KAoS Policies for Web Services

Andrzej Uszok, Jeff Bradshaw, IHMC, Pensacola, FL ({auszok, jbradshaw}@ihmc.us)

1. Introduction

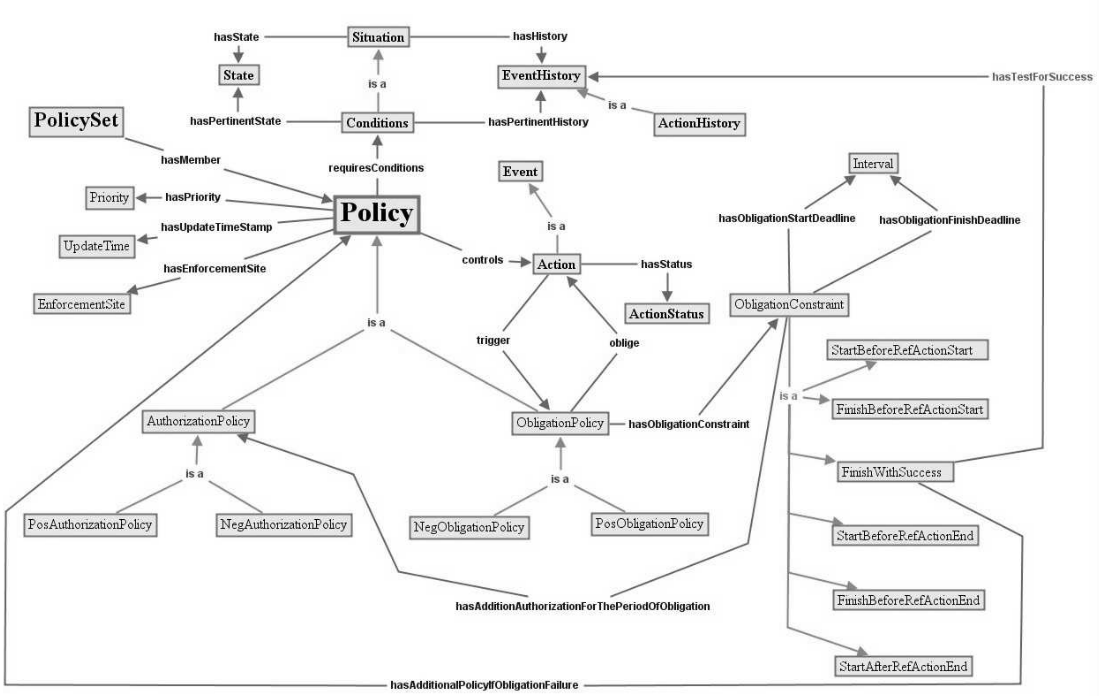
In this document we present the universal ontology-based KAoS [2, 10] Policy Service and aim to explain how it can effectively supported Web Services needs functionality. We also present the integration of KAoS with Semantic Web Services [1, 5, 8] running inside the Tomcat servlet container (*http://jakarta.apache.org/tomcat*). Additionally, we will show our integration with TrustBuilder[1], which is used for trust negotiation between a client and a service. This is, however, the first step towards the full integration as TrustBuilder currently is using its own syntax for policy expression. We are working towards the usage of KAoS to express the TrustBuilder policies, as well.

We experiment with will be policies in the context of the OWL-S services in the scope of the CoSAR-TS[2] (Coalition Search and Rescue Task Support) project. The services defined for this project include rescue resources, medical facilities, notification mechanism, and so forth, which are constrained by different policies on how they may be used in a particular coalition context.

2. KAoS Policy and Policy Service

KAoS has pioneered the use of Semantic Web language—in this case OWL—to represent policy[3]. KAoS services have been extended to work equally well with both agent-based (e.g., CoABS Grid, Cougaar, SFX, Brahms) and traditional clients on a variety of general distributed computing platforms (e.g., CORBA, Grid Computing (Globus GT3)).

Diagram Presenting KAoS Policy Ontology



[1] http://isrl.cs.byu.edu/TrustBuilder.html

[2] http://www.aiai.ed.ac.uk/project/cosar-ts/, http://ontology.ihmc.us/CoSAR-TS/Demos/CoSAR-TS_Demo_Concept.htm

[3] A comparison among two semantically-rich representations of policy (KAoS, Rei) and amore traditional policy language (Ponder[3]) can be found in [9]

KAoS uses ontology concepts encoded in OWL to build policies (see *http://ontology.ihmc.us/ontology*). These policies constrain allowable actions performed by actors (clients or agents). The KAoS Policy Service distinguishes between authorizations (i.e., constraints that permit or forbid some action) and obligations (i.e., constraints that require some action to be performed when a state- or event-based trigger occurs, or else serve to waive such a requirement). The applicability of the policy is defined by a class of situations which definition can contain components specifying required history, state and currently undertaken action. In the case of the obligation policy the obligated action can be annotated with different constraints restricting possibilities of its fulfillment.

KAoS offers generic and already well tested mechanism in at least the following areas.

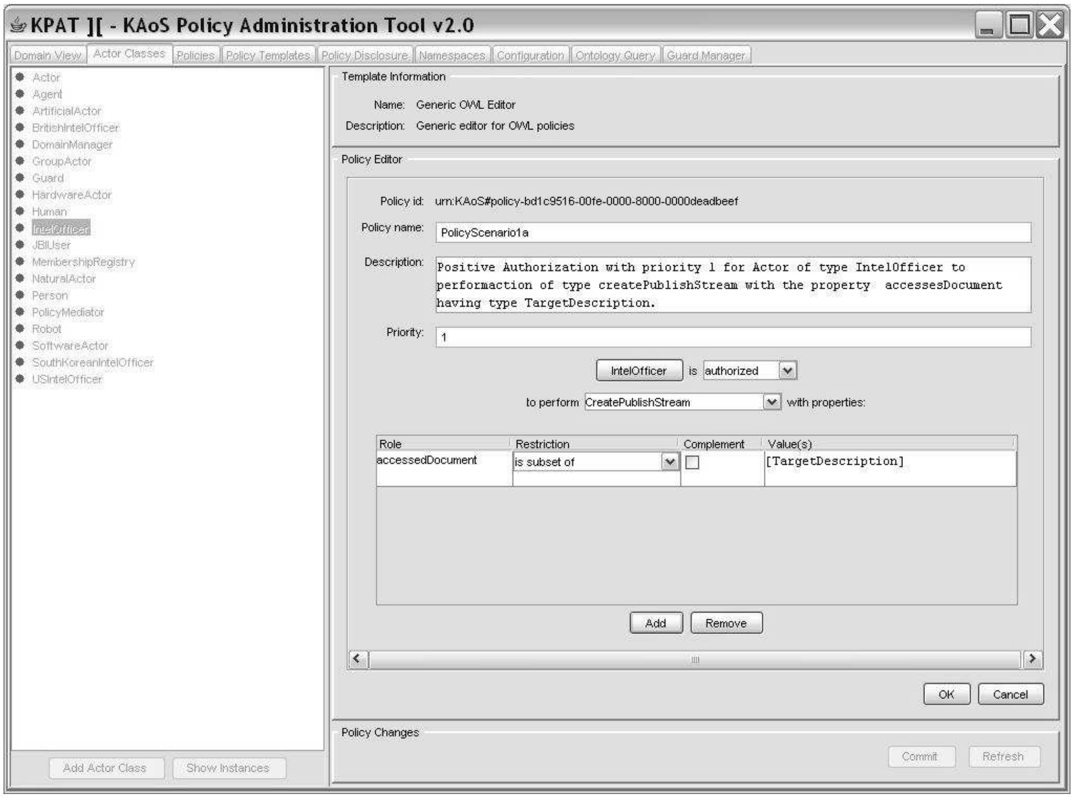
Policy System Bootstrap and Configuration
During its bootstrap, KAoS first loads the KAoS Policy Ontology (KPO) defining concepts used to describe a generic actors' environment and policies within this context (*http://ontology.ihmc.us*). Alternatively, a previously saved configuration containing namespaces, policies, etc. can be loaded. Such a configuration can be saved at any time during system execution, to preserve the created policies and ontology definitions.

Ontology Namespace Browsing and Management:
KPAT allows for browsing of loaded ontologies: examining their content; classes, properties, instances and imported namespaces. It also allows dynamically adding new ontologies on the fly extending concepts from the generic ontology, with notions specific to the particular controlled environment.

Domain and Actor Class Creation
KAoS allows for different ways of expressing the subject of the policy, either through explicit domain membership or implicitly by values of properties defined as ontological actor classes. Alternatively polices can be defined to for abstract classes (roles) but for concrete instances of actors. Dynamic creation of concepts is possible both programmatically and through KPAT.

Policy Creation
Again policies can be created using API or KPAT GUI. The tool guides a user through a creation process using ontology defined ranges to always narrow user choices to the most appropriate set of values; only these valid in the given context. The OWL encoding of the created policy is generated using popular and well supported by HP Lab Jena toolkit (*http://www.hpl.hp.com/semweb/jena.htm*), however the complexity of this representation is hidden from the users and in great degree from the connected applications.

Policy Distribution
After a policy is created it has to be distributed, through the KAoS Directory Service, to Guards, which are policy decision points located close to the running entities; usually a Guard is associated with a Java VM. In the process of policy distribution first the Description Logic [7] classification algorithm is use to find out if given policy should be distributed to a particular Guard controlling given actors (KAoS has to determine if they potentially fall into the scope of the given policy). Then the policy is translated from the OWL format to more efficient form of hash tables and the necessary subsumption results on its concepts (that means subclasses and subproperties) are cached into the policy packaged send to the Guard in order to make the policy decision process efficient.



- Actor
- Agent
- ArtificialActor
- BritishIntelOfficer
- DomainManager
- GroupActor
- Guard
- HardwareActor
- Human
- IntelOfficer
- JBIUser
- MembershipRegistry
- NaturalActor
- Person
- PolicyMediator
- Robot
- SoftwareActor
- SouthKoreanIntelOfficer
- USIntelOfficer

Template Information

Name: Generic OWL Editor

Description: Generic editor for OWL policies

Policy Editor

Policy id: urn:KAoS#policy-bd1c9516-00fe-0000-8000-0000deadbeef

Policy name: PolicyScenario1a

Description:

```
Positive Authorization with priority 1 for Actor of type IntelOfficer to
performaction of type createPublishStream with the property  accessesDocument
having type TargetDescription.
```

Priority: 1

IntelOfficer is authorized

to perform CreatePublishStream with properties:

Role	Restriction	Complement	Value(s)
accessedDocument	is subset of	□	[TargetDescription]

KPAT interface - policy creation

Policy Disclosure

When many policies have been defined, policy disclosure (i.e., finding out which policies apply to a given situation) becomes complex. Similarly to the policy distribution mechanism, policy disclosure exploits the Description Logic classification inferencing. KAoS functionality in this area is not limited to the determination if a given action is authorized or forbidden or if it triggers some obligations but also allows for an exploration of policy related options. For instance; it is possible to find values which the given property of a partially specified action is allowed to take. The appropriate API is available to be used in the code, so an entity can describe its indented action using ontological concepts and submit it to the KAoS Guard for a policy decision. In addition, KPAT contains the graphical interface that allows building an action definition hypothetically performed by a given actor and also allows testing, from different perspective, how this action would be affected by the policies. What is important is that the complexity of the OWL policy syntax and interaction of numerous policies in the disclosure process is transparent for the endpoint applications.

Policy Analyses

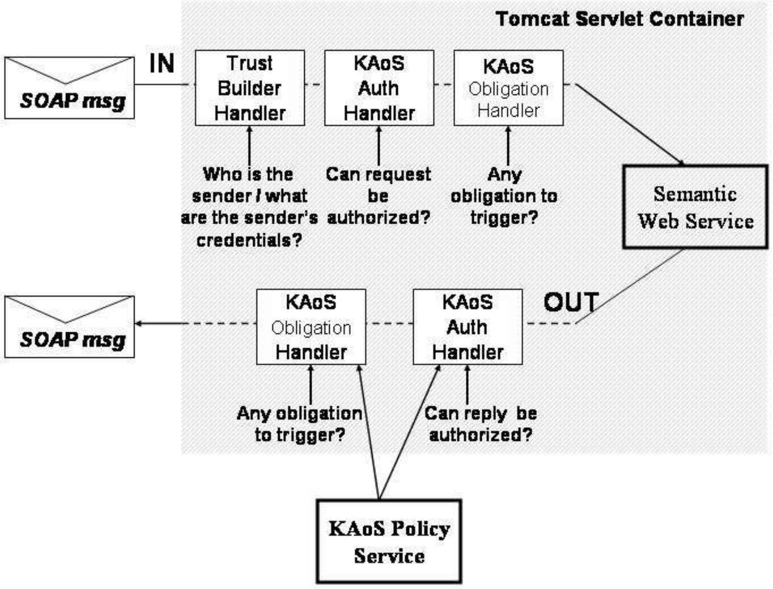
Policies being introduced by many users and at different time or as an effect of merger of two policy sets can and usually are involved in unexpected and unintended conflicts or overlaps. KAoS allows to analyze policies' interactions and, if necessary, to modify them. The Description Logic subsumption reasoning is used to find relations between classes of situations and actions controlled by different policies in order to determine if they are in conflict. The KPAT graphical interface for this functionality is available.

Policy Situation Awareness

A policy system in order to make a decision has to be aware about the situation context in which it takes this policy decision. The system has to know and be able to classify subject of the action, type of the action, its properties and their values. KAoS allows a system to register different entities important from the perspective of the policy decision, such as actors, users, streams, documents, etc. KAoS also register history of some actions, for instance attempts to open a publish streams as it can important from some policy perspective. Still of course, it is not possible to build a system, which would be capable to understand any situation. What is however possible is to build a well-design framework allowing for easy extension of the generic functionality with specialized classifier and other component. KAoS is developed with this idea in mind. For instance it is possible to add a classifier which has the knowledge about a particular message format and is able to determine if a particular message falls in a particular category. This information is then used by a generic KAoS policy disclosure algorithm.

3. KAoS Semantic Web Services integration

The figure below presents elements controlling a SOAP request before it reach its target Semantic Web Service running as a servlet within Tomcat. Likewise, the SOAP response is similarly controlled.



The first handler on the way of SOAP request entering Tomcat is the TrustBuilder handler which negotiates initial credentials of a client and registers the client within the local KAoS in specified domains and with negotiated actor classes. This will allow KAoS to classify the client and apply appropriate policies to its actions. This handler can also renegotiate the trust with the client in case some of its requests are rejected. As an effect the client registration within KAoS changes so different policies will apply to its actions.

The next are two KAoS handlers that use WSDL information about the called method in the SOAP message to find out in the annotated WSDL file available through the included in the message URL the reference to the OWL-S ontology defining this service. This allows building the ontologically annotated description of the request and asking KAoS about its policy decision regarding this request. The process of translation between OWL-S ontology and KAoS Policy ontology is presented in [11].

First, the authorization handler checks if the given request is allowed. Then if request passed the authorization handler, the obligation handler checks if the request triggers some additional obligations. It is assumed that these obligations refer to some Web Services which will be called by the handler. For instance a policy may require consultation or registration of performed transactions in some logging service available on the Web.

Software Availability

The web site: *http://ontology.ihmc.us/* contains the OWL ontology used by KAoS Policy Service.
The KAoS GUI – KPAT can be currently run from the Web using the Java Web Start technology from the following link: *http://norma.coginst.uwf.edu:8080/coalition/KPAT-TCP.jnlp*.

Acknowledgements

This material is based on research sponsored by the Defense Advanced Research Projects Agency (DARPA) CoABS, DAML, and Ultra*og programs the US Air Force Research Laboratory under agreement numbers F30602-00-2-0577 and F30602-03-2-0014. Thanks to the other members of the KAoS project team: Renia Jeffers, Gianluca Tonti, Maggie Breedy, Larry Bunch, Matthew Johnson, Hyuckchul Jung, Shri Kulkarni, James Lott and William Taysom. We are also grateful for the contributions of Lars E. Olsen, Austin Tate, Jeff Dalton, Mark Burstein, Pat Hayes, Luc Moreau, Niranjan Suri, Paul Feltovich, Richard Fikes, Jessica Jenkins, Bill Millaar, Deborah McGuinness, Rich Feiertag, Timothy Redmond, Rebecca Montanari, Sue Rho, Ken Ford, Mark Greaves, Jack Hansen, James Allen, and Robert Hoffman.

References

[1] Ashri, R., Payne, T. R., & Surridge, M. (2004). Towards a Semantic Web Security Infrastructure. *AAAI Spring Symposium on Semantic Web Services*. Stanford University,

[2] Bradshaw, J. M., Uszok, A., Jeffers, R., Suri, N., Hayes, P., Burstein, M. H., Acquisti, A., Benyo, B., Breedy, M. R., Carvalho, M., Diller, D., Johnson, M., Kulkarni, S., Lott, J., Sierhuis, M., & Van Hoof, R. (2003). Representation and reasoning for DAML-based policy and domain services in KAoS and Nomads. *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference (AAMAS 2003)*. Melbourne, Australia, New York, NY: ACM Press,

[3] Damianou, N., Dulay, N., Lupu, E. C., & Sloman, M. S. (2000). *Ponder: A Language for Specifying Security and Management Policies for Distributed Systems, Version 2.3*. Imperial College of Science, Technology and Medicine, Department of Computing, 20 October 2000.

[4] Denker, G., Kagal, L., Finin, T., Paolucci, M., & Sycara, K. (2003). Security for DAML Web Services: Annotation and Matchmaking. In D. Fensel, K. Sycara, & J. Mylopoulos (Ed.), *The Semantic Web—ISWC 2003. Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, USA, October 2003, LNCS 2870.* (pp. 335-350). Berlin: Springer.

[5] Fensel, D., Hendler, J., Lieberman, H., & Wahlster, W. (Ed.). (2003). *Spinning the Semantic Web.* Cambridge, MA: The MIT Press.

[6] Kagal, L., Finin, T., & Joshi, A. (2003). A policy-based approach to security for the Semantic Web. In D. Fensel, K. Sycara, & J. Mylopoulos (Ed.), *The Semantic Web—ISWC 2003. Proceedings of the Second International Semantic Web Conference, Sanibel Island, Florida, USA, October 2003, LNCS 2870.* (pp. 402-418).: Springer.

[7] Li, N., Grosof, B. N., & Feigenbaum, J. (2003). Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information Systems Security (TISSEC)*, 1-42.

[8] McIlraith, S. A., Son, T. C., & Zeng, H. (2001). Semantic Web Services. *IEEE Intelligent Systems*, 46-53.

[9] Tonti, G., Bradshaw, J. M., Jeffers, R., Montanari, R., Suri, N., & Uszok, A. (2003). Semantic Web languages for policy representation and reasoning: A comparison of KAoS, Rei, and Ponder. In D. Fensel, K. Sycara, & J. Mylopoulos (Ed.), *The Semantic Web—ISWC 2003. Proceedings of the Second International Semantic Web Conference, Sanibel Island, USA, 2003, LNCS 2870.* (pp. 419-437). Berlin: Springer.

[10] Uszok, A., Bradshaw, J. M., Jeffers, R., Suri, N., Hayes, P., Breedy, M. R., Bunch, L., Johnson, M., Kulkarni, S., & Lott, J. (2003). KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. *Proceedings of Policy 2003.* Como, Italy.

[11] Uszok, A., Bradshaw, J. M., Jeffers, R., Johnson, M., Tate, A., Dalton, J., & Aitken, S. (2004). Policy and Contract Management for Semantic Web Services. IEEE Intelligent Systems, July/August, p. 32-41,