Prepared by:

**Poplin Working Group**

**MITA Reference Architecture**

Service Definition: Service Registration and Discovery

Final Draft

Version 1.0

October 20, 2017

Record of Changes

| Version | Date | Author / Owner | Description of Change |
| --- | --- | --- | --- |
| 0.1 | 08/31/2017 | Jeff Cook, Dave Hill, Pace Ricciardelli | Initial draft |
| 0.2 | 09/01/2017 | Jeff Cook, Dave Hill, Pace Ricciardelli | Separated research findings from service definition content |
| 0.3 | 09/19/2017 | Jeff Cook, Dave Hill, Pace Ricciardelli | Updated based on feedback from Poplin working group |
| 0.4 | 10/18/2017 | Jeff Cook, Dave Hill, Pace Ricciardelli | Version reviewed by MITRE Technical Review Board (TRB) |
| 1.0 | 10/20/2017 | Jeff Cook, Dave Hill, Pace Ricciardelli | Final Poplin v1 draft; all feedback incorporated |

Table of Contents

[1. Introduction 1](#_Toc496257939)

[1.1 Purpose 2](#_Toc496257940)

[1.2 Audience 2](#_Toc496257941)

[1.3 Document Organization 2](#_Toc496257942)

[2. Process Model 3](#_Toc496257943)

[3. Object Model 6](#_Toc496257944)

[4. Resource Model 10](#_Toc496257945)

[5. API Specification 10](#_Toc496257946)

# Introduction

The Poplin project is a Medicaid reference architecture being developed, under the MITA Governance Board and sponsored by CMS, by key states, vendors, and other stakeholders. Poplin defines a standard, extensible set of business processes, object definitions, and application programming interfaces (APIs) for State Medicaid Agencies. The [Poplin GitHub site](https://github.com/MITA-Governance-Board/Poplin) provides further detail about the Poplin reference architecture and implementation. This paper defines a *Service Registration and Discovery* shared service for Poplin.

Service registration addresses the mechanism for services to register their location (host and port) so that they can be found by clients who want to use them. Service discovery addresses the mechanism that clients use to obtain service locations (via the service registry). The service registration and discovery approach is therefore a critical piece of any service-oriented or microservice architecture as it provides the “awareness” between service requesters (i.e., service consumers) and service providers.

Service registries are constantly evolving resources that catalog information about the services available for use in an enterprise. These registries constantly communicate information about resources (metadata) that the enterprise wants to make available for discovery. This metadata typically includes parameters defining the resource’s purpose, capabilities, and limitations. The metadata must be searchable by criteria-driven queries, and stored and maintained in a consistent format.

Service registries provide not only awareness of available resources and services, but also a standards-based infrastructure for consuming the requested services once discovered. Each service’s purpose and capabilities must be clearly expressed to be correctly identified and interpreted by both human and software consumers. Well designed and maintained Service Registries will help state Medicaid enterprises operate more effectively, by promoting reuse and avoiding re-development of existing capabilities.

Poplin defines this shared service in a way that does not dictate tools or technologies. Poplin does, however, provide a fully executable open source stack that SMAs can use in whole or in part to realize their service registration and discovery capability.

The Poplin reference open source stack leverages Kubernetes, a popular existing open source solution automating deployment, scaling and management of containerized applications, to implement service registration and discovery. Kubernetes was written by Google, and is currently used by small- as well as large-scale operations, has been integrated with several other microservice-based open source technologies such as OpenShift and Docker, and has been shown to be simple, secure, fast, and reliable. Kubernetes uses Swagger and OpenAPI technologies to provide a JSON-based RESTful API, and has a strong open source community with commercial support.

This document specifies the business processes achieved through the Service Registration and Discovery Service, the objects that are used, and defines the resources and APIs that can be used by client software.

## Purpose

The primary purpose of the Poplin Service Registration and Discovery shared service is to enable an executable, open source, standard mechanism for State Medicaid Agencies (SMA) to build and deploy discoverable services in their enterprise architecture. The shared service provides two key capabilities:

* Service registration addresses the mechanism for services to register their location (host and port) so that they can be found by clients who want to use them.
* Service discovery addresses the mechanism that clients use to obtain service locations (via the service registry).

## Audience

The entire Poplin reference architecture and implementation are open source and available to the public ([Poplin GitHub site](https://github.com/MITA-Governance-Board/Poplin)). The primary audience of the Poplin service definitions are State Medicaid Agencies (SMAs) and their vendors, specifically system architects, developers, and IT operations personnel supporting their state’s move to a modular, microservices-based enterprise architecture. The Poplin shared services, however, are not specific to the Medicaid domain and could be leveraged by any enterprise.

