# POC■0 Achievements & Implementation Summary
This document summarizes the major milestones, fixes, configuration corrections, and successful outcomes of completing **POC■0 of the Universal Web + Mobile Microfrontend Platform**.
---
# ■ Overview
POC■0 was designed to validate the most technically difficult part of the entire architecture:
> **A fully universal, platform■agnostic microfrontend that runs natively on React Native and loads dynamically from a remote server using Module Federation v2 + Re.Pack.**
This has now been **successfully achieved**.
The system can:
- Build a **mobile remote MFE** using React Native + Re.Pack
- Serve the remote bundle + manifest
- Load it dynamically into the **mobile host** over the network
- Render a shared universal component (`HelloUniversal`)
- Share state, events, and props across MF boundary
- Execute on Hermes with no globals like `exports` / `require`
- Fully support React Native primitives (View, Text, Pressable)
- Work reliably on Android emulator networking (10.0.2.2)
This officially completes **POC■0**.
---
# ■ Major Achievements
## 1. Mobile Host Successfully Booted with Re.Pack
The RN Metro bundler was replaced with Re.Pack via Rspack, with correct:
- `resolve` configuration for RN internals
- Hermes compatibility
- Correct target: `target: Repack.getRepackTarget(platform)`
- Working dev server
- Working Module Federation runtime
This removes all Metro limitations.
---
## 2. Mobile Remote MFE Built & Served Correctly
A remote bundle is now generated at:
- `HelloRemote.container.js.bundle`
- `__federation_expose_HelloRemote.bundle`
- `mf-manifest.json`
The manifest exposes:
- `./HelloRemote` → resolves to `HelloUniversal`
---
## 3. Host Successfully Loaded Remote MFE at Runtime
The ScriptManager workflow was validated:
1. Resolve manifest
2. Load manifest
3. Resolve remote entry
4. Load remote entry script
5. Fetch federation expose chunk
6. Execute bundle in Hermes
7. Render RN component tree
This proves:
- Working MF runtime
- Working RN integration
- Working Hermes-compatible execution
- Working cross-bundle module resolution
---

## 4. Universal Component Strategy is Proven
The remote loads and renders a shared universal component:
- Built using **React Native primitives only**
- Compatible with both mobile and web
- Uses shared utils package
- Fully cross-platform
---
## 5. Props & Events Verified
Pressing “Press Me”:
- triggers remote code
- increments remote state
- re-renders inside host
This validates:
- host → remote prop passing
- remote → host UI updates
- MF boundary is transparent to React
---
## 6. Networking & Bundler Edge Cases Solved
During debugging, platform handled:
- Android emulator 10.0.2.2 vs LAN IP
- Missing bundle paths
- Incorrect manifest publicPath
- Hermes global absence (`exports`, `require`)
- Duplicate asset emission
- Babel ENOENT issues
- Yarn symlink edge cases
- Target mismatch between web/mobile builds
All resolved.
---
# ■ What POC■0 Validates
| Requirement | Status |
|---|---|
| RN host with Re.Pack | ■ Complete |
| MFv2 mobile runtime | ■ Complete |
| Build remote RN bundle | ■ Working |
| Serve bundle + manifest | ■ Working |
| Load remote MFE at runtime | **■ COMPLETE** |
| Universal RN component rendering | **■ COMPLETE** |
| Shared libraries resolution | **■ COMPLETE** |
| Network-based remote loading | **■ COMPLETE** |
| Hermes evaluation success | **■ COMPLETE** |
| No Metro involvement | **■ COMPLETE** |

**POC■0 is officially 100% COMPLETE.**
---
# ■ Conclusion
This milestone proves:
> **The Universal Web + Mobile MFE Platform works.
A single microfrontend can now run on Web, Android, and (later) iOS with shared code and runtime federation.**
This is the hardest milestone—and now it’s done.
■ Onward to POC■1.
# Architecture Diagrams — Universal Web + Mobile MFE Platform (POC■0)
Below are **text■based architecture diagrams** suitable for GitHub docs.

