Gem Session 22 August – Backwards compatibility

Jean-Michel Pittet, Costin Genescu, Vlad Petcu, Dominik Suess

Bē
Jon Noorlander

Agenda

Upgrade Gem sessions

Upgrade vision

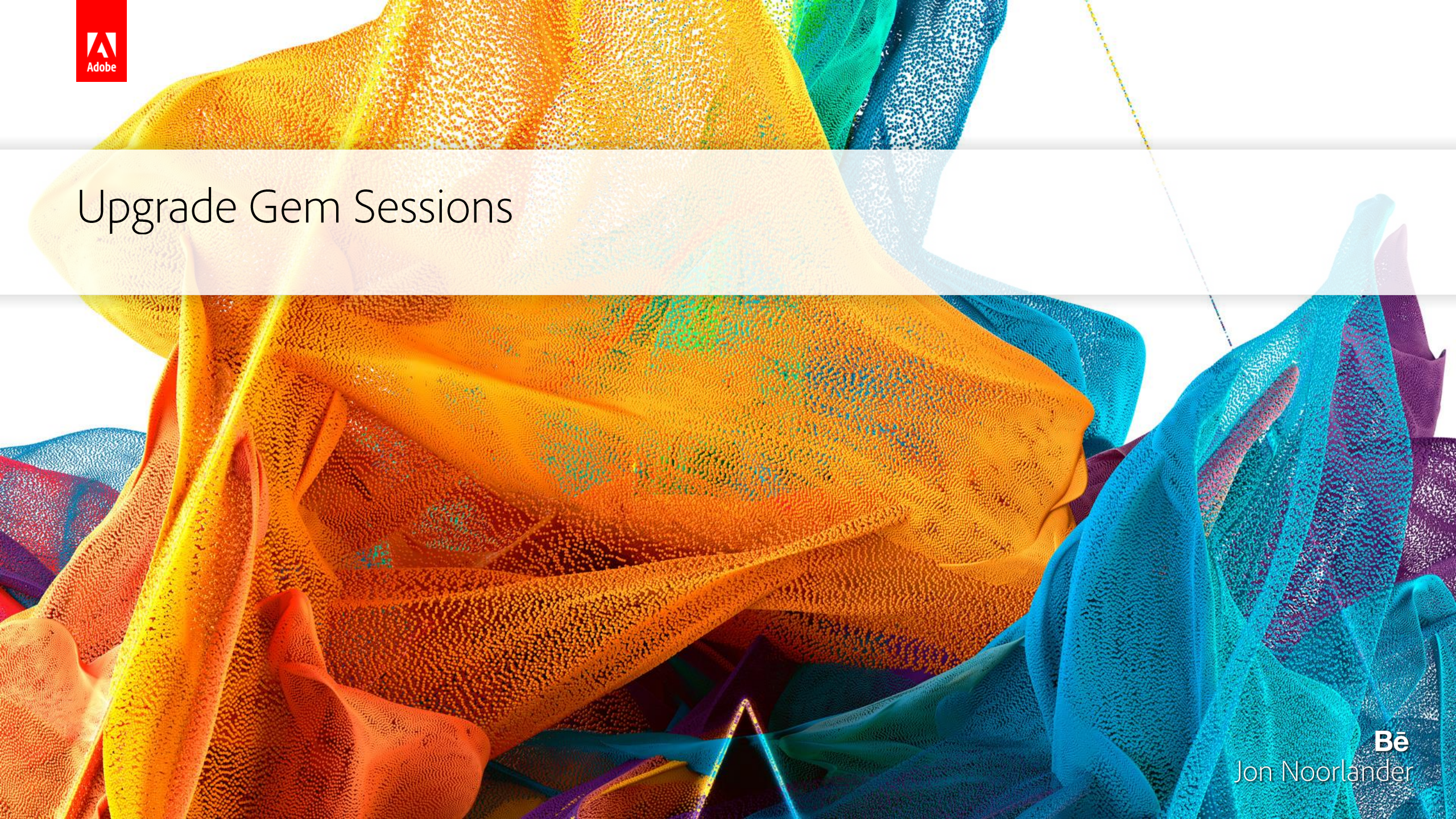
Compat package

Surface API

Q&A

Adobe

Upgrade Gem Sessions



Jon Noorlander

Upgrade Gem Sessions

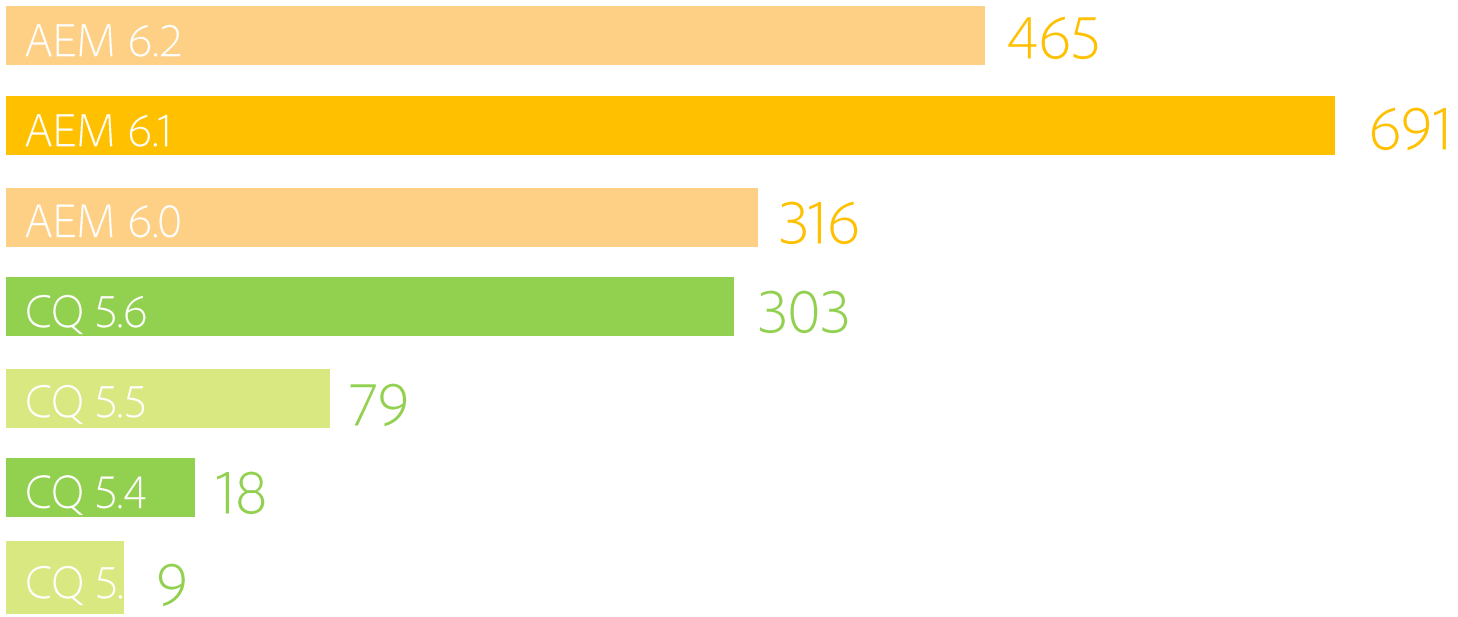
- 22 August – Backwards compatibility, Surface API – this one
- 29 August – Pattern detection, Testing
- 20 September – Mockingjay 6.4
- Sometime in October – Upgrade/Backwards compatibility

Upgrade vision

Jean-Michel Pittet

Jon Noorlander

Current Status



* Based on FY16 DayCare stats

AMS Customers (August 2017)	#
5.x	10
6.0	18
6.1	109
6.2	195
6.3	59

Source	Target	# Upgrades executed or planned
5.x	6.1	4
	6.2	4
	6.3	1
6.0	6.2	8/18
6.1	6.3	15/123

High cost of upgrade

- High development cost due to incompatibilities
- High load on CSEs due to back and forth support issues during the upgrade
- AMS cannot scale (195 upgrades from 6.2 to 6.4, 250+ from 6.3 to 6.5)
- AEM cannot compete at lower price points

No value by new AEM versions to existing customers

Upgrade vision

1

AEM 6.5 is available. Please upgrade now!

Cancel Upgrade

2

Updating

3

Welcome to AEM 6.5

Thank you for upgrading to the latest version!

