



The Future of Insurance Starts Here

Policy Overview

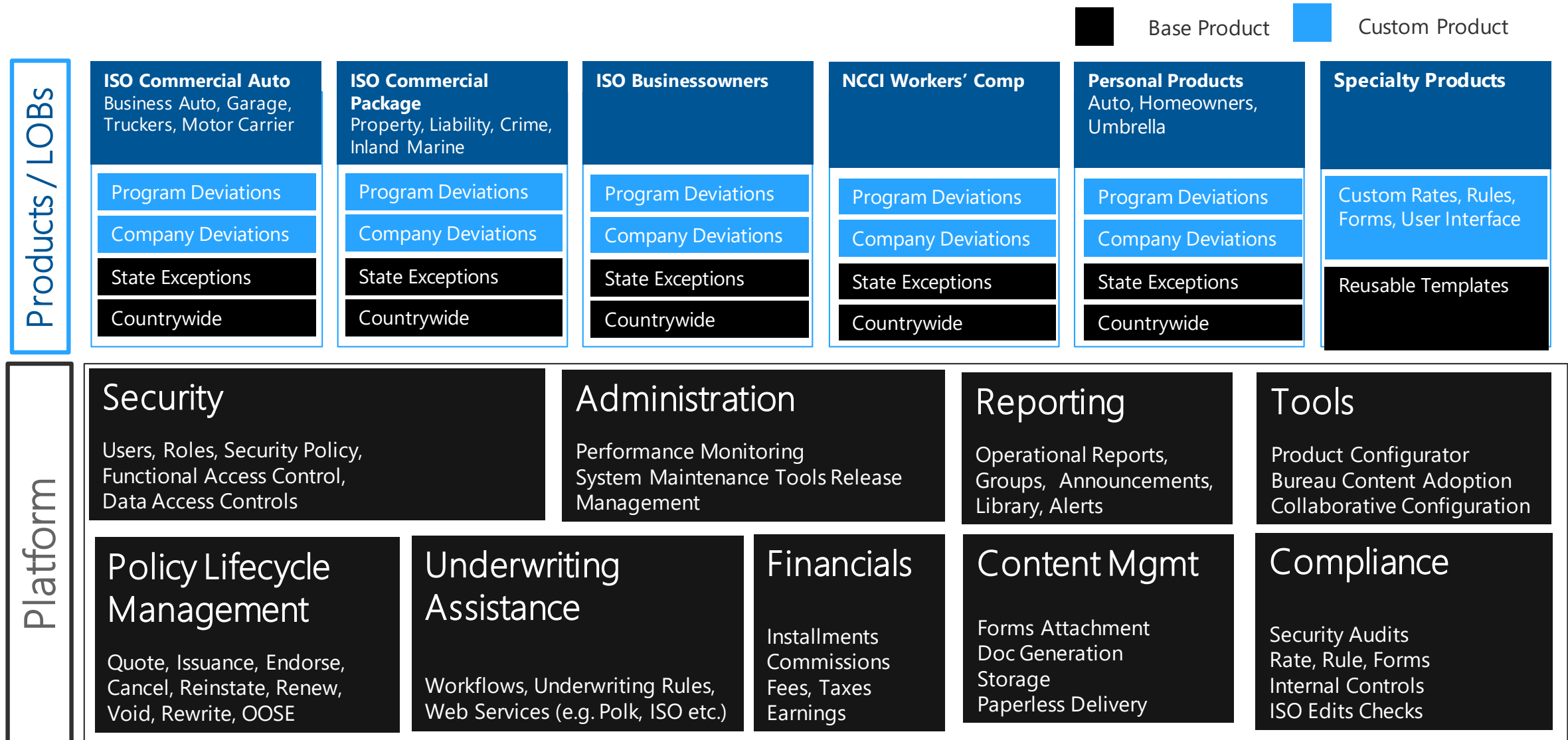


AGENDA

- Policy Admin Architecture
- Policy Framework Overview
- Lines of Business
- Deployment Architecture
- Metadata-driven Product Configuration
- Base & Custom Pack
- Content Inheritance Architecture
- Release Cycle Management
- Product Cache