In POC■1 these can be converted to draw.io / Mermaid.
---
## 1. Universal Microfrontend Platform — High■Level Overview
```
+------------------------------+
| Monorepo (Nx) |
| universal-mfe-yarn-seed |
+------------------------------+
|
---------------------------------------------------------------------
| | |
+---------------+ +------------------+ +--------------------+
| Web Shell | | Mobile Host | | Microservices |
| (Rspack) | | (Re.Pack + RN) | | Node.js services |
+---------------+ +------------------+ +--------------------+
| | |
Remote MFEs Remote MFEs Auth / API / Payments
```
---
## 2. Mobile Host Architecture (Re.Pack + RN + MFv2)
```
+---------------- Mobile Host (Android/iOS) ----------------+
| |
| React Native runtime + Hermes |
| Re.Pack Dev Server (8081) |
| Rspack bundler for RN (index.bundle) |
| |
| Module Federation Runtime |
| ------------------------------------------- |
| - loads manifest from remote |
| - resolves URL: http://IP:9004/HelloRemote.container... |
| - loads secondary chunk: __federation_expose_...bundle |
| |
| App.tsx (Host Shell) |
| ■■■ → ScriptManager.loadRemote("Hello...") |
| |
+-------------------------------------------------------------+
```
---
## 3. Mobile Remote (HelloRemote) Build Pipeline
```
packages/mobile-remote-hello/
|
repack.remote.config.mjs
|
+---------------------------+
| Rspack (RN-target build) |
+---------------------------+
|
Dist folder output:
- HelloRemote.container.js.bundle
- __federation_expose_HelloRemote.bundle
- index.bundle
```

- mf-manifest.json
```
---
## 4. Module Federation Data Flow (Mobile)
```
Mobile Host Remote (Hello)
| |
|-- Fetch manifest ----------------->|
|<-- mf-manifest.json ------------- |
|
|-- GET HelloRemote.container.js -->|
|<-- Bundle (global entry) ---------|
|
|-- GET exposed chunk --------------|
|<-- __federation_expose...bundle --|
|
+--> Remote Component Renders
```
---
## 5. Universal Shared Libraries
```
@universal/shared-utils
- getGreeting()
@universal/shared-hello-ui
- HelloUniversal (RN primitives only)
→ Works in Web (RNW) & Mobile (RN)
```
---
## 6. POC■0 Proven Architecture
```
• Web & Mobile both run MFEs
• No coupling between MFEs
• Host loads remote at runtime
• MFv2 wiring functional
• RN + Rspack + Re.Pack bundle chain validated
• RNW for web remote validated
• Shared libs resolve correctly
• Hermes executes MF chunks successfully
```
# Lessons Learned — POC■0 Retrospective
POC■0 was intentionally painful — it exercised every edge of RN + MFv2 + Rspack + Re.Pack.
Here are the distilled learnings.
---
## 1. React Native + Module Federation is extremely sensitive to bundler outputs
**Key issues hit:**
- Wrong bundle filenames (`index.js` vs `index.bundle`)
- Wrong publicPath (`auto` breaks Hermes)
- Remote URL resolution using wrong IP
- Extra chunks not being resolvable
- Rspack emitting *web-style* chunks for RN
**Takeaway:**
RN is not Webpack. Every MF bundle must be deliberately shaped for Hermes.
---
```

## 2. Re.Pack must be used consistently for ALL mobile builds
Switching between:
- `rspack build`
- `repack build`
caused inconsistent output and missing RN stubs.
**Takeaway:**
All mobile remotes MUST be built using **Re.Pack**, never Webpack-style commands.
---
## 3. Shared libraries must be pure RN primitives
Errors like:
```
TypeError: Cannot read property 'OS' of undefined
TypeError: Cannot read property 'default' of undefined
```
were caused by:
- A remote bundling `react-native-web`
- Host expecting `react-native` only
**Takeaway:**
Shared UI MUST be RN■primitives only and version-aligned.
---
## 4. mf-manifest.json must reference RN bundles, not web bundles
MFv2 generates:
- `remoteEntry.js`
- `HelloRemote.container.js.bundle`
- Secondary chunks: `__federation_expose_*.bundle`
Host must load the `.bundle` versions.
**Takeaway:**
Manifest rewriting rules must be part of POC■1 tooling.
---
## 5. Android Emulator networking requires strict rules
- `localhost` → **host machine inside emulator = 10.0.2.2**
- Rspack dev server sometimes binds IPv6 first
- Webpack dev server auto■redirects to IPv6
**Takeaway:**
For mobile MFEs: always use `adb reverse` or fixed IPv4 address.
---
## 6. Repack stubs are required for missing RN DevTools internals
Missing modules required:
```
ReactDevToolsSettingsManager.js
NativeReactDevToolsRuntimeSettingsModule.js
```
Without these, remote bundles fail evaluations.
**Takeaway:**
Stub files MUST be included in all mobile remotes.
---
## 7. Node_modules hoisting affects RN gradle builds
RN expects native Gradle plugin at:
```
node_modules/@react-native/gradle-plugin
```
When hoisted, Android build breaks.
**Takeaway:**

