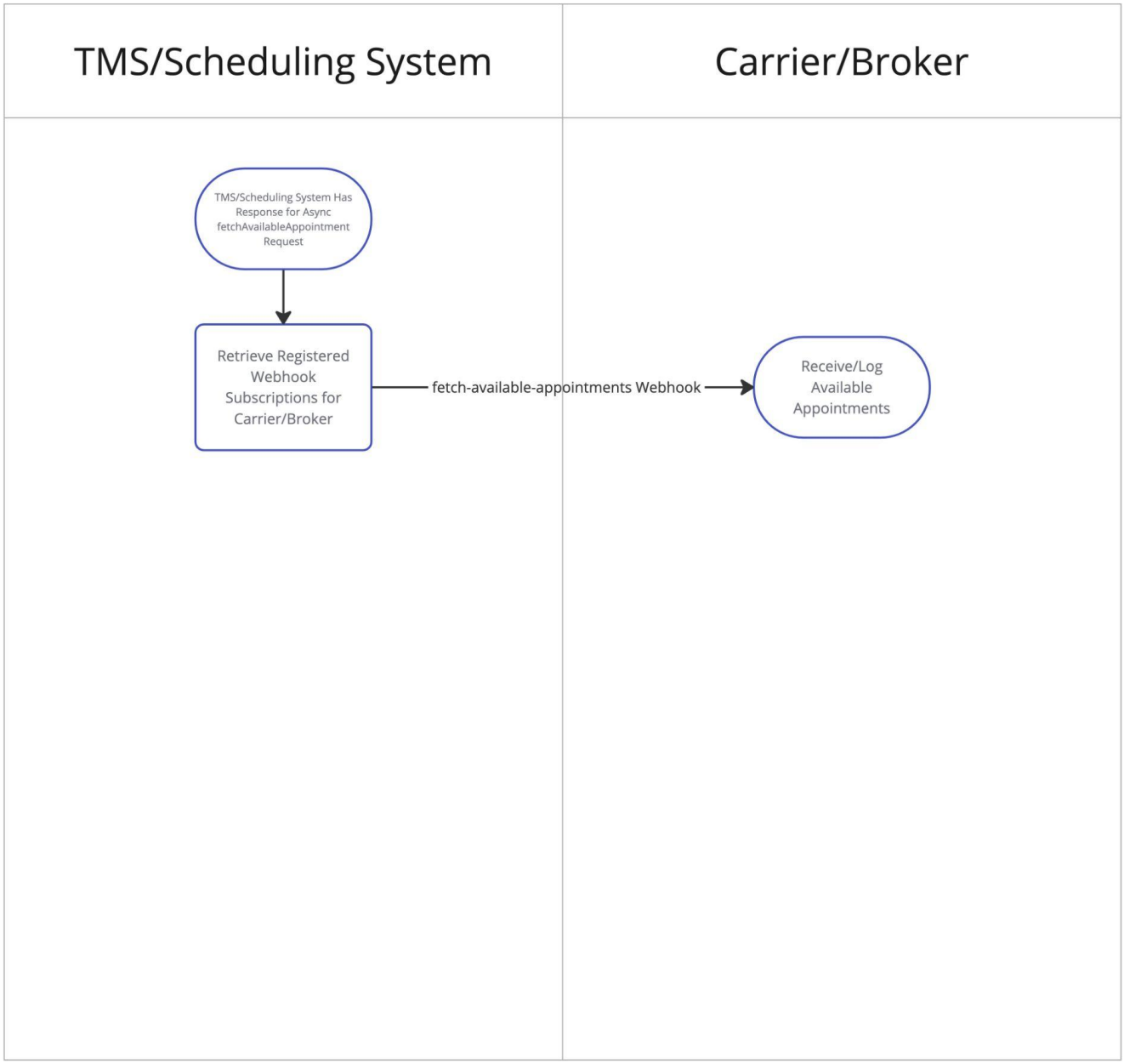


Fetch-Available-Appointments

As a TMS / appointment scheduling solution, I expect to send asynchronous information about available appointments to carriers / brokers who are subscribed to the “fetch-available-appointments” webhook event.

- In a request, the TMS / appointment scheduling solution MUST provide information about the available appointments including a request identifier and when the results were produced.
- In a request, the TMS / appointment scheduling solution can provide information for two scenarios.
    - If the information passed by the carrier / broker was rejected, the TMS / appointment scheduling solution MUST be able to provide failure information. For this scenario:
        - The response MUST contain a status of failure.
        - The response MAY contain problem detail objects outlining why a particular failure occurred.
    - If a list of available appointments can be provided, the TMS / appointment scheduling solution MUST be able to provide success information. For this scenario:
        - The request MUST contain a success status, the available appointments (specific appointments and/or appointment windows), and available appointment response types.
            - Available appointment response types SHOULD indicate if they can be scheduled synchronously or asynchronously, if the facility is first come, first served, or if they are preset and require confirmation.
        - The request MAY contain available appointment identifiers for scheduling and rescheduling, dock group and door assignment, and the location identifier.
        - The request MAY contain location address information, including street address, region, locality, country, and postal code.
        - The list of available appointment options SHOULD be restricted by necessary exclusions including, but not limited to, equipment type, commodity, consignee, “must-pickup-on” and “must-deliver-by” dates, etc. It is important to ensure dock groups are properly configured so that only eligible times are returned.
        - If there are no available appointments for the information passed by the carrier / broker, the list of available appointments in the success request SHOULD be empty.
- In a response, the carrier / broker MAY confirm the fetch-available-appointment webhook event information was received.

Figure 19: Receive Fetch-Available-Appointments Events via Webhook



End of Document