Policy Development Process and Supporting Components

Purpose of this Document

This is an overview discussion on how to use Security and General Business Policies in a controllable fashion.

Additionally, it is conceptual background and guidance for requirements on a Policy Management System tool. Such a tool would maintain a common repository of Policies and be able to consume and render them in a deployable format for any implementation language. Otherwise, even federated Policy Management would become extremely burdensome to maintain.

Components Supporting Policy Development and Operations

Regardless of a policy language chosen, standard or proprietary, there is a set of 12 components, called, **The Duty Dozen** here, that one must address.

The Duty Dozen are interconnected to form a Policy development method, end to end. This method covers all steps from the defining Business Processes all the way to Deployment of Policies into Operations, followed by the most difficult step of all, Deprecation of Policies into retirement.

A Duty Dozen: Development of Business Process to Operations

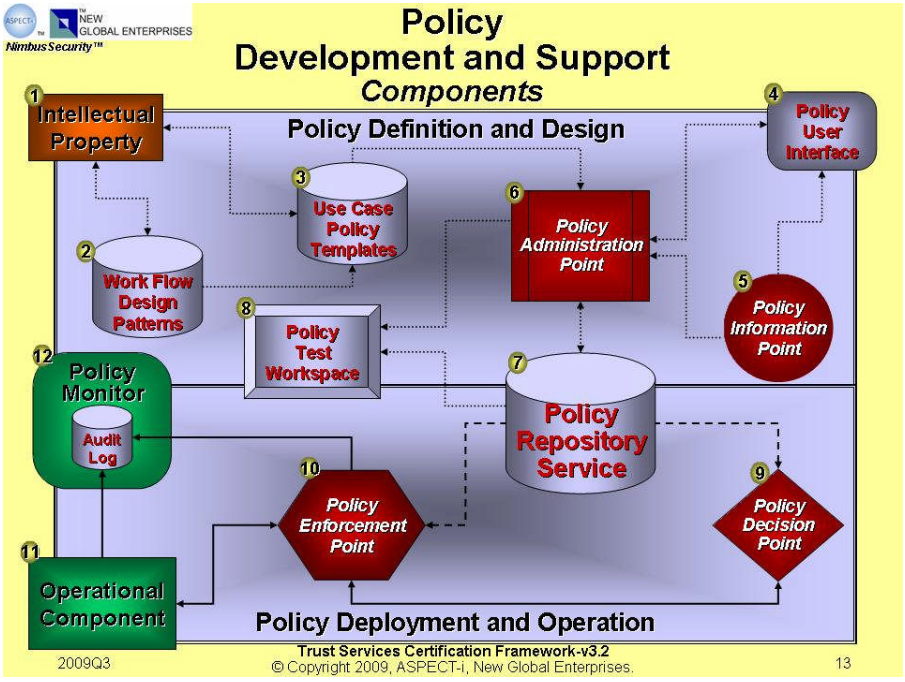

Figure 1: Overview of Development and Support for Biz Process to Operations

Figure 1 depicts how Enterprise Intellectual Property in the form of Business Processes is used to define and select Work Flow Design Patterns and Use Case Policy Templates.

This is accomplished through a Policy User Interface that gathers Policy Information from Points of Presence and allows the *Creation* through *Deprecation* of Policies through any Policy Administration Point of Presence.

Policy Administration Points interface to Policy Repository Services which maintains all in-flight, extant and retired policies and policy suites. The Repository supports the Policy Test Workspace which is where in-flight policies are moved through the life cycle up to Deployment and Operation.

There are Policy Decision Points which support Policies at the Points of Enforcement in Operational Components.

All, this culminates in the Policy Monitor which shadows the Audit Log.

With respect to the Data Flow Diagram on page 17 of the XACML 2.0 Specification (http://tinyurl.com/j73hb**)**, this architecture virtualizes the Policy Information Point.

