# A Framework for Obligation Fulfillment in REST Services

WS-REST 2011

John Field
Senior Technologist
EMC Office of the CTO

Joint work with Steven Graham and Tom Maguire

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Agenda

- Introduction
  - Motivating Use Case
- Modeling Obligations
  - Definitions, Architectural Roles
- Extending Spring Security
  - Design and Implementation
- Obligation Design Patterns
  - Metadata, Compatibility with REST Constraints
- Obligation-enhanced Operations
  - Development, Test, Deployment.
- Conclusion
  - Q&A, and Open Discussion

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# What is an Obligation?

- An Obligation is an expression of non-functional or cross-cutting requirements
  - The scope of which transcends any specific service…
  - …But for which the service bears an enforcement responsibility.
- Example: The regulatory requirements imposed upon processing of Electronic Health Records (EHR).

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# A Motivating Use Case: E-Health Records

- Example:
  - *“A patient’s records may be released to a 3rd party, but only on a secure channel, and in a redacted form, and only with an appropriate notification to the parent or guardian within 24 hours”.*
- The implied requirements go beyond a simple “Permit” or “Deny” authorization decision, and include:
  - Privacy/Redaction
  - Records Retention
  - Patient and/or guardian notifications
  - Complex conditional authorization rules
  - Etc.

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# A Motivating Use Case: E-Health Records

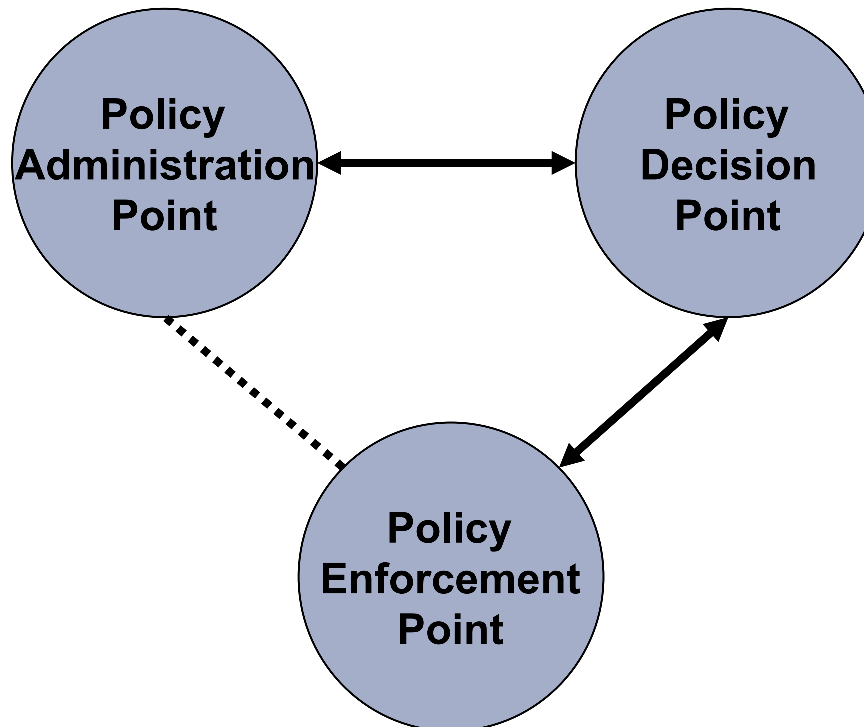
- These types of requirements are also subject to change, and may not even be known until deployment time.
  - Making design, development, and product roadmap planning even more difficult.
- Developers need a way to satisfy these requirements, while ***avoiding tangling*** the non-functional and cross-cutting GRC code into their core service logic.
  - Question: Can we help by creating an appropriate developer framework?
- Finally, note that use cases like this create challenges for many different stakeholders, not just developers:
  - Product Managers
  - Security Administrators
  - Auditors