Policy Admin Architecture



Framework Overview

- Core Services (Security, CMS, Dev Studio Runtime, Installer)
- Policy Admin
 - Core Transactions Processing (New Business, Issuance, Cancellations, Endorsements, OOSE etc.)
 - Product Metadata Execution (Rules, Rating & Dynamic UI Rendering)
- Other Frameworks
 - Tiles Framework
 - Dashboard, Folder, Reports Framework
 - Workflow Framework
 - Notes, Files & Email Framework
 - Document Generation Framework
 - Document Storage Framework
 - Entity Management (Producers, Programs, Market Segment etc.)
 - Interfaces, Web Services
 - Automated Transaction Processing with Batch Management
 - Performance Management (Instrumentation & Diagnosis)

Lines of Business

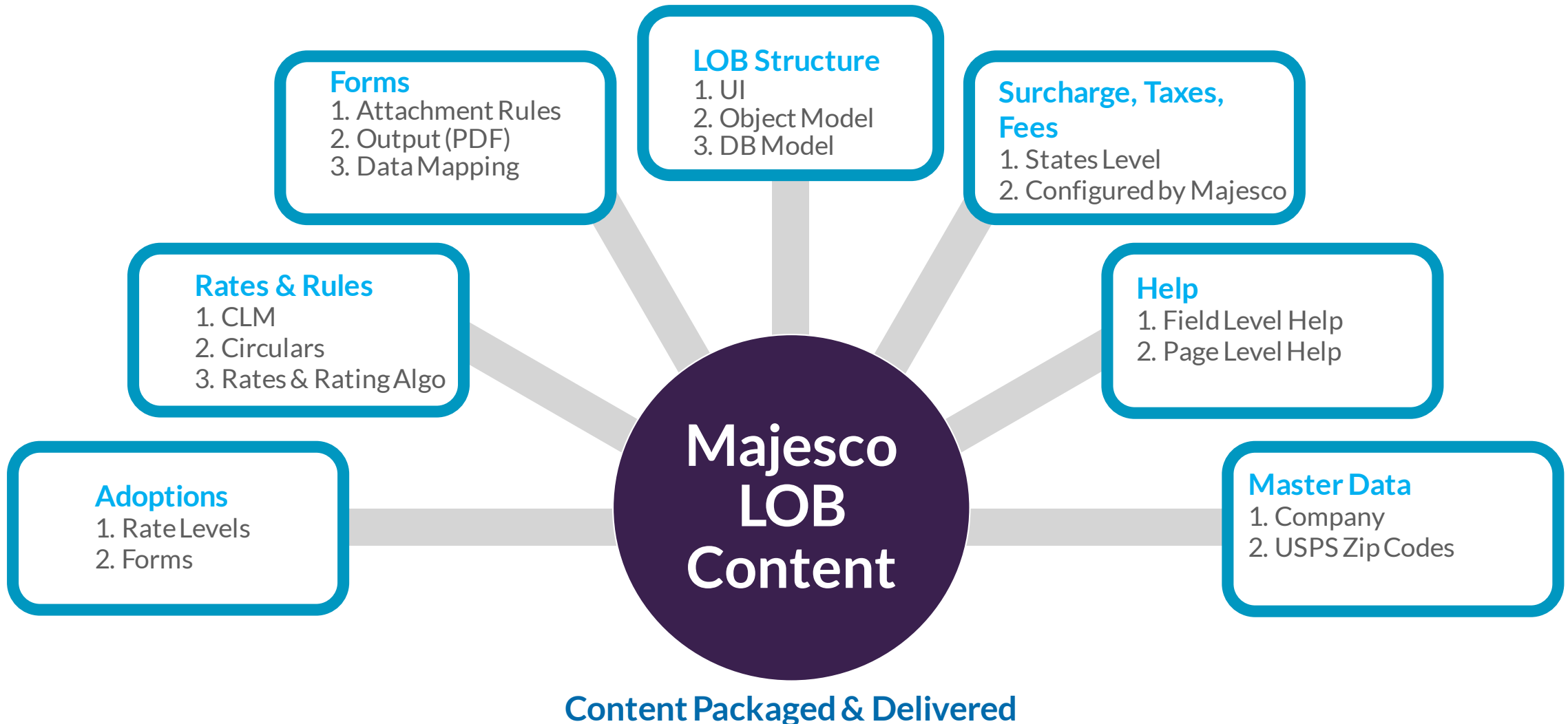
Bureau Lines of Business (ISO/NCCI)

