



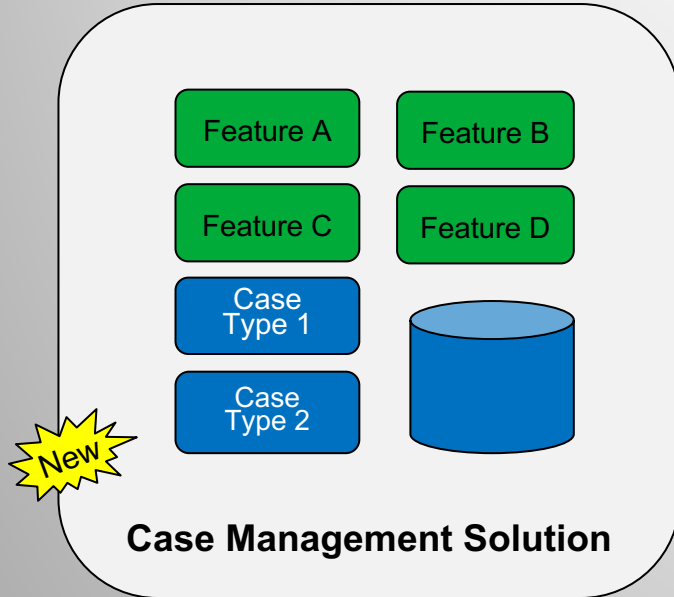
Slay the Monolith: Our journey to Federate Case with JBPM

Presenters

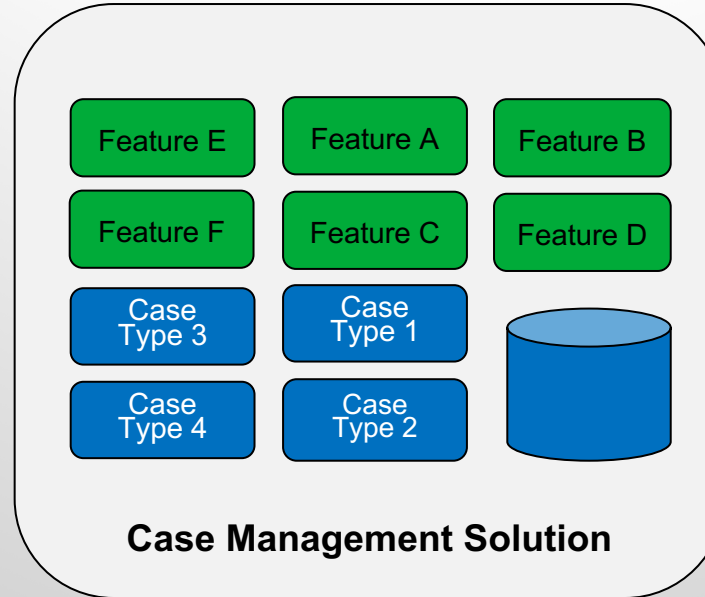
Michelle Kelo, Director, Enterprise Architect, Capital One