Upgrade vision implications

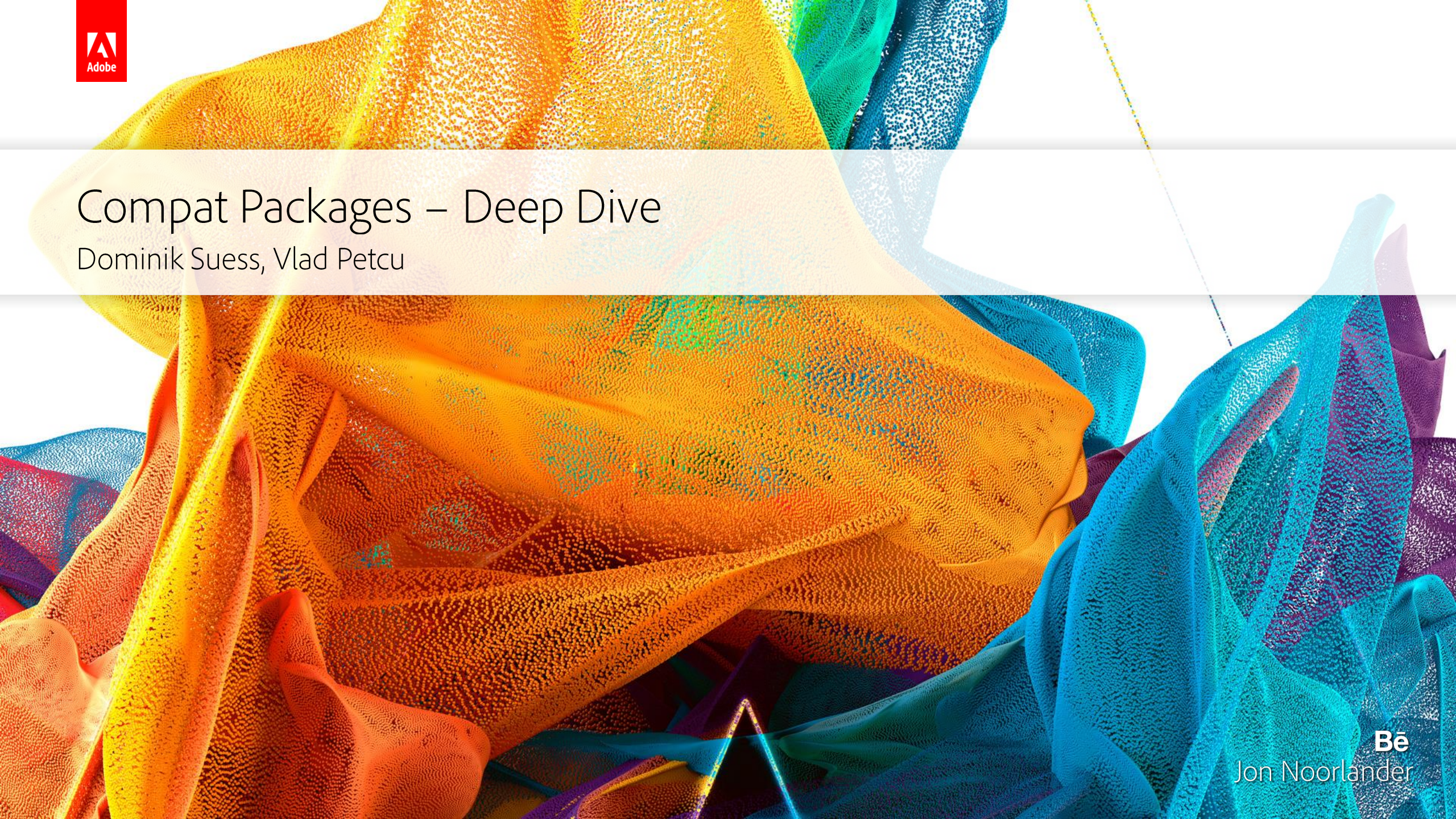
We cannot ask customers to change anything in their code during the upgrade.

JM Q&A

Q&A

Compat Packages – Deep Dive

Dominik Suess, Vlad Petcu



Backwards compatibility - CQ-4194070

A release is backwards compatible with the previous version if a customer does not have to change the code/customizations or recompile when doing the upgrade.

For 6.4, we will offer backwards compatibility with 6.3 release for both bundles and content via compat package.