| | | |
|-----------------|-------------------|--------------|
| Commercial Auto | General Liability | Property |
| Inland Marine | Crime | Workers Comp |
| BOP | | |

Non Bureau Lines of Business(Custom)

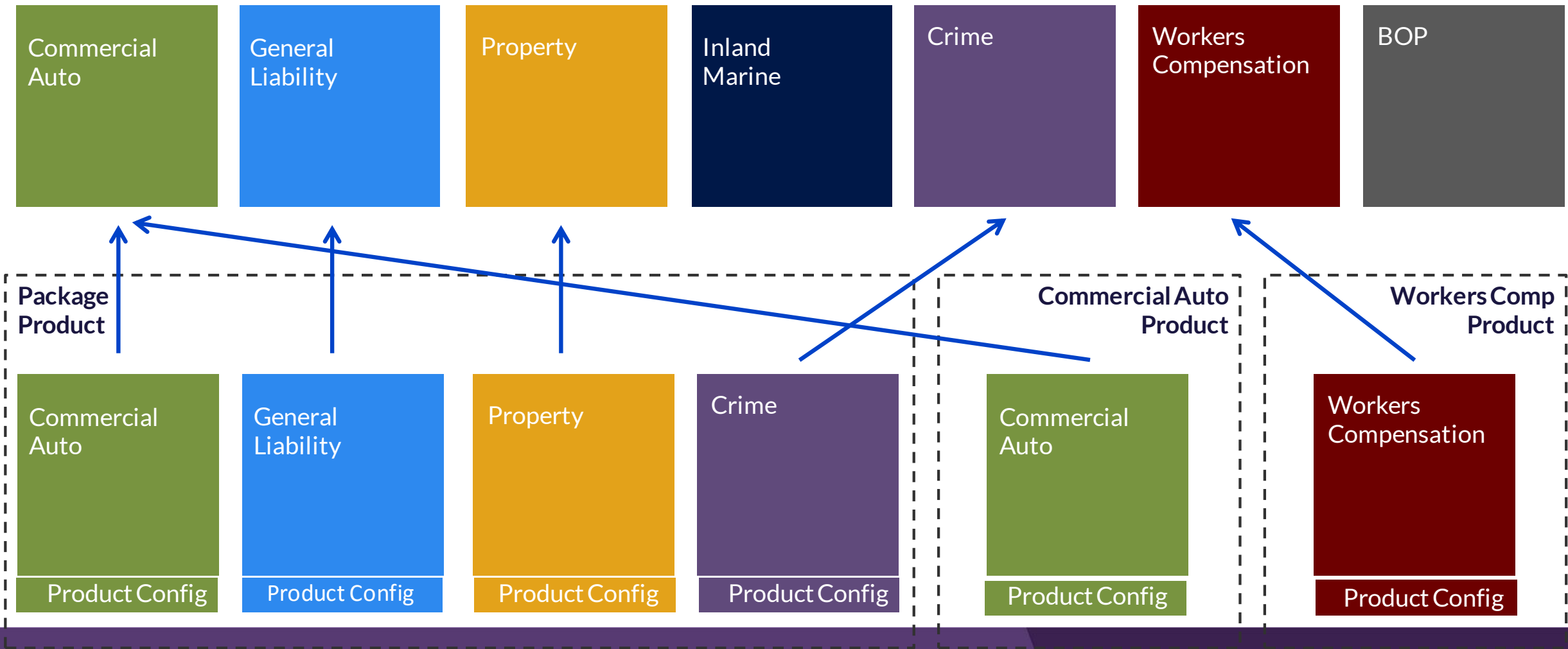
| | | | | | |
|-----------------------------|-------------------------------|-----------------|------------------|--|----------------------|
| Large Property E&S | Employment Practice Liability | E&O | D&O | Bond | Auto Physical Damage |
| Fiduciary Liability | Lawyers Liability | Kidnap & Ransom | Church Specialty | Umbrella Personal & Farm Commercial | Group Four Plan |
| Misc Professional Liability | Excess Liability | Cyber Liability | Hail | Office Liability UK Market | Defense Shield |