## Document Organization

This document is organized as follows:

Table 1. Document Organization

| Section | Purpose |
| --- | --- |
| Section 2: Process Model | Describes the dynamic aspects of the service area |
| Section 3: Object Model | Describes the logical aspects of the service area |
| Section 4: Resource Model | Describes the important elements or key abstractions for the service area |
| Section 5: API Specification | Describes how the API should look structurally for the service area |

# Process Model

The basic pattern for service registration and discovery involves a process where a service provider submits a service definition to a discoverable registry where service consumers can find and use services. This conceptual process is illustrated in **Figure 1** below.

****

Figure 1. Conceptual Process Model

Service registration and discovery implementations comprise a variety of capabilities. A basic set of these capabilities are highlighted in the use case model in **Figure 2**. The Client, or Service Consumer, discovers and uses services at run-time. However, in a microservices architecture, this service consumer is often another collaborating, orchestrated service.

Many of the service registration and discovery capabilities are utilized by the development and operations teams to register and deploy secure, containerized services across the infrastructure.

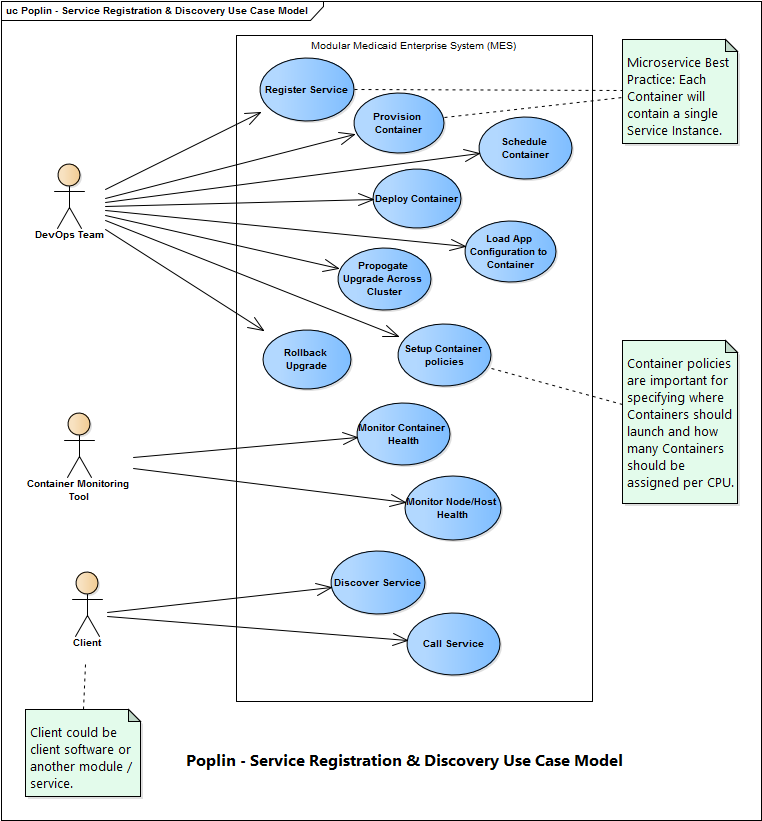
****

Figure 2. Use Case Model

The executable Poplin reference architecture will be initially deployed via OpenShift and kubernetes. For more information on the Poplin reference implementation please see the [Poplin GitHub site](https://github.com/MITA-Governance-Board/Poplin).

Many applications running for long periods of time eventually transition to broken states, and cannot recover except by restarting them. Kubernetes provides [*liveness probes*](https://kubernetes-v1-4.github.io/docs/user-guide/pod-states/#container-probes) to detect and remedy such situations. The following process diagram (**Figure 4**) illustrates the basic health check pattern for service containers and nodes.