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Modeling Obligations

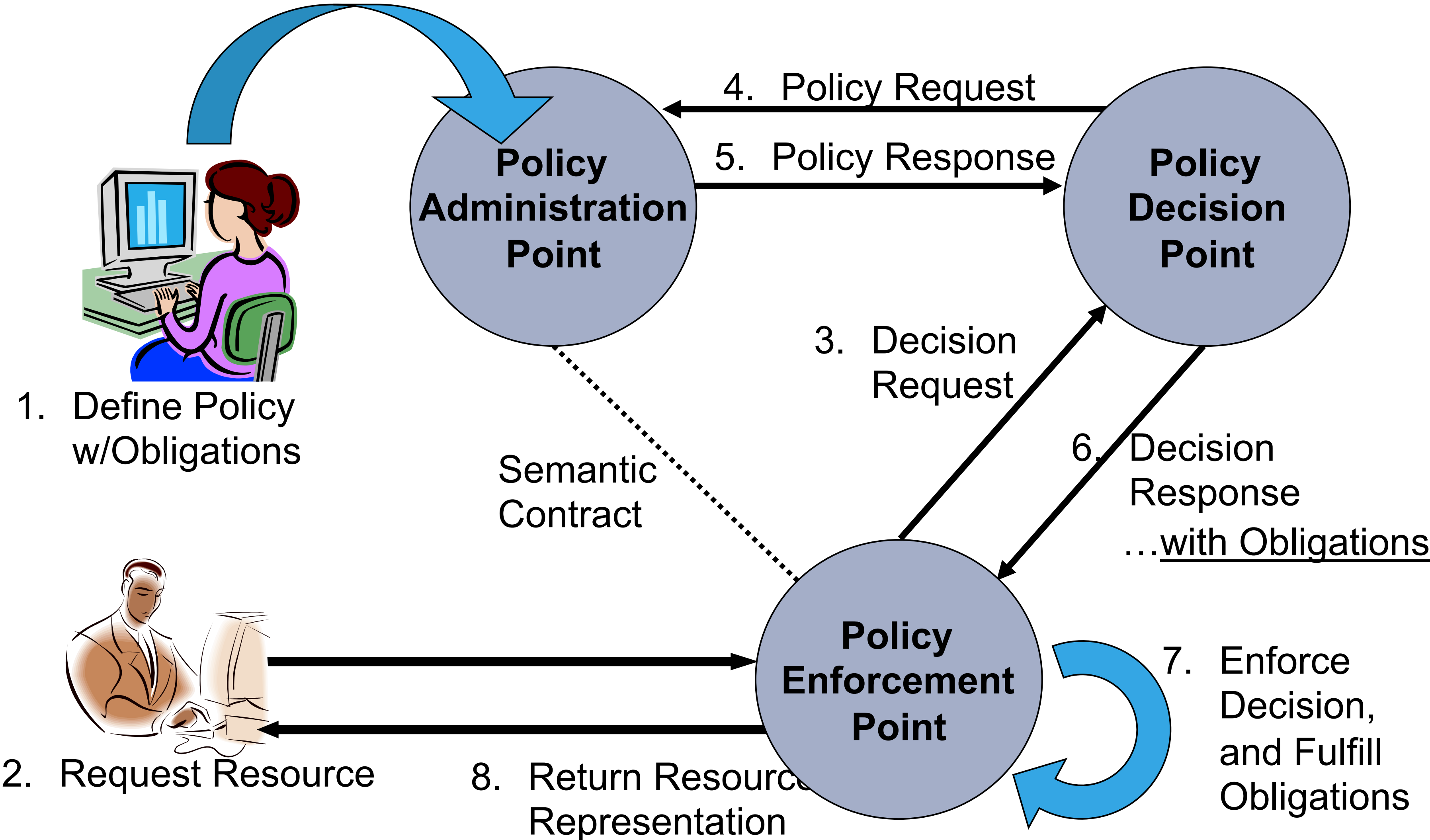
- Obligations can be modeled as an extension to the classical conceptual model used for security enforcement.




EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

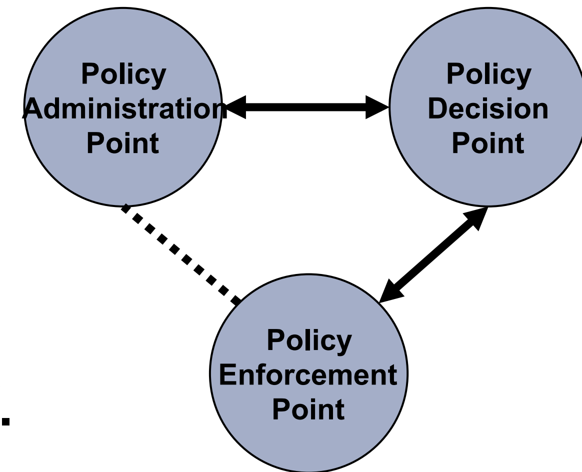
# Conceptual Model for Obligations




EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Conceptual Model for Obligations




- The PAP **defines** the Obligations.
- The PEP **fulfills** the Obligations.
- The PDP is a trusted intermediary.
  - Instantiates the obligation assertions,
  - Does not otherwise interpret or enforce the obligation.
- Obligations extend the interface contract between the PDP and the PEP, and the semantic contract between the PEP and the PAP.

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# A Standardized Model for Obligations

- The OASIS XACML Technical Committee has defined a standard conceptual model for obligations.
  - Boolean response + Obligation URN + attribute metadata.
  - …unordered collection of strongly typed attributes, with values associated at runtime.
    - `Permit`…but do `urn:emc:cto:obl:poc:1` with `attributes`
    - `Deny`…and do `urn:emc:cto:obl:poc:2` with `attributes`
- Our implementation was *inspired* by the XACML core standard, but is not a complete implementation.
- Our design choices were driven by the desire to support REST developers who are already using a services framework such as Apache CXF or Spring MVC, along with Spring Security.

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Developer Enablement via the Spring Framework

- Effective use of obligations requires support via a developer framework.
  - Framework-based enforcement ensures appropriate:
    - Separation of concerns in the Java code
    - Separation of skills and responsibilities between application development and application deployment.
- The Spring Framework is the de facto standard for Java developers
  - Thus, we explored what could be done to provide support for obligations via Spring Security.

