Pilot Implementing Delivery Readiness

[Build & publish Pilot central assets](#_2g60olwe6fd0)

[Activities](#_vr1savmbi2a6)

[API Specification Template](#_vr1savmbi2a6)

[API Taxonomy](#_qdqkzmh8zudc)

[Architecture Review Report](#_wwtnoz85qtqe)

[Security Review for all of Pilot’s Deployments](#_sto0bfcuitnl)

[API Design Review Checklist](#_668vcl50zurl)

[API Implementation Review Checklist](#_1pts3iqfv1du)

[Code Review Checklist](#_rngt3kllwfpa)

[API URL Design and Naming Conventions](#_xr33xp56r1ig)

[API Versioning](#_hq0c6v52sah6)

[API Versioning Workflows](#_2d3jr3d86gzi)

[CloudHub Naming Strategy](#_8jguid9o58cu)

[Batch Application Template](#_vpskpdz349ut)

[Pub Sub Template using Anypoint MQ](#_48r87jv8n6ch)

[API Framework Template](#_kat8b75f7mgc)

[All In ONE Mule API Template !!](#_j6j6hxxdauvo)

[Central repository for sharing Assets](#_dgkmjxiytdei)

[Define the Pilot Delivery Engagement Model](#_lhnuygca9auc)

[Engagement Charter](#_gyumiutdrk7l)

[Operational Model Template](#_vncabr6bwrm3)

[Stakeholder Map](#_3gldhlvts8rx)

[Develop a List of Metrics and KPIs for Day 2 Ops Support](#_9bsmu7xn7frz)

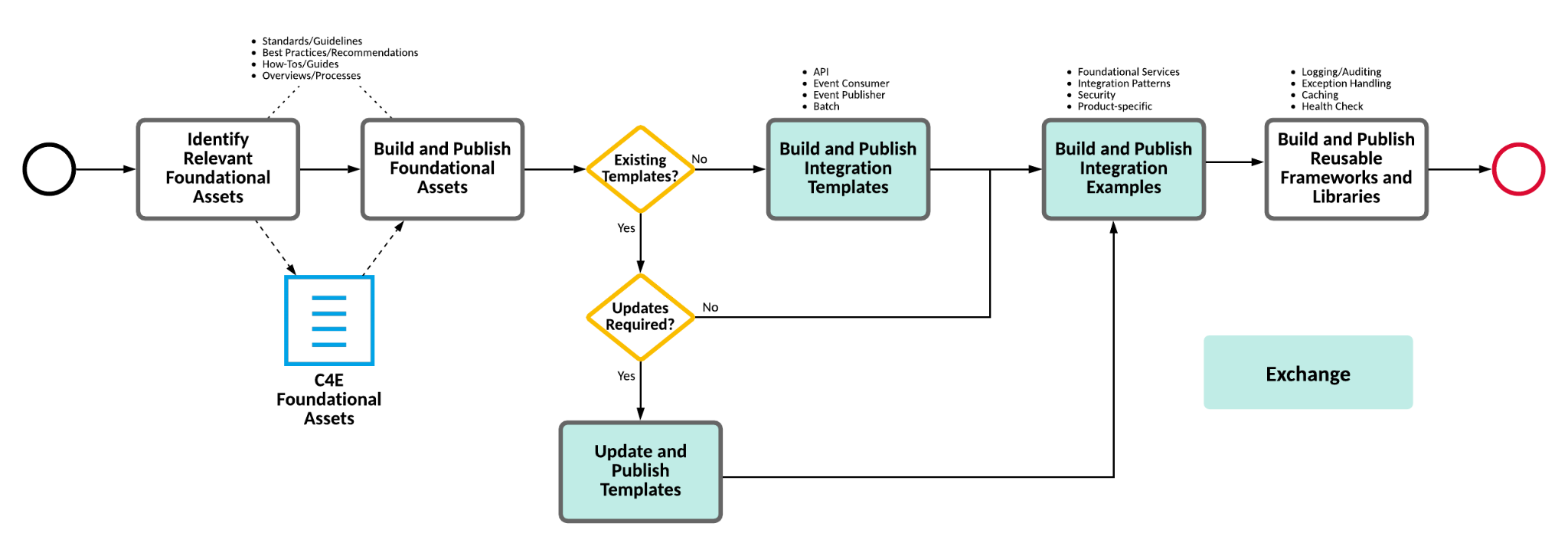
[Build an KPI System API](#_tly7514dekh1)

