Dev Spec Template

The following document is the template dev teams should use for the design review process.

Author & Contributors:	**[name]**
Revision:	1.0 – Jan 1, 2021
Related documents:	
Approver:	

Contents

Introduction

[1 paragraph elevator pitch. The elevator pitch should set context.]

A typical Teams application requires access to user data to be useful. The Teams SDK provides access to a SSO token which has limited use across Microsoft services – such as Microsoft Graph. Our goal is to provide a simple way for developers to easily access Microsoft services like Microsoft Graph in their Teams application.

Problem Statement

[Identify the customer and their problem. Problem statements should be written in terms of their jobs to be done]

Pro developers building LOB apps and ISVs building LOB apps are our target customers. These developers are often familiar with app development and tend to focus on writing business logic. Understanding the nuances of Azure authentication and how to configure Azure services to work together can be a challenge, and ultimately detracts from their goal – to build LOB apps. Today these developers face several challenges when they want to build new or bring their existing LOB apps to Teams:

1. *It is hard to call Microsoft Graph*
2. *B*
3. *C*

Goal / Vision

[What is the outcome? Write this both in terms of jobs to be done and in terms of deliverables. Be as specific as possible.]

Our toolkit will provision and set up all desired Microsoft services for the developer, removing their need to manually configure services. We will make it easy for developers to access Microsoft services of their choice in their business logic with a single line of code. With this investment we believe we will significantly reduce the complexity for our developers to leverage Azure in their Teams apps.

We will do this by providing the following features:

1. *Feature A*
2. *Feature B*
3. *Feature C*

Definitions

[Any key terms that need to be defined?]

Term	Definition

Assumptions

[Are there any assumptions that you make in your design? These can be assumptions about anything:

1. Customer info, segment, data, etc
2. Scope
3. Technology]

We make the following assumptions:

1. *Our customers have a separate Azure tenant from their Office tenant*
2. *We will focus on single tenant applications*
3. *Developers are familiar with NodeJS*

Key measurements and success criteria

[List the key metrics to measure, and the success targets of these metrics. Not all metrics need to be associated with a success target, but all success targets should be metrics.]

Metric / Key result	Description	Success target
A		N/A
B		B < 10
C		C > 15

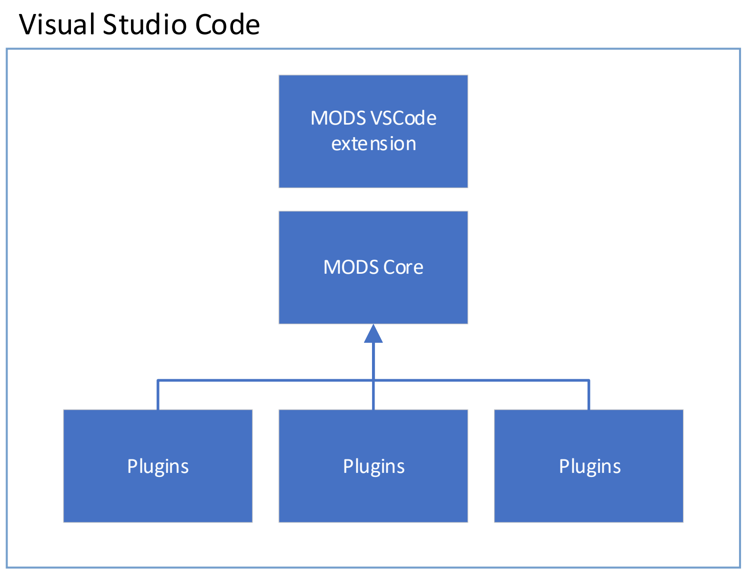
Design

High Level Architecture

[In this section, you describe the architecture in block diagrams. There are two considerations here:

1. Describe how the subject of the spec (a block) fits into its larger system (block diagram)
2. Describe how the subject of the spec (block diagram) breaks down into smaller components (blocks)

It is useful to provide both – one frames the subject of the spec against a bigger picture, and the other frames the subject of the spec in more detail.]