EMC² where information lives

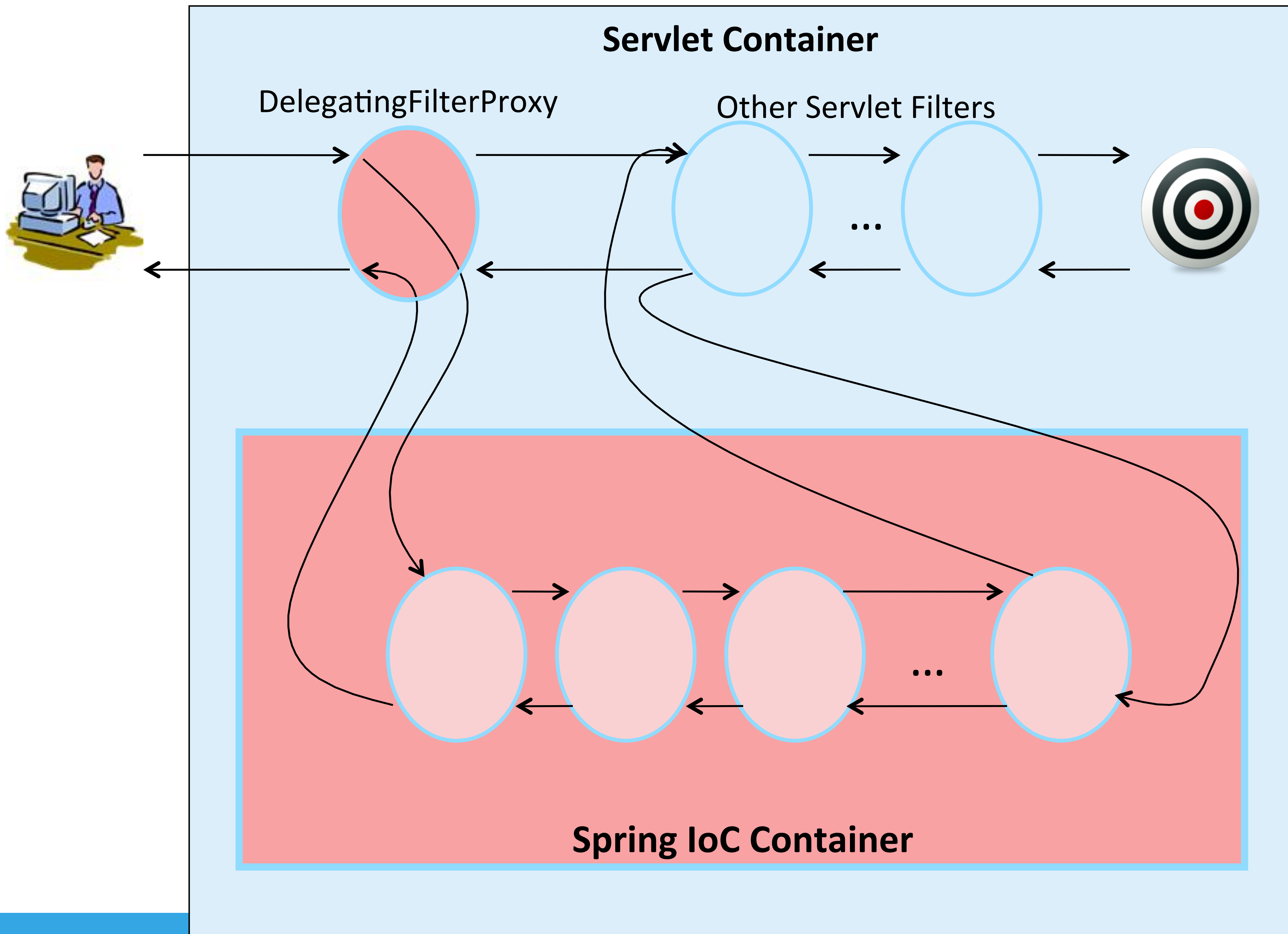
© Copyright 2011 EMC Corporation. All rights reserved.

# Extending Spring Security

- Spring Security's features are extensible through the use of dependency injection.
  - We designed and implemented the new obligation-aware components, and then configured these using DI.
- The new components are designed to co-exist with the installed base.
  - But to ensure that fulfillment occurs, deployers must be sure to configure the obligation-aware components first.

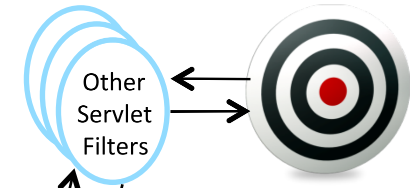
EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.




EMC
where information lives
© Copyright 2011 EMC Corporation. All rights reserved.

# Inside the Spring Security Filter Chain




Security Context Persistence

Authent-ication

Session Management

Author-ization

Allowed

Denied

Securit y Contex t

- Authentication
- ROLE_XXX

Access Decision Manager

where information lives

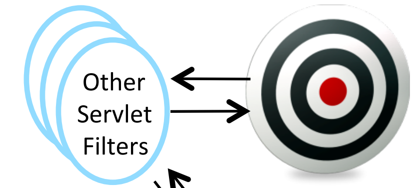
© Copyright 2011 EMC Corporation. All rights reserved.

# Supporting Obligations in Spring Security



© Copyright 2011 EMC Corporation. All rights reserved.

