Solution Overview: DN Host Application

Architecture: Raj

Objective

A Host or Shell application serves as a base web application that binds MFEs that make up the functionalities application. Host shall also provide functions that are essential for proper working of all micro-front-end fragments/views such as user login, view routing, feature flags, permissions, user analytics etc.

Dealer Navigator (DN) with its numerous application offerings currently is designed to have multiple host or shell web applications. This requires the common functions to be shared between all host applications. This document analyses and describes key problem areas with the current design of DN host applications and provides solutions that focus on performance, ease of maintenance and feature scalability.

Current DN Host/Shell Application Architecture

Tbd

Problem Statements

Maintenance

The common functions from Dealer Platform(DP) are made available for all DN host applications via ‘DN Shell Lib’ an angular library. This creates a number of side effects limiting the ease of maintenance and adding release friction.

1. Every update the ‘DN Shell Lib’ will warrant an update/adoption in all host applications. Some of these updates can be as simple as updating the version of lib consumed by the host application, while others can demand code changes in the consuming host for proper working.
2. Updates to the ‘DN Shell Lib’ and cascading the change across all DN host applications will require more effort to communicate, align and plan releases.

Performance

Navigation from one DN view to another can trigger moving from one host application to another and is subjected to the following side effects,

1. Re-fetch and re-render of the entire application including Navbars, header and footer. HTML, JS bundles and any other static assets are re-fetched during the navigation. This not only increases the overall asset transfer size between user’s device and the web server but also abrupts smooth and swift transition between pages/views.
2. Data and context of the view is recreated upon view navigation even if the view prior to this navigation had most if not all necessary context. Examples of such contextual data includes dealer info, permissions, feature toggle data etc.

Host and MFE Communication Model

Host application and its micro front-ends often exchange pieces of information that are necessary for proper functioning of the application. This often is a set of properties within the MFEs whose values are assigned by the Host application through a contract with individual MFEs. This however has led to problems like,

1. Maintenance of MFE properties and Data Mapping (at code level)
2. Explosion of properties sometimes redundant values
3. Backward compatibility of older property names

Scalability

‘App Shell lib’ contains shared web components across all DN hosts and common capabilities or functionalities such as Auth, Feature highlight, Permissions, Feature flags etc. Some of these capabilities are owned by different federated teams and can lead to dependency issues if not coordinated appropriately between teams. Scaling features in any

of these capabilities can lead to restrictions due to the nature of consumption (public contracts) in numerous host application code bases. e.g. modifying a parameter in a public API method from Shell Lib for Feature toggle capability can be expensive to accomplish.

Ease of Migration

Although attempts have been made in the past to unify the DN host application and migrate the view/page MFEs to 'DN Shell', the host application maintained by Dealer Platform, the progress was minimal and far from the then target architecture. One of the primary causes for this slow down of migration is lack of 'backward compatibility' when upgrading common capabilities to newer and target architecture. It required both Dealer Platform and other federated teams to align the adoption timeline for a migration to be successful.

A 'backward compatible' solution with a migration window, could ease federated teams with Host application to plan and execute the migration to 'DN Shell'.

Solution and Architecture

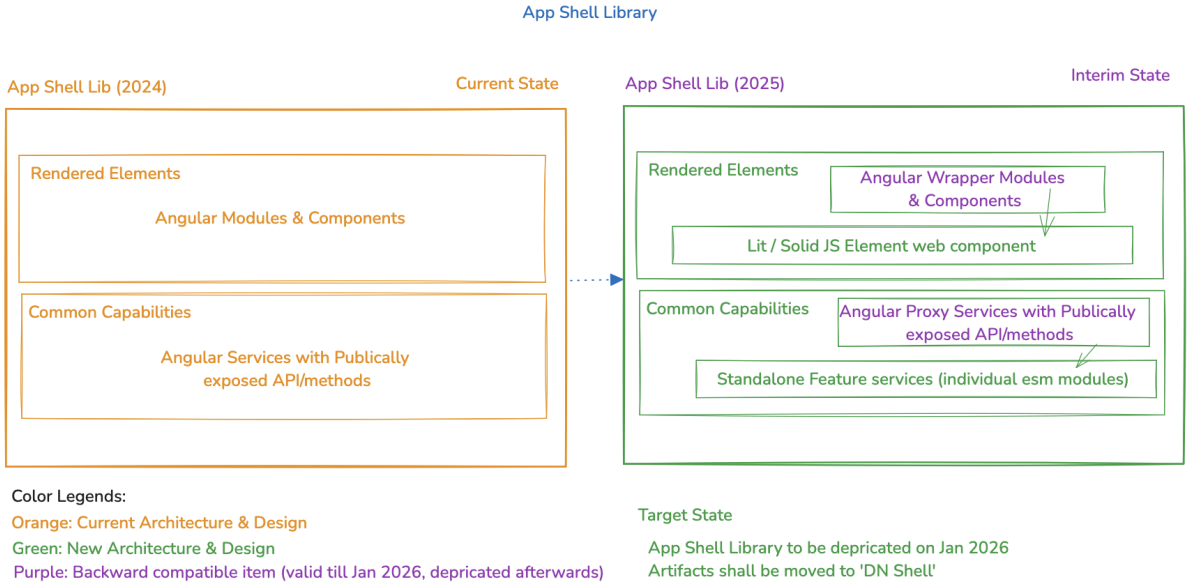
The section addresses the issues in Problem Statements section and describes the architecture of different entities in 3 states,