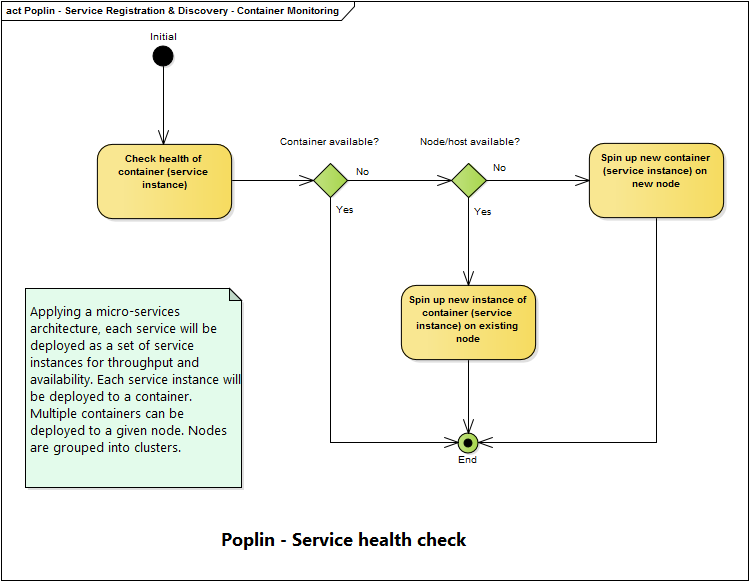


Figure 4. Service Health Check

# Object Model

Service registration and discovery facilitates the matchmaking process between Service Requester goals and Service Provider capabilities. The Poplin metamodel defines service consumers as *Performers*. The *Performer’s* needs should be independently formalized as *Goals* in accordance with their internal requirements, isolated from the peculiarities of the Service Provider infrastructure, data or behavior models. The conceptual object model in **Figure 5** below illustrates some of the key types in the service domain and their relationships, e.g., *Performers* (i.e., service consumers), their *Goals*, and the *Services* they consume.

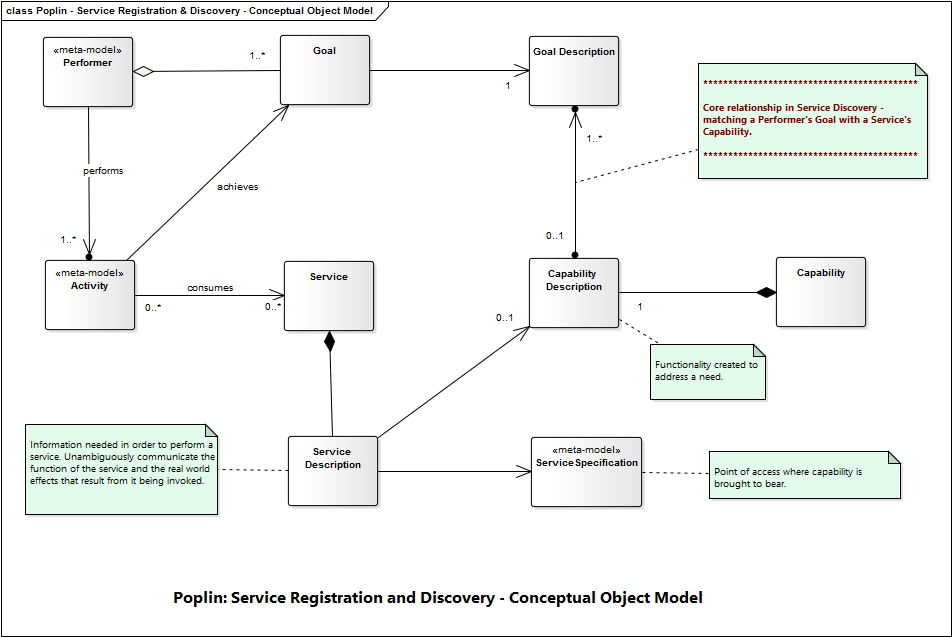


Figure 5. Conceptual Object Model

As an example implementation of this conceptual model, **Figure 6** illustrates a kubernetes microservice implementation.

