# Proposed Model for Natural Language ABAC Authoring

Ronald C. Turner
XpressRules LLC
1818 W. Francis Ave. Ste 250
Spokane, WA 99208 USA
1+(509) 467-0668
Ron.Turner@XpressRules.com

## ABSTRACT
Authorization policy authoring has required tools from the start. With access policy governance now an executive-level responsibility, it is imperative that such a tool expose the policy to business users—with little or no IT intervention—as natural language. *NIST SP 800-162* [1] first prescribes natural language policies (NLPs) as the preferred *expression* of policy and then implicitly calls for automated *translation* of NLP to machine-executable code. This paper therefore proposes an interoperable model for the NLP's human expression. It furthermore documents the research and development of a tool set for end-to-end authoring and translation. This R&D journey—focusing constantly on end users—has debunked certain myths, has responded to steadily increasing market sophistication, has applied formal disciplines (e.g. ontologies, grammars and compiler design) and has motivated an informal demonstration of autonomic code generation. The lessons learned should be of practical value to the entire ABAC community. The research in progress—increasingly complex policies, proactive rule analytics, and expanded NLP authoring language support—will require collaboration with an ever-expanding technical community from industry and academia.

## Keywords
ABAC; natural language policies; business rules; XACML authoring tool; policy semantics; RDF analytics; SPARQL queries

## 1. MOTIVATION
This paper and the research and development it represents are shaped by three milestones in the evolution of authorization architectures.

In 2003 the Organization for the Advancement of Structured Information Standards (OASIS) ratified V1.0 of Extensible Access Control Language (XACML) as an abstraction layer to enable portable access policy expression that is independent of computer-executable code. Its syntax and verbosity represented costly development effort. But the good news from the start was that "XACML is intended primarily to be generated by tools." The bad news was "These tools aren't here yet."[2]

Over the next eight years access policies became a matter of intense administrative concern. In 2011 a team of several agencies' chief information officers released the Federal Government's implementation of Identity, Credential, and Access Management (FICAM) Roadmap and Implementation Guidance v2.0 [3], which defines a comprehensive life cycle for access control policies. FICAM's level of discourse about access control is clearly now at the executive and business-user level.

The 2014 National Institute of Standards and Technology (NIST) *Special Publication 800-162 "Guide to Attribute Based Access Control (ABAC) Definition and Considerations* defines natural language policy (NLP) as "Statements governing management and access of enterprise objects. NLPs are human expressions that can be translated to machine-enforceable access control policies. NLP is thus an additional abstraction layer, this layer to isolate the non-IT business user from formal authorization policies.

This paper investigates the two-fold market-facing challenge to the ABAC tool provider: (1) defining a right-fitting model for the policy's *human expression* and (2) developing an automated *translation* from NLP to machine-executable form.

## 2. POLICY MYTHS
There are (at least!) four iconic myths swirling in the space of info security architecture, misperceptions that hinder the effort toward effective NLPs.

**Bubblehead Policy Author**

*Myth:* S/He works in heroic isolation, grinding out authorization policies, which will be snatched up momentarily to be enforced by the PIP and PAP. An effective authoring tool will therefore reduce the pre-deployment time to near-zero.

*Reality:* Access policies are invariably the work of a *team*. And no technology can replace the role of team deliberation and exchange.

**One-size-fits-all Policy Language**

*Myth:* Unlike IT, business users (BUs) engage one another using business-user natural language ($NL_{BU}$). Our task is to discover and describe a sub-set of $NL_{BU}$ so that we can develop a repeatable process for reducing $NL_{BU}$ to machine-executable form.

*Reality:* The policy authoring team ranges from senior management--those with no tolerance for technical notation—to gifted developers for whom programming languages are totally natural. So the "language" in "NLP" in fact presents as a continuum of dialects. So rather than simply to "discover and describe," our task is instead to "select and formalize" a single domain-specific language (DSL) that will allow for comprehensive NLP governance by business users.

**Ark of the Covenant** (aka "Policy Store")

*Myth:* Like the stone tablets of old, policies come down from On High; they simply *happen*. The iconic and serene "Policy Store" appears in every security architecture. It may receive new data from the Bubblehead Policy Author (above), and it will supply those data to the PIP and PAP. But the Policy Store zone of the architecture is a quiescent and rather uneventful neighborhood.

*Reality:* "Store" is an outright misnomer. Policies in fact represent a high-traffic complex within the security infosphere. As such the "Policy Store" itself comprises a rich architecture of its own.

**IT Defensive Tackles** (Developers fearing job endangerment)

*Myth:* Significant IT effort is invested in information security, with large numbers of developers coding endless maintenance revisions to hard-coded security software. The developer and her organization fear that to automate security processes would endanger IT budget and positions.

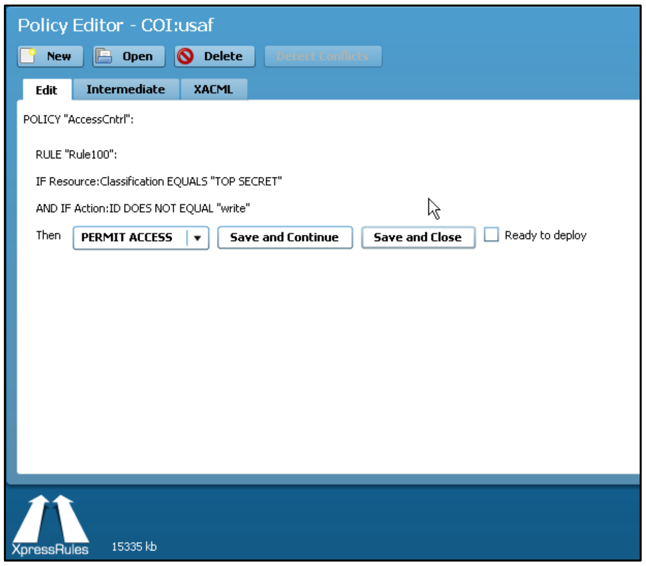
*Reality:* IT organizations everywhere are saturated with projects far more urgent that writing maintenance code. And even if a compelling ABAC solution is available, IT fears most their own added burden of installation, learning curves, and total cost of ownership (TCO). IT would welcome the relief of business users themselves assuming a meaningful share of responsibility for their own assets' security.

## 3. THE "EMERGING MARKET"

From its origin XpressRules Studio has been shaped by market demand. The company's research and development has responded to three sequential demands, each iteration serving to augment the architecture and functionality of the Studio.

### 3.1 Drive Cross-Domain Guards with NL

Our company's requirement for NL authoring came first from the US Air Force Research Laboratory (AFRL) in 2005. AFRL was seriously exploring the new OASIS Extensible Access Control Markup Language (XACML) standard for enforcing rules in their cross-domain guard. They soon realized that XACML required a NL interface. Under a Small Business Innovation Research (SBIR) contract, we delivered to the U.S. Air Force Research Laboratory (AFRL) an XML Cross-Domain Guard ($X^2DG$)



POLICY "AccessCntrl":

RULE "Rule100":

IF Resource:Classification EQUALS "TOP SECRET"

AND IF Action:ID DOES NOT EQUAL "write"

Then PERMIT ACCESS

**Figure 1. Policy Editor for XpressRules AFRL Prototype**

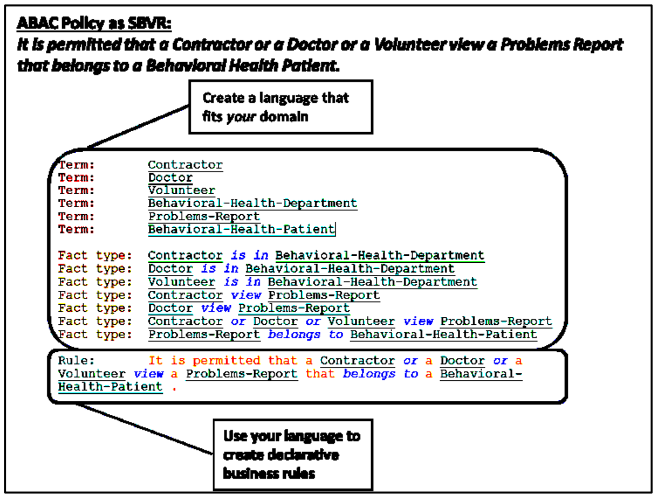
prototype. The policy editor console (Figure 1) depicts a rulebuilding session using the XpressRules prototype.