Contracts and boundaries

[In this section, you describe the contracts for the blocks. Like above, you should describe:

1. The contract and the boundaries defining the subject of this spec as it relates to the larger system

2. The contracts and boundaries of the components making up the subject of this spec

Contracts and boundaries should be clear and well defined. They can include but may not be limited to:

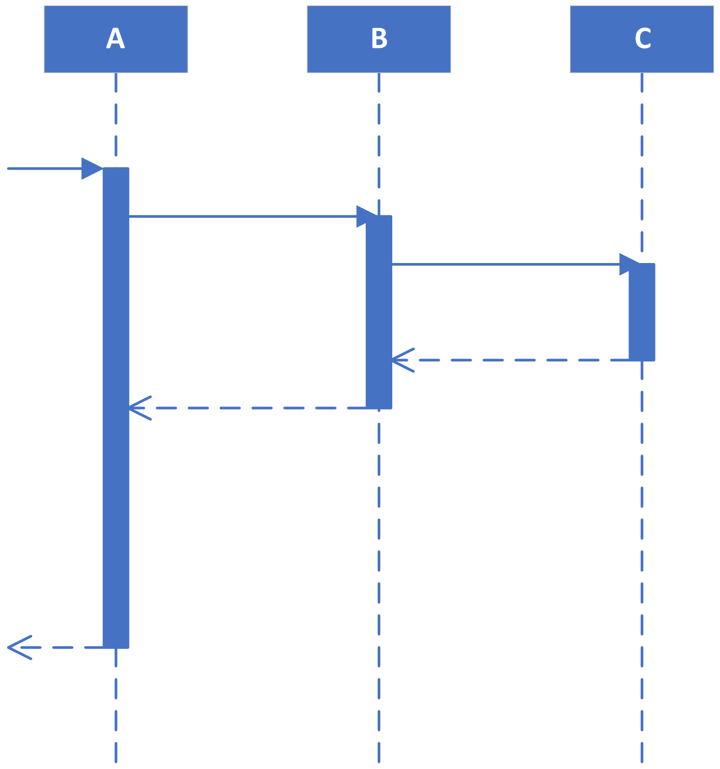
1. APIs and events
2. Data schemas
3. Files, blobs, and queues / streams]

Flows

[In this section, you describe how data and control flows.]

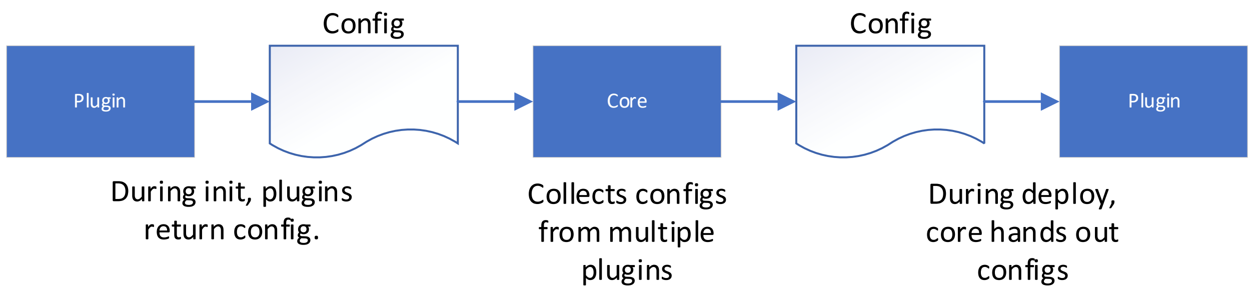
Control Flows

[Control flows are best illustrated with swim lane diagrams. These flows are meant to illustrate the "workflow" of control through your system. Control flows should include systems that call you and systems that you call]



Data Flows

[Data flows are meant to illustrate how data changes as it passes through the system]


Config
Config
Plugin
Core
Plugin
During init, plugins return config.
Collects configs from multiple plugins
During deploy, core hands out configs

Metrics

[In this section, you should document the metrics that your feature will emit, along with the criteria when they will be emitted. This should line up somewhat to your metrics and key results from the section above]

Testing

[In this section, you should describe your testing strategy. This should minimally include unit tests, but should also include integration / end-to-end / UI testing as necessary]

Work breakdown

[In this section, you should break down your work into a reasonable set of work items. Include links to ADO where appropriate. Add estimates where appropriate]

Other Considerations

Dependencies

[Is your feature dependent on another feature / design?]

Trade-offs

[Include any trade offs you considered]