What makes a LOB?



LOB Products Thru LOB Packaging

Line of Business



Deployment Architecture

- **Source Environment(Dev Studio):**
 - This is Development environment.
 - This environment have Dev Studio which is used by developer to develop functionalities.
- **Target Environment(Policy V10):**
 - The development activity performed on Dev Studio are installed(deployed) on this environment.
 - The newly developed functionalities/ changes can be verified on this environment.
 - Policy V10 Application is available on this environment.
 - From Development and Testing perspective there can be 2 target environment.
 - For Development it is Integration environment whereas for testing it would be QA environment.
 - For more information please visit below link for reference

Policy : <https://confluence.majesco.com/display/MPCL/Majesco+Policy>

Common : <https://confluence.majesco.com/display/MPCL/Common+Functionality>

Metadata-driven Product Configuration

What is Metadata-driven Development?

- The platform uses a metadata-driven development model to help developers become more productive. It means that the most functionality are defined as metadata in a database rather than being hard-coded in a programming language.
- Metadata-driven development is exactly the same model for how Web browsers work. Instead of hard coding the definition of a Web page in a free-form programming language, a Web page author first defines the page as HTML, which is itself a kind of metadata.

Source: Salesforce.com for description of Metadata-driven Development

Example

- When a user requests a page, the Web browser renders the page using the metadata provided in the HTML tags. Even though the HTML/browser combination does not allow authors as much formatting power as they might get in a regular publishing tool, it simplifies the work of publishing content to a wide audience and increases the Web page author's overall productivity.
- Like Web pages that use JavaScript or Flash to add functionality to HTML pages, Policy 2015 platform also provides ways for more advanced developers to add custom functionality through PL/SQL.

Automatic Dependency Injection

- Typical application design a special and careful programming to trigger business rules depending on what and when data is getting changed which often leads to defects. Some system are also not able to track what data is changed and requires all business rules to be invoked which results into poor performance.
- Example: Value of field A is equal to sum of value of field B and field C and value of field A must be less than 100. There are two rules defined on field A – 1) $A = B + C$ 2) $A < 100$.
- System automatically registers dependency on field B and C for field A.
- When value of B or C changes, system automatically invalidates field A and invokes execution of rule (timing of invocation of rule will depend on intelligent algorithm to avoid early execution of rules for better user performance).

Object Model (aka Product Hierarchy)

How is Object Model, UI Model and DB Model are managed automatically?

- **What is Object?**

Object is the overall definition of the type of information stored together. For example, the address object allow you to store information regarding address.

- **What is Field?**

A part of an object that holds a specific piece of information, such as a city, state, zip in the address object.