[KPIs Supported](#_djgg0zlb9313)

[C4E Metrics](#_ut1fwhjg394)

[Details](#_ou3984w2dr6r)

# Build & publish Pilot central assets



## **Activities**

* Identify Relevant Foundational Assets
* Build and Publish Foundational Assets
* Build/Update and Publish Templates
* Build and Publish Integration Examples
* Build and Publish Reusable Frameworks and Libraries

#### API Specification Template

An asset providing a basic RAML template for a REST API Specification. Intended for use with the Common RAML Fragments asset. This asset is intended to be used as a template only and should be reviewed and customized as required.

<https://github.com/mulesoft-catalyst/api-specification-template/tree/master>

#### API Taxonomy

* Report & Visualize exchange assets via taxonomy
* Manage exchange taxonomy via API and UI
* Gaps in Mule Anypoint API Metadata and how to manage them
* <demo> Exchange [GraphiQL Interface](https://anypoint.mulesoft.com/graph/api/v2/graphql) to design & build exchange reports
* <demo> Leveraging Postman to execute and report
* Publish a taxonomy Playbook

#### Architecture Review Report

A template to use as a report in an architecture review engagement. An architecture review should focus focuses on the following key areas of the MuleSoft Anypoint Platform:

* Performance
* Application Architecture
* Technical Architecture
* Code Review (if required as a separate deliverable)

Much of these details are presented in the recommendations doc – <https://docs.google.com/spreadsheets/d/1-jvdkQN1DlZ0PylopD5DzKg8Lha4ruMv20Vr2oU5EpU/edit#gid=0>

Template of the architecture review – <https://docs.google.com/document/d/1QQLQXDOHvKYv8n-jtsgQ-YX9pD8BjWdh/edit>

#### Security Review for all of Pilot’s Deployments

This asset provides a checklist for security review covering different areas like network, transport, API access etc.

<https://docs.google.com/spreadsheets/d/1H88HFGuoHm1ohuxMRes2zBf_gzQdkbiWPdk5mtCv2xg/edit#gid=549885634>

#### API Design Review Checklist

Checklist to assess if API design meets quality standards. This helps peers and architects to quickly do that to support the SDLC.

<https://docs.google.com/document/d/1vjfbjzgfjlbCaFXTnMEYnByuGR9lwThS6pCZ-T4Vk0w/edit>

#### API Implementation Review Checklist

<https://docs.google.com/document/d/12ebULTo5Qr71SUXE2YRK2pupol-IaupnCNwC5QqYOPo/edit>

Checklist to facilitate peer reviews on the implementation code. Assuming there is automated tooling available for syntax and such, this list helps development leads to quickly pass over implementation and assess whether it can be allowed to move to the next step in the SDLC.

#### Code Review Checklist

Checklist to be used as guidance for code review engagements. It covers all the aspects of the implementation of Mule applications.

<https://docs.google.com/spreadsheets/d/1TJCeU7hpU99aFsom8j9eJvMDzqESH9pK/edit#gid=251432689>

#### API URL Design and Naming Conventions

set of recommendations to start setting you APIs' naming conventions and design APIs' URLs.

<https://drive.google.com/drive/folders/1TrVQcIGU5ycHZTdnYabD8wn3m9Q8S3FU>

The structure of the URL is central to how APIs are organized and categorized within your enterprise domain. A strong URL taxonomy helps to categorize APIs across functional domains and regions while also providing insight into API relationships. A well-defined URI also helps to govern the lifecycle of your API through versioning practices.

#### API Versioning

An API should be designed for the long-term, but since change is inevitable, the API will never be completely stable. API change management can be addressed by a comprehensive versioning strategy. The strategy should include a periodic API deprecation and retirement schedule as well as up-to-date documentation describing the technical changes per version as well as the business requirement necessitating the change in a sufficient level of detail.