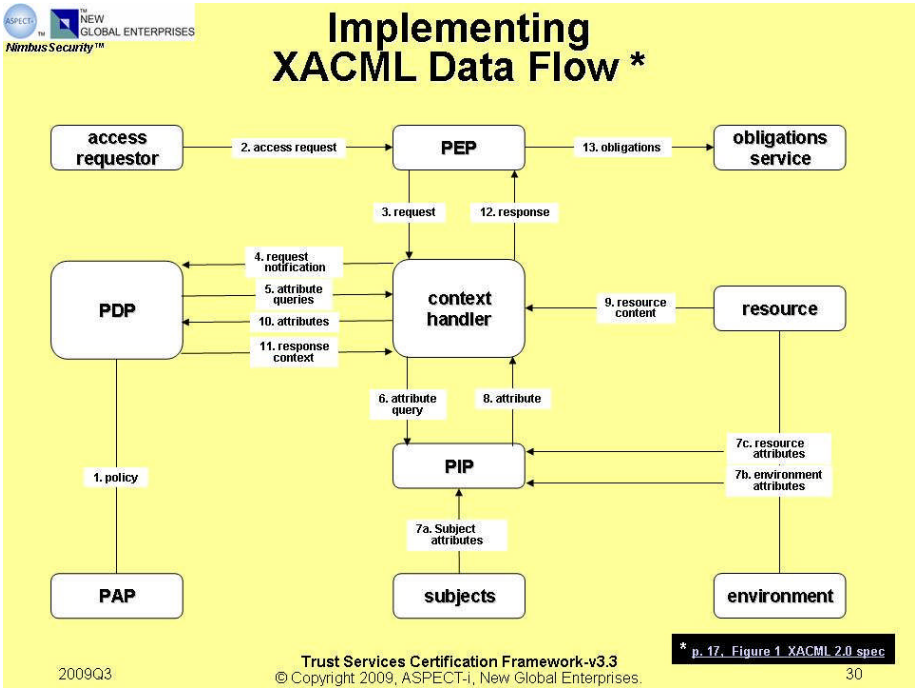

Figure 2: XACML Data Flow

Our Figure 1 is conformant to the XACML architecture in the 2.0 spec. This is a topic of another paper itself and would only distract from the discussion here.

To complete the "picture is worth a thousand words" theory of content, the following 12 sections cover the relationships amongst the components of Figure 1.

1. Intellectual Property: Business Process

Business Processes unique to an Enterprise make up this critical Enterprise IP. These determine the patterns of Work Flow and Use Case Policies that are relevant to Operational Deployment and Monitoring.

2. Work Flow Design Patterns

There are many differing Work Flow Patterns which need to be accommodated: people/auto agent/combo, simple serial tasks, parallel threaded (forked/joined), embedded, etc. Order2Cash BP requires a parallel thread Work Flow. Auto provisioning of a grid is a simple serial Work Flow.

An example of a Work Flow Pattern, without accompanying annotations of deployment options