Tarek Aridi, Sr. Manager, Solutions Architect, Capital One

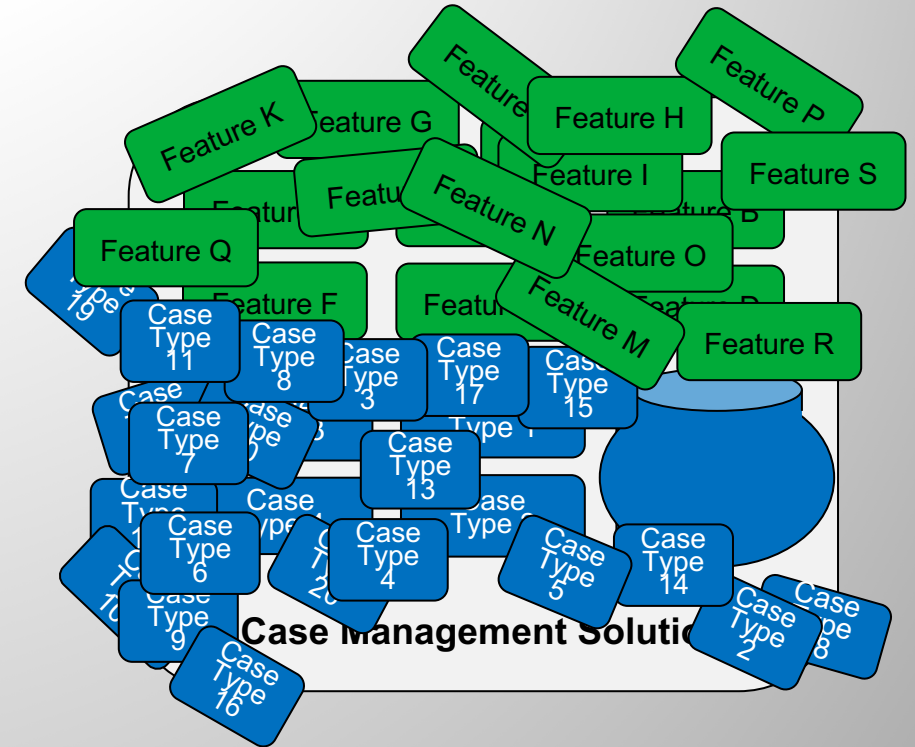
Problem Statement: *Most monolithic applications start out as simple solutions, easy to develop test and deploy... BUT as they grow larger and more complex they become harder and harder to maintain...*



- New Case Solution!
- Support OOTB case types & features
- Few or no customizations needed!
- Easy to deploy, and keep current



- Everything is going well!
- New case types are added and a few custom features.
- Still easy to develop, test and deploy.
- Maintenance and upgrades performed without issue



- Over time addition LOB's case types and features are added
- Many customizations
- Increased complexity
- Solution upgrades become large, expensive efforts

The Solution Strategy: Federation using a Case Framework

- **Federation**

- Decentralized case solutions enables better independence & control for the LOB/Domains
- Decoupled solutions reduce complexity and risk
- Allows for better resource efficiency and scale

- **Case Framework**

- Is a software tool set, providing the capability building blocks for creating case management applications
- Provides *out-of-the-box* core case features which enable our business partners to quickly build and deploy low cost customized solutions
- Is being built as an inner-sourced effort
- Leverages JBPM as its foundation

Why JBPM?

- Is truly open source
 - The Enterprise version (BPM Suite) has the same capabilities as the community version
- Enterprise version is fully supported by Redhat
- It hosts a rich set of capabilities that are key needs for case management
 - BPM using the latest BPMN2.0 specification
 - Native rules engine (Drools),
 - Business Activity Monitoring for Dashboards & Reporting
- It is a mature and proven product (14 years*) with a vibrant development community
- Is flexible and easily extensible

Architecture Key principles guiding our technology approach

Architecture Principle	Expected Outcome
<ul style="list-style-type: none">• Decompose monolithic applications into smaller manageable services• Design for Reuse	<ul style="list-style-type: none">• Improved flexibility and faster time to market• Reduced complexity• Enables Federation & team independence
<ul style="list-style-type: none">• Leverage Open Source technologies where they are mature and well supported	<ul style="list-style-type: none">• Focused resources on higher-value, differentiated capabilities• Lowers TCO
<ul style="list-style-type: none">• Select the right tool for the right use case	<ul style="list-style-type: none">• Avoidance of unnecessary or unnatural tool modifications
<ul style="list-style-type: none">• Pivot to cloud and modern technologies	<ul style="list-style-type: none">• Reliable, Resilient & Scalable infrastructure

Analysis of the capabilities common across LOB use-cases identified the target feature set for the framework

Core Case

- Create Case
- Search Case
- Retrieve & View Case
- Update Case
- Route Case
- Classify/Reclassify Case
- Assign Case
- Transfer Case
- Prioritize Case
- Pend Case
- Add Case Note
- View Case Document
- Match & Attach Case Document
- Decision Case
- Publish Case Events & Decisions
- Close Case
- Manage Parent/Child Case
- Send Case Correspondence

Agent Effectiveness

- Enable Customer View
- Get Next Case
- Guided case workflow / next best action
- Workflow Automation
- Integrated fulfillments
- Systematic Case alerts
- Case visibility during case transfer
- Flexibility in queue priority
- Configurable Role based access and fulfillment
- Capture Agent (& Customer) feedback
- Auto Notes