<https://docs.google.com/document/d/1cxB9Dt6un1VC95PE_3CXTQ2Hui_HbQS8/edit>

#### API Versioning Workflows

Not all changes require a new version. API versioning strategies can be categorized into two categories: backwards-compatible and non-backwards-compatible (or "breaking").Small changes that are backwards-compatible typically do not require a new version. Since a backwards-compatible change does not change the behavior of the existing API (even though a new resource has been added or the data types supplied by the API request/response have been expanded), the API can utilize the same major version as described. Once the initial major version has been deployed, newer ‘backwards-compatible’ deployments can use the same major version. Altering the API contract in this way requires changes to the following systems:

* Design Center
* Exchange
* Anypoint Studio
* API Manager

Changes that are not backwards-compatible usually require the consumer to alter the way their clients interact with the API prior to interaction. Changes that are not backwards-compatible (or "breaking") should be well-documented so that potential consumers fully understand the differences between the API major versions. Breaking changes can be captured as a major version change (e.g. v1 to v2). Altering the API contract in this way requires changes to the following systems:

* Design Center
* Exchange
* API Manager
* Anypoint Studio
* Runtime Manager

<https://docs.google.com/document/d/1XSvgB68OoVN8rTPk6CQINgrWUV96BFpY/edit#heading=h.gjdgxs>

#### CloudHub Naming Strategy

An asset detailing the naming conventions for Design Centre, API Manager, Mule project application naming, API URIs and CloudHub runtime manager

<https://docs.google.com/presentation/d/1RJibfUUGmZJKn2l4OVpbfy3k2-xF8Icp/edit#slide=id.p1>

#### 

#### Batch Application Template

Consider using this template as it contains all the necessary scaffolding code, which should help cut down significant development time by letting you just focus on the core batch processing steps.

Details

* Batch processing is handling and processing large amounts of data in batches or chunks
* Usually, jobs are run at regularly scheduled times (e.g., overnight) or on an as-needed basis
* Mule Batch Processing Application using Batch job scope
* Data stored in persistent queues for reliable processing and retries

Links

GitHub Repository: <https://github.com/mulesoft-catalyst/batch-template>

#### Pub Sub Template using Anypoint MQ

Consider using this template as it contains all the necessary scaffolding code, which should help cut-down significant development time by letting you just focus on the core batch processing steps.

Links

<https://github.com/mulesoft-catalyst/pubsub-messaging-amq-template>

#### API Framework Template

An API framework that incorporates basic error handling, logging, and properties configuration. The framework comes with a Mule application template that is ready to be used to start building RESTful and SOAP based APIs.

Links -

<https://github.com/mulesoft-consulting/mulesoft-template-api>

<https://github.com/mulesoft-consulting/design-api-template>

This repository contains all the template projects as a form of [git submodules](https://git-scm.com/book/en/v2/Git-Tools-Submodules) to facilitate the management of the entire set.

##### [All In ONE Mule API Template !!](https://github.com/mulesoft-consulting/all-in-one-mule-templates)

These are the related projects used:

* [common-parent-pom](https://github.com/mulesoft-consulting/common-parent-pom)
* [mule-application-template](https://github.com/mulesoft-consulting/mule-application-template)
* [common-traits-lib](https://github.com/mulesoft-consulting/common-traits-lib)
* [dw-library-log-mapper](https://github.com/mulesoft-consulting/dw-library-log-mapper)
* [health-check-app](https://github.com/mulesoft-consulting/health-check-app)
* [dw-library-error-mapper](https://github.com/mulesoft-consulting/dw-library-error-mapper)
* [mule-application-template](https://github.com/mulesoft-consulting/dw-library-error-mapper)

#### Central repository for sharing Assets

A C4E Central Repository typically consists of a variety of technologies including:

* a Content Management System for storing and reviewing C4E documentation, best practices, onboarding materials, and standards.
* a Source Code Management System for storing common code-based frameworks and libraries
* Anypoint Exchange for storing application templates, examples, connectors, and API specifications.