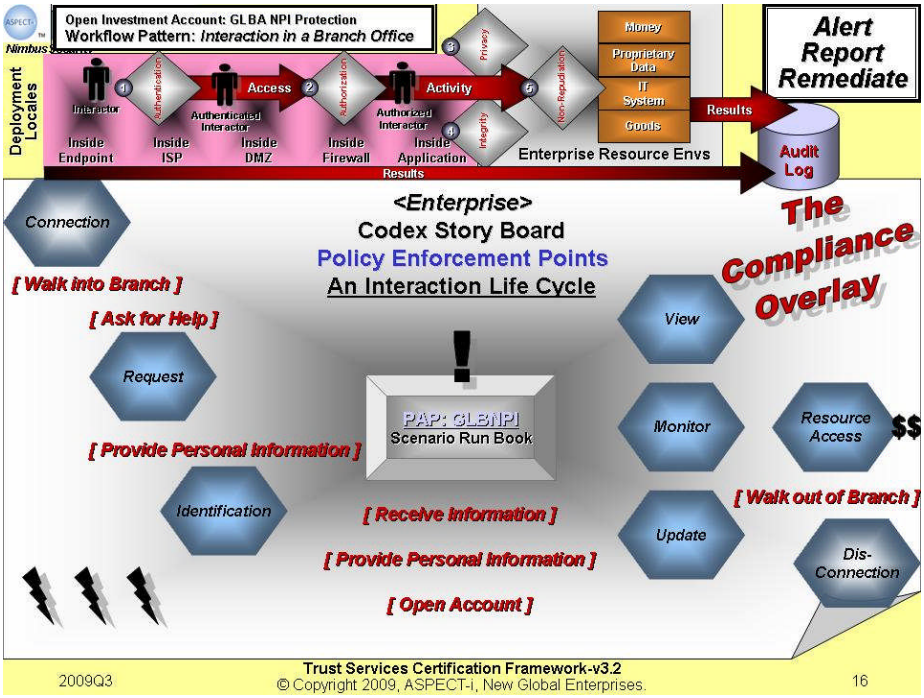

Figure 3: Alert, Report, Remediate on Enterprise Resources

3. Use Case Policy Templates

For each Use Case, there will be templates per suites of policies that are applied at different Points of Enforcement within a Work Flow.

Consider the above generic Compliance Overlay Work Flow Pattern: ***Interaction in a Branch Office***.

The Points of Enforcement are the blue hexagons: Connection, Request, Identification, View/Monitor/Update Enterprise Resources, Access to those Resources, and, lastly, Disconnection. These are the places from which critical business events are generated.

Each Policy Enforcement Point (blue hexagon) contains a suite of "boiler plate" policies properly abstracted with substitution variables, much like shell variables in a DOS BAT file, perl/python/php/ruby script or Bourne/C/Korn Shell script.

In a fashion, the Environment elaborates as a set of ***<variable>=<value>*** pairs which are applied to the salient template to produce the Operational Master copies of deployable PEPs and their associated PDPs. These are all stored and extracted from the Policy Administration Point per the choices of deployment language/appliance modalities for the Policies. (cf. section below on ***Policy Repository Services***.)

4. Policy User Interface

The Policy User Interface is delivered in a variety of forms—GUI, Web Service API, or libraries of methods for various languages.

Interaction through this interface invokes various functional capabilities in the life cycle of Policy Suites (cf. section below on ***Policy Life Cycle Development***.)

5. Policy Information Point

Policy Information Points are systems and stores of identity, entitlement data and other controls for system Interactors. These data include LDAPs, Active Directories, /usr Directories as well as data bases and flat files.

These data are imported into the process of creating and storing Policies within the Policy Administration Point.

6. Policy Administration Point

The Policy Administration Point is central to Policy Development and Support. As the name implies, it is the place where policies are administered and managed. It actuates the creation, storage, modification, testing, deployment and archiving of Policies.

It is able to import Work Flow Design Patterns and concomitant Use Case Templates. In support of the PAP content production, it brokers Policy Repository Services to the Policy User Interface.

7. Policy Repository Service

The Policy Repository Service (PRS) is the core of Policy Development and Support. In addition to storing, retrieving and archiving Policies, the PRS can import/export XACML forms of Polices. It provides deployment renditions in languages like Java, C/C++, C# and scripting forms like perl/python/php/ruby.

The PRS can be accessed through Web Services and APIs alike.

8. Policy Test Workspace

The Policy Test Workspace is just that—the place to test and assure the functioning of Polices. It is a mini Operations Environment which assures proper functioning when Policies are deployed into OPS.

The PTW contains Unit, Integration and System Test areas. It is the Test Staging area. This is infrastructure deployed like applications—a great advantage in controlling complexity. Infrastructure cannot be easily advanced as EVERYTHING that is dependent on it cannot be regression tested in a reasonable amount of time.

9. Policy Decision Point

A Policy Decision Point is that place in the system where a Policy is evaluated at run time and its results produced for consumption by a Policy Enforcement Point.

One PDP can serve many PEPs, implying it must support multiple threads as well as multiple users.

10. Policy Enforcement Point

Policy Enforcement Points are either located within or invoked locally from within Operational Components.

The key requirement here is that the PEP operation must fail hard and coincidentally with the failure of the invocation of the Policy Decision.