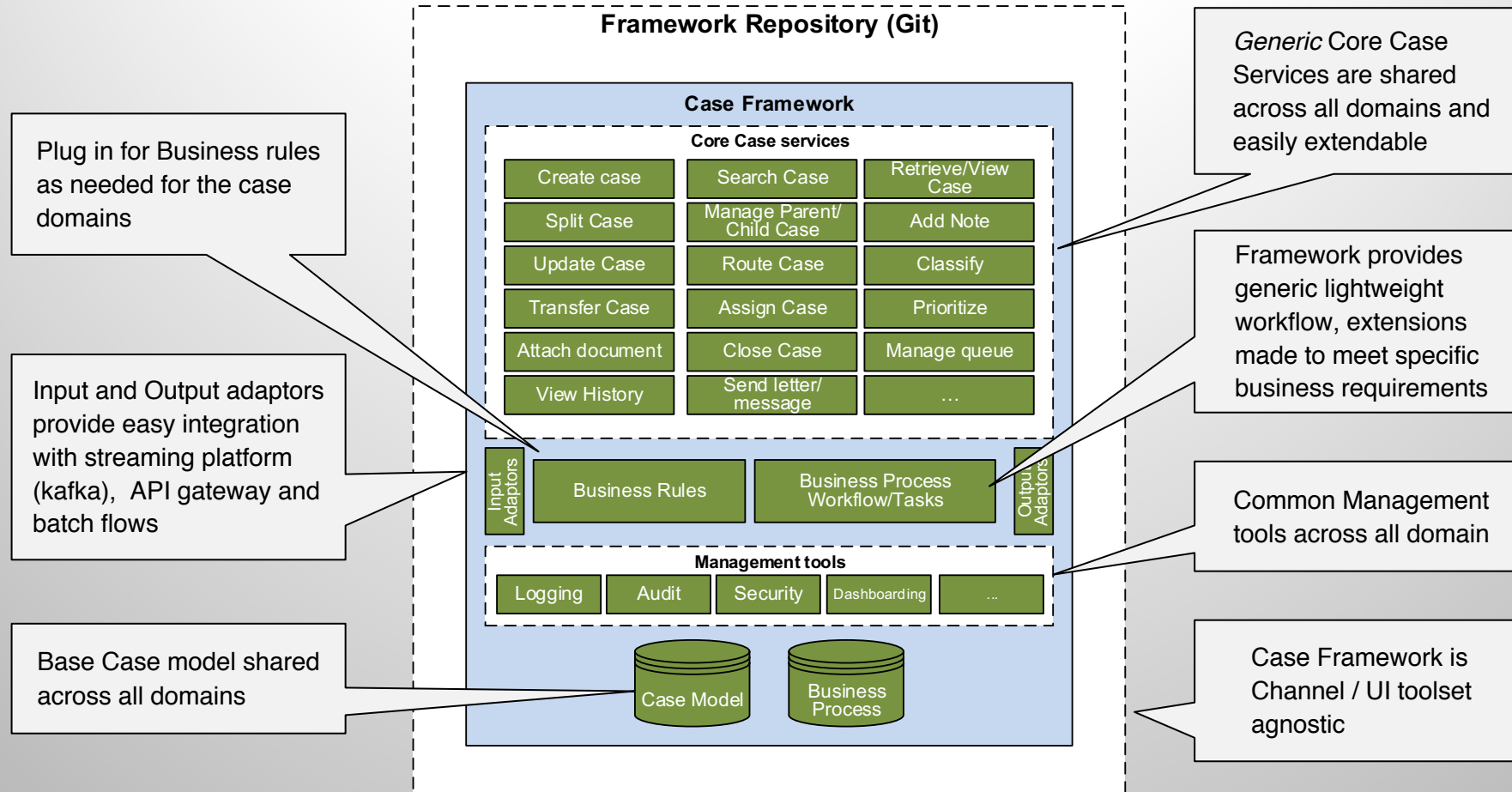
Monitoring & Reporting

- End-to-end case history
- Dashboarding
- Queue Depth Reporting
- SLA Reporting & alerts
- Analytics support
- Audit Trail – Business and Agent view into steps performed

Security & Maintenance

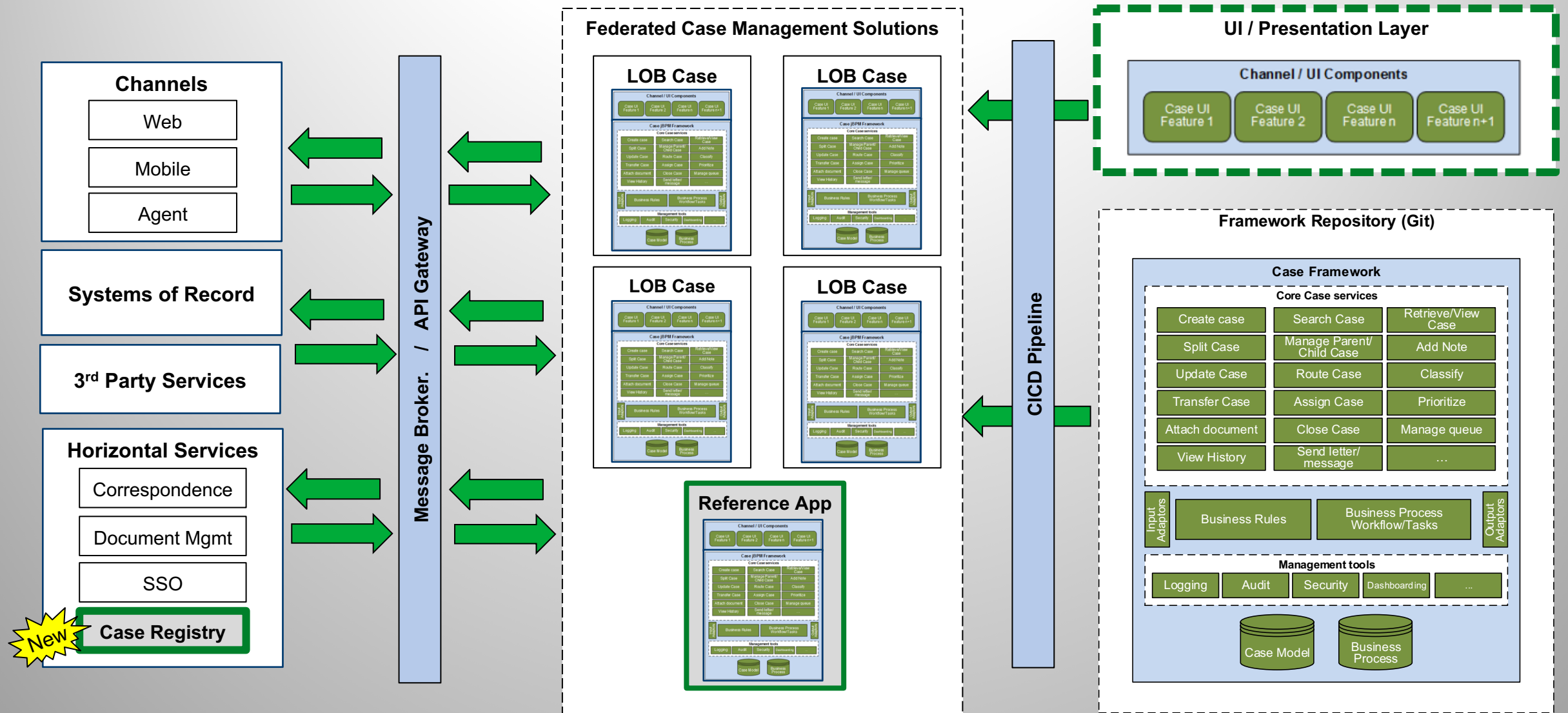
- Roles & Access Management
- Feature entitlement

The Core Case Management framework serves as a tool set of extendable Core Case Capabilities



Roughly 80% of Case Capabilities are Shared
with the remaining 20% as extended or unique capabilities per LOB domain

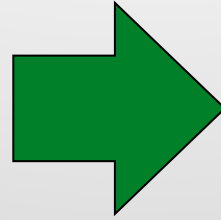
Target State: Federated Case Solutions built on JBPM based Framework



Our End-State Case Management strategy will provide LOB case autonomy via a lightweight, easily extendable JPMS framework

Starting State

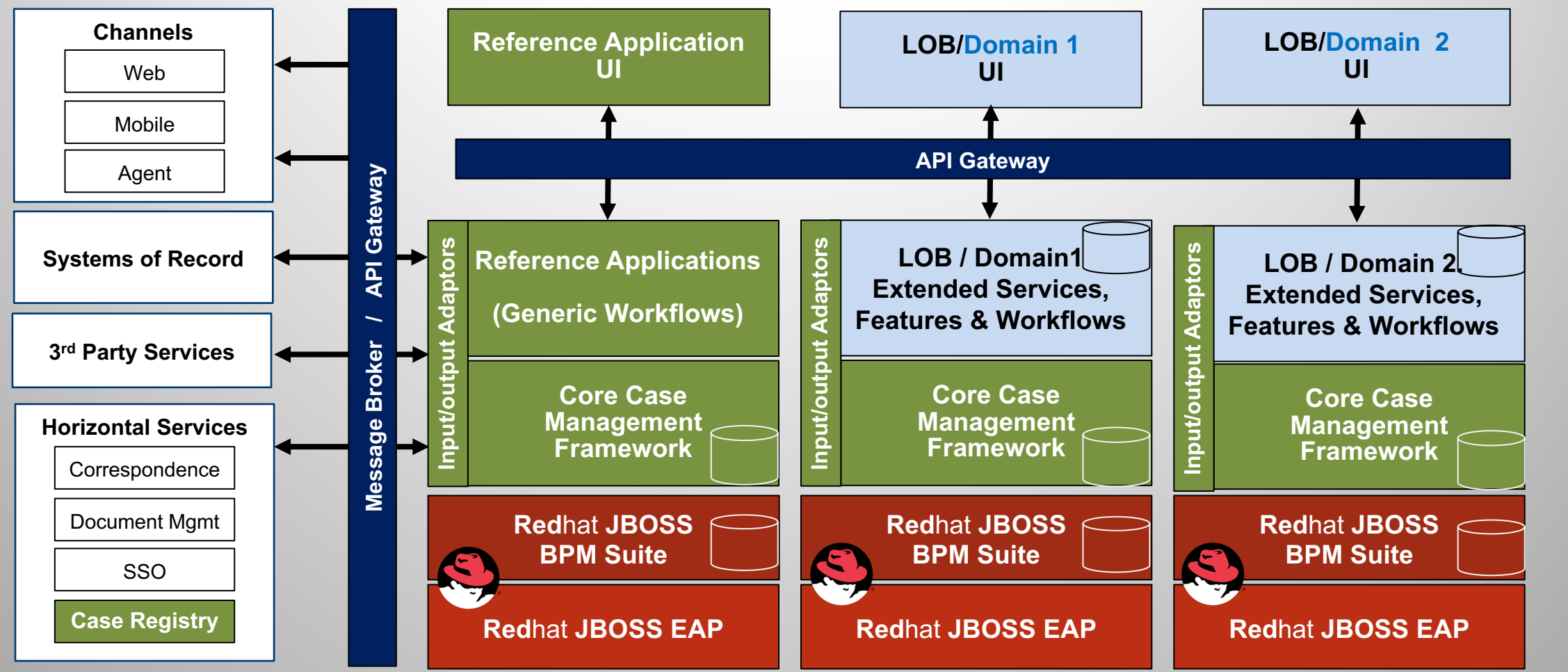
- Monolithic Case Platform
- High cost commercial based solution
- Centralized Development & Deployment
- Inflexible Case model & core services
- UI tightly bound to case engine
- Batch-oriented analytics



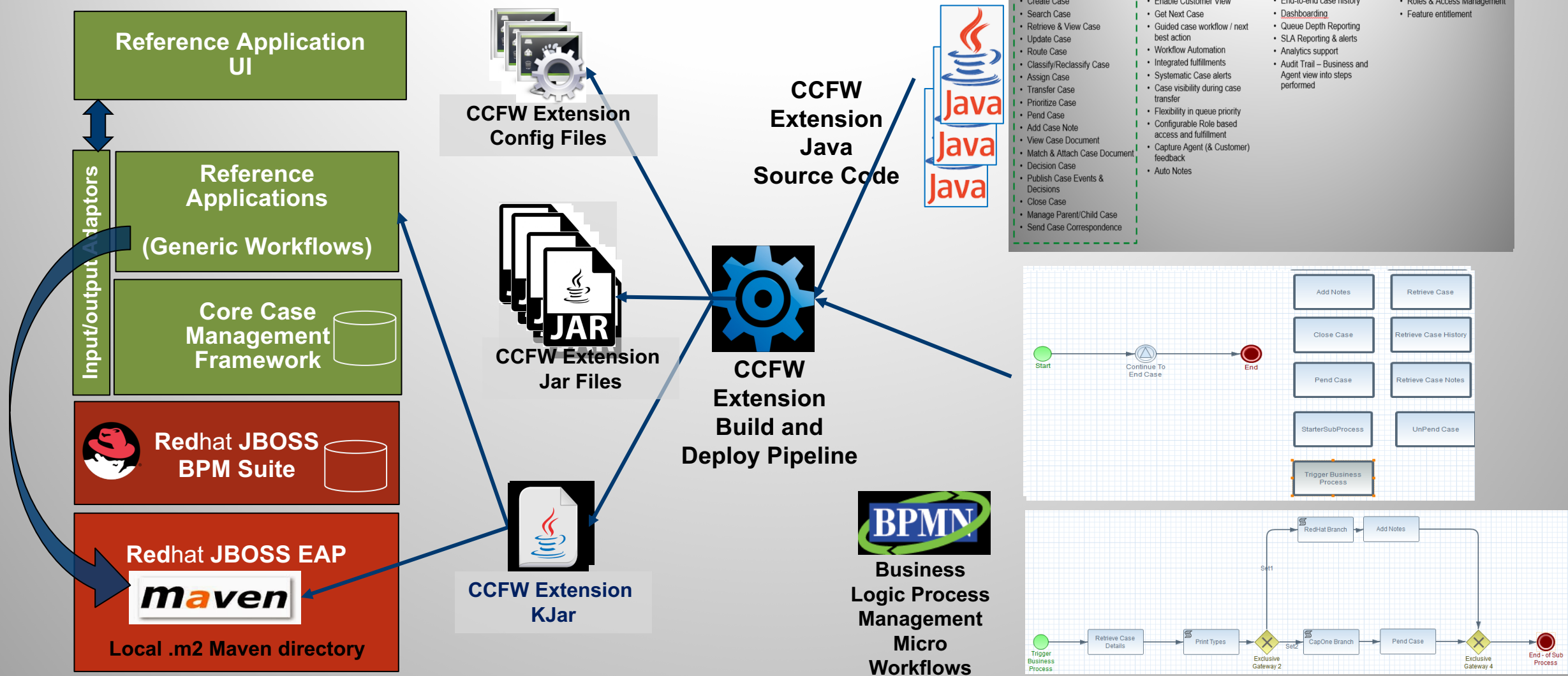
Target State

- Domain-owned Case applications with a Central Registry
- Open Source (JBPM) based framework
- Federated Development & Deployment with inner-sourcing model
- Lightweight, extensible case model & core services
- UI agnostic case engine
- Real-time analytics & events enabled via streaming platform

