NMFTA In-Transit Visibility API Product Requirements

Overview

In-Transit Visibility is the capability to track freight movements through milestones, events and location updates in real time, from pickup to delivery. It is vital to the lifecycle of a load because it provides insight into where a shipment is at each stage and alerts all parties of important milestones or issues. This capability ties into the broader Digital Standards Development Council open API standards initiative, which aims to digitize data exchange in freight transportation. By adopting standard APIs, the industry can reduce manual processes, improve data accuracy, increase visibility into shipments, enable real-time notifications of shipment changes, and drive process automation.

Shippers today continue to demand real-time updates at every key milestone – before pickup, during transit, and upon arrival. In response, industry leaders are making automated digital tracking a core part of operations, replacing phone calls and emails with API-based status sharing. This broad industry trend underscores how critical in-transit visibility has become. It also reinforces the NMFTA's mission to standardize visibility.

By implementing this API, all parties – shippers, carriers, and intermediaries – share a single source of truth about a load's status. In-transit updates, when combined with other NMFTA standard APIs (e.g. electronic tendering and eBOL), form a continuous digital thread from load creation to completion, greatly enhancing transparency and efficiency in freight movements.

<u>Value Proposition</u>

- **Shipper**: Gaining in-transit visibility allows shippers to easily get updates on the status and location of their shipments, reducing misunderstandings, human error, and theft. For example, instead of waiting for manual updates, a shipper can see when a load is picked up, in transit, or delayed, and even receive location updates before pickup occurs. This proactive insight gives shippers peace of mind and the ability to address exceptions quickly.

- **Carrier**: The In-Transit Visibility API allows carriers to provide digital status updates to relevant parties using a standard protocol. This approach enables tracking information to be shared with shippers and third-party logistics providers (3PLs), supporting automated communication of updates such as departure, arrival, or delay notifications, and contributing to operational transparency.

- **3rd Party:** The In-Transit Visibility API will allow third parties of all kinds to receive and send necessary shipment information to ensure a seamless flow through the life cycle of a shipment. Due to the nature of their business and having to interact with multiple parties in the transportation industry, they stand much to gain by simplification of their business processes, which the In-Transit Visibility specification will enable. Overall, the In-Transit Visibility standard helps 3rd parties coordinate more effectively and provide better service with less overhead.

Definitions

The following terms are used in the In-Transit Visibility API specification. Definitions align with the NMFTA Digital Standards Development Council (DSDC) Glossary of Terms:

Application Programming Interface (API): An API is a set of rules using HTTP that enables software applications to communicate with each other, exchanging data, features, and functionality. In transportation, APIs can be used to integrate systems for tracking shipments, managing logistics, and automating processes.

Event: An occurrence that updates or is related to an entity (such as a shipment). In the context of visibility, a “status event” refers to a milestone or update in the shipment journey (for example, Pickup, In Transit, etc.). Each event typically has a type (what happened), a timestamp (when it happened), and often a location.

Status Update: A message or record of a status event that provides new information on a shipment’s state. Status updates are generated by the carrier’s system as the load progresses (e.g., a departure, a delay notification, a delivery confirmation). A status update usually includes an event type and relevant details like time and location. (Note: “Status Update” is a descriptive term; it encapsulates one or more Events as defined above.)

Location: A specific geographic place relevant to the movement, storage, or tracking of goods or vehicles. This can refer to an origin, destination, transfer point, facility (such as a warehouse, terminal, or distribution center), or the real-time position of Equipment.

Location Update: A specific type of status update that provides the real-time position or location of a shipment in transit. This could be given as GPS coordinates or a city/location name, indicating where the freight is at a certain time. Location updates may be provided periodically between major events to increase transparency on the shipment’s progress.

Appointment: A scheduled range of time for the pickup or delivery of goods. It ensures that the carrier and receiver are coordinated for efficient handling of shipments. Appointments ensure the carrier and the facility (shipper or consignee) are coordinated for loading or unloading.

Shipment: An item or collection of items that is planned and moved between a single origin and destination location. A Shipment can consist of one or more legs (e.g., a shipment from an origin of vendor or supplier location split through a hub to a destination DC). One or more Shipment(s) are created from a single transport order (e.g., purchase order, sales order, transfer order).

Load: A freight service provided by a Carrier to move goods from one or more Stops to one or more Stops. It represents the transportation of one or more Shipments. Loads are typically planned, tracked, and managed as discrete units.

Carrier: Any person, organization or government responsible for the transport of goods by any means of transport. In this context, the carrier is the party providing status and location updates for the shipment.

Shipper: Any person or legal entity by whom or in whose name, or on whose behalf, a contract of carriage of goods has been concluded with a carrier. Additionally any person by whom or in whose name, or on whose behalf, the goods are delivered to the carrier in relation to the contract of carriage.