The prototype was a contractual success, automatically transforming NL to error-free XACML and "decompiling" XACML back to NL for effective audit. This was impressive in multiple demonstrations, but the prototype was "brittle," far from market-ready:

1. Rule building allowed only for tightly constrained selections by trained MLS specialists.
2. The interface was built with Adobe's ActionScript.
3. The underlying architecture relied heavily on MySQL and extensive stored procedures.
4. The development environment, although driven by a strict agile methodology, required high-level Java programming expertise for any changes.
5. However pleasing the user interface (UI), the prototype was essentially a dedicated XACML compiler-decompiler.

### 3.2 Define Policies as Business Rules

Every demonstration of XpressRules to business users prompted the audience—accurately or not—to equate access rules with *business rules*. To capitalize on the immediate market recognition of business rules, we discovered a promising fit between XpressRules' semantics-rich expressions and Semantics of Business Vocabulary and Rules™ (SBVR™) [4]. SBVR is a project of the Open Management Group (OMG). It restricts itself to structured English, rules of which begin with "It is permitted that…," "It is prohibited that …" and other "modal operations." Figure 2 depicts the SBVR approach to formulating business rules: (1) define Terms, (2) use the Terms to construct Fact Types, then (3) combine the Facts to build declarative Rules.



**ABAC Policy as SBVR:**

***It is permitted that a Contractor or a Doctor or a Volunteer view a Problems Report that belongs to a Behavioral Health Patient.***

Create a language that fits *your* domain
Use your language to create declarative business rules

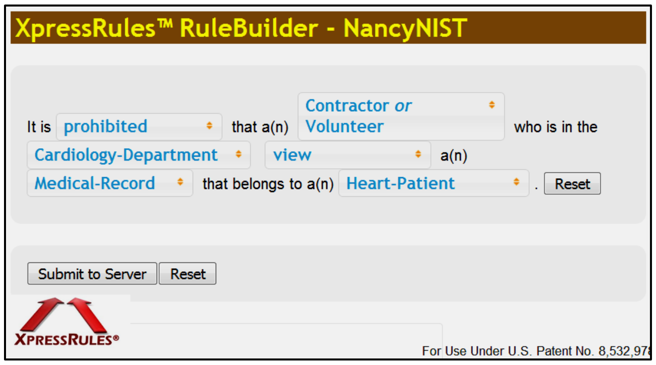
```
Term:        Contractor
Term:        Doctor
Term:        Volunteer
Term:        Behavioral-Health-Department
Term:        Problems-Report
Term:        Behavioral-Health-Patient

Fact type:   Contractor is in Behavioral-Health-Department
Fact type:   Doctor is in Behavioral-Health-Department
Fact type:   Volunteer is in Behavioral-Health-Department
Fact type:   Contractor view Problems-Report
Fact type:   Doctor view Problems-Report
Fact type:   Contractor or Doctor or Volunteer view Problems-Report
Fact type:   Problems-Report belongs to Behavioral-Health-Patient

Rule:        It is permitted that a Contractor or a Doctor or a
Volunteer view a Problems-Report that belongs to a Behavioral-
Health-Patient .
```

**Figure 2. SBVR Depiction of an ABAC privacy rule**

Based somewhat on the SBVR paradigm, the company released V2.0 of XpressRules, built on a much more sustainable architecture: 100% open-source and freely available components, 100% open-standards compliance for all information processing, dynamically (programmatically) generated NL→XACML/JSON and XACML→NL transformers, and zero client-side install or client footprint, Figure 3 depicts the V2.0 console for creating an access rule.


XpressRules™ RuleBuilder - NancyNIST

It is prohibited that a(n) Contractor or Volunteer who is in the Cardiology-Department view a(n) Medical-Record that belongs to a(n) Heart-Patient .

**Figure 3. XpressRules RuleBuilder Console**

XpressRules V2.0 successfully demonstrated round-trip (NL→XACML/JSON→NL) policy authoring to a variety of industries. And it did so in seconds instead of the weeks typically required in an IT-centric environment. This invariably evoked immediate excitement. But the market's response identified the gap between our successful R&D and the capabilities required for enterprise-level uptake:

1. **"Analytics hooks."** A serious policy store requires that policy auditors perform both forensic and proactive (i.e. forward-looking) analyses of access. Furthermore the logical integrity of a policy store relies on rigorous and comprehensive semantics-based analysis.

2. **Localized (Re)configuration.** Enterprise-level policy organizations may require that they themselves—without vendor reliance—be able to modify or rebuild the total authoring environment at will: specialized vocabulary and syntax, browser consoles, and options within the generated code (XACML/JSON).

3. **Scalability.** Familiar ABAC use cases typically do not address the size or complexity of industry-scale policies: thousands of options for a single attribute selection, dynamic attribute stores, arbitrary types of attributes for each rule category—subject, action, resource and environment.

4. **Deployment.** In theory a policy is a document. In management practice, the policy must therefore operate *at least* according to the requirements for document management: version control, version merging, source control, check-in/-out, locking/unlocking, history tracking, fallback and more.

## 3.3 Represent ABAC as Semantic Models

The most recent R&D has yielded an ABAC authoring environment that is totally ontology-driven. [5] And in the spirit of business-level user experience (UX), the underlying work flow is totally hidden from the end user. This ontology-motivated architecture enables important feature enhancements including the following:

- Local authorized configuration stewards themselves may now reconfigure or rebuild the Studio in seconds without IT intervention.
- Dynamically-generated expandable tree drop-downs allow for user selection from entire taxonomies of ABAC attributes.
- A compact ontology-based "minimal storage unit" allows for enterprise policy stores that will enable FICAM-style policy life cycle management.

Semantic models "under the hood" support ABAC for at least two industry-critical requirements: agile reconfiguration and taxonomic representation.

### 3.3.1 *(Re)configuration by the "Syntax Layer"*

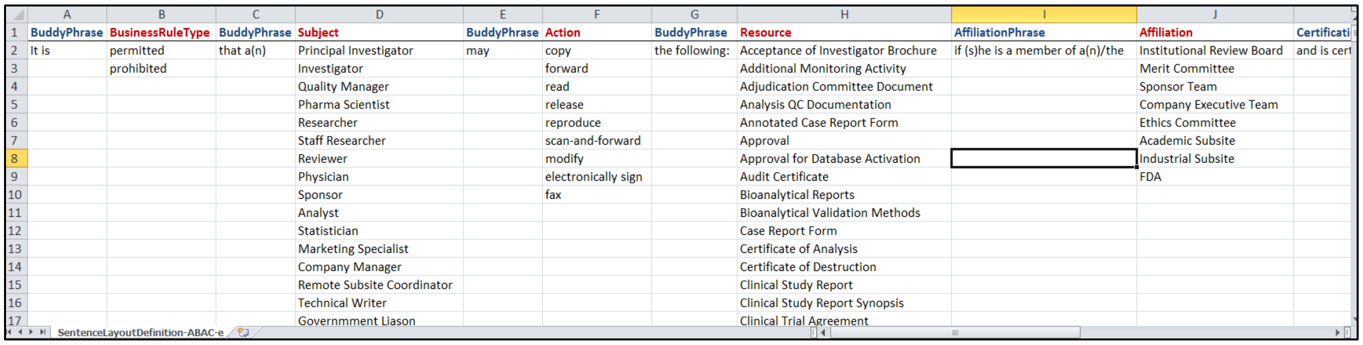
A scalable ABAC authoring solution requires that a customer-side configuration specialist—not the vendor—manage the authoring environment: site-specific vocabularies and syntax, attribute options, taxonomies from industry-specific standards groups, live feeds from external back-end attribute exchanges, and customized browser consoles that incorporate all of these for the local end user.

Unlike out-of-the-box templates used in many authoring tools, XpressRules has relaxed the notion of "template" such that a local configuration specialist defines "sentence classes" whose members are instantiated by options and attributes. Figure 4 is a spreadsheet representation of a sentence class. A spreadsheet is but one approach to defining a class and requires minimal skills. The configuration specialist need not be IT-capable.

The items in the "red" columns in Figure 4 are the drop-down options to appear in the end-user's console. These options—255 for "Resource" in this case—can be supplied by a back-end attribute exchange (BAE) server or may be automatically generated from an industry group's specification. Or they may be entered manually by the specialist. In The items in the "blue" columns are the linking phrases supplied by the specialist.