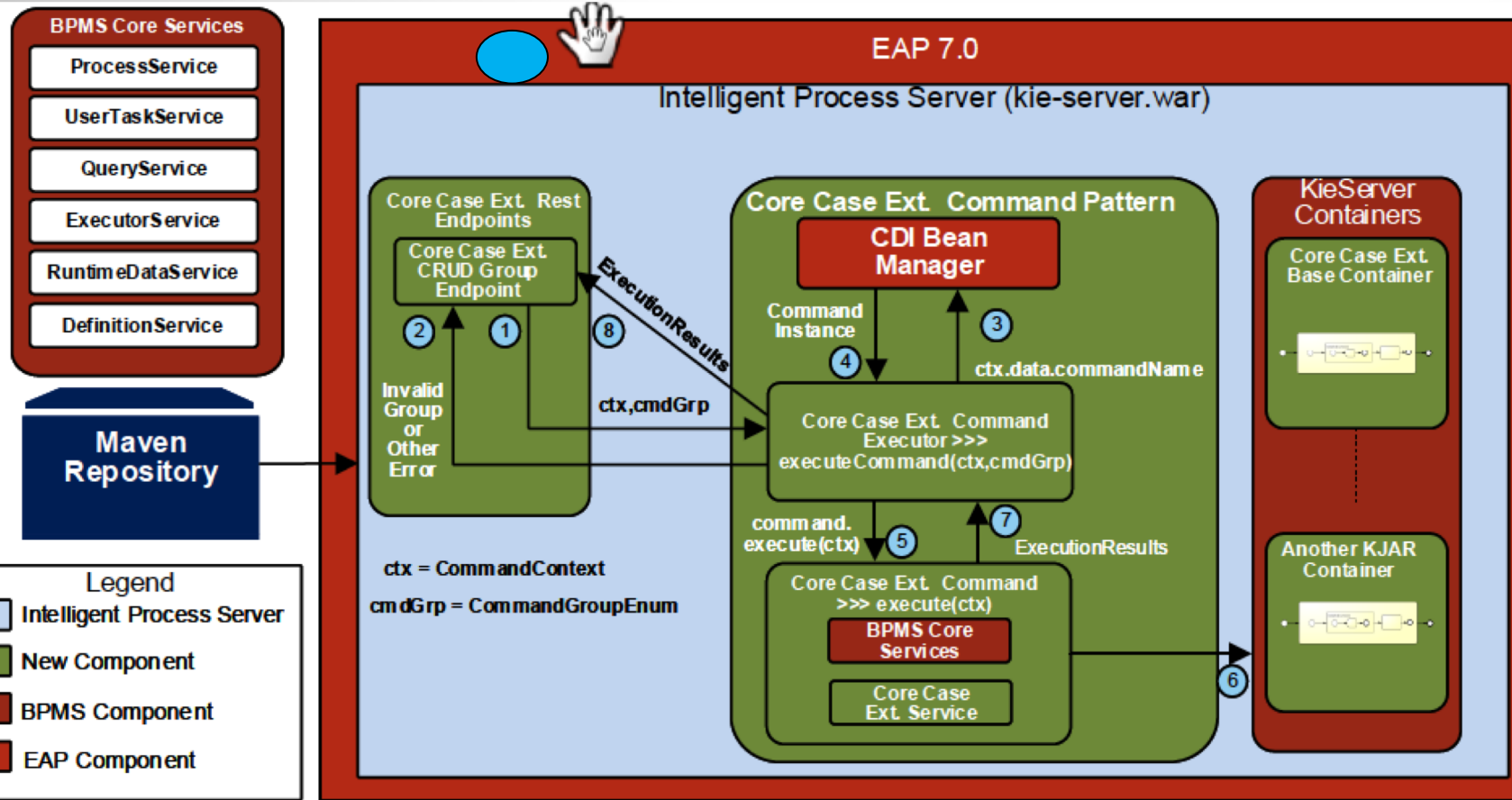
The Core Case Framework architecture consists of loosely coupled component layers built on Red Hat's BPM foundation



How we extended jBPM



Core Case Framework Component Architecture



Core Case Framework Extension Component Architecture

1. The Core Case Framework (CCFW) Rest endpoint Ext. processes incoming requests.
2. It validates requests' eligibility and content via the Command Executor. Return error response as needed.
3. The Command Executor initiates command instance.
4. The instance of the command class is created in Command Executor.
5. The execute method of the command passes the input payload to the CCFW jBPM extension.
6. The executed command performs case management actions against deployed kjar containers in the kieserver runtime. (Persists/manipulates case data into DB)
7. Returns results of the command execution
8. Returns Execution Results to the REST endpoint and subsequently to the calling clients.

Core Case Framework Demo

[Demo](#)