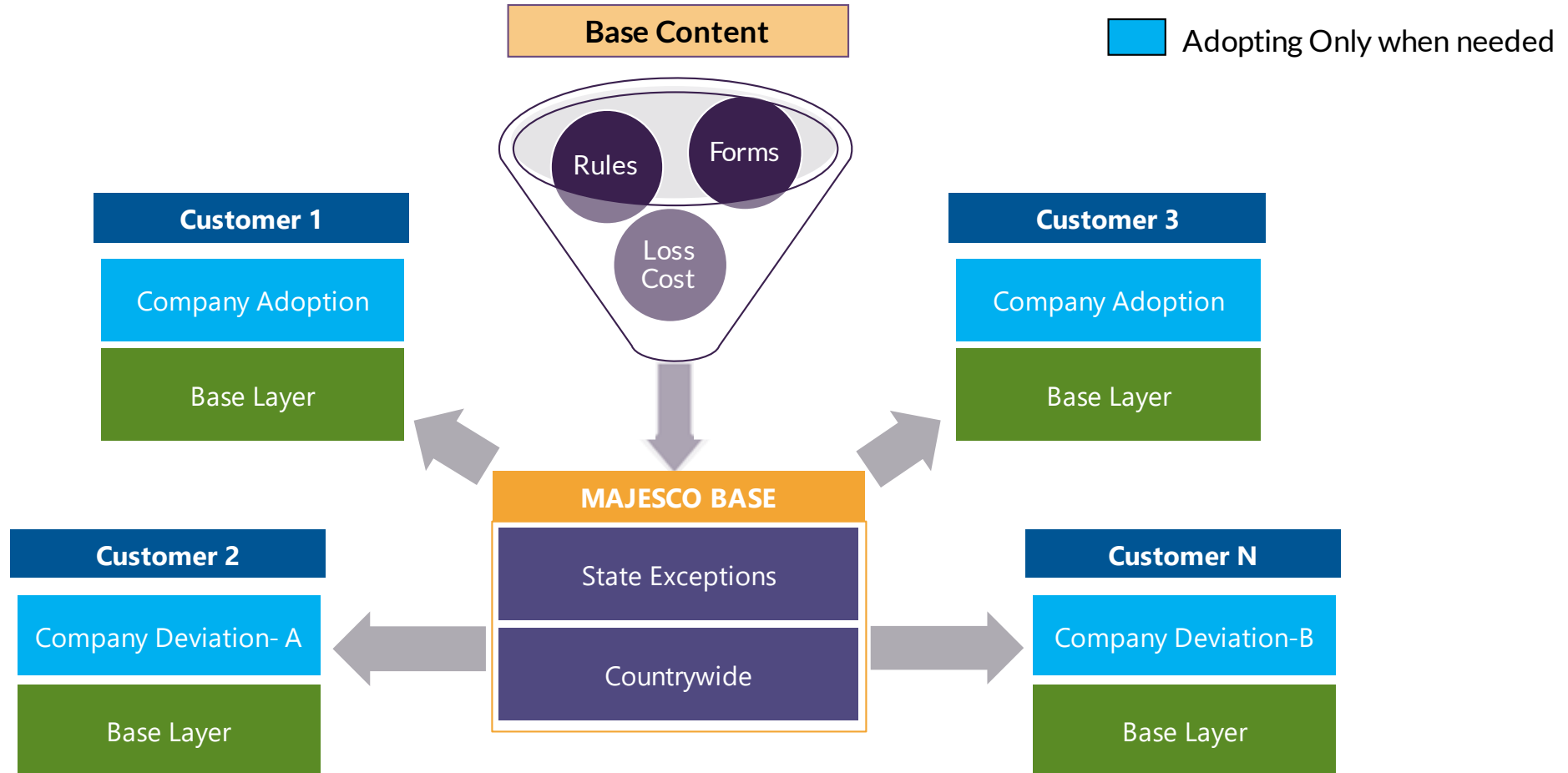
- **What is Object Model (aka Product Hierarchy)?**

Composition of multiple objects in hierarchical manner starting with a root-object.