mobile-host must keep RN dependencies local (non-hoisted).
---
## 8. Final Proof
Despite all challenges:
- Remote bundles load
- Hermes executes federation chunks
- Remote UI renders on RN
- Shared libs function cross■platform
This validates the **entire Universal MFE architecture**.
---
# POC■0 → POC■1 Transition Plan
POC■1 focuses on **stabilization**, **tooling**, and **scalability**.
Below is a clean, actionable roadmap.
---
# 1. Stabilize Build System (High Priority)
### ■ Deliverables
- Replace all raw Rspack usage with **Re.Pack wrappers**
- Introduce unified build scripts:
- `build:mobile-remote`
- `start:mobile-remote`
- `serve:mobile-remote`
- Standardize output filenames:
- `.container.js.bundle`
- `__federation_expose_.bundle`
### ■ Goal
Zero ambiguity about what bundle the host loads.
---
# 2. Introduce Federation Manifest Rewriter
### Why
MFv2 emits web-centric manifests. We need RN-centric ones.
### Tasks
- Implement a post■build script:
- Convert remoteEntry paths → RN bundle paths
- Remove web-only fields
- Ensure valid URLs for federation chunks
- Validate for both dev & prod builds.
---
# 3. Shared Libraries Hardening
### Tasks
- Freeze RN version across all packages
- Freeze React version across all MFEs
- Add CI check: **fail if any MFE imports DOM elements**
### Deliverables
- `/shared/validation/no-dom-imports.js` script
- `/shared/validation/version-lock-check.js`
---
# 4. Mobile Host Enhancements
### Objectives
- Add Remote Registry
- Add dynamic remote config loader
- Add error boundaries around MF loading
- Provide a host navigation bridge to MFEs
### Example API

```ts
Host.registerRemote({
id: "profile",
url: "http://10.0.2.2:9010/Profile.container.js.bundle",
});
```
---
# 5. Web Host Enhancements
### Deliverables
- Web shell with MFv2 (Rspack) routing
- Basic auth flow stub
- Remote loading UI skeleton
---
# 6. Developer Tooling (POC■1 Quality of Life)
### Tools to add
- `dev:mobile` single command (runs host + remote)
- `dev:web` command
- Automatic IP detection for Android remotes
- Watch mode rebuild for mobile-remote
---
# 7. CI/CD Only for POC■1 (Not POC■0)
### Deliverables
- Pipeline for mobile-remote build artifacts
- Pipeline for host
- Publish remote bundles to:
- S3
- or simple static hosting folder
- Automated manifest publishing step
---
# 8. Documentation
### Deliverables
- `/docs/poc1/build-system.md`
- `/docs/poc1/remote-conventions.md`
- `/docs/poc1/host-api.md`
- `/docs/poc1/developer-flow.md`
---
# 9. POC■1 Exit Criteria (Definition of Done)
### ✔ Mobile Host loads 2+ remotes
### ✔ Web Shell loads 2+ remotes
### ✔ Shared libs used across all MFEs
### ✔ Manifest rewriting fully automated
### ✔ CI builds remote bundles
### ✔ Dev environment: 1 command → everything runs
---
# 10. POC■2 Preview (For Later)
- Payments domain wiring
- Global event bus
- Auth service
- Navigation unification
- Versioned MF loading
- Production hosting standards
---