# What about Obligations related to “Access Denied”?




Obligation Enhanced Security Context Persistence

Authent-ication

Session Management

Autho izatio

Deny Obligation Fulfiller Filter

Obligation Fulfiller Filter

Obligation Fulfillers

Obligation Fulfillers

DENIED and…

Obligation Enhanced Security Context

PiP

- Authentication
- ROLE_XXX
- Obligations

PaP

Obligation Enhanced Access Decision Manager

Fulfiller Registry

Fulfiller Registry

where information lives

© Copyright 2011 EMC Corporation. All rights reserved.

# General Obligation Design Pattern

Q: When should software designers use obligations?

A: When there is a requirement for a supplementary policy enforcement action that is:

- Application-centric, but not application-specific.
- The supplementary enforcement is concurrent with the processing of a specific user transaction.
- Enforcement action may be independent of the outcome of the user transaction.
- Enforcement requirement specified at deployment time.
  - Requirement may not yet be known at application design time

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Metadata for Obligations

***Representing and Managing both the Implict and Explicit Properties***

- No standards exist for the metadata properties that characterize the obligation behaviors.
- Our analysis led to the following table:

| Obl. Category | Obl. Type | Directly Modifies Request? | Directly Modifies Response? | Fires Before Resource Access? | Fires After Resource Access? | Side Effects? | Synchro-nized* |
|---|---|---|---|---|---|---|---|
| **GRC** | **Policy Mgmt** | **No** | **No** | **Yes** | **No** | **Yes.**<br>**PAP updated** | **No** |
| **Security, SIEM** | **Audit Logging** | **No** | **No** | **No** | **Yes** | **Yes.**<br>**Logs updated** | **No** |
| **SLA** | **Resource Mgmt** | **No** | **Yes** | **No** | **Yes** | **None** | **No** |

*Synchronized before response sent to client?

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Obligation Management Consistent with REST

- Obligations must themselves be considered REST resources.
  - They are addressable, cacheable, resources.
- The linked data representation of the obligation should include
  - Implicit metadata properties, as described above
  - Explicit relationships with other resources, i.e. a type, such as a Patient application or record.
- The obligation management system must exhibit the appropriate emergent properties:
  - Loose coupling through the uniform interface, scalability, visibility

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Compatibility With HATEOAS Constraint

- Obligations can have a direct impact on the resource representation that is delivered to the client.
  - The client may or may not be made aware that obligation fulfillment has occurred.
- Either scenario can maintain complete adherence to the HATEOAS constraint.
  - The representation must contain a consistent metadata set, via appropriate links, and media types.
  - i.e. one could represent the required fulfillments through the use of specific media types
    - `Content-Type`, maybe `Accept` headers.