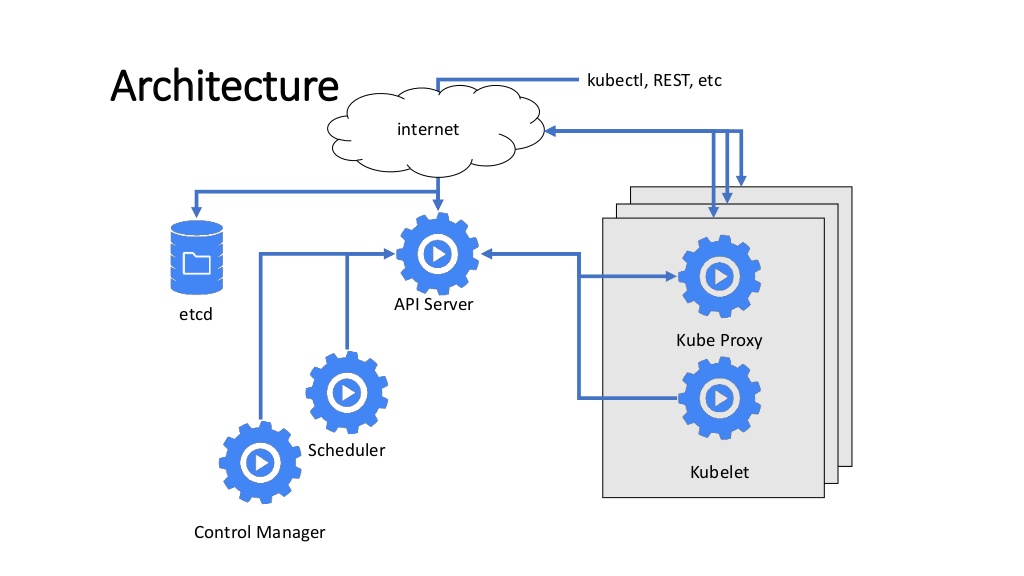


Figure 6. Example Architectural View - Kubernetes Microservices [[1]](#footnote-1)

Some best practices of any service registration and discovery architecture include:

* Services should be deployed using containers (e.g., Docker, Kubernetes Pods)
* Services should be atomic and not grouped with other services in a single container
* Multiple containers can be deployed per node
* Multiple nodes can be grouped into a cluster
* For service discovery, Domain Name Servers (DNS) are used to map RESTful service names to IP addresses

**Figure 7** below provides insight into how bindings are leveraged to connect clients to service endpoints. Bindings combine a concrete protocol with a data format specification (typically for a particular port type). Endpoints are URLs where RESTful services can be accessed by a client. There may be multiple endpoints required to cover different protocols.

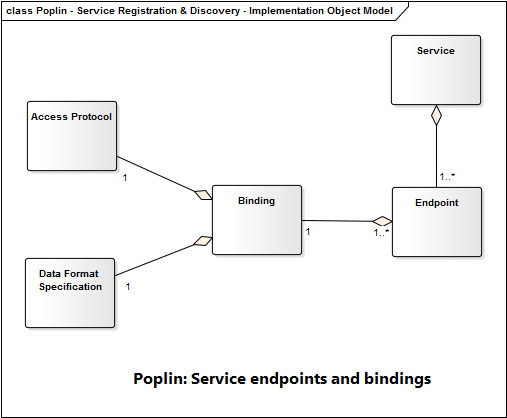


Figure 7. Service Endpoint Binding Object Model

Two common activities in any application architecture include deploying and rolling back application updates. This process can be complex in highly distributed microservice architectures. Kubernetes provides robust, automated capabilities to perform these updates and rollbacks, as depicted in **Figure 8** below.

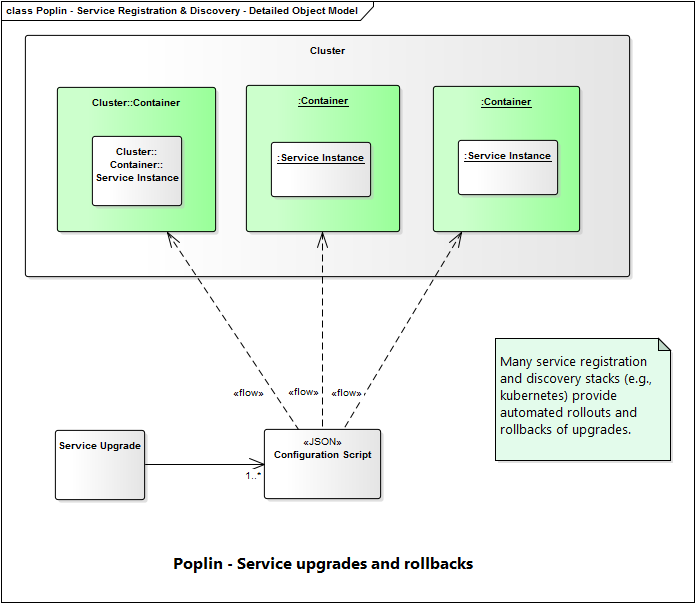


Figure 8. Service Container Upgrades and Rollbacks

# Resource Model

The [API specification for Kubernetes](https://kubernetes.io/docs/reference/) provides an excellent resource model for the implementation of service registration and discovery. Open the latest API specification to view the resource definitions.

# API Specification

The [API specification for Kubernetes](https://kubernetes.io/docs/reference/) provides a comprehensive example of the required interfaces and metadata for the service registration and discovery API. Open the latest API specification to view the API descriptions.

1. <https://www.slideshare.net/PlovDev/plovdev-2016-kubernetes> [↑](#footnote-ref-1)