Receiver / Consignee: The receiver is the person or entity that accepts final receipt of goods, also known as delivery. They are responsible for ensuring the goods are delivered as expected. Could also be referred to as Consignee.

Third Party (3PL/Broker): A third-party logistics provider or broker involved in the shipment but not as the shipper or carrier. They facilitate or manage shipments on behalf of a client. In tracking, a 3PL might subscribe to updates to keep all stakeholders informed. (In general terms, a third party to a shipment is any party that is neither the shipper nor the consignee but has a role in the transaction.)

Reference Number: A unique identifier assigned to a shipment or transaction for tracking purposes. In this API, the term “reference number” can refer to any key identifier used to look up the shipment’s status. Examples include a PRO number, booking ID, shipment ID, BOL number, purchase order number, etc. The reference number is how the shipper, carrier, and 3PL will refer to the shipment in API calls.

Unique Load Identifier: A generic term for the key ID that uniquely identifies a shipment across systems.

Full Truckload (FTL): Full Truckload refers to a transportation mode where a truck carries a Load for a single customer, meaning the journey is reserved for that customer only. This is typically used for large Shipments that can fill an entire truck.

Less Than Truckload (LTL): Less Than Truckload is a shipping service for smaller freight loads that combines shipments from multiple customers to optimize costs and efficiency. It is used when shipments do not require the full capacity of a truck.

Estimated Time of Arrival (ETA): Estimated time of arrival can be sent from carrier based on current location of the truck and remaining transit hours.

Proof of Delivery (POD): Confirmation that the shipment was delivered and received by the consignee. This often takes the form of a signed delivery receipt, either paper or electronic.

These definitions ensure a common understanding of key terms used in the In-Transit Visibility API specification and align with industry-standard terminology.

Best Practices

When implementing or using the In-Transit Visibility API, it's important to understand which data fields are required vs. optional and how to maximize the value of the shared information. The following best practices should be observed for effective implementation:

- **Unique Shipment Identifiers**: Always include a clear unique identifier for the shipment in any visibility request or update. The API standard supports multiple types of reference numbers that could uniquely identify a shipment (for example, PRO number, shipment/load ID, booking number, BOL number, purchase order, etc.). Rather than mandating a single identifier, the standard allows flexibility – the specific ID used may differ by carrier or mode. It is crucial that the shipper/3PL and carrier agree on which identifier(s) will be used for tracking. As a rule of thumb, provide all reference numbers that might help identify the shipment if there is any ambiguity. For instance, an LTL carrier might require the PRO number, whereas a truckload carrier might use an internal load ID; including both in your tracking request (if available) can ensure the carrier's system finds the correct load. Clearly communicating in your integration documentation which identifiers are required or supported is recommended so that shippers and 3PLs know what to supply.

- **Optional Information**: Many data fields in the API are optional (not required to constitute a valid request or event notification). However, it is highly recommended to provide as much optional information as possible whenever it is available. More information typically results in a better experience for all parties involved. For example, if a location update event allows an optional latitude/longitude and an optional city name in addition to a required timestamp, including those details will greatly enhance the clarity and utility of the update. Similarly, if a status event has an optional field for a reason code (explaining, say, a delay) or an estimated time of arrival (ETA) for the next milestone, populating those fields will improve downstream planning and communication. In short, sending richer data (beyond just the minimum required) enables more robust and proactive visibility solutions.

- **Key Milestone Events**: Ensure that status events cover the critical milestones in the shipment's journey. At minimum, provide updates for when the shipment is picked up, when it's in transit (including any major delays or departures/arrivals at intermediate stops), and when it is delivered.

- **Data Consistency & Standards**: Use standard codes and formats for events, timestamps, and locations as much as possible. Consistency in how data is provided ensures that all integrators interpret the status the same way. For instance, use standardized event type names or codes provided by the NMFTA specification for events like “Pickup”, “In Transit”, “Delay”, “Delivered”, etc., rather than custom terms. Likewise, provide location information in a consistent format (e.g., always include city/state or coordinates, and use agreed-upon location identifiers for facilities if available). Standardization across all users of the API will reduce errors and confusion, especially as data is shared between multiple parties.

- **Security & Authentication**: Although not a field within the payload, it’s a best practice to enforce secure authentication for all API interactions. Typically, each request should include valid credentials (such as an API key or OAuth token) identifying the caller. This ensures that only authorized parties can access tracking information for a shipment. For example, a 3PL should only be able to track shipments they are involved with. Many systems use OAuth2 bearer tokens for secure access. It is important to follow the carrier’s authentication requirements. Additionally, all data exchange should occur over HTTPS to protect sensitive information like shipment locations and status from interception.