Constructing an asset taxonomy is strongly recommended to help users find the correct artifacts in the C4E central repository. Asset categories can be defined based on several different business and technical characteristics, including:

* Integration Type
* Business Function
* Business Process Supported
* System Supported
* Access (Internal/External)
* Medium (document vs. code vs. video)

Asset categories can also be combined to create a multi-dimensional view of each asset. Whichever approach is chosen, a consistent and extensive taxonomy dramatically improves the user’s ability to browse and discover assets.

Work so far – <https://docs.google.com/presentation/d/1TpxaoclQq4XMxxFgTD7m6RuzMRSWkGR7/edit#slide=id.p6>

# Define the Pilot Delivery Engagement Model

Many of the products and services provided by the C4E should realize the goals set forth by IT leadership. For example, to reduce operations costs, the C4E can provide routine application design reviews to ensure that APIs are being adopted and reused. To ensure higher quality applications, the C4E can conduct routine code reviews to identify errors and defects in applications prior to promotion into production.

In most cases, a C4E serves as an internal consulting group that provides services to a range of integration programs, projects, and initiatives. The C4E develops integration guidelines, standards, processes, and toolsets and promotes their usage across the IT organization. Ultimately, the C4E should be the heart of the integration practice for the client.

If a C4E Delivery Team is attached to the C4E, then it provides application delivery services such as API design, development, testing, and general maintenance and support.

# 

#### 

#### Engagement Charter

The Charter Document provides information and the scope about client’s C4E. It covers the following:

- C4E Mission Statement

- C4E Overview

- C4E Objectives

- Business Benefits

- C4E Services

- Operating Model

- C4E Roles

- Measuring Success

- MuleSoft Training and Certification Options

- Reference Materials

- Risks, Constraints, Dependencies

- Platform Stakeholders

<https://docs.google.com/document/d/1kKSqOKVSbo2M7nE5SHQsXJyB-XxPySOK/edit>

#### Operational Model Template

At a higher level, it captures the key areas we need to be focusing on with our clients and/or partners to define and agree on the model.

Use this template to open the communication around the model before production deployment happens. Also, reserve sufficient time to get the agreement on the model and for the documentation of the individual areas.

This template helps to lead discussions with clients and partners on what are the responsibilities of the different individuals within the team in DevOps terms (e.g. developers, architects, infrastructure, deployment, support, etc.). The template is just a starting point and should drive a detailed documentation of the different aspects of the model (e.g. CI/CD, permissions or user management, application support model, monitoring, etc.).

<https://docs.google.com/spreadsheets/d/1QkVDjPNex0FeSJafcyEVzyWcLINUDnMKVe2Vh0w7rd8/edit#gid=0>