Object Model (continued)

- **How does Policy V10 provides productivity through automatic synchronization of Object, UI & DB Models?**
 - Object and fields are defined in metadata
 - At install time, metadata management API generates a table for every new object, column for every new field and a foreign key for every hierarchical relation
 - At runtime, each object is rendered as either a screen or a section of a screen in form or tabular layout depending on UI property of the object
 - Benefits: No need to manage multiple models and mapping them. No code / instructions needed to save data to database since platform manages the ORM mapping
- **WYSIWYG way for defining Object Model (aka Product Hierarchy)**
 - Visual ways to define object model (aka product hierarchy) by defining and placing them on Screens in Dev Studio

LOB Base & Custom Layer



Base & Custom Pack

- Product has a base layer and custom layer. There is a separate environment on which Base team works and a separate environment on which the Custom team works.
- Custom LOB does not require Base Product (eg. Umbrella Policy) as it is not there in Base Product so development starts from scratch.

Versions for Base & Custom Pack

- **Base Pack**
 - The release version of base pack is having format as <YY>.<MM>.<Patch#>
 - Example : 19.1.0, 19.2.0, 19.2.1 etc.
 - Base release is for all customer.
- **Custom Pack**
 - The release version of custom pack is having format as <YYMM>.<Patch#>.<Day of the week>
 - Example : 1901.0.0, 1902.0.0, etc.
 - if there is a patch released on Tuesday then the release # will be 1902.1.2
 - Custom pack is only for specific customer.