- **Frequency of Updates**: Consider the timing and frequency of update requests or pushes. If using a polling model (where the shipper’s system regularly requests updates), avoid excessive polling that might unnecessarily tax the carrier’s API. Conversely, polling too infrequently could result in missed timely information. Balance this based on the shipment’s typical transit time (for instance, polling every 15 minutes for a long-haul that lasts days may be reasonable, whereas a local delivery might warrant more frequent checks). Ideally, if the carrier API supports webhooks or push notifications for status events, use that feature to get real-time updates without polling. From the carrier’s perspective, whenever a meaningful event occurs (pickup, delay, arrival, etc.), they should emit/push that update immediately. The goal is real-time or near-real-time visibility with efficiency in mind.

By following these best practices, participants ensure the must-use fields are always present and correct (so the system functions), while optional fields are leveraged to their fullest extent (so the system functions well). Also, aligning with industry-standard event definitions and security protocols makes integration smoother for all parties.

Requirements

Synchronous Visibility: Request(Pull)

As a Shipper or authorized 3rd Party, I want to initiate synchronous tracking for a shipment by requesting in-transit visibility from the carrier's system for that load.

- The Shipper/3PL MUST provide a **unique shipment identifier** for the load they wish to track. This is critical: the carrier needs to know which shipment's visibility data to enable. The identifier should be one that the carrier recognizes in their system. For FTL shipments, it will typically be a load ID, booking number, or similar reference that was provided by the carrier (for example, the load or dispatch number returned when the load was tendered or when the eBOL was created). If the shipper has multiple reference numbers and the API allows, it's good practice to include as many as necessary (see Unique Shipment Identifiers in Best Practices). This ensures that even if, say, one reference is not found, another might match the carrier's records.

- The request can also indicate the **shipment's origin and destination** in some form, if required for identification or routing. In many cases, if a unique shipment identifier is provided, additional origin/destination info isn't needed for the carrier to identify the shipment. The standard allows providing an origin and destination either as location codes known to both parties or as address details. This helps prevent any ambiguity (for example, if two shipments had the same reference number, matching origin/destination could clarify which one to track).

- The Shipper/3PL MUST provide the **requested pickup date** (or shipment start date) if that is relevant for the mode. This helps the carrier confirm which shipment is being referenced, especially in scenarios where reference numbers might repeat over time. The requested pickup date must be in a standard date/time format.

- Optionally, the Shipper/3PL MAY **specify preferences** for what they want in the tracking response or how they want to receive updates. For instance, there might be a flag to request an initial status snapshot in the create response (as opposed to simply an acknowledgment). Another example: the requester could indicate if they intend to poll for updates or if they have a callback URL to receive push notifications (webhook). The standard should allow extensions – e.g., a boolean like includeLatestStatus or a field for callbackURL.

Synchronous Visibility: Response

As a Carrier, I want to acknowledge the tracking initiation request and provide initial tracking information or confirmation to the requester.

- In response to a visibility request, the Carrier MUST indicate **whether the request was successful**.
    - **TrackingIntiatied: True**
        - When the response is True, then all shipment events should be transmitted in a synchronous response starting from the pickup event of the shipment followed by all the events that have occurred until the time of the visibility request.
        - If the shipment is already delivered at the time of the request, the response from the carrier should include all the unique events occurred from pickup to delivery event.
        - All event times should be sent in local time that relates to the time zone of the event location with an Offset of UTC time.
        - Date Time format should be ISO 8601 in the response.
        - Always provide history of events regardless of the time of the request.
    - **TrackingIntiatied: False**
        - If the request is rejected, the response MUST include a reason or error code explaining why.
        - Common reasons for failure might be:
            - Invalid shipment ID (the carrier could not find any shipment with the given identifier)
            - Unauthorized (the requesting party is not allowed to track that shipment),
            - Bad Request (required information was missing).
        - Providing a clear error message helps the shipper/3PL correct the issue (e.g., fix the ID and retry).

Async Create Visibility: Request (PUSH)

As a Shipper or authorized 3rd Party, I want to initiate tracking for a shipment by requesting in-transit visibility from the carrier's system for that load.

- In a Create Visibility request, the Shipper/3PL MUST **identify themselves in the message** so that the Carrier knows who is requesting the tracking data. This could be implicitly handled by authentication (for example, the API credentials used will indicate the identity/organization), or it may be an explicit field in the request body (such as a "requestingParty" or a role indicator). Identification ensures the carrier can verify authorization and log who is subscribing to the tracking information (e.g., distinguishing a request from the actual shipper vs. a 3PL acting on the shipper's behalf).