Unlike the restrictive "canned" templates or fill-in forms commonly found in policy authoring tools, the ABAC Sentence Class allows the specialist to build customized rule-building consoles for *any* ABAC expression. He or she only needs to understand the following:

- The columns represent the sequential fields of a NLP expression.
- Five of the "red" fields are mandatory:



| | A | B | C | D | E | F | G | H | I | J | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | BuddyPhrase | BusinessRuleType | BuddyPhrase | Subject | BuddyPhrase | Action | BuddyPhrase | Resource | AffiliationPhrase | Affiliation | Certificati |
| 2 | It is | permitted | that a(n) | Principal Investigator | may | copy | the following: | Acceptance of Investigator Brochure | if (s)he is a member of a(n)/the | Institutional Review Board | and is cert |
| 3 | | prohibited | | Investigator | | forward | | Additional Monitoring Activity | | Merit Committee | |
| 4 | | | | Quality Manager | | read | | Adjudication Committee Document | | Sponsor Team | |
| 5 | | | | Pharma Scientist | | release | | Analysis QC Documentation | | Company Executive Team | |
| 6 | | | | Researcher | | reproduce | | Annotated Case Report Form | | Ethics Committee | |
| 7 | | | | Staff Researcher | | scan-and-forward | | Approval | | Academic Subsite | |
| 8 | | | | Reviewer | | modify | | Approval for Database Activation | | Industrial Subsite | |
| 9 | | | | Physician | | electronically sign | | Audit Certificate | | FDA | |
| 10 | | | | Sponsor | | fax | | Bioanalytical Reports | | | |
| 11 | | | | Analyst | | | | Bioanalytical Validation Methods | | | |
| 12 | | | | Statistician | | | | Case Report Form | | | |
| 13 | | | | Marketing Specialist | | | | Certificate of Analysis | | | |
| 14 | | | | Company Manager | | | | Certificate of Destruction | | | |
| 15 | | | | Remote Subsite Coordinator | | | | Clinical Study Report | | | |
| 16 | | | | Technical Writer | | | | Clinical Study Report Synopsis | | | |
| 17 | | | | Governmment Liason | | | | Clinical Trial Agreement | | | |

**Figure 4. Syntax for an ABAC Sentence Class**

1. Business Rule Type
2. Subject
3. Action
4. Resource (255 in this instance)
5. Environment

- The ABAC field definitions to the right of "red" field 4 (Resource) appear in the following order:
  1. Subject Attributes$_1$, Subject Attributes$_2$ , . . Subject Attributes$_n$, (arbitrary number of Attribute types and columns, here two: "Affiliation" and "Certification.")
  2. Action Attributes$_1$, Action Attributes$_2$ , . . Action Attributes$_n$, (arbitrary number)
  3. Resource Attributes$_1$, Resource Attributes$_2$ , . . Resource Attributes$_n$, (arbitrary number)
  4. Environment Attributes (Same as above)

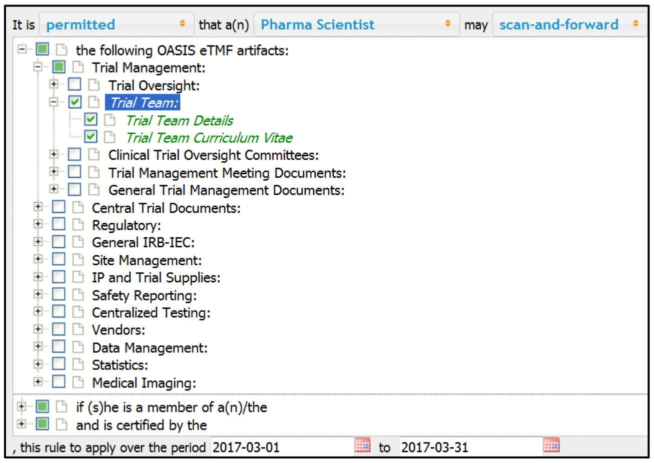
In the underlying architecture, all rule- and policy-building interaction with XpressRules occurs in the Application Layer (Open Systems Integration model, Layer 7). But the configuration specialist's dialogue is within the Syntax Layer (Layer 6, "Data Representation"). Interpreting the fields in a syntax definition—their order, their meaning, their types—and then executing numerous internal transformations is delegated to the Syntax Layer. This delegation allows XpressRules to support the syntaxes of any number and variety of sentence classes (e.g. Patient Consent Directives, ATM deposits and withdrawals, Physical Access to Secured Facilities).

### *3.3.2 Selection from attribute taxonomies*

A typical RuleBuilder console now appears as in Figure 5. The Rule's 255 Resource options appear in the taxonomy expansion of "artifacts."

The selections depicted in Figure 5 generate XACML, JSON and reverse-compiled NL audit for this policy:

```
It is permitted that a(n) Pharma Scientist may
scan-and-forward the following: Trial Team
Details or Trial Team Curriculum Vitae if (s)he
is a member of a(n)/the Merit Committee and is
certified by the American Board of Colon and
Rectal Surgery, this rule to apply over the
period 2017-03-01 to 2017-03-31.
```


It is permitted that a(n) Pharma Scientist may scan-and-forward
- the following OASIS eTMF artifacts:
  - Trial Management:
    - Trial Oversight:
    - Trial Team:
      - Trial Team Details
      - Trial Team Curriculum Vitae
    - Clinical Trial Oversight Committees:
    - Trial Management Meeting Documents:
    - General Trial Management Documents:
  - Central Trial Documents:
  - Regulatory:
  - General IRB-IEC:
  - Site Management:
  - IP and Trial Supplies:
  - Safety Reporting:
  - Centralized Testing:
  - Vendors:
  - Data Management:
  - Statistics:
  - Medical Imaging:
- if (s)he is a member of a(n)/the
- and is certified by the

, this rule to apply over the period 2017-03-01 to 2017-03-31

**Figure 5. XpressRules RuleBuilder Console with dynamic taxonomy drop-downs**

The underlying source code and data for the console in Figure 5—1175 lines of HTML, JSON, JavaScript, jQuery, AJAX and CSS—is in itself of little interest. The notable phenomenon is that the entire console, including the expandable taxonomy fields, is created by a semi-skilled specialist in real time. This level of dynamic configuration would be impossible without the rich structure and semantics provided by the rule's ontology.

## 4. POLICY LANGUAGE DEFINITION

This discussion of the language of NLP is based XpressRules LLC's perception of current and emerging market readiness for authoring NLPs with the following characteristics:

1. Are suitable for life cycle management
2. Conform to the guidance of *NIST SP 800-162*
3. Impose minimal and right-fitting constraints on policy stewards
4. Maintain a declarative (vs. procedural) syntax
5. Preserve intact the semantics of ABAC attributes

We suggest that such NLPs can support the majority of authorization use cases.

### 4.1 NLP for Comprehensive Management

The "L" in "NLP" is a multi-faceted participant in multi-layered policy life cycle management. Figure 6 depicts some important aspects of managing NLP:

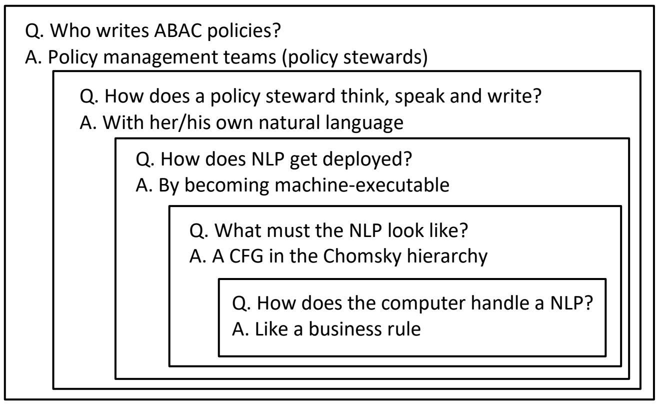

Q. Who writes ABAC policies?
A. Policy management teams (policy stewards)
Q. How does a policy steward think, speak and write?
A. With her/his own natural language
Q. How does NLP get deployed?
A. By becoming machine-executable
Q. What must the NLP look like?
A. A CFG in the Chomsky hierarchy
Q. How does the computer handle a NLP?
A. Like a business rule

**Figure 6. NLP Within a Policy Management Architecture**

These questions suggest the interrelated facets and requirements of a full-bodied enterprise-level policy management service. The interrelatedness reminds us that the definition of the language has important ramifications throughout the entire human → machine work flow.

### 4.2 Natural Language Policy in SP 800-162

"Natural Language Policy (NLP)" is prominent in the *NIST SP 800-162* ("Guidance"), occurring no less than 29 times! The informal definition [§2.4.1 ] is informative:

> Natural Language Policies (NLPs) are high-level requirements that specify how information access is managed and who, under what circumstances, may access what information.