This is implied by the data flow above from page 17 of the XACML 2,0 Spec. The issue addressed is how it is known that Policy Decision is Atomic.

11. Operational Component

Operation Components are methods invoked from within deployed processes. It was from within these Operational Components that the PEP is either resident or invoked via a transactional proxy.

12. Policy Monitor Audit Log

The Policy Monitor Log is the repository for all audit data. It correlates the results from the operational Component and the PEP in the case the PEP is external from the Operational Component.

Policy State Life Cycle from Definition to Master Operational Copy

This section is a contribution to the Open Narrative that is dubbed, ***NimbusSecurity***, nimbus being the form of rainmaking cloud. Rainmaking is a term borrowed from investment banking which means generating huge amounts of high value activities.

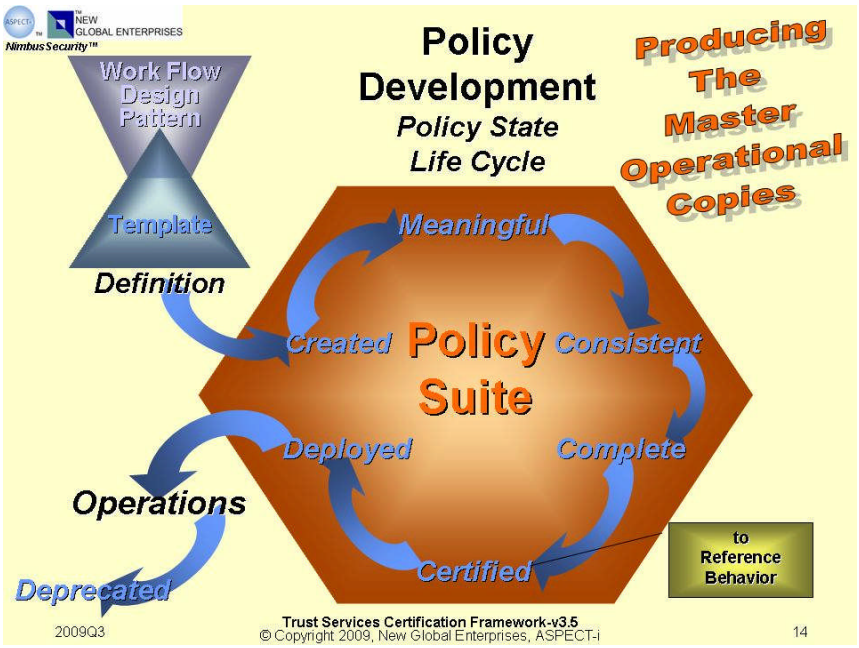

Figure 4: Policy Development; Policy State Life Cycle

Policy Suite Life Cycle

Sets of XACML policy, page 19, 2.0 Specification (http://tinyurl.com/j73hb**)**, are extended as policy suites of guarded commands which are congruently XACML Rules of Conditions and the consequential Effects, cf. the bottom of the Figure 5 diagram of XACML Policy Language Model below:

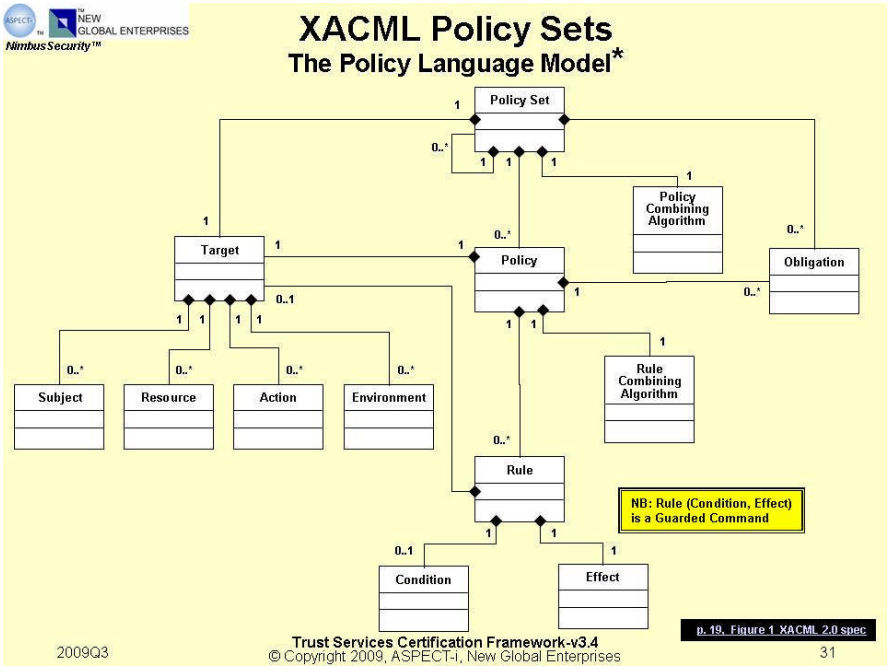

Figure 5: XACML Policy Language Model

Dubbed ***NimbusSecurity***, it includes a number of Security Work Flow Design Patterns from which derived Template artifacts seed and drive the Definition of Policy Suites. This requires a supporting Policy Repository Service per Section 7 above.

The list immediately below describe the states of policies and policy suites (XACML Policy Sets) that are developed as part of an Enterprise Business Policy Codex Policy Enforcement Points per the Gramm Leach Bliley Act Non-Public Information compliance example detailed in Figure 3 under Section 2 Work Flow Design Patterns above.

The policy state meanings of work-in-process policies with respect to operational Enterprise Systems:

- **Created**

  Newly minted policies that are well formed syntactically.

- **Meaningful**

  Terms and target references of the policies exist and are well defined.

- **Consistent**

  Policies do not conflict with extant, in force policies deployed or proposed for deployment.

- **Complete**

  Policies do not contain gaps over the range of covered target references.

- **Certified**

  Policies conform to the desired Reference Behavior which can be determined by automated/manual methods.

- **Deployed**

  Policies that are operationally in force in the various SDLC process environments, Unit Test, Integration Test, Staging Test, or Production.

- **Deprecated**

  Policies that are to be avoided and/or replaced for a variety of reasons such as design flawed but kept for backward compatibility (cf., Wikipedia Deprecation entry (http://en.wikipedia.org/wiki/Deprecation) for a fuller description of use in software development).

Reference Behavior

Reference behavior is the centerpiece of system governance. It is used to certify conformance with the in force Enterprise Standards and Reference Architecture.

These include the officially sanctioned way:

- to enforce security policies,
- to comply with regulations,
- to ensure availability,
- to achieve performance, and,
- to assure ease of use.

Conclusions

Current State of the Art

External Cloud is better developed than Internal Cloud because it is somewhat of a green field while Internal Clouds must deal with legacy integration.

External Cloud providers are maturing as viable business operational facilities and have been quite useful in low security circumstances as with Analytics Calculation Facilities.

External Cloud Providers offer walled gardens at the moment, but integration technologies and methods are available to use. Large vendors like the usual suspects are talking Cloud while the Amazons and RackSpaces are walking Clouds.

Internal IT organizations are learning about Clouds and what it means to architect internal systems as multi-tenant and multi-landlord.

The Security issues are well under way to resolving and Operational Monitoring is quickly following on its heels.

We are still doing Production Prototypes through 2009 with Pilots launched by the end of 2010. In 2011, we will see a much more rapid adoption in Cloud deployments of Business Services.

Going Forward with the Long View

We are entering an era when we have both the knowledge and the computing power to really prove things about deployments BEFORE they go into production. This is knowledge and reasoning as the ultimate security infrastructure.

To reason about deployments, more formality is required. But, the constant challenge is to make this formalism accessible. We need "Tools. Tools. Tools."

A US partisan would say, "Who says the US is becoming devoid of tool and die makers?" This virtual manufacturing demands new specialized tools and dies. Now we use "Self-Service" for tool and "Template" for die.

IT is the Tool and Die making of the new manufacturing.