- The Shipper/3PL MUST provide a **unique shipment identifier** for the load they wish to track. This is critical: the carrier needs to know which shipment's visibility data to enable. The identifier should be one that the carrier recognizes in their system. For FTL shipments, it will typically be a load ID, booking number, or similar reference that was provided by the carrier (for example, the load or dispatch number returned when the load was tendered or when the eBOL was created). If the shipper has multiple reference numbers and the API allows, it's good practice to include as many as necessary (see Unique Shipment Identifiers in Best Practices). This ensures that even if, say, one reference is not found, another might match the carrier's records.

- The request MUST indicate the **shipment's origin and destination** in some form, if required for identification or routing. In many cases, if a unique shipment identifier is provided, additional origin/destination info isn't needed for the carrier to identify the shipment. The standard allows providing an origin and destination either as location codes known to both parties or as address details. This helps prevent any ambiguity (for example, if two shipments had the same reference number, matching origin/destination could clarify which one to track).

- The Shipper/3PL MUST provide the **requested pickup date** (or shipment start date) if that is relevant for the mode. This helps the carrier confirm which shipment is being referenced, especially in scenarios where reference numbers might repeat over time. The requested pickup date must be in a standard date/time format.

- Optionally, the Shipper/3PL MAY **specify preferences** for what they want in the tracking response or how they want to receive updates. For instance, there might be a flag to request an initial status snapshot in the create response (as opposed to simply an acknowledgment). Another example: the requester could indicate if they intend to poll for updates or if they have a callback URL to receive push notifications (webhook). The standard should allow extensions – e.g., a boolean like includeLatestStatus or a field for callbackURL.

This Create Visibility step parallels the concept of a shipper "subscribing" to tracking for a shipment. It aligns with industry API flows where a shipment/order can be created or registered for tracking via API, after which events will be available.

Async Visibility: Response

As a Carrier, I want to acknowledge the tracking initiation request and provide initial tracking information or confirmation to the requester.

- In the response to a Create request, the Carrier MUST indicate **whether the request was successful**.

-
  - **TrackingIntiatied: True**
    - When the response is True, then all shipment events should be transmitted in a synchronous response starting from the pickup event of the shipment followed by all the events that have occurred until the time of the visibility request.
    - If the shipment is already delivered at the time of the request, the response from the carrier should include all the unique events occurred from pickup to delivery event.
    - Always provide history of events regardless of the time of the request.
  - **TrackingIntiatied: False**
    - If the request is rejected, the response MUST include a reason or error code explaining why.
    - Common reasons for failure might be:
      - Invalid shipment ID (the carrier could not find any shipment with the given identifier)
      - Unauthorized (the requesting party is not allowed to track that shipment),
      - Bad Request (required information was missing).
    - Providing a clear error message helps the shipper/3PL correct the issue (e.g., fix the ID and retry).

- If the tracking request is accepted, the Carrier MUST **provide a timestamp** confirming when the tracking was activated or the request was processed. This timestamp essentially marks the start of the visibility session. It's usually the current time at the carrier's system. This is useful for auditing and to let the requester know the request went through at a specific time.

- The Carrier MUST **echo back the shipment's reference identifier** in the response, exactly as recognized in their system. For example, if a shipper requested tracking for load "ABC123" and the carrier's internal ID for that shipment is also "ABC123", the response should include that ID to confirm which shipment is now being tracked. If the carrier normalized or changed the ID, the response should include the identifier that must be used for subsequent tracking calls. This ensures both sides are referring to the same shipment ID going forward.

- The Carrier SHOULD return the **current status information** for the shipment as part of the response, if available and if requested. Often, when tracking begins, there may already be one or more events recorded in the carrier's system (for example, the pickup might have already occurred and there's a "Picked Up" event logged). It is very useful to provide the latest known status event(s) immediately so that the shipper has an initial state without needing to immediately call the update endpoint.
  - If supported, the response might contain an array of events (or at least the most recent event) with details like event type, timestamp, and location. For instance, the response could say: "accepted": true, "latestEvent": {type: Pickup, time: 2025-05-01T14:30Z, location: "Origin City"}.
  - If the API does not support including event data in the create response, then the response will be an acknowledgment and the client will follow up with an Update request to get status.

- **Error Handling**: If the create request was malformed (e.g., JSON errors, missing required fields) the API response should return a 400-level HTTP error with details. If the shipment identifier was provided but not found, a 404 Not Found might be appropriate, or a 200 with a rejection status and a message "Shipment ID not recognized". Additionally, if the shipment was found but is in a state that cannot be tracked (for example, already delivered long ago and purged from active records), the response might indicate that as the reason for rejection.

In summary, a successful Create Visibility response establishes that "Shipment X is now being tracked". After this, the expectation is that the shipper/3PL can query for updates until the shipment is completed or is otherwise terminated. This explicit handshake is analogous to confirming a subscription to tracking events.