1. **Current State** - Design implemented by the entity at present
2. **Interim State** - Design that shall be implemented during the 'migration window' for backwards compatibility. Host Applications teams shall be able to plan and migrate to target state design during this window
3. **Target State** - Final architecture and design that fully addresses the Problem Statements

Goals,

1. Modernize 'DN Shell' Host Application to improve performance, scalability, maintenance and MFE communication model
2. Facilitate smooth transition of all other DN Host Applications to migrate its views/pages as runtime pluggable MFEs in 'DN Shell' maintained by Dealer Platform

Common Capability web components and functions



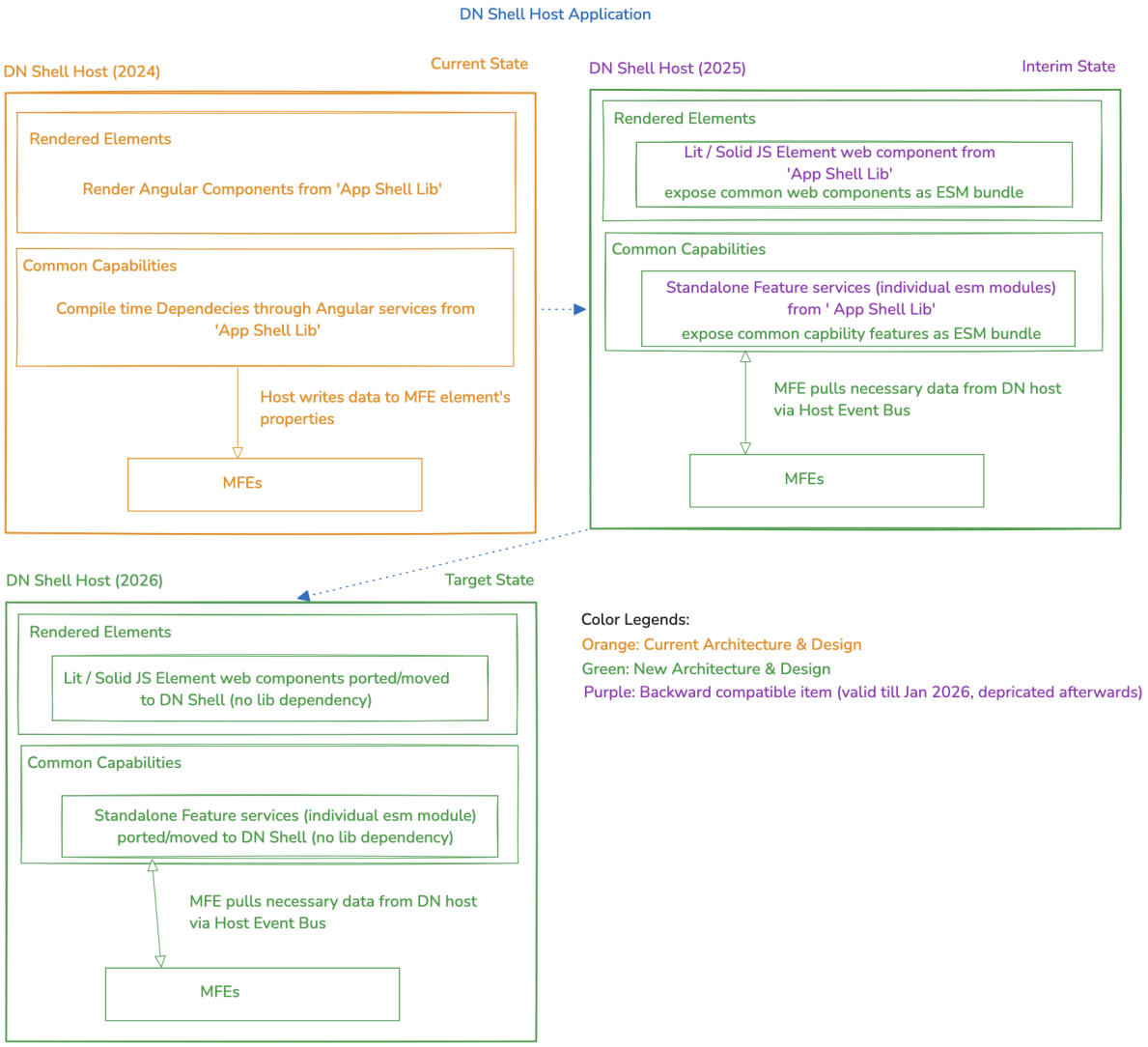
Color Legends:
Orange: Current Architecture & Design
Green: New Architecture & Design
Purple: Backward compatible item (valid till Jan 2026, depricated afterwards)