Steps to Create Base & Custom Pack

Steps to Create Base & Custom Pack

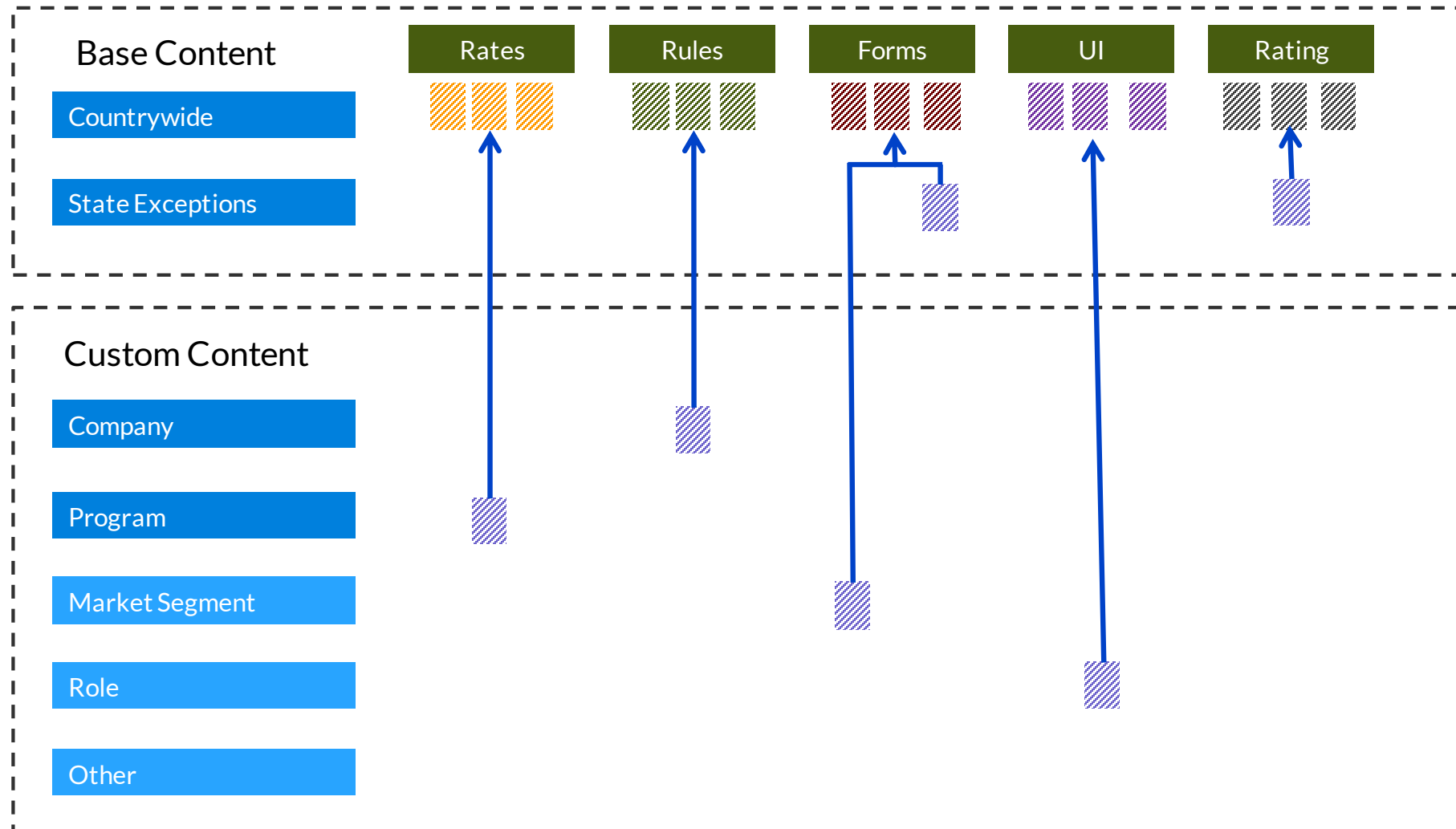
- **Base Pack**

- <https://confluence.majesco.com/display/MPCL/Base+Packs+Plugin>

- **Custom Pack**

- <https://confluence.majesco.com/display/MPCL/Custom+Packs+Plugin>

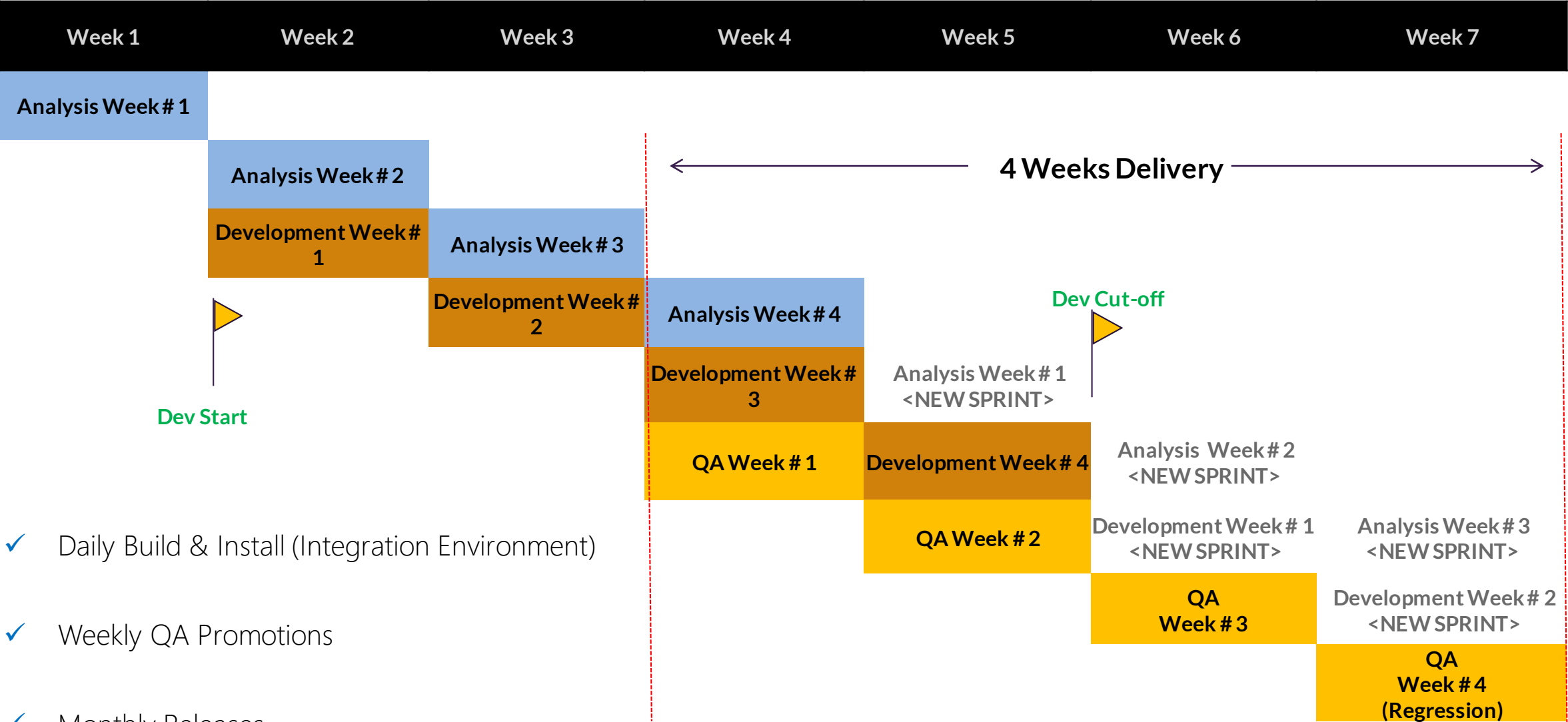
Content Inheritance Architecture



Content Inheritance Architecture



- Base Content is only driven on state & countrywide attributes.
- Only rates , rules ,UI ,forms , rating can be inherited from base content.
- If we want to apply condition based on different attributes such as company , program , role, market-segment etc we can inherit base content and provide customization.

Release Cycle Management



- ✓ Daily Build & Install (Integration Environment)
- ✓ Weekly QA Promotions
- ✓ Monthly Releases

Release Cycle Management

| | Base Pack  | Custom Pack  |
|----------------------------|--|---|
| Content | Base Content | Custom Content |
| Release Schedule | Monthly | As Needed |
| Needed for Bureau Products | Yes | Yes |
| Needed for Custom Products | No | Yes |
| Managed By | Majesco | Majesco Client Services AND/OR Customer AND/OR Third Party |

WEEKS

Development Week

JAN-2017

| SUN | MON | TUE | WED | THU | FRI | SAT |
|-----|-----|-----|-----|-----|-----|-----|
| 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| 29 | 30 | 31 | 1 | 2 | 3 | 4 |

Week 1

Week 2

Week 3

WEEKS

Development Week

FEB- 2017

| | SUN | MON | TUE | WED | THU | FRI | SAT |
|--------|-----|-----|-----|-----|-----|-----|-----|
| Week 3 | 29 | 30 | 31 | 1 | 2 | 3 | 4 |
| Week 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Week 1 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| | 26 | 27 | 28 | 1 | 2 | 3 | 4 |
| | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