Any exception will be personally approved by JM

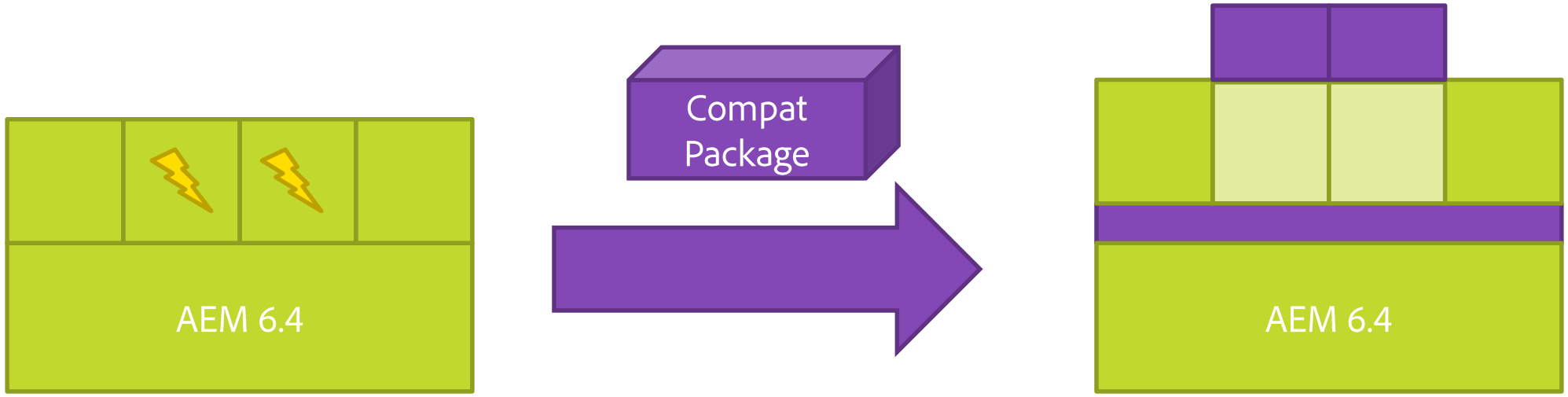
Compat packages

- There will be ONE compat package shipped
  - Compatibility ON/OFF
  - Engineered in granularity of git repositories but shipped as cumulated compat package.
  - PROS: reduced complexity – easy to test
  - CONS: masks 6.4 features
    - Teams to track the list of features that are impacted by the compatibility mode ON.
      - https://wiki.corp.adobe.com/display/WEM/6.4+Features+impacted+by+enabling+6.3+compatibility+mode

Compat Packages is a last resort solution to manage removal of code & content and prevent new customers to build on legacy.

Teams are expected to work on sustainable patterns preventing the need for breaking changes in the future.

Compat Packages



Plain AEM 6.4

AEM 6.4 +
Compat Package

Breaking Change – for new features

Removed AEM 6.3 API & Impl (Java + Content)

Compat packages

DEMO

Compat package

- RTC for compat package routing next week.
- We'll support backwards compatibility for upgrades to N+2 version
  - We'll release the 6.3 compat package with 6.4 release
- Testing will be covered in 29th August session

!!! AGAIN !!!

Compat Package is a last resort solution to gracefully handle removal!

Prevent the need for further replacements for future releases!

Zero touch upgrade – Surface API

Dominik Suess

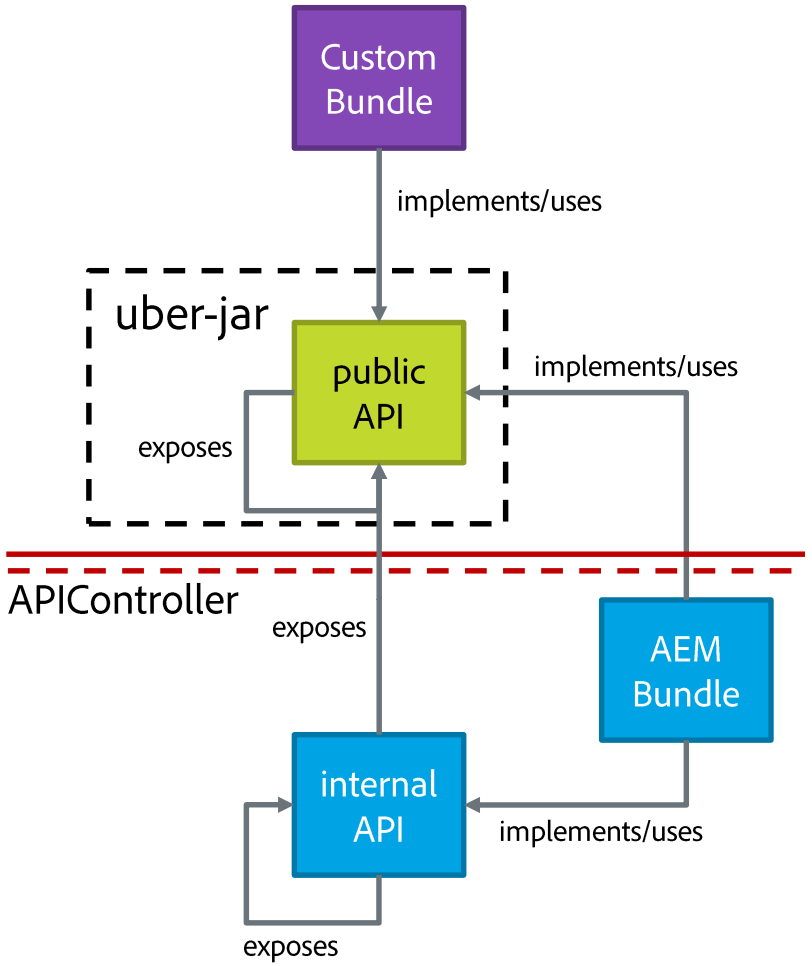
Surface API

- Mapping AEM
  - **Customer usage patterns**
  - **What to deprecate**
  - **How many customers are impacted**
- Extension points
- Extension model
- Anti patterns (Pattern Detector - next Gem Session)

GOAL FOR 6.4: REDUCE THE AEM API SURFACE

Governing Exposed Java API via Uber Jar

- Uber JAR reflecting full exposed API
  - Internal API needs to be excluded and protected via API Controller
  - Status Quo:
    - no package level protection via API Controller
  - Next Steps
    - Teams will get review tasks for java packages in bundles they own
  - What you should do:
    - Protect bundles that contain no public API via API Controller
    - Split public & internal API packages in distinct bundles and protect internal one
- Compat Uber JAR to expose additional API provided via compat package or hidden post GA

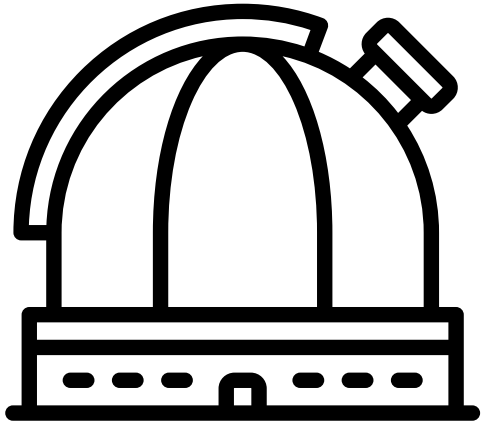


Governing Exposure of Application Content

- Classification of content under /libs by teams
  - Proposal about final & internal content classification and corresponding annotation practices during the next days
- Next Steps:
  - Distribution of content path lists to teams for classification
  - **Categorization of content-paths and add classification to content by teams**
  - Setup of iterative delta reviews & capturing of exposure metrics
  - Implementation of health checks for usage breaches in context of pattern detector – *no runtime enforcement*

public	may be overlayed and used by customers	**default**
final	no overlays supported - only usages • *substructures of a final marked node are internal by default*	*(no classification of substructures required)*
internal	**no usage direct or indirect** (superType) by customers *(e.g. code/content in apps)*	*(excluded for BC checks unless public in the past)*

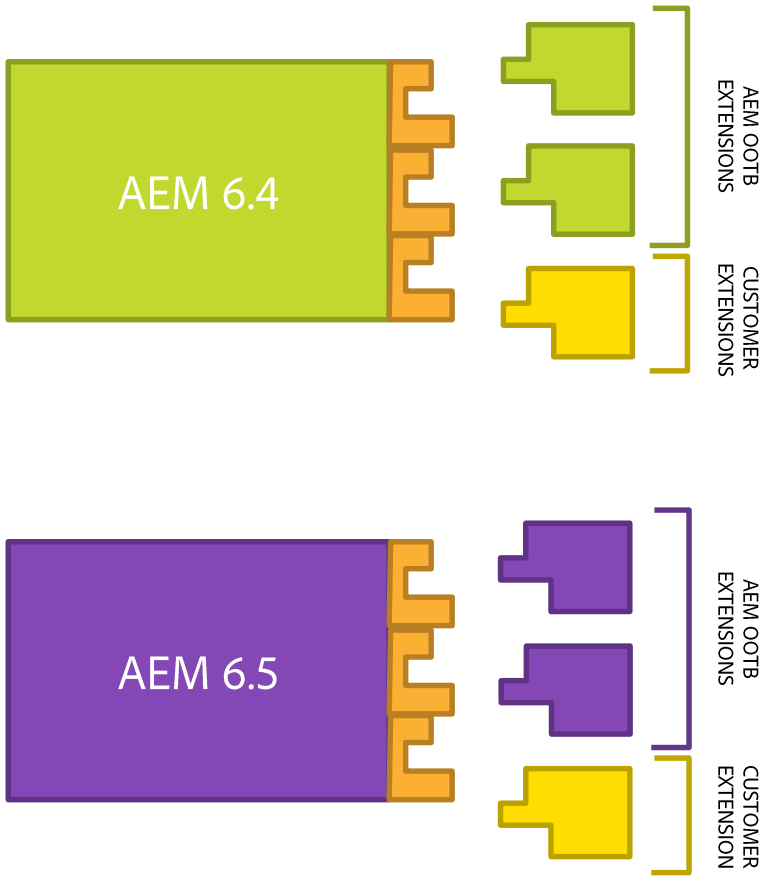
Policies in place



- ContentModelIT
  - Prevent unintentional breaking changes
  - Make exception handling and moves to compat packages traceable
- BundleVersion Reference Check
  - Prevent accidential introduction of major version bumps (e.g. 3rd party libs)
  - Make exception handling and moves to compat packages traceable

- Those policies and checks are generic and exceptions are expected:
  - **False negative: RTC to clarify why breach creates no compatiblity issue**
  - **Unavoidable breaking change: JM approval**

Sustainable Extension Architecture



- Plan extensibilty – don't let it happen
- Analyze MAEM data (for required extension points)
- Ask Consulting for field background
  - Slack: #ask-consulting
- Avoid structural lock-in
- Take care of testcoverage for extension points

- Participate in extension point practice experiments & discussions

Further Backward Compatiblity Concerns

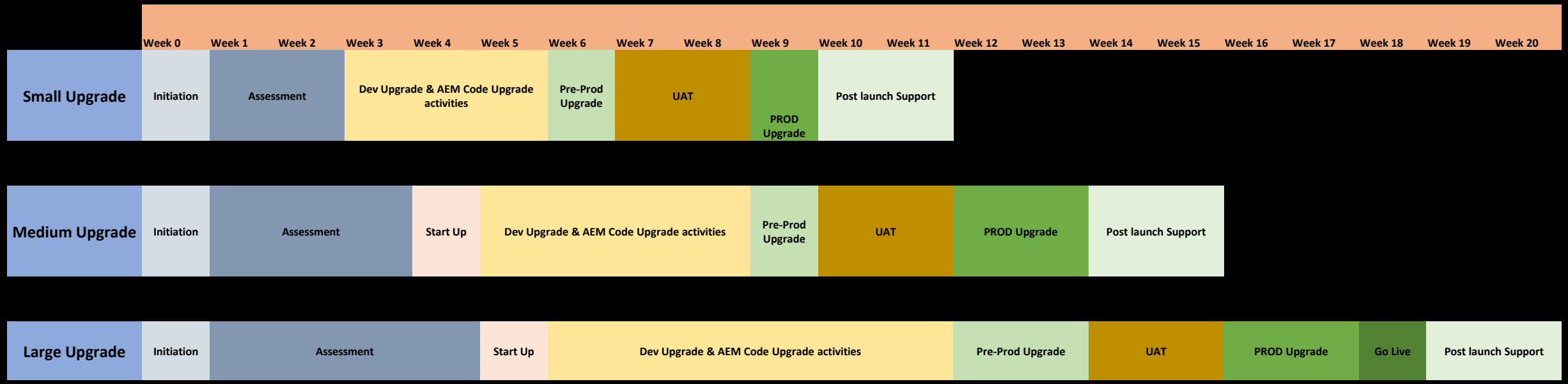
Clientlibs & Scripts	• Follow best practices promoted by Foundation UI team • Don't build on internals (same usage as by customers/partners) • Use existing building blocks as far as possible • Get in touch with Foundation UI team for generic extensions
HTTP API	• Treat inbound and outbound data as public API • Stick with public API to build consoles • Treat UI consoles as one of multiple interfaces for the HTTP API (headless first) • Consider HAF & follow current state of /api practices
Managed Content	*Proposal (pending/open discussion):* • *Annotate managed content as private (like for libs)* • *Hide private content from CRXDE and require "show system internals" action for inspection*

