# Gaps in TypeSpec Directory Structure Guidelines - Evaluation of liftrneon Service

## Overview

This document evaluates the new TypeSpec directory structure guidelines against the existing liftrneon service directory to identify gaps and ambiguities that may present challenges during restructuring efforts.

## Identified Gaps and Ambiguities

1. **Top-Level Directory Naming Confusion**
   * **Issue**: The guidelines state that the folder immediately under specification should be the RPNamespace name, but the current structure has liftrneon which doesn’t match the actual RP namespace, which appears to be Neon.Postgres.
   * **Question**: Should the top-level directory be renamed from liftrneon to Neon.Postgres? Or is liftrneon an acceptable “orgName” that should be retained?
   * **Impact**: This creates confusion regarding whether to restructure in place or create a new directory structure.
2. **Service Name Determination**
   * **Issue**: The current structure has Neon.Postgres.Management as the service directory, but it’s unclear what the appropriate service name should be in the new structure.
   * **Question**: Should this be renamed to something like NeonPostgres, or should the current name be preserved?
   * **Impact**: Uncertainty about the correct service name affects the entire restructuring process.
3. **Operations Service Requirement**
   * **Issue**: The guidelines mandate an operations service folder, but the current structure doesn’t have one.
   * **Question**: What content should go into this new operations service folder? How much of the existing code needs to be extracted to form this service?
   * **Impact**: Service teams may not know how to create this required operations service from existing code.
4. **Models Directory vs. models.tsp File**
   * **Issue**: The current structure has a separate Neon.Postgres.Models directory with multiple model files, while the guidelines suggest a single models.tsp file.
   * **Question**: Should multiple model files be consolidated into one models.tsp file, or is a models directory with separate files acceptable?
   * **Impact**: Significant refactoring might be required if consolidation is mandatory.
5. **LiftrBase Dependencies**
   * **Issue**: The current TypeSpec files have references to LiftrBase.Data, but the guidelines don’t address how to handle external or shared dependencies.
   * **Question**: How should shared dependencies like LiftrBase be structured and referenced in the new directory structure?
   * **Impact**: Import paths may break if dependencies aren’t properly handled.
6. **Examples Organization**
   * **Issue**: Examples exist in both the TypeSpec structure and under resource-manager, but the guidelines are not clear about how to consolidate or reconcile these.
   * **Question**: Should examples from both sources be combined, or should they be kept separate based on their origin?
   * **Impact**: Potential duplication or loss of examples if not properly migrated.
7. **Relative Path Configuration in tspconfig.yaml**
   * **Issue**: The guidelines don’t provide specific guidance for updating relative paths in tspconfig.yaml files when restructuring.
   * **Question**: How should paths like emitter-output-dir, output-file, and examples-dir be updated to align with the new structure?
   * **Impact**: Build errors or warnings may occur if paths are not correctly updated.
8. **Multiple API Versions Handling**
   * **Issue**: The service has multiple API versions (2025-03-01-preview, 2025-03-01), but the guidelines aren’t explicit about how these should be represented in the TypeSpec files.
   * **Question**: How should the restructured TypeSpec files handle multiple API versions? Should there be separate directories or files?
   * **Impact**: API versioning might not work correctly if not properly implemented.
9. **README.md Requirements**
   * **Issue**: The guidelines require a readme.md file in the service directory with specific autorest YAML blocks, but don’t detail what these blocks should contain.
   * **Question**: What specific content and YAML blocks are required in the new service directory’s readme.md file?
   * **Impact**: Autorest generation might fail if the readme file isn’t properly configured.
10. **Cleanup of Old Structure**
    * **Issue**: The guidelines don’t address how to handle the transition period or cleanup of the old structure.
    * **Question**: After restructuring, should the old directories be immediately removed, or should there be a transition period with both structures?
    * **Impact**: Potential disruption to existing workflows if old structures are removed too quickly.

## Recommended Clarifications

1. Provide clear guidance on whether the top-level directory name (liftrneon) should be changed to match the RPNamespace.
2. Clarify the naming convention for the service directory, particularly when the current name doesn’t match the expected format.
3. Provide concrete examples of how to extract and implement the required operations service.
4. Specify whether model files can remain separate in a models directory or must be consolidated into a single file.
5. Offer guidance on handling external or shared dependencies.
6. Clarify the expected organization of examples from different sources.
7. Provide specific guidance for updating relative paths in configuration files.
8. Detail how to handle multiple API versions in the restructured TypeSpec files.
9. Provide a template or example of the required readme.md file with autorest YAML blocks.
10. Outline a recommended approach for transitioning from the old structure to the new one.