EMC²
where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

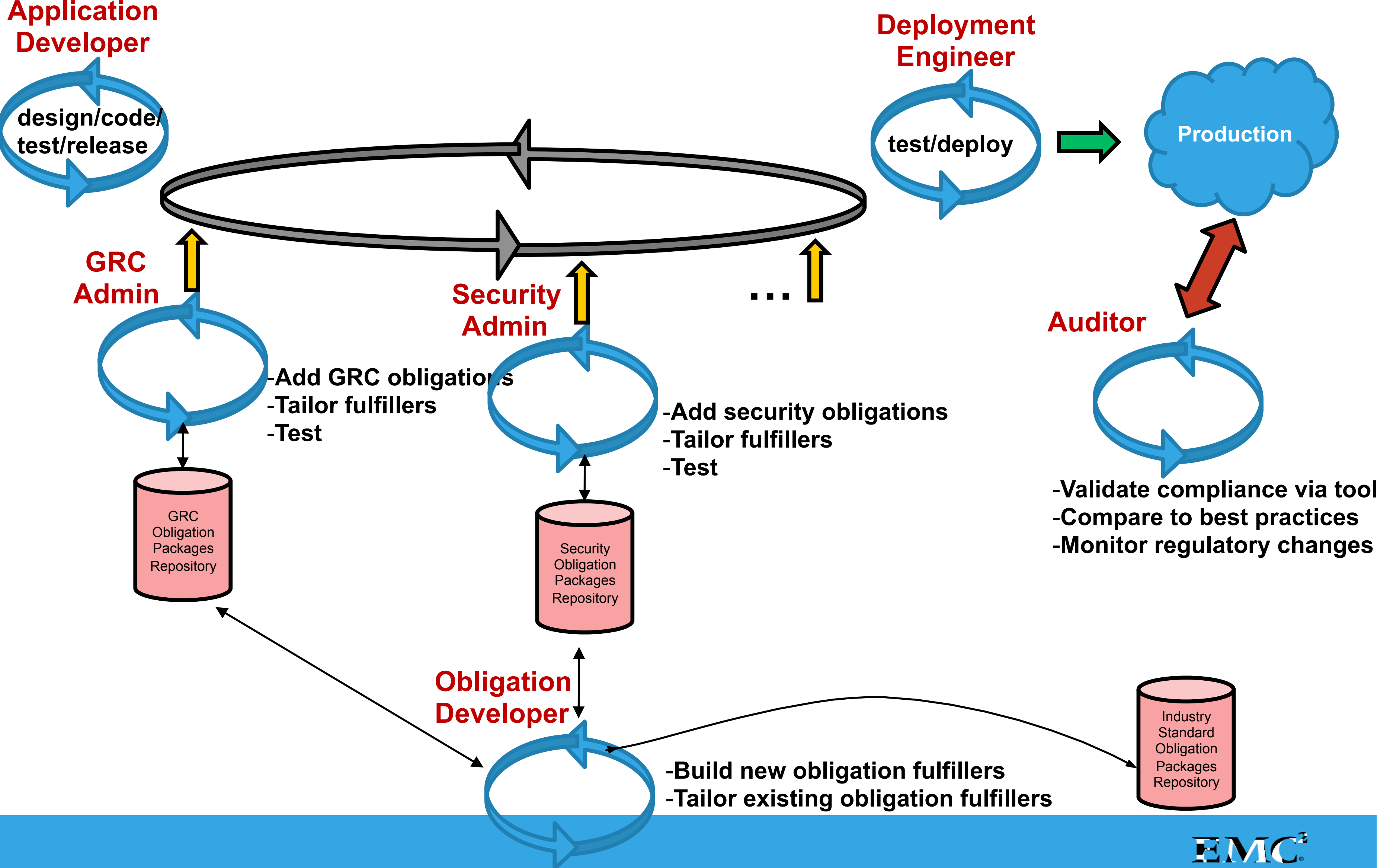
# Compatibility with the Layered System Constraint

- Our current prototype is based upon an in-process model.
  - Targeted to the installed base of Java + Spring Framework developers.
- One can also envision other fulfillment models.
- The layered system and uniform interface constraints of REST ensure that clients and servers may rely upon arbitrary intermediaries to provide fulfillment.
  - This approach may have advantages when in-process fulfillment is not convenient or achievable.
- An out-of-process fulfillment option again highlights the importance of defining obligations as addressable resources
  - As the intermediary may be a shared component supporting more than one administrative domain, i.e. in a Cloud Service.

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Obligation-enhanced Development




EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Operational Benefits of Obligations

- For the developer
  - obligations provide a consistent and standardizable mechanism to enable injection of application-centric GRC controls at design, development or deployment time.
- For the security admin
  - obligations provide a mechanism to enable applications to effectively leverage an existing XACML-based enterprise entitlement management infrastructure.
- For the deployment engineer
  - obligations can also be used to customize a generic application in support of specific corporate policies.
    - including adjustments required for different localities or geographies.
- For the auditor
  - obligations are an effective way to factor and package the code that implements any application-independent or well-known controls.

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# Q&A

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.

# THANK YOU

EMC² where information lives®

© Copyright 2011 EMC Corporation. All rights reserved.