MAKE IT AN EXPERIENCE

Deliverables

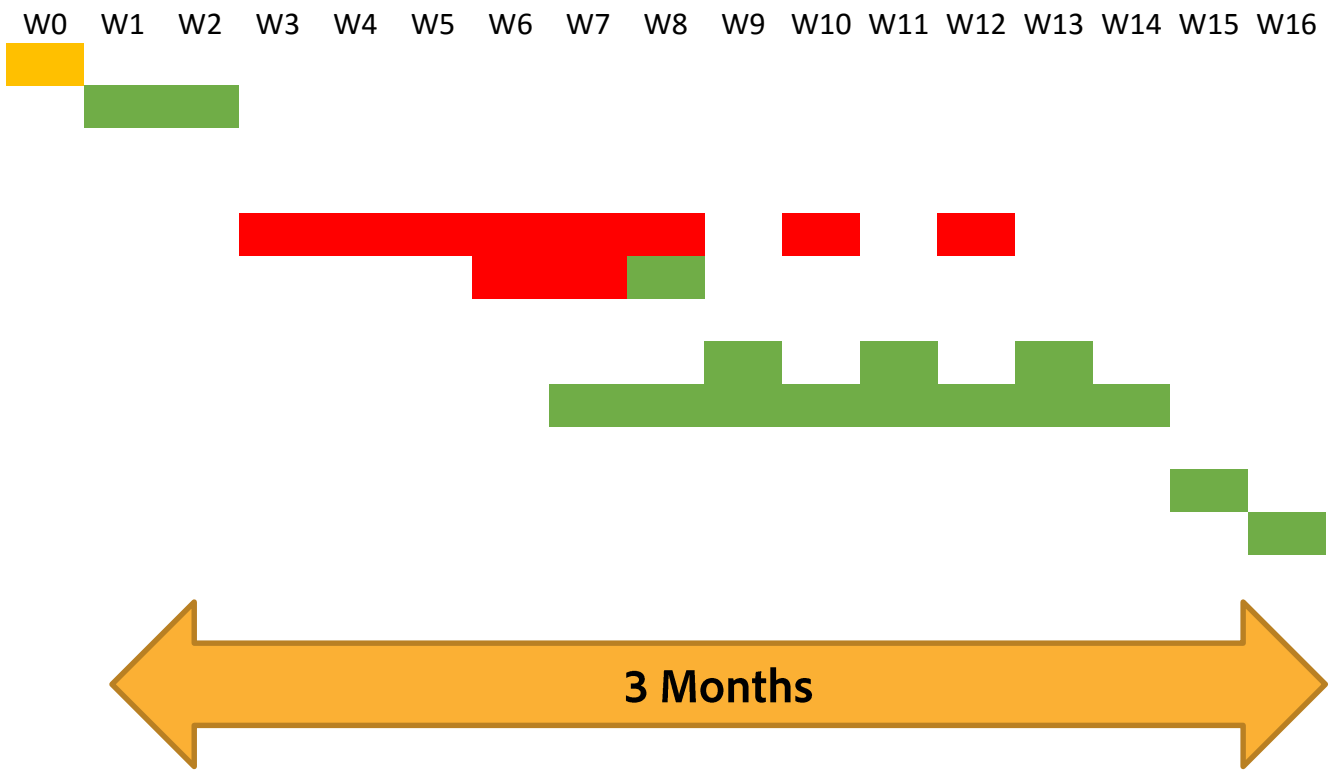
Jira	Deliverables	Teams work required
CQ-4194070 Framework for zero touch upgrade	**Documentation** for AEM surface API **Compat Packages** PoC and lifecycle **KPIs** for upgrade process Define a way to calculate the cost of maintaining BC	Surface API documentation Generate and test compat packages Handle exceptions to the backwards compatibility rules
CQ-4194071 Enforce backwards compatibility for AEM feature developers	Enable automatic validation of backward compatibility by targeting the API extension points as defined by CQ-4194070 Enable Engineers to detect breaking changes (as classified CQ-4194070) at build time Enable a framework for the E2E validation of the customer upgrade (3 clones provided by AMS)	Add tests for enforcing the surface API QA team must create the framework for testing compat packages
CQ-85533 Pattern-Detector for potentially breaking customisations	**Tool** that detects patterns that are known to break upgrades **The pattern library**	Contribute to the pattern library

Project Plan



Upgrade project flow

	CSE	Customer	Integrator
Recommendation for Upgrade (CSE)	x		
Setup of Project with Integrator		x	
Integrator to estimate adjustments			x
Offer for application adjustment			x
Adjusting Application	x		x
Upgrade Test Creation	x	x	x
Loadtests, Functional Tests, UAT, OAT etc.			
Upgrade Testruns	x		x
Upgrade Execution Plan	x	x	x
Downtimes, Communication Plan …			
Upgrade Execution	x	x	
GoLive	x	x	

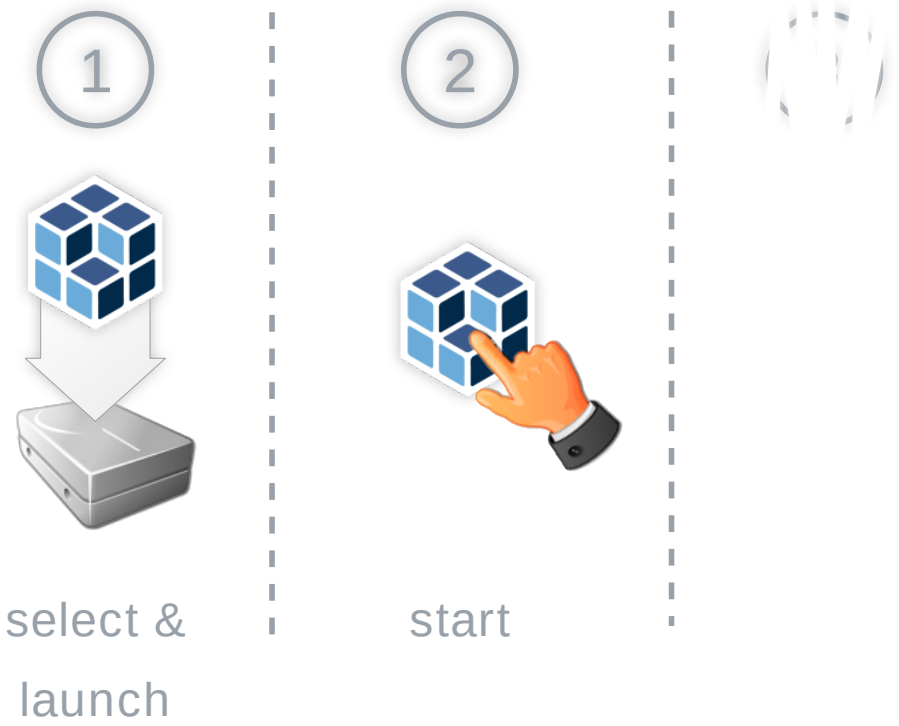


Avg Upgrade Cost

- 2-10 Dev
- **Dev + Functional Testing(no new features): ~30-50%**
- Time: 3-9 months
- **Avg cost: $ 500k**

Upgrade vision

Upgrade. Double-Click.



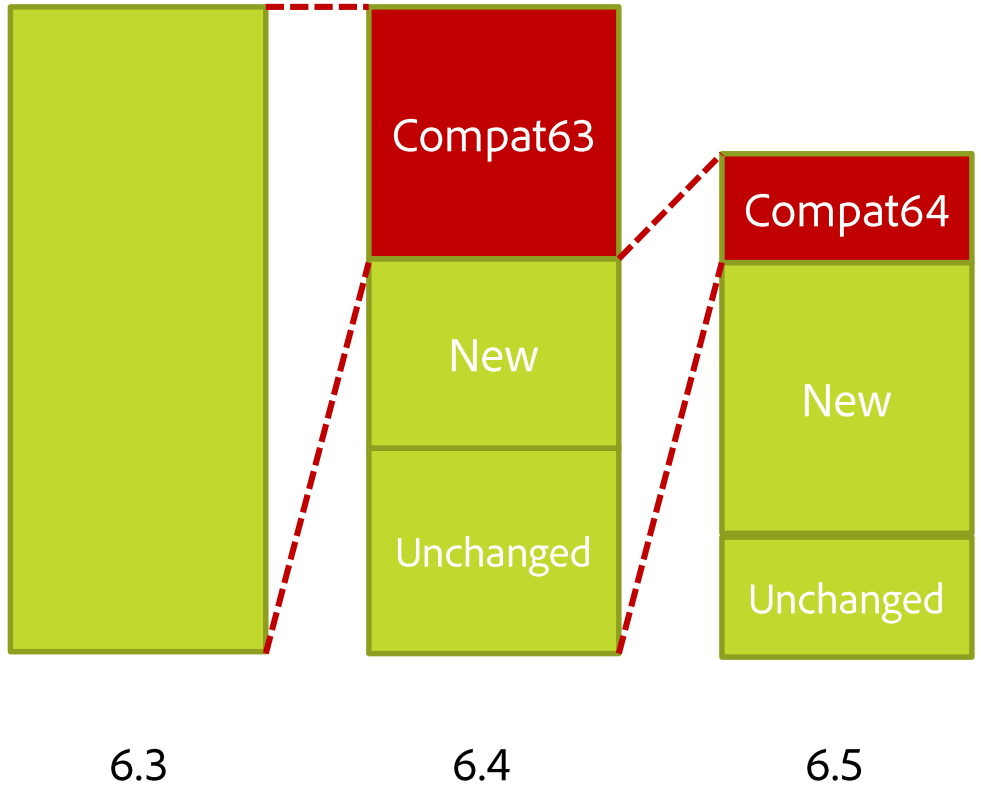
X

Adobe

Adobe

Scope for backwards compatibility in 6.4 vs. 6.5+

Public API Surface



Bigger scope in terms of compat packages coverage in 6.4 vs. 6.5
Wild card given to 6.3 customers
Higher net cost