Journal of Universal Computer Science, vol. 27, no. 12 (2021), 1300-1324
submitted: 11/2/2021, accepted: 9/8/2021, appeared: 28/12/2021 CC BY-ND 4.0

Big Data between Quality and Security: Dynamic Access Control for Collaborative Platforms

Mohamed Talha
(Cadi Ayyad University, National School of Applied Sciences, Marrakesh, Morocco
https://orcid.org/0000-0002-1586-3074, mohamed.talha@icloud.com)

Anas Abou El Kalam
(Cadi Ayyad University, National School of Applied Sciences, Marrakesh, Morocco
https://orcid.org/0000-0001-7714-4801, a.abouelkalam@uca.ac.ma)

Abstract: Big Data often refers to a set of technologies dedicated to deal with large volumes of data. Data Quality and Data Security are two essential aspects for any Big Data project. While Data Quality Management Systems are about putting in place a set of processes to assess and improve certain characteristics of data such as Accuracy, Consistency, Completeness, Timeliness, etc., Security Systems are designed to protect the Confidentiality, Integrity and Availability of data. In a Big Data environment, data quality processes can be blocked by data security mechanisms. Indeed, data is often collected from external sources that could impose their own security policies. In many research works, it has been recognized that merging and integrating access control policies are real challenges for Big Data projects. To address this issue, we suggest in this paper a framework to secure data collection in collaborative platforms. Our framework extends and combines two existing frameworks namely: PolyOrBAC and SLA-Framework. PolyOrBAC is a framework intended for the protection of collaborative environments. SLA-Framework, for its part, is an implementation of the WS-Agreement Specification, the standard for managing bilaterally negotiable SLAs (Service Level Agreements) in distributed systems; its integration into PolyOrBAC will automate the implementation and application of security rules. The resulting framework will then be incorporated into a data quality assessment system to create a secure and dynamic collaborative activity in the Big Data context.

Keywords: Big Data Quality and Security, Collaborative Platform, Dynamic Access Control, IoT, PolyOrBAC, SLA-Framework, Smart City, WS-Agreement
Categories: C.2.4, I.2.1, K.6.5, L.4.0
DOI: 10.3897/jucs.77046

1 Introduction

Internet of Things (IoT) often refers to a growing number of objects connected to the Internet making it possible to bring together new masses of data on the network and therefore, new knowledge. The term IoT has a universal character to designate connected objects with various uses that can constitute Smart Environments. Smart City is a known example of smart environments which refers to an urban area using connected objects and Information and Communication Technologies (ICT) to collect and analyze data with the aim of improving the quality of urban services and optimizing their costs. The exchange of data between the various actors of a smart city constitutes a collaborative environment hosting very large volumes of data. IDC (International

Data Corporation) predicts that by 2025 the amount of human-produced data will reach 163 ZettaByte (1 ZB = 10^{21} bytes), of which more than 95% will be created by connected objects [Reinsel, 17]. Big Data is a direct consequence of this “digitization” of human life and leads to a new era of technologies dedicated to the storage, analysis, transmission and exploitation of these large volumes of data. However, these new technologies have brought new challenges and problems, especially in terms of data quality and security.

Many definitions of the term "Big Data" refer to the **5V** model to denote **V**olume (huge amounts of continuously growing data), **V**elocity (data produced and processed at very high speed), **V**ariety (heterogeneity of data and their sources), **V**eracity (consistency and reliability of data) and **V**alue (benefits that can be attained from this large amount of data). Data Quality is therefore an essential aspect for any Big Data project. A collaborative environment generally constitutes a network of organizations (or actors) in which each can be supplier or consumer of data. This supplier / consumer relationship is governed by an Access Agreement (or contract) made up of a set of functional and non-functional requirements governing the conditions of access to data between organizations. This calls for thinking about dynamic access control systems to serve these types of environments. Access Agreements, on the other hand, must be managed automatically, using a Machine to Machine concept, throughout their lifecycle (from negotiation to monitoring).