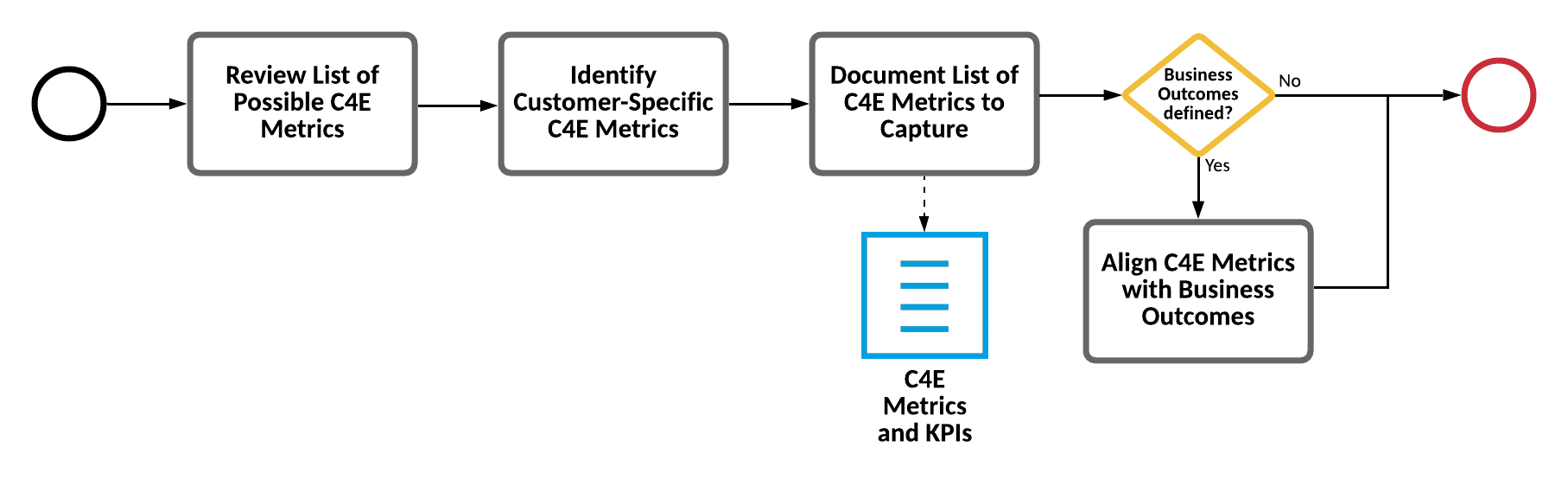
#### Stakeholder Map

Document the primary business units, specific projects, system integrators or partner resources, Center for Enablement (C4E) leads, key stakeholders, and a comprehensive list of the advisory and consultancy services provided.

Sample Stakeholder Map (FPL) - <https://docs.google.com/presentation/d/1p8z1KoshoHVMHmRxyYkTyQaTxnb6w9oIsa-gXxWhqXU/edit#slide=id.g1f5ac75716f_0_2278>

# Develop a List of Metrics and KPIs for Day 2 Ops Support

An essential step during the delivery readiness is identifying the metrics to measure, the cadence of measurement, and the method of reporting. For example, for each metric, create a measurement plan that defines the data used in defining the metric, how the data is utilized to calculate the metric, and how it should be displayed.



### Build an KPI System API

#### KPIs Supported

* Number of Active APIs
  + Provides the number of active APIs by type for the organization.
  + *CloudHub only*
  + API types are identified by naming suffix (*but is configurable by properties files*):
    - System API -> 'system-api'
    - Process API -> 'process-api'
    - Experience API -> 'experience-api'
* Number of Active APIs per Environment
  + Provides the number of active APIs by type for the environment.
  + *CloudHub only*
* API Reuse Percentage
  + Provides the percentage of active APIs that have more than one consumer per environment.
* Average Number of Consumers
  + Provides the average number of consumers of active APIs per environment.
* Top 10 - Number of Consumers
  + Provides the top 10 APIs ranked on total number of consumers per environment.
* Average Number of API Versions
  + Provides the number of API versions by environment.
* Top 10 - Number of API Versions
  + Provides the top 10 APIs ranked on total number of versions by environment.
* vCore Usage
  + Provides the number of vCores used across Production and Sandbox environments.
* Runtime Versions Used
  + Provides the number of applications using a specific Mule runtime per environment.
  + *CloudHub only*
* Number of Exchange Assets
  + Provides the the total number of Exchange assets by type for the organization.
* Overall Asset Satisfaction
  + Provides the ratings of all assets currently offered in Exchange by asset type for the organization.
* API Policies Used
  + Provides the policies applied to each individual API.

Versions

* Mule 3: <https://github.com/mulesoft-catalyst/anypoint-kpi-system-api>
* Mule 4: <https://github.com/mulesoft-catalyst/anypoint-kpi-system-api-mule4>

### C4E Metrics

C4E Metrics can be broken down into the following areas: Asset Reuse, Asset Availability & Satisfaction, Evangelism, Employee Onboarding and Governance.

#### Details

C4E Metrics can be broken down into the following areas:

* Asset Reuse
  + Helps to track shorter delivery lifecycles and the development of higher quality deliverables.
* Asset Availability & Satisfaction
  + Helps to track the quality and suitability of the integrations, APIs, and assets.
* Evangelism
  + Helps to ensure that best practices and standards are being followed.
* Employee Onboarding
  + Helps track employee productivity and can deliver high-quality applications.
* Governance
  + Helps ensure that best practices are being followed.

Template is at <https://docs.google.com/presentation/d/1Q1cNuZ2rH6oOoGdtlD8uRFCGxC_gRwCy/edit#slide=id.p32>