Async Update Visibility: Request

As a Shipper or 3rd Party, I want to retrieve the latest status updates for a shipment that I am tracking, so I can get up-to-date visibility information (or, in push scenarios, as a Carrier I want to send a new status update for a shipment to interested parties).

- In an Update Visibility request, the Shipper/3PL MUST provide the **shipment's unique identifier** again, to specify which shipment they are inquiring about. This is the same ID used in the Create step (e.g., PRO or load ID) and ensures the carrier knows which tracking record to query.

- If the API endpoint is structured as /visibility/{shipmentId}, the shipmentId in the URL fulfills this requirement. In a GET request, it might be part of the path or a query parameter; in a PUSH scenario (where the carrier sends an update), the identifier would be in the payload to tell the receiver which shipment this event belongs to.

- The Shipper/3PL MUST **identify themselves** (via authentication or an identifier in the request) just as in the Create call.
    - This is mainly to maintain security – the carrier verifies that the requester of updates is the same party that initiated tracking (or otherwise authorized).

- The request MAY include **parameters to filter or scope the update data**.
    - For example, the client might include a parameter like sinceTimestamp (asking for any events that occurred after a certain time, to avoid getting duplicate data they already have).
    - Another parameter might be eventTypes (if the requester is only interested in certain event categories).

- The request MAY be **repeated periodically**. The shipper's system can call the Update request as often as needed to stay up to date. Some might do it on a schedule (e.g., every 15 minutes, or every time their user opens a tracking dashboard). The API should handle these repeat calls idempotently – i.e., if nothing new happened, it should harmlessly return either no new data or the same last data. If something changed, it returns the new info.

- **Push Variant**: Although the standard is described as if the shipper/3PL is requesting updates, it's worth noting that the underlying data model supports push as well. In a push scenario, the carrier would use the equivalent of the Update call to send a new event (or events) to the subscriber's endpoint. The content of the message would be the same (one or more events, identified by shipment). The main difference is who initiates. The NMFTA API standard should be implemented as either a pull (GET updates) or push (POST event to subscriber) system depending on bilateral agreements. The important thing is that the data format remains consistent in both cases.

Async Update Visibility: Response

As a Carrier, I want to provide the most recent tracking information for the shipment when an update is requested (or when pushing an update), so that the shipper/3PL stays informed of the shipment's progress.

- In an Update response, the Carrier MUST **return all relevant status events and location updates for the shipment** since the start of tracking (or since the last known event, if a filter was used). This is typically provided as a list of events, ordered chronologically. Each event in the list MUST include at least an:

        - **Event Type** (what happened)
        - **Event Timestamp** (when it happened).
        - **Location** (where it happened) if applicable.
    - For instance, an event could be represented as: { type: "Departed Pickup", time: "2025-05-01T14:30Z", location: "Springfield, MO" }. A location update event might be something like: { type: "LocationUpdate", time: "2025-05-01T18:00Z", coordinates: "35.05,-92.39" }. The exact structure should be defined in the API schema, but fundamentally the client should receive a structured list of events with those key details.

- The Carrier MUST **ensure the events are in the correct sequence** (usually sorted by time) **and that no events are omitted**. If events have unique IDs or sequence numbers, those should be included too, to help the client detect if any gaps occurred.
    - In scenarios where events might be recorded out-of-order in real life (say a delay event is logged after a departure event with an earlier timestamp), the carrier should still output them in chronological order of when they occurred or provide a sequence that the client can sort.

- If the shipment is currently **in transit**, the response should contain **all events up to the present**. If the shipment has been delivered, the final event in the list would be a "Delivered" event.

- The Carrier MUST **include a "last updated" timestamp in the response** to indicate when the data was generated.
    - For example: lastUpdateTime: 2025-05-02T09:00Z. This is useful for the shipper to know how fresh the information is, especially if they might not poll again for a while.

- The Carrier SHOULD **include ETA and appointment-related information** in the update data whenever it's available and relevant.
    - For example, an event type could be "ETAAdjusted" with details, or the next scheduled stop event could carry an updated ETA field. Similarly, if a delivery appointment gets scheduled or changed, an event like "Delivery Appointment Set" with the appointment time should be part of the stream. This allows the shipper/consignee to be alerted to schedule changes.

- **Stop-Level and Location Events**: It's important that the visibility feed covers both stop-based status events and in-transit location updates.
    - Stop-based events include things like Arrival at Origin, Departure from Origin, Arrival at Destination, and Delivery Completed, as well as arrivals/departures at any intermediate stops (terminals, relay points) if applicable.
    - Location updates are typically mid-route pings (e.g., "Truck at latitude/longitude X at time Y"). The combination of these gives a full picture.

- If no new events have occurred since the last update (in a polling scenario), the Carrier MAY **return an empty list or a special indicator to show that tracking is up-to-date**.
    - For instance, the response could be { events: [], note: "No new updates as of current time" } or simply a 204 No Content status. This prevents the client from misinterpreting a lack of change. If using a sinceTimestamp query, the expectation is that only new events (after that timestamp) are returned; if there are none, an empty result makes sense.

- **Error Handling**: If the update request cannot be fulfilled (e.g., if the shipment ID was missing or invalid in the request), the Carrier should return an error (400 Bad Request or similar) with an explanation. If the tracking session was never created or was already canceled, the response might be an error or a message indicating no active tracking (perhaps a 404 Not Found for that shipment's tracking). Under normal circumstances, after a successful create, such errors should not occur unless something like an authentication token expired (in which case a 401 Unauthorized would prompt the client to re-authenticate).

The update response is the core of the visibility service – it provides a timeline of the shipment's journey. The information from this response can be used to drive notifications, feed customer-facing tracking pages, trigger downstream processes (like warehouse prep when a delivery is imminent), and so on. By adhering to this structure, all parties get a consistent and complete set of tracking data.

Async Cancel Visibility: Request

As a Shipper or 3rd Party, I want to cancel tracking for a shipment – either because the shipment itself is canceled, or because it has been delivered and I no longer need to poll for updates (in some systems, this may simply acknowledge that tracking can be closed).

- In a Cancel Visibility request, the Shipper/3PL MUST **provide the unique shipment identifier** for the shipment whose tracking should be canceled. This tells the carrier exactly which tracking session to terminate. It is similar to the create and update calls in that the key to identify the shipment is required.

- The Shipper/3PL MUST **identify themselves** (as in prior calls) to ensure the request is authorized. The carrier will verify that the identity requesting cancellation is the same party that initiated tracking or otherwise has rights to do so. This prevents unauthorized cancellation of tracking by a third party.

- The request SHOULD **indicate the reason or context for cancellation**, if applicable. While not mandatory, providing a reason code or note can be useful.

    - Typical reasons might be “Shipment Canceled” (if the load was called off), “Delivered” (if using the cancel as an acknowledgment of completion), or “Duplicate Tracking” (if this tracking request is being withdrawn perhaps due to a consolidation of tracking). The API might have a field for reason, or it could simply be a best practice for logging.

- **Idempotency**: The Cancel operation should be idempotent. If a client issues Cancel for a shipment that is not (or no longer) being tracked, the API should handle it gracefully (for example, respond with a message “tracking already stopped” or just a success with no effect). This way, if a cancel call is accidentally made twice, it doesn’t cause an error or require special handling.

The Cancel Visibility call here serves a similar purpose in the visibility domain: it tells the carrier that the shipper is done tracking this shipment (either because the job is done or the shipment is called off).

Async Cancel Visibility: Response

As a Carrier, I want to confirm that tracking has been terminated for the specified shipment, so that the shipper/3PL knows no further updates will be sent.

- Upon a valid cancel request, the Carrier MUST **respond with a confirmation** that the shipment’s tracking has been canceled. This can be a simple acknowledgment, e.g., a 200 OK with a message “Tracking canceled for shipment XYZ”. The confirmation should clearly reference the shipment (by ID) so it’s clear what was affected.

- If the cancel request included a reason, the Carrier SHOULD **acknowledge that reason**.
    - For example, if reason was “Shipment Canceled”, the carrier might reply with “Tracking canceled – shipment marked as canceled.” If reason was “Delivered”, the reply might be “Tracking closed – shipment already delivered.” This lets the shipper know the carrier understood the context.

- **After cancellation**, no new updates should be provided for that shipment. The tracking session is considered terminated. If the shipper tries to call Update after this, the carrier may either return an empty result or an error indicating tracking is not available. Essentially, the Cancel confirms that the visibility feed for that load is complete.

- **Error Handling**: If the carrier cannot find an active tracking session for the given ID (for instance, if tracking was never started or was already canceled), the carrier should still respond in a clear manner. Possibly a 404 Not Found (if it treats it as “no such tracking”) or a 200 OK with a message “No active tracking to cancel for shipment XYZ” – either way conveying that at the time of the request, nothing was active to cancel. This is related to idempotency: it shouldn't be a fatal error if you try to cancel something that isn't there. Only truly invalid requests (like missing ID) should prompt an error response.

Once the cancel is confirmed, the lifecycle for that shipment's visibility tracking is concluded in the context of this API. If the shipment itself was canceled mid-transit, the physical implications (like returning freight to the shipper) are outside the scope of the API, but the digital record will show an end-of-life (potentially with a "Canceled" event as the last record). If the shipment was delivered normally, the Cancel call is mostly a formality to indicate "no need to keep this subscription live anymore."