The formal definition is highly challenging because it assumes a complex translation process:

> ***Natural Language Policy (NLP):*** *Statements governing management and access of enterprise objects. NLPs are human expressions that can be translated to machine-enforceable access control policies.*

In order to accomplish this "translation," the Guidance directs that "NLPs must be codified into Digital Policy (DP) algorithms or mechanisms." Digital Policy is defined as both (1) a cluster of activities and (2) an intermediate form:

> ***Digital Policy (DP):*** *Access control rules that compile directly into machine executable codes or signals. Subject/object attributes, operations, and environment conditions are the fundamental elements of DP, the building blocks of DP rules, which are enforced by an access control mechanism.*

The DP is thus the policy's primary expression. And so the Guidance rightly advises that "DPs [with their enforcement capabilities] . . . need to be managed, stored, validated, updated, prioritized, deconflicted, shared, retired, and enforced." These capabilities are collectively termed Digital Policy Management (DPM), an activity that is widely understood among policy officers. And in "Policy Management Strategy" below we shall return to DPM's tasks and capabilities.

The Guidance furthermore specifies the DP as an *intermediate implementation* for translation. In contrast, the XpressRules approach decouples the DP-as-intermediate-form-and-process altogether. In abstracting away the translation process, all implementation details (e.g. number of intermediate processes and definitions of intermediate data structures) are left to the discretion of the product architects. In actual fact the XpressRules Studio dynamically creates upwards of forty such structures and executes over thirty processes during its various "translations."

SP 800-162 was not intended to be a treatise on natural language. Nevertheless the Guidance discusses certain properties and characteristics of natural language that play a major role in the implementation of NLP. The following list summarizes that discussion:

- NLPs are expressed in human understandable terms.
- [NLPs] may not be directly implementable in an Access Control Mechanism (ACM).
- NLPs may be ambiguous.
- [Ambiguous NLPs are] hard to derive in formally actionable elements.
- Enterprise policy [that is based on these ambiguous NLPs] may be difficult to encode in machine-enforceable form.

To paraphrase more loosely:

- NLPs fortunately (and finally) allow for human stakeholders to take control of their own policies.
- NLPs unfortunately may be ambiguous, making it difficult to render policies in machine-executable form.
- Even unambiguous NLPs may not be *directly* implementable in machine-executable form.

Taken at face value, the success of ABAC that is both (1) human-understandable and (2) machine-executable might seem problematic at best and daunting at least.

Sensing the urgency of NL in the rules marketplace, the company adapted the list above and subsequent discussion of NLP in *800-162* to create its top-level product requirements for Authoring Studio. And in so doing, XpressRules promotes the NLP to center stage:

1. A human-understandable NLP is the unique and authoritative expression of a policy and is the primary player in every stage of the policy's life cycle.
2. The NLP must—without human intervention and within seconds—be implementable in an ACM.
3. To qualify as a policy's authoritative embodiment, the NLP must be unambiguous. But resolving ambiguity must furthermore take into account all applicable rule outcomes.
4. As an unambiguous expression the NLP must ultimately be amenable to *logical analysis* (of the policy's own consistency and correctness) and to *data analytics* (for both forensic and proactive views of the effects of the policy's execution)
5. NLP authoring in itself, consisting of the usual CRUD activities, demands sufficient effort and attention so as to constitute its own sphere of prominence within the PAP.
6. NLP authoring must treat the attributes (for subject, action, resource, environment) as model-based "editing coequals" with subjects, action, resource and environment.
7. A comprehensive and proper architecture and implementation for NLP authoring—because of natural language—must radically reduce the policy's development and maintenance costs

## 4.3 Constraining the NLP Grammar

In the spirit of hiding complexity from business users, it is inappropriate to draw undue attention to the grammar that drives their rule authoring solution. At the same time it is imperative not simply to say that the language has a grammar but to specify what the grammar is and how it drives the authoring of NLP expressions. The approach for this project is to *constrain* the NLP's context-free grammar (CFG) in such that every NLP expression (1) represents a *deterministic* CFG and furthermore (2) represents an *unambiguous* DCFG. The practical advantages to this "grammar-first" approach to NL constraint are (1) that it allows for an expansive universe of NLPs and (2) it assures that these policies are rigorously *auditable*. This grammar-driven workflow furthermore allows XpressRules to support user-configured NLPs (1) whose vocabularies and syntaxes that are autonomically converted to executable computer code and (2) whose internal representations fully express an ABAC semantics. We first review the two-step CFG→unambigousCFG→unambigousDCFG *grammatical* constraint. Next we discuss the important *usability*-motivated constraint for rules as *declarative* expressions. In the next section we describe the theory and apparatus required for adequate semantic expression.

### *4.3.1 NLP as unambiguous context-free language*

Grammars are of immense importance for languages of any variety: programming, natural spoken or natural written. Grammar is therefore the foundational object of analysis in programming languages, compiler theory and automata theory. The mathematics and formality surrounding grammars is of great interest and is likely familiar to most of the readers who have though seriously about NL. In our authoring universe, a grammar is similarly all-important. In very non-mathematical terms, a grammar has the following characteristics and properties:

1. It consists of a sequence of math-like expressions, defining in a succinct manner all the strings (spoken utterances and written sentences)—and *only* those strings— that we are allowed to *generate* with the language.
2. A grammar is therefore a language's *constraint*.
3. The number of a language's sentences is practically infinite.
4. In addition to *generating* sentences, the grammar can work in reverse, *parsing* (interpreting) the sentences.
5. If we match up one of the sequential (one-dimensional) strings in the language with the grammar for that

language, the grammar-as-parser determines the following:

a. Does this string really belong to this language?
b. If so, what is *the significance*—either semantic meaning or syntactic function—of each segment of the string?

6. The definite article in "*the significance*" (5.b.) is critical. We need in every case for the parser to provide only a single result for each string.

This chatty list is a relatively accurate description of an *unambiguous grammar*. XpressRules demands unambiguous grammars because (per item #6 above) it relies on its parsers to return only a single result for each NL sentence.

To be slightly more formal, here is the ambiguity test for a grammar:

Using only this grammar try to say or write two or more expressions that differ in their "significance" (semantic meaning or syntactic function). If you cannot, then the grammar is unambiguous.

### *4.3.2 NLP as unambiguous deterministic CFG (DCFG)*

Our authoring environment demands a CFG that is more than unambiguous. The grammar must furthermore be a *deterministic* context-free grammar (DCFG). Below is a user-modifiable segment of the grammar for one of XpressRules' "sentence templates."

An XpressRules administrator uses a pre-defined "sentence template" to create a locally-configured unambiguous DCFG. Here is a portion of the localized Tree Transformation Lanauge [TXL] [6] DCFG created during the configuration step, later to be repeatedly consumed by the XpressRules parser:

```
define br_action_phrase
     [br_action]
   | [br_action_phrase] or [br_action]
end define

define br_action
     copy
   | forward
   | read
   | release
   | reproduce
   | scan-and-forward
   | modify
   | electronically sign
   | fax
end define
```

The grammar for each such template is deterministic because every "terminal node" in the grammar (e.g. "release" in this example) is defined prior to the grammar's deployment for building NL policies and transforming NL→XACML/JSON. Constrained by the pre-defined "sentence template," end users thus create deterministic context-free sentences ("sentential forms"). Here is one such sentence:

```
It is permitted that a(n) Quality Manager
may release the following: Approval or
Study Registry Documents if (s)he is a
member of a(n)/the Company Executive Team
and is certified by the American Board of
Physical Medicine and Rehabilitation,
this rule to apply over the period 2016-
12-01 to 2017-02-28.
```

## 4.4 NLPs as Declarative Expressions

Because every NLP in XpressRules is thus constrained by its deterministic grammar, the task of parsing a NLP is likewise deterministic (adhering to a "straight-line" algorithm). The parser's computational burden is therefore drastically reduced. The benefit of this reduced-burden "closed-loop" work flow is that XpressRules itself can reliably configure and create major components of the NL Processor software on the fly.

We chose to embed TXL, a freely-available parsing and transformation product that fully supports CFGs. In doing so we bypassed a NL parsing implementation altogether. TXL-based parsing allowed us to concentrate on the non-trivial task of writing an *unambiguous* DCFG for parsing the widest domain of NL ABAC policies feasible.

In §2.2 we referenced the project's earlier guidance from SBVR, according which we selected a narrow set of "model operations" with which to structure our NL expressions: "It is permitted that…" "It is prohibited that..," and "It is obligatory that…" We selected that set because it is an adequate "semantic cover" for most ABAC rules.

The XpressRules project is keenly aware of the much broader field of business rules, of which ABAC rules are a proper subset. We seek to implement whatever type of NL business rule expression for ABAC that will be right-fitting for a business-level author. Our most helpful and mature guidance in this pursuit is the book *Writing Effect Business Rules: A Practical Method*, by Graham Witt [7]. The chapter "How to write quality natural language rule statements." The author presents a convincing argument for the use of *templates*, of which XpressRules "sentence types" are an example. His "Complete Taxonomy of Rules" furthermore helps us to keep our project conformant to a maturing business rules mainstream. Most importantly we monitor such guidance carefully to implement the widest possible "palette" of business user-level expressions.

On the other hand we have continually asserted that our NL rules be *declarative* and not *procedural*. Declarative English sentences are of the form subject-verb-object. And a few programming languages are declarative (e.g. SQL and Prolog). But the majority of programming languages are procedural. Procedural languages invariably define their control flows using some form of **`If-Then-Else`**. All procedural expressions therefore require some form of grouping (or nesting). For example, without some grouping mechanism this pseudo-code snippet [8] would be ambiguous and therefore impossible for a computer to execute:

```
if (x < 0) sign = -1; else if (x == 0) sign
= 0; else sign = 1;
```

The expression is ambiguous because it is not clear which **`if`** belongs with which **`else`**.

Does reformatting alone resolve the ambiguity?

```
if x < 0
   sign = -1;
else
   if (x == 0)
      sign = 0;
   else
      sign = 1;
```

Yes, somewhat, but only for a knowledgeable C++ programmer. And that is only because the programmer knows this "secret" (i.e. has this external knowledge): an **`else`** always belongs to the closest **`if`** without an **`else`**. And the C++ compiler also knows this secret. But for a non-IT business user it is totally unreasonable to require external knowledge of an expression's hidden assumptions.

Resolving such ambiguity in a NL expression is a somewhat more doable using parentheses (or square brackets or curly braces or even indentation alone). Consider the familiar Christmas narrative:

> "And they came with haste and found Mary and Joseph and the baby lying in a manger." (Luke 2:16)

There is no clarifying punctuation even in the original Greek. Yet, regardless of which language, every reader interprets the passage as

> "And they came with haste and found **(**Mary and Joseph**)** and **(**the baby lying in a manger**).**"

but never as

> "And they came with haste and found **(**Mary and Joseph and the baby**)** lying in a manger."

But even if parentheses can resolve ambiguities how user-friendly are they really?

Here is a pseudo-NL condition, part of a problem posted by a C++ programmer [9]:

> If sex is male and age is between 18 and 35 and previous military experience OR you can do more than 50 pushups

Multiple on-line exchanges and corrections by four mentors yielded the following C++ expression ("`&&`" means "and" and "`||`" and means "or"):

```
if ((sex == "m") && ((age >= 18) && (age <=
30))) apply=true;
else if ((sex == "f") && ((age >= 18) &&
(age <= 32))) apply=true;
else if (((sex == "m") && ((age >= 18) &&
(age <= 35)) && ((military == "yes") ||
(pushups >= 50)))) apply=true;
else if (((sex == "f") && ((age >= 18) &&
(age <= 40)) && ((military == "yes") ||
(pushups >= 30)))) apply=true;
else apply=false;
```

Nesting here is only to a level of four, while real-life code frequently requires levels twice that deep. Our conclusion is that if even moderately nested procedural code flows are challenging for programmers, they must have no place at all in a NL for non-IT business users. Consequently XpressRules has opted for NL expressions that are always declarative and not procedural. On the other hand, see §9.3 "Expanded NL Support" for our current work in behalf of users in a wider "user spectrum."

## 4.5 ABAC Semantics

A NLP—like human speech and unlike a fill-in form—is a one-dimensional unidirectional signal stream. Since our CFG is both deterministic and unambiguous, our parser must derive accurately the relationships between subject and subject attributes, action and action attributes, resource and resource attributes, and environment and environment attributes. In doing so the parser captures these semantic relationships from the ABAC NLP, which XpressRules then translates dynamically into an ontology, a *data map*. Figure 7 depicts a portion of the ABAC semantic relationships inferred from the following simple (user-role-only) rule for a management portal:

> **`It is permitted that a CPM Advisor may access CP&E Report(s) or Project View(s) or Portfolio Milestone Reporting View(s).`**

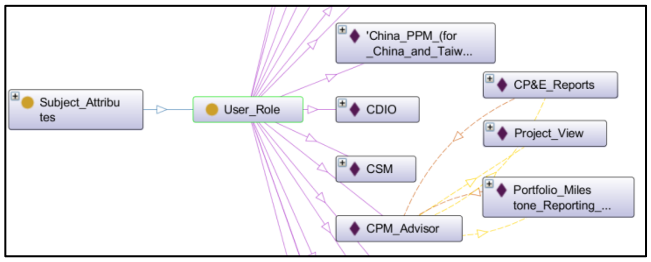


**Figure 7. Semantics of an ABAC Expression**

**Immediate application: reconfiguration.** XpressRules consumes the ontology's underlying Resource Description Framework (RDF) code in order (1) to rebuild its browser-based editing consoles and (2) to recode all of the translation programs required for the NL→XAML/JSON→NL "round trip."

**Secondary application: analytics.** Once the underlying ABAC semantics of an NLP store have been captured as RDF data maps, policy analysts may then query the data map database for ad hoc analytics to answer broad questions about the policies' implementation. (See §9.2 "Intentional" Analytics)

## 5. POLICY STEWARD EXPERIENCE

We submitted this paper because our core research problems, our central development agenda, our most important lessons learned, and our commercialization issues of greatest concern were and continue to be motivated by the *end users' natural language*:

- Who is the typical policy steward and/or attribute steward user of the ABAC tool and what is his/her "natural language"?
- Can a "typical user" even be identified, or are "policy steward" and "attribute steward" in fact a collective spectrum of "users"?
- How does a policy steward think about ABAC?
- What is the acceptable level of constraint we should exert on the NL we claim to support?
- Does NL apply only to localized *vocabularies* for subjects, actions, resources, environments and attributes of an ABAC policy, or must we allow for localized NL governance of the policy's *syntax* as well?
- How do we regard the IT policy steward for whom—as an experienced policy analyst—"natural" means "familiar," "everyday" and "easy-to-understand"? I.E. Should we not relax the definition of NL to support an IT-level "natural language" that is already machine-recognizable, "programmatic" and procedural?
- How does a large enterprise user manage the life cycles of policies-as-NL?

All of our research— to date and in progress—focuses on these seven large questions. We urge that ABAC effort everywhere continue to focus on business-level users, the true ABAC policy stewards.

## 6. DEPLOYMENT

How an access control solution is deployed is of course highly site-specific. But the architecture in Figure 8 suggests how the highest-level building blocks might be distributed so as to assure (1) a zero footprint on the client, (2) a secure "execution enclave" for all run-time processing (isolation of business users from the server's internals), and (3) a simple data exchange protocol (FTP).

## 7. POLICY MANAGEMENT OBJECTS

The immediate concern in policy *governance*—in response to *SP-162's* call to store, update, validate, deconflict, share and retire—is how best to *manage* a policy as a digital entity. How should it expose itself? What is its format? How do we protect it? How do we maintain its integrity? How do we store it? In this sense a policy is but another digital asset to be managed. Each enterprise IT group will likely have a digital asset management (DAM) solution in place. Here we suggest five approaches for consideration, each with advantages and disadvantages:

1. *Documents*. NLP representations themselves, managed—with revision control, versioning, check-in/-out—by OTS document management solutions.
2. *Code base*. XACML/JSON representations treated as computer code in a secure repository such as GIT, Mercurial or Bazaar for version control**.**
3. *Tokens*. Compact, encrypted and interoperable representations for cross-enterprise (re)deployment
4. *Database records*. SQL storage entities as searchable policy metadata
5. *GraphDB*. RDF-type tuples that allow direct queries for analytics over the policy store. OTS solutions: Direct SQL-like NoSQL data map queries (Neo4J, Titan)

## 8. LESSONS LEARNED

NLP and ABAC have required newcomer technologies for their implementation. So it is no surprise that the findings over the three phases of XpressRules' *research* have driven a constantly-evolving *development* agenda.

### 8.1 User

*Assumption*: a minimally-trained business user clicks on pull-down menu options to create machine-executable policies (XACML) and then deploys those policies in near real-time.

*Reality*: A policy's life cycle is extensive and is accordingly governed by an extensive community of responsible policy stewards.

### 8.2 Policy

*Assumption:* Per the OASIS definition "Policy" is an interoperable, standards-compliant machine-executable collection of rules.

*Reality:* "Policy" in practice is an entity consisting of multiple representations, depending on its status within its governance life cycle.

### 8.3 Verifiability

*Assumption:* A policy's ecosphere is "closed-loop." It is production- or combat-ready when (1) its semantics maps precisely to the use case against which it is formulated and (2) it conforms to propositional logic tests for contradiction, tautology, and redundancy.

*Reality:* Policy governance can at best enact a "tightened loop." (1) every use case undergoes constant change because it reflects unforeseeable conditions in its environment; (2) "closed-loop" propositional logic in itself can only mitigate semantic leaks.

### 8.4 Entity

*Assumption:* "It is radically less costly to manage NL business rules than to maintain computer code." A NL policy, though "living" and revisable, is nevertheless a *stateless* document. So

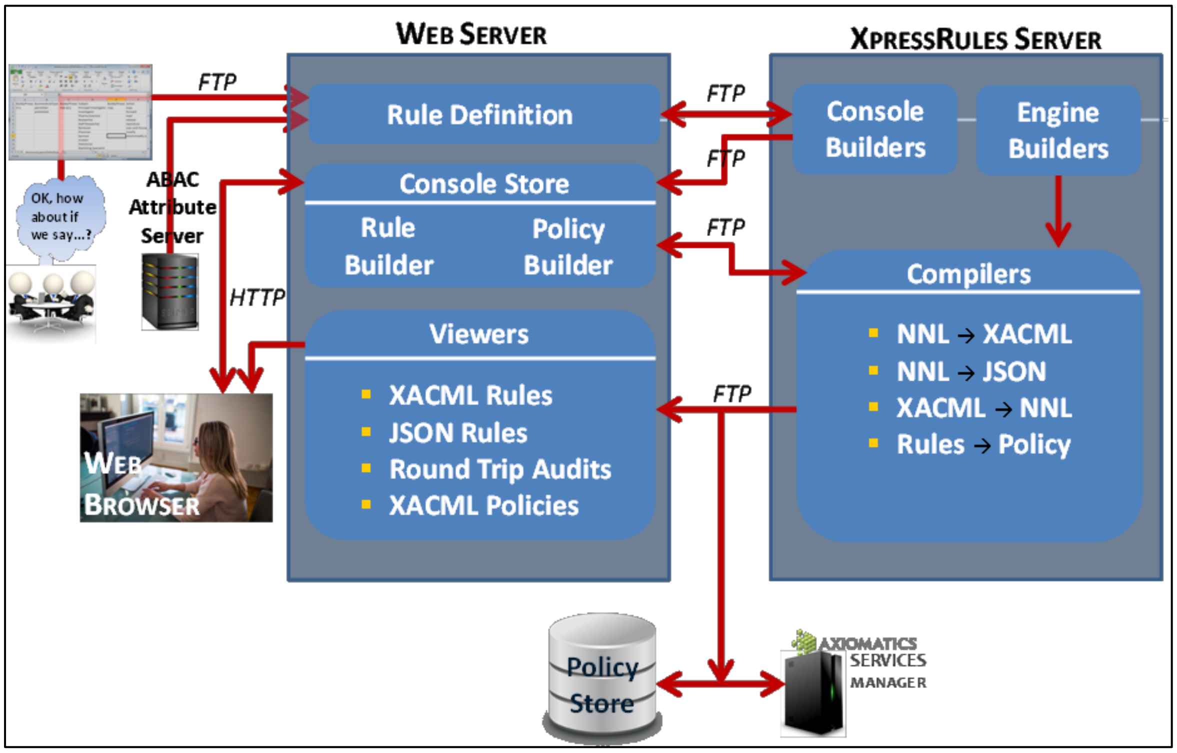


**Figure 8. An ABAC Deployment Architecture © Ronald C. Turner**

policy-as-BR governance therefore is as straightforward as document management.

*Reality:* No matter how the business rule is expressed—as NL or Java or XACML or JSON or come proprietary rules language—the rule is *stateful*, tightly coupled to the changing realities of the enterprise.

## 8.5 Analytics

*Assumption:* Analytics as low-level *analysis*. The prototype's data structures in themselves are sufficient to support both forensic (backward-facing) and *somewhat* predictive (forward-facing) analysis by auditors and risk managers. Besides "predictive" ("Who all can see what all?") is the elusive Holy Grail of access control, so why or how could we attempt a solution with XpressRules?

*Reality:* Analytics as *query-able semantic data mapping*. The prototype's data structures already were ontology-based, so it became essential for XpressRules to provide "analytics hooks" whereby risk analysts might use existing semantics technologies to query the policies—as semantic maps—to achieve more accurate predictions.

## 8.6 (Re)configurability

*Assumption*: Since client-side XpressRules is a web tool (zero footprint on the client machine) and since the user experience relies solely on browser encoding, some lower-level IT helper—over a few hours—will "simply" modify HTML and Javascript to configure the screens for the local UX.

*Reality:* A viable ABAC product requires constant real-time, non-IT, end-to-end reconfiguration. The entire product installation—computer code (eight languages), data structures, NL vocabulary and syntax—must regenerate itself totally, within seconds and with no human intervention. This "raised bar" forced the product's configuration toward becoming 100% *autonomic*, requiring only a few clicks by the end user.

## 8.7 Attributes

*Assumption:* XpressRules is for authoring policies, not managing attributes. The tool will simply ingest attributes from other entities (e.g. backend attribute exchange [BAE] providers) So it is out of scope for XpressRules to concern itself with attribute authoring and management.

*Reality:* ABAC semantics treats rule vocabularies, rule syntaxes and rule attributes as a homogenous whole. Therefore XpressRules—in addition to ingesting attributes from BAE and other externalized sources— must allow for the policy author him-/herself to introduce and manage attributes, again without IT intervention.

## 8.8 Market

*Assumption*: Authorization, logic-based access control (LBAC), and ABAC in particular are emerging and maturing, much in the way that authentication, identity management and its (SAML) standard evolved to full commercial uptake. Therefore ABAC likewise will soon become a ubiquitous commodity, thus creating a concomitant and spontaneous demand for NL ABAC.

*Reality:* With notable exceptions many key decision-makers—including those in security, compliance, privacy and risk management—still struggle to understand the difference even between authentication and authorization. Successful ABAC vendors must accordingly be eloquent ABAC evangelists and charismatic ABAC thought leaders. As for rules providers (like XpressRules LLC), there is an even more serious barrier to entry: the perception that an ABAC NLP is yet another [plain old] *business rule*, just another instance of a well-understood commodity in a very large and mature market. The challenge for the rules vendor therefore is to articulate the "what's-the-difference-between" for *business rules* generally and *natural language access policies* in particular.

## 9. RESEARCH IN PROGRESS

NLPs and NL authoring encompass technologies whose requirements have yet to be defined by a barely-emerging market. Consequently our company will continue to work from a "rolling" R&D agenda. Here are but three of such agenda.

## 9.1 Multi-dimensional policies

This paper (and XpressRules Policy Studio so far) has concentrated on ABAC *rules* as vs. ABAC *policies*. This important distinction for access policies has existed for over a decade. The "Policy Language Model" of the XACML 3.0 standard (Figure 9 below) clarifies three distinct layers: Rule, Policy and Policy set.

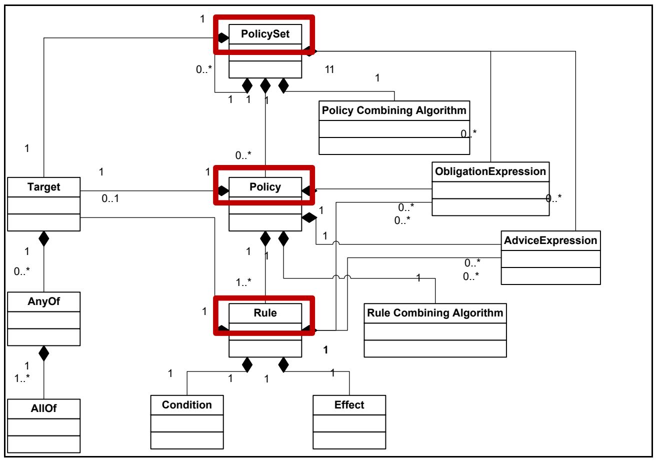


**Figure 9. XACML "Policy language model" (layers in red)**

The UML notation captures the bare bones of the model's hierarchy: one or more Rules constitute a Policy, and Polic(ies) constitute a single Policy Set. But the UML's formality conceals two important authoring corollaries and two associated research agenda:

**Corollary$_1$:** The Rule Author must be attentive to a lower, more fine-grained level of policy detail than that of the Policy Author. Correspondingly the thought process for a *business-level* Rule Author will invariably be different from that of a Policy Author.

**Research$_1$:** Discern from actual customers the best-fitting user experience for each level—mode of expression, problem-solving pattern, and how each type of author approaches his/her task.

**Corollary$_2$:** A NLP approach prescribes NL grammar (vocabulary and syntax) across the entire policy language model. But the differing policy "components" for the layers of the language model means that there will be differing NL grammars as well.

**Research$_2$:** Determine and define a right-fitting grammar for each level of NL expression: Rule, Policy and Policy set. We anticipate the need for a domain-specific language (DSL) for NL-Policy that is different from NL-Rule.

The results of this research will allow XpressRules to support a flexible policy architecture for customer-specific "policy contours" (Table 1).

**Table 1. Policy depth and Rule complexity**

| | Rule complexity | |
|---|---|---|
| **Policy depth** (complexity of Policy Set structure) | **Shallow policies** **Simple rules** | **Shallow policies** **Complex rules** |
| | **Deep policies** **Simple rules** | **Deep policies** **Complex rules** |

## 9.2 "Intentional" Analytics

"Analytics" is summarized above under Lessons Learned. With an ontology-driven design we have inserted "analytic hooks" into the design of the product. In doing so we are first illustrating the insight that *attribute analytics* provides for a rule set. We suggest a fully developed *policy analytics* will provide a similar level of insight into the overall policy. So again, an R&D agenda:

**Problem$_1$:** The term "analytics" is overused, misunderstood, too widely applied and most often grossly misapplied. Without a clear definition there can be no meaningful policy analytics.

**Research$_1$:** Listen to enterprise-level architects, IT directors and Risk Managers to formulate concise and actionable definitions of "analytics." I.E. Determine what the industry actually understands and genuinely needs from "policy analytics."

**Problem$_2$:** Much of the "analytics conversation" is based on fragmentary observations drawn haphazardly from too wide a spectrum of market activities.

**Research$_2$:** Assemble customers' actual scenarios and use cases requiring ABAC policy analytics. Formulate queries of mission-critical value based on those use cases.

**Problem$_3$:** Underutilization of "outlier" data. We tend to associate "analytics" with "semantics" and therefore fail to look beyond ontologies for semantic data mining.

**Research$_3$:** Identify and exploit *all* data, data structures, data relations, URLs and attribute values to mine the semantics required for meaningful queries.

**Two-Fold Work Flow.** An XpressRules Studio session actually comprises two separate and parallel operations: (1) compiling the rule author's NL options to *generate computer code* (XACML, JSON, XACML→NL Affidavit) and (2) dynamically *augmenting the ontology* (data map) of a multi-rule policy so that an analyst may query the semantics of the policy.

**Market-Motivated Use Case.** A work group uses XpressRules to view access provisioning as specified by the policy's attributes. The work flow comprises the following artifacts:

### 9.2.1 *Business User's Source Document*

Following a recent meeting of the Risk Management Group, an assistant transcribes access policy data from the meeting room white board as follows:

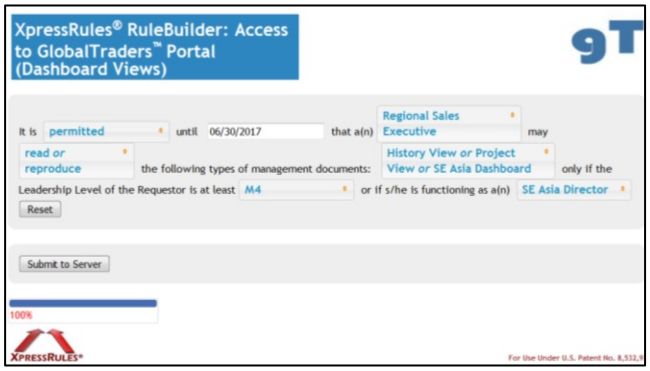
**Table 2. Roles required for Dashboard access**

| User Role | | Dashboard |
|---|---|---|
| 1. | SE Asia Director | History View,Project View,SE Asia Dashboard |
| 2. | PDQM | History Milestone Reporting View,History Vie |
| 3. | Global DPM | Project Update/Project View,History Milesto Reporting View |
| 4. | Singapore DPM | PP&I Reports |
| 5. | China DPM | PP&I Reports |
| 6. | DPM Consultant | PP&I Reports,History Milestone Reporting View,History View |
| 7. | Global QPM | Project View,PP&I Reports |
| 8. | US Log Affair PM | Project Update/Project View,History Milesto Reporting View |
| 9. | CQM | PP&I Reports |
| 10. | CQMA | PP&I Reports |
| 11. | CQL | PP&I Reports |
| 12. | CQIO | PP&I Reports |
| 13. | CQC | PP&I Reports |
| 14. | CTM | PP&I Reports |
| 15. | CMO | PP&I Reports |
| 16. | Supply Forecaster | PP&I Reports |
| 17. | Director SCM | PP&I Reports |

A requestor may access certain "Dashboard" reports or views (right column) only if her/his role matches the "User Role" in the corresponding left column.

### 9.2.2 *Auto-Generated RuleBuilder Console*

The rule author uses an auto-configured RuleBuilder console to write 17 rules, one for each row of the Roles/Dashboard table. The screen depicted in Figure 10 corresponds to the row describing access by a Regional Sales Executive—having the role of SE Asia Director—to three of the Dashboard views.



XpressRules® RuleBuilder: Access to GlobalTraders™ Portal (Dashboard Views)

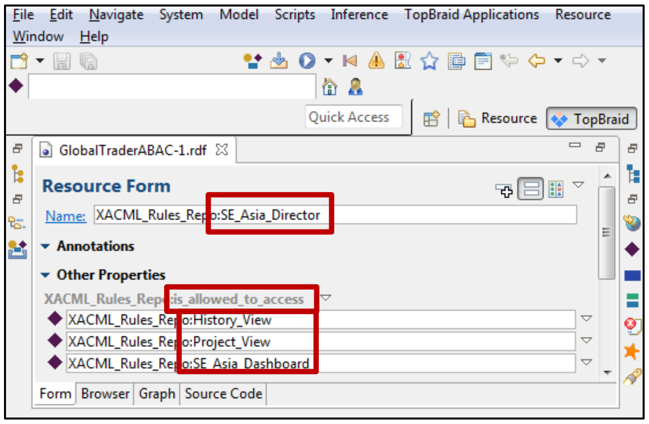
It is permitted until 06/30/2017 that a(n) Regional Sales Executive may read or reproduce the following types of management documents: History View or Project View or SE Asia Dashboard only if the Leadership Level of the Requestor is at least M4 or if s/he is functioning as a(n) SE Asia Director

**Figure 10. RuleBuilder for Dashboard Portal**

### 9.2.3 *Dynamic Revision of Data Map*

Once the rule author has completed all 17 rules, the Studio's PolicyBuilder combines those rules set into a machine-executable ABAC policy. In addition, PolicyBuilder will generate a query-ready RDF data map of the policy's rule set.

The semantic web tool TopBraid allows us to view the computer-enabled semantics expressed in Table 2. For each selection within each rule XpressRules has used the object property "is allowed to access" to associate user role with the proper resources. Figure 11 depicts the semantic association expressed in Row 1 of the table.



**Resource Form**

Name: XACML_Rules_Repo:SE_Asia_Director

Annotations

Other Properties

XACML_Rules_Repo:is_allowed_to_access
- XACML_Rules_Repo:History_View
- XACML_Rules_Repo:Project_View
- XACML_Rules_Repo:SE_Asia_Dashboard

**Figure 11. Semantic (RDF) View of Rule**

The RDF data map now contains both object property associations ("is allowed to access" and "can be accessed by") necessary to describe the semantic relations among all of the roles and all of the views and reports. The RDF rendition of the policy is now available for business users' queries.

### 9.2.4 *SPARQL-ready RDF*

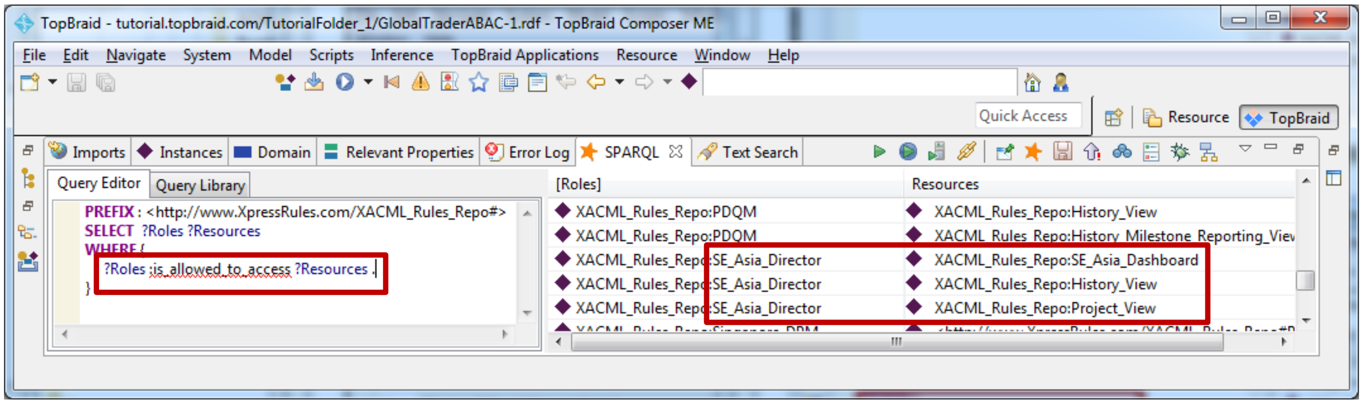
Once ConsoleBuilder has consolidated the rules, the policy's RDF file (data map) can be queried in much the same manner as a SQL database. The standard tool for such queries is SPARQL (W3C Recommendation, 21 March 2013). The screen in Figure 12 depicts the "allowed-to-access" relationships for the SE Asia Director (red box). Figure 13 depicts the results of the (inverse) "can-be-accessed-by" query.

## 9.3 Expanded NL Support

We have firmly adhered to *declarative* expressions for ABAC (for enforcing authorization *events*). But we are fully aware of the much larger market for general business rules (for governing work *processes*). There are many mature COTS BR authoring tools, some claiming to be "natural language." Most of these NL tools however are of the IF-THEN-ELSE (procedural) variety, which we know to be problematic for business users. Consequently we will continue to monitor market interest and to explore how even procedural rules might be "domesticated" to a NL form that is accessible and attractive to non-IT business users.

## 10. ACKNOWLEDGMENTS

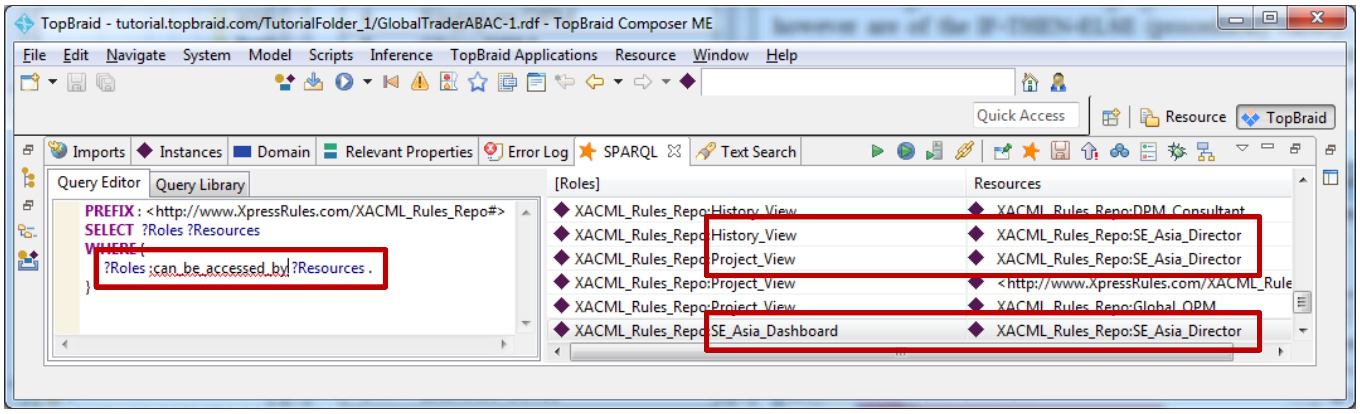
The author wishes to thank Edward Altman and Graham Witt for their visionary insight, workplace awareness and most helpful comments.


TopBraid - tutorial.topbraid.com/TutorialFolder_1/GlobalTraderABAC-1.rdf - TopBraid Composer ME
File Edit Navigate System Model Scripts Inference TopBraid Applications Resource Window Help
Quick Access
Resource
TopBraid
Imports
Instances
Domain
Relevant Properties
Error Log
SPARQL
Text Search
Query Editor
Query Library

```
PREFIX : <http://www.XpressRules.com/XACML_Rules_Repo#>
SELECT  ?Roles ?Resources
WHERE {
    ?Roles :is_allowed_to_access ?Resources .
}
```

| [Roles] | Resources |
|---|---|
| XACML_Rules_Repo:PDQM | XACML_Rules_Repo:History_View |
| XACML_Rules_Repo:PDQM | XACML_Rules_Repo:History_Milestone_Reporting_Viev |
| XACML_Rules_Repo:SE_Asia_Director | XACML_Rules_Repo:SE_Asia_Dashboard |
| XACML_Rules_Repo:SE_Asia_Director | XACML_Rules_Repo:History_View |
| XACML_Rules_Repo:SE_Asia_Director | XACML_Rules_Repo:Project_View |

**Figure 12. Querying "Who is allowed to access what?"**


TopBraid - tutorial.topbraid.com/TutorialFolder_1/GlobalTraderABAC-1.rdf - TopBraid Composer ME
File Edit Navigate System Model Scripts Inference TopBraid Applications Resource Window Help
Quick Access
Resource
TopBraid
Imports
Instances
Domain
Relevant Properties
Error Log
SPARQL
Text Search
Query Editor
Query Library

```
PREFIX : <http://www.XpressRules.com/XACML_Rules_Repo#>
SELECT  ?Roles ?Resources
WHERE {
    ?Roles :can_be_accessed_by ?Resources .
}
```

| [Roles] | Resources |
|---|---|
| XACML_Rules_Repo:History_View | XACML_Rules_Repo:DPM_Consultant |
| XACML_Rules_Repo:History_View | XACML_Rules_Repo:SE_Asia_Director |
| XACML_Rules_Repo:Project_View | XACML_Rules_Repo:SE_Asia_Director |
| XACML_Rules_Repo:Project_View | <http://www.XpressRules.com/XACML_Rule |
| XACML_Rules_Repo:Project_View | XACML_Rules_Repo:Global_OPM |
| XACML_Rules_Repo:SE_Asia_Dashboard | XACML_Rules_Repo:SE_Asia_Director |

**Figure 13. Querying "What can be accessed by whom?"**

## 11. REFERENCES

[1] NIST Special Publication 800-162: http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.sp.800-162.pdf

[2] XACML—A No-Nonsense Developer's Guide: http://www.idevnews.com/stories/57

[3] FICAM Roadmap and Implementation Guidance v2.0: https://www.idmanagement.gov/IDM/servlet/fileField?entityId=ka0t0000000TNNBAA4&field=File__Body__s

[4] Semantics of Business Vocabulary and Rules™ (SBVR™): http://www.omg.org/spec/SBVR/Current

[5] An interactive demo of the XpressRules Policy Studio is available at http://abac.xpressrules.com/ABAC_Studio.html

[6] J.R. Cordy, "The TXL Source Transformation Language", *Science of Computer Programming* 61,3 (August 2006), pp. 190-210.

[7] Witt, G. 2012. *Writing Effective Business Rules: A Practical Method*. Elsevier (Morgan Kaufmann), Waltham, MA.

[8] Based on example at https://www.macs.hw.ac.uk/~pjbk/pathways/cpp1/node99.html

[9] Final example derived by multiple contributors at this site: http://www.cplusplus.com/forum/beginner/25622/