Target State

App Shell Library to be depricated on Jan 2026
Artifacts shall be moved to 'DN Shell'

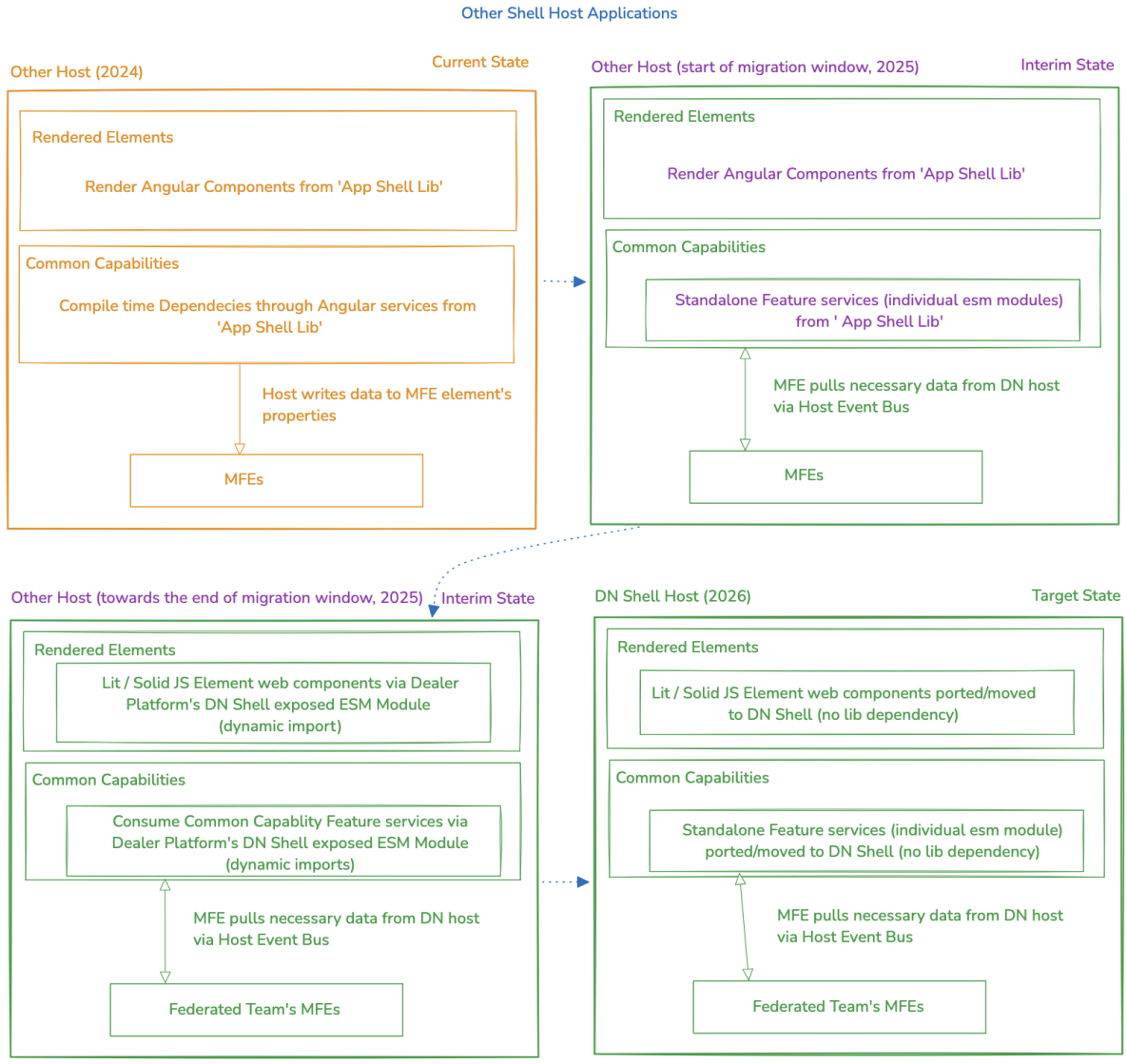
<tbd explain>

DN Shell Host Capabilities and Communication



<tbd explain>

Other DN Application Hosts



Color Legends:
Orange: Current Architecture & Design
Green: New Architecture & Design
Purple: Backward compatible item (valid till Jan 2026, depricated afterwards)

<tbd explain>

Host and MFE Communication Model