References

Dates & Date/Times

This API Standard will reference ISO 8601 for all dates and date-times

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.

In-Transit Visibility Typical Flow

Process Flow Summary: Using the above operations, a typical end-to-end scenario might look like this:

A shipper tenders a load to a carrier via the standard tender API (or other means) and receives a unique shipment ID in return.

The shipper (or 3PL) then calls Create Visibility with that ID to initiate tracking. The carrier's system responds affirmatively and may provide the first status (e.g., "Pickup scheduled" or "In transit").

As the truck picks up and moves the freight, the carrier's system logs events: departure from origin, perhaps a location ping in transit, arrival at destination, etc.

The shipper's system periodically calls Update Visibility to fetch these new events (or the carrier pushes them as they occur).

All parties now see a timeline of status events and location updates in sync – for instance, a 3PL dashboard, the shipper's internal system, and the carrier's tracking page would all reflect the same information.

This continues until the shipment is Delivered, at which point the final event is captured. The shipper might then call Cancel Visibility to formally stop tracking (or the carrier auto-terminates the feed after delivery).

The carrier confirms cancellation, and the digital visibility session is closed. Throughout this process, because everyone used the standard API, the data exchanged (statuses like "Picked Up", locations, timestamps) were immediately understood by all systems without custom integration work.

This seamless flow is exactly what the NMFTA In-Transit Visibility standard is designed to achieve: a universally interoperable, real-time tracking experience for freight shipments from end to end.

Figure 1: In-Transit Typical Flow

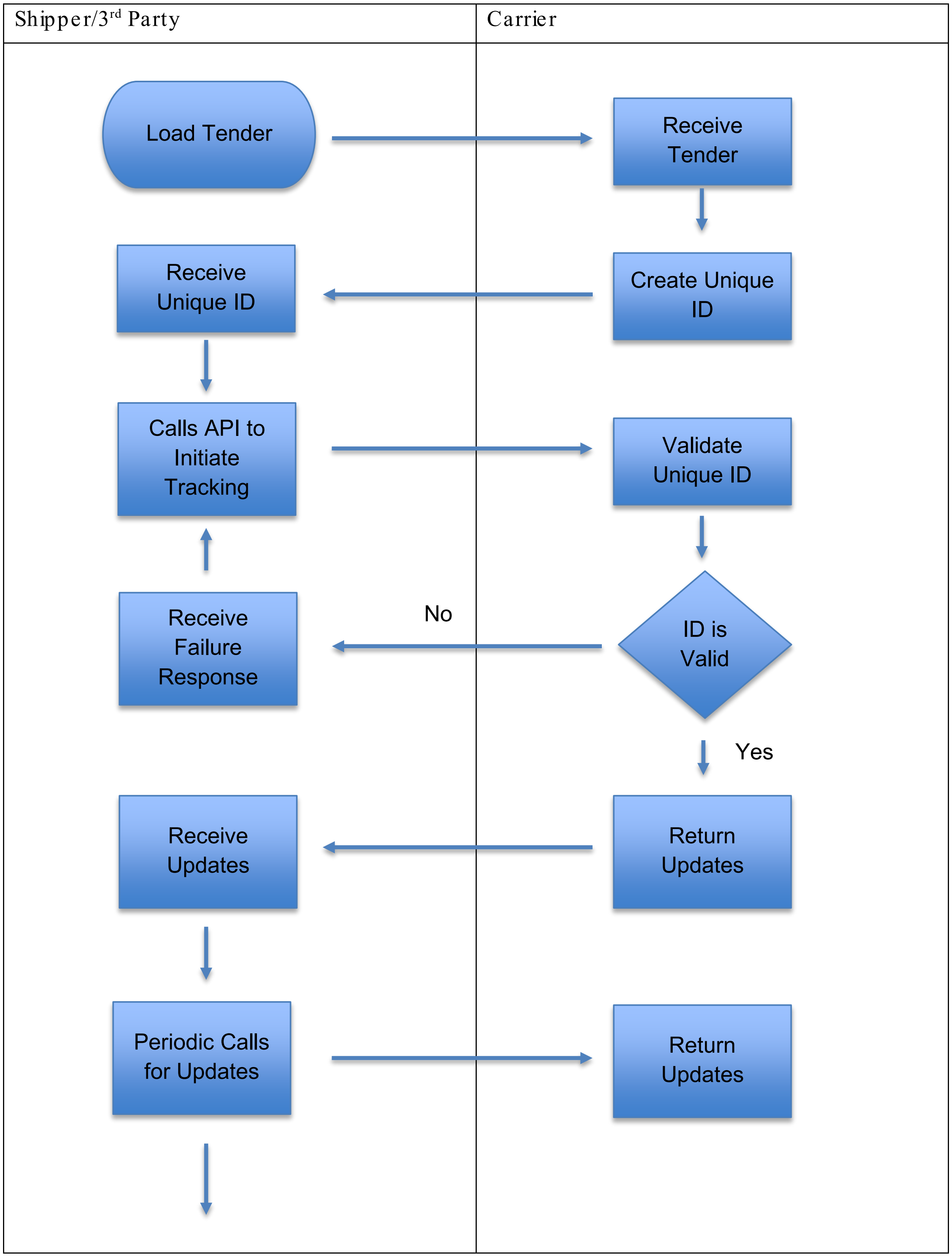
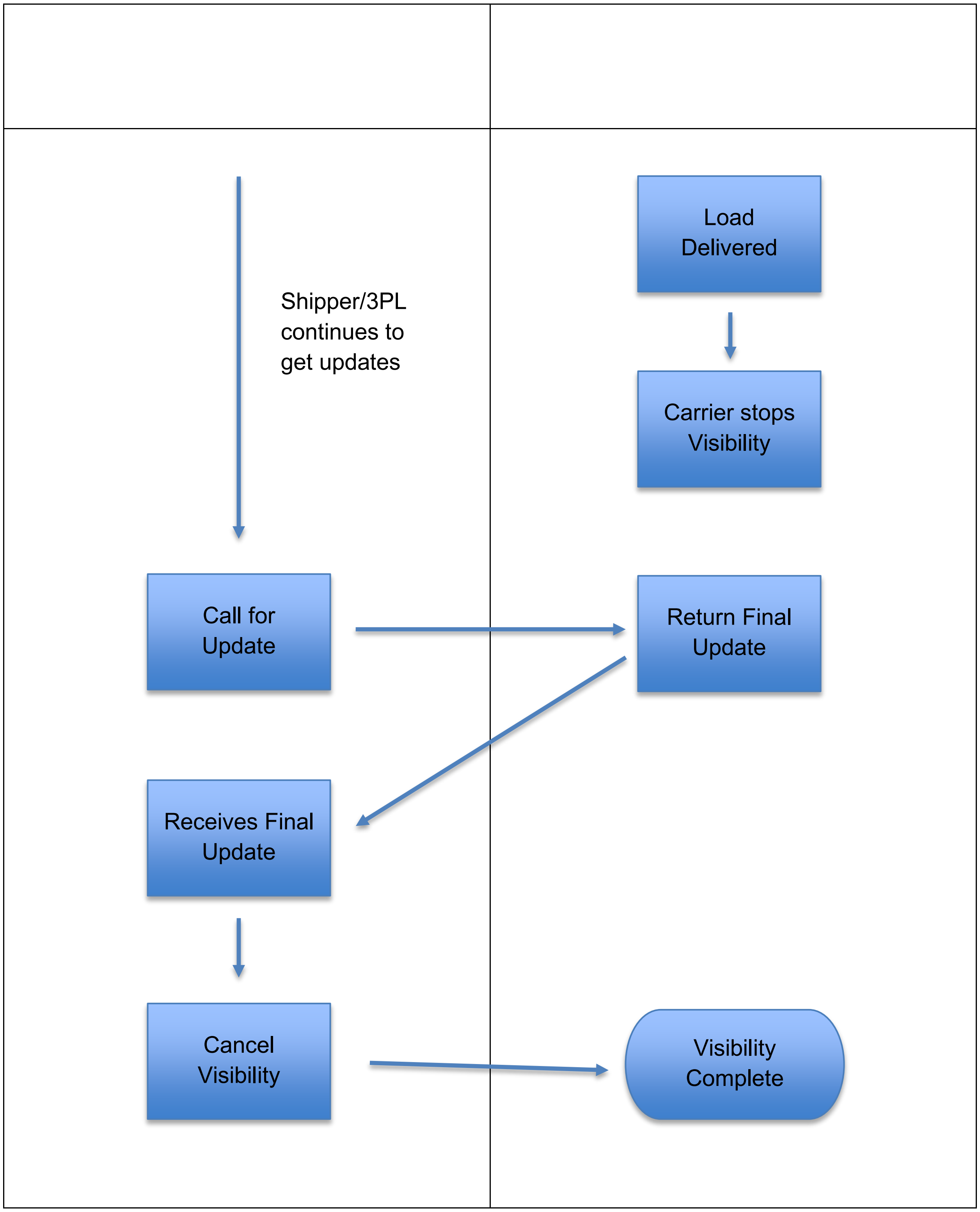


Figure 1: In-Transit Typical Flow (Continued)


Shipper/3PL continues to get updates
Load Delivered
Carrier stops Visibility
Call for Update
Return Final Update
Receives Final Update
Cancel Visibility
Visibility Complete