In [Talha, 19a], we explained how, in the Big Data context, Data Quality Management System and Data Security System can be conflicting. Indeed, the assessment and improvement of data quality may require read and write access to data collected from multiple external sources that could impose their own security policies. This can lead to conflicting situations: should certain security tolerances be allowed to simplify data quality management, or, should we remain firm from a security point of view but at the expense of data quality? The implementation of automatic systems to integrate and merge the security policies associated with the data represents a major challenge in Big Data [Bertino, 15]. In [Talha, 19b], we addressed this problem and listed a set of challenges for Data Quality and Data Security. We also proposed in [Talha, 20] an approach for assessing data quality in the Big Data context through the accuracy dimension. Our model abstracts from the data security system and considers that the data quality assessment process can access all the data it needs from an organization's data lake. Obviously, this cannot be accepted in a professional context. We will see through this article how to solve this problem.

The objective of this paper is, first of all, to demonstrate the purpose and ways to extend and combine two existing solutions: the PolyOrBAC Framework, that we have proposed in [Abou El Kalam, 07] and [Abou El Kalam, 09], and the WS-Agreement Specification [Andrieux, 04] through its implementation, the SLA-Framework [SLA-Framework, 16]. The PolyOrBAC Framework is intended for collaborative systems and enables the expression of local access control policies, for any organization in a collaborative environment, as well as collaboration rules involving several organizations. PolyOrBAC suffers from a major limitation: the negotiation and creation of access agreements is a "static" process carried out upstream by the actors involved in the collaborative activity. As for the WS-Agreement Specification, it is currently the standard in terms of contracts established between suppliers and consumers of web services. This specification defines a protocol and language for dynamically negotiating, renegotiating, creating and monitoring access agreements in distributed systems. In this paper, we propose to extend and combine the PolyOrBAC Framework and the SLA-Framework in order to automate the mechanism of managing access agreements throughout their life cycle and thus dynamically control access in collaborative platforms. We will then use our Data Accuracy Assessment Process, proposed in [Talha, 20], to call on this new framework which will make it possible to secure the data collection from external sources.

The rest of this article is organized as follows. Section 2 is devoted to the review of certain related works, in particular those extending the PolyOrBAC Framework to make access control dynamic in collaborative platforms or even those using WS-Agreement to integrate security aspects into SLA contracts. Section 3 presents the PolyOrBAC Framework and the WS-Agreement Specification in order to better understand the purpose of this work. Section 4 is dedicated to the presentation and illustration, through a case study on a Smart City, of the new automated PolyOrBAC Access Negotiation Protocol. Section 5 demonstrates how we extended the SLA-Framework to be able to cover the PolyOrBAC Framework requirements. Section 6 presents our secure framework to assess data accuracy in the Big Data context. We then discuss this research work in section 7 in order to highlight its pros and cons. Finally, we conclude and present our future work in section 8.

2 Literature Review

Collaborative environments are characterized by the involvement of several organizations in an activity to achieve a common goal. This collaborative activity is generally governed by an agreement between partners on a set of constraints such as the resources to be shared, the duration of the contract, security rules, etc. The organizations participating in this type of activity are often independent of each other and the users, unknown in advance by the system offering the services, may at any time engage in malicious behavior (at the start or during the collaborative activity). In addition, the competition that may exist between these organizations may encourage some to take malicious action. Consequently, to protect the collaborative platform, the sharing of data and resources must be based on restriction rules, thus forming an interoperability security policy.

To achieve this objective, among the most discussed solutions in literature is the establishment of a "trust" management system between the entities (organizations and/or users) involved in a collaborative activity. Usually, establishing a trust management system involves calculating a "trust score" based on a set of measurable criteria. Trust can be assessed differently depending on the research context and the applied trust criteria; it can be a binary value (specify if the entity is trustworthy or not), multiple values (e.g. "very low", "low", "medium", "high") or a continuous value between 0 and 1. Trust can be applied in different directions: either it is the consumer who applies a trust management system, or the reverse, or in both directions. In some cases, it is the consumer who must ensure that the service provider has a level of trust that guarantees the quality and security of the data. To do this, the calculation of the trust score is based on criteria such as reliability, availability, safety, integrity and maintainability [Sun, 11], recommendations from other data consumers [Habib, 11] as well as compliance with quality of service (QoS) clauses [Saleh, 14]. In other cases, it is the data provider who sets up a trust management system to secure external access.

This is the case of multi-organization environments. The most widely used trust criteria to protect against malicious actions are the "reputation" of external entities and the "recommendations" of other actors in the collaborative system. Reputation represents the degree of compliance with the security rules imposed by data providers; it is typically measured by the behavior and transaction history of users.

Many approaches reuse or extend the OrBAC model [Abou El Kalam, 03] or the PolyOrBAC Framework in order to implement a trust management system. The OrBAC model provides the ability to define permissions, obligations and prohibitions depending on the access context of a given organization. By applying a clear separation between concrete entities (subject, action and object) and abstract entities (role, activity and view), the OrBAC model offers the possibility for different organizations to implement their security policies independently of each other. However, OrBAC has a limit regarding collaborative systems; it does not manage external access. This limit has been resolved thanks to the PolyOrBAC Framework which, through web services and access agreements pre-negotiated between providers and consumers of web services, allows a subject to access the data of an organization to which he/she does not belong. However, OrBAC and PolyOrBAC do not explicitly manage "trust" between the different entities of a collaborative platform. In [Ait Aali, 15], the authors propose the Trust-PolyOrBAC model which integrates a Certification and Authentication Authority (CAA) with which any entity wishing to access the services of other entities must authenticate to obtain an identifier that it will use in its interactions with other entities. The CAA certifies each new organization in the collaborative system and records all types of transactions and historical configurations between organizations. Any interaction between two organizations is based on a "trust score" that must exceed a threshold predetermined by the CAA. This trust score is calculated based on a set of parameters, namely the satisfaction of organizations after each transaction and the reputation of the organizations solicited for their recommendations. In [Belbergui, 16], the authors extend the OrBAC model by introducing a Trusted Third Party (TTP) to control interactions between access requesters and the cloud platforms hosting the requested resources. In their approach, the authors are inspired by the French system for the management of driver's license points; each new user is assigned an initial credit of 10 points by the TTP and all his/her behavior is recorded in an audit log. The user is penalized for each instance of malicious behavior, losing a certain number of points depending on the severity of the breach of the security policy. On the other hand, if his/her behavior remains correct for a certain time and after a certain number of non-malicious uses of the system, he/she will be rewarded and collect some points. When calculating the access decision in OrBAC, two users with the same role do not necessarily have the same access rights; security rules are enabled or disabled based on trust settings. Based on the same principle, the TOrBAC [Ben Saidi, 12] and Multi-Trust_OrBAC [Ben Saidi, 13] models integrate a confidence index into the rules of the OrBAC model. The approach is that, for each access request, the user will first have to connect to the TTP to create a session and get his/her up-to-date confidence index. This index is initialized by the trust manager and assigned to each new user. Then, after each attempt by a user to violate the security policy, his/her confidence index will be decremented by the TTP. Finally, the Trust-OrBAC model [Toumi, 12] assesses trust using the notion of "situation" which corresponds to the desire to carry out an activity on a view and the notion of "time" represented by a time interval during which the assessment trust for an entity does not change. The Trust-OrBAC model extends

OrBAC by assigning, by the system administrator, a trust class to each role. Users and organizations also belong to trust classes calculated based on the history of their behavior. This allows, during the evaluation of an access request, to perform mapping allowing calculating the roles that can be assigned to the access requester.

All the works presented so far attempt to manage the security of collaborative activities by integrating the notion of "trust" by extension of the OrBAC model or the PolyOrBAC Framework. In our opinion, since the expression of a security rule in OrBAC is based on one, and only one organization, it will not be possible to adopt OrBAC to manage access from one organization to another. This is the case, for example, with Saidi et al. who first proposed the TOrBAC model [Ben Saidi, 12] to manage the confidence index and then extended it in Multi-Trust_OrBAC [Ben Saidi, 13] to cover multi-organization environments. In addition, research extending the PolyOrBAC Framework manages to establish, at a certain level, a dynamic approach to manage data security through the concept of "trust" but this remains insufficient for collaborative environments such as Grid Computing, Infrastructure as a Service (IaaS) and more for Smart Environments. If we take PolyOrBAC as an example, collaborative activities are governed by "static" agreements negotiated and concluded beforehand between the entities. These entities must therefore have contact before the actual collaborative activity takes place. However, in a smart environment, with a large number of objects connected and unknown to each other, this static approach may not be suitable. The resolution of this problem consists of (*i*) automating the negotiation of security rules and (*ii*) integrating security rules into specific clauses of SLA agreements. In accordance with this principle, Stankov et al. [Stankov, 12] argue that SLAs can be seen as an instrument to build "trust" between providers and consumers of cloud computing services, especially at the initial stage of collaborative activity, before a relationship is formed.

One of the most promising solutions to achieve this goal is the WS-Agreement Specification. Ludwig et al. [Ludwig, 06] were the first to use this Specification to negotiate SLAs paving the way for the use of distributed resources in shared environments. WS-Agreement offers a reliable mechanism to create solid electronic agreements between different entities interested in the establishment of dynamic collaborations which include mutual obligations, thus making it possible to fill the existing gap when trying to establish a relationship of trust between these entities [Ludwig, 06]. In this context, Smith et al. [Smith, 07] present a Web Service oriented approach, based on the WS-Agreement Specification, allowing the implementation of a fine-grained security configuration mechanism according to the requirements provided by a user. The use of the WS-Agreement makes it possible to negotiate traditional service parameters such as the configuration of resources and the quality of service, as well as security parameters such as the level of encryption and sandboxing. When a customer wants to submit a job or use a service, he/she can create an instance from a WS-Agreement template to specify his/her exact security and performance needs. The WS-Agreement Specification can also be combined with attribute-based access control models to allow service providers and consumers to specify their security policies in both directions (since the commitments are mutual). Li et al. [Li, 16] propose integrating the attributes and their values in an SLA contract through OrBAC rules. Their approach is that, through the WS-Agreement Specification, IaaS providers and consumers exchange offers and counter-offers until an agreement is reached regarding the level of service as well as the security requirements associated with the infrastructure. Thus, when applying the allocation of resources, the broker, who is the intermediary between the provider and the consumer of the services, takes into consideration not only the classic requirements found in an SLA (QoS, guarantees, etc.) but also the security rules to protect the various components of the infrastructure.

However, these approaches do not explicitly implement the WS-Agreement Specification up to the level of elimination of any human intervention when discovering the services, negotiating the terms of the agreement, validating and signing of the SLA. The solution that we propose in this article makes it possible to meet all these requirements by extending the WS-Agreement Specification, via the SLA-Frameworks. Thus, our solution can be operated in a smart environment in which connected objects can conclude access agreements. The explicit integration of the trust management system through criteria such as reputation and recommendation will be an additional functionality which will be taken into account in a future work.

3 Background

3.1 OrBAC (Organization Based Access Control)

Security policies specify the access authorizations to passive entities (objects) by active entities (subjects) and regulate the actions carried out on an information system. In [Abou El Kalam, 03], we proposed the OrBAC model which defines a set of concepts that enables the specification of multiple security policies within an organization. The concept of "*Organization*" in OrBAC can be seen as a structured group of active entities, i.e. "*Subjects*", each playing certain "*Roles*". The concept of "*Object*" mainly represents non-active entities such as files, emails, web services, etc. Since it is necessary to structure objects and add new ones to the system, the OrBAC model introduces the concept of "*View*". Intuitively, a view corresponds to a set of objects that satisfy a common property. Moreover, in OrBAC, the concept "*Action*" mainly encompasses computer actions such as Read and Write. Just as roles and views are abstractions of subjects and objects, OrBAC defines the "*Activity*" as an abstraction of actions. Finally, OrBAC defines the "*Context*" to specify the concrete circumstances in which organizations grant permissions to perform activities on views.

Regarding relationships between the previous concepts, OrBAC defines the relationship "*Permission*" between organizations, roles, views, activities and contexts: if 'org' is an organization, 'r' is a role, 'a' is an activity, 'v' is a view and 'c' is a context, then the relationship "*Permission(org, r, v, a, c)*" means that the organization 'org' grants the role 'r' the permission to perform the activity 'a' on the view 'v' in the context 'c'. Relationships "*Prohibition*", "*Obligation*" and "*Recommendation*" are defined following the same principle. At the concrete level, access control must make it possible to describe the concrete actions carried out by the subjects on the objects. In order to model concrete permissions, OrBAC introduces the "*Is_Permitted*" relation between subjects, objects and actions: if 's' is a subject, 'α' is an action and 'o' is an object, then "*Is_Permitted(s, α, o)*" means that the subject 's' has the permission to perform the action 'α' on the object 'o'. Relationships "*Is_Prohibited*", "*Is_Obligated*" and "*Is_Recommended*" are defined in the same way.

3.2 PolyOrBAC Framework

OrBAC has certain limitations for collaborative systems. Indeed, it is not possible to represent rules that involve several independent organizations, or even autonomous sub-organizations of a particular collaborative system. It is therefore impossible to associate permissions with users belonging to other partner organizations. To get around this limitation, we have proposed the PolyOrBAC Framework [Abou El Kalam, 07] [Abou El Kalam, 09] based on the OrBAC model. Intended for platforms collaborating via web services, this framework consists of two main phases:

- **Negotiation of Collaborative Access Rules**: any organization wishing to expose data begins by developing the necessary web services. Then it declares them in the UDDI (Universal Description Discovery and Integration) registry so that they are accessible to other organizations. Any organization interested in a web service found in the UDDI contacts its supplier in order to negotiate and reach a common access agreement. These organizations define the security rules for using the web service; these security rules are then stored in the provider's PAP database (Policy Administration Point in the XACML sense [XACML, 13]). A PolyOrBAC security rule consists in considering, by the web service provider, "virtual" subjects to represent "real" subjects coming from an external organization. To better understand this rule, let's take an example: suppose we have a web service called ‘ws’ developed by the Org1 organization. Access to ‘ws’ is restricted to users of Org1 who have the role ‘r’. Suppose there is an access agreement between Org1 and another Org2 organization interested in ‘ws’. The collaboration rule consists in that Org1 adds in its PAP database a "virtual" subject, for example ‘partner 1’, which has the role ‘r’. Any user from Org2 wishing to call ‘ws’ will have to prove to Org1 that he/she is authorized by Org2 to be embodied by the virtual subject ‘partner 1’.
- **Access to Collaboration Web Services**: any subject coming from an external organization (web service consumer) must present an encrypted and signed ticket proving that he/she is authorized by his/her organization to represent a "virtual" subject allowing him/her to call a web service of the provider. The access request is then checked by the organization providing the web service, through its PEP (Policy Enforcement Point in the XACML sense), in accordance with its security policy and the agreement established with the consumer.

PolyOrBAC has the advantage of being a framework that can be extended so that each organization can adopt the access control model that suits it best. Collaboration activity is provided through calls to web services by external users as part of an access agreement. However, PolyOrBAC has a major drawback: the negotiation and creation of access agreements must be automated. Indeed, nowadays we cannot accept human interventions in negotiating access between connected objects. The integration of the WS-Agreement Specification into the PolyOrBAC Framework seems to address the automation of access agreement negotiations. In the next section, we will briefly present an overview of this specification.