WEEKS

QA Week

JAN-2017

| | SUN | MON | TUE | WED | THU | FRI | SAT |
|--------|-----|-----|-----|-----|-----|-----|-----|
| Week 1 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| Week 2 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Week 3 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| Week 4 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
| Week 5 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| | 29 | 30 | 31 | 1 | 2 | 3 | 4 |

WEEKS

QA Week

FEB- 2017

| | SUN | MON | TUE | WED | THU | FRI | SAT |
|--------|-----|-----|-----|-----|-----|-----|-----|
| Week 1 | 29 | 30 | 31 | 1 | 2 | 3 | 4 |
| Week 2 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Week 3 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| Week 4 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| | 26 | 27 | 28 | 1 | 2 | 3 | 4 |
| | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

What is Product Cache??

- Each product has a lot of rules (configurations) which change the runtime behaviour (i.e. how the UI should be displayed, when the rules are executed).
- At start-up, the system reads these configurations, arranges them in system use-able format (objects) and stores this information in memory and disc.
- Once the cache is generated, the product can be used to create Quotes/Policies. If the cache is not generated for a product or is not present in memory, then on first usage of that product, it is created (if not present on disc) and loaded into memory.

What is Product Cache??

- The cache contains:
 - Hierarchy of objects
 - Dependency information of each field (which field depends on which field)
 - Rules for each field
- Please refer below link:
<https://confluence.majesco.com/display/MPCL/Product+Cache>



THANK YOU!