3.3 WS-Agreement Specification

The term "Negotiation" is generally defined as the process of discussion between two or more parties in order to reach a common agreement defending the interests of each. By analogy, we define access negotiation as a process of exchanging information allowing data providers and consumers to negotiate a set of terms in order to reach a common agreement framing all the modalities of access to the data. Automating negotiations requires formalizing the definition of each term of the access contract so that machines can understand them. Much work has been done in recent years to automate contract negotiations in various fields such as Internet of Things ([Marino, 18], [Li, 19a], [Li, 19b]), Cloud Computing ([Shojaiemehr, 19], [Scoca, 17], [Labidi, 17], [Baig, 17]) as well as distributed environments ([Li, 14], [Tseng, 16], [Castro, 15], [Coutinho, 16]).

In a collaborative platform, data providers and consumers operate in a dynamic context governed by a set of rules, conditions, obligations and guarantees formalized in a contract, the SLA. This is a type of electronic contract that describes services and specifies QoS properties that must be maintained by a provider during service provision. These properties define the SLO (Service Level Objectives) which are measurable terms that are monitored throughout the life of the contract. Regarding SLA contract management techniques (negotiation, renegotiation, creation and monitoring), many solutions exist, among which we find:

- **WSLA (Web Service Level Agreement)**: this is a specification created by IBM for the creation and monitoring of SLAs. It defines a flexible and extensible language to allow service consumers and providers to define and specify SLA parameters [Ludwig, 03].
- **WSOL (Web Service Offering Language)**: this XML-based specification allows providers to define multiple service levels for the same service in different instances. The particularity of this approach is that an offer of a service represents a single class of services with a specific QoS. A consumer can therefore search for a desired service and select the desired instance based on the level of service that matches his/her needs [Tosic, 02].
- **WS-Agreement (Web Service Agreement)**: this is a standard of the Grid Resource Allocation and Agreement Protocol Working Group (GRAAP-WG) of the Open Grid Forum (OGF). It is a specification that defines a protocol and language for dynamically negotiating, renegotiating, creating and monitoring two-way SLAs (between consumer and supplier) in distributed systems [Andrieux, 04].

WS-Agreement and its extension WS-Agreement Negotiation [Wäldrich, 11] are the only SLA specifications standardized and accepted by a large community [Marino, 18], [Li, 14]. In addition, to our best knowledge, these are the only solutions that bilaterally manage negotiable SLAs. Therefore, our study is based on the WS-Agreement Specification to automate the negotiation and creation of PolyOrBAC access agreements. Regarding the implementation of the WS-Agreement Specification, we found two main solutions:

- **WSAG4J Framework [WSAG4J, 12]**: this is a generic implementation of the WS-Agreement Specification. This framework supports common functionality to negotiate, renegotiate, create and monitor agreements and allows users to quickly build and deploy web services based on the WS-

Agreement Specification. WSAG4J follows a declarative approach to support and manage the entire lifecycle of an agreement, from the definition of an agreement template, the deployment of models in factories, all the way to the management of the agreement. Unfortunately, this framework was released in its latest version 2.0.0 in 2012 and has not been actively maintained since (expired certificates, no more support, weak community, etc.).

- **SLA-Framework [SLA-Framework, 16]**: this is another implementation of the WS-Agreement Specification. It is an open-source project that enables management of the lifecycle of SLA agreements. Currently in version 1.1, this framework only supports one-shot negotiation. In order to allow a negotiation loop (offers and counter-offers), it will be necessary to improve this functionality of the framework or, to use it as is, to manage exchanges history between the providers and the consumers of the web services. In order to not get stuck at this point, let's assume that an offer made by the consumer can only be accepted or refused by the provider (we dispense with the possibility of counter-offers). We therefore chose this implementation to extend it to cover the needs of the PolyOrBAC Framework.

The WS-Agreement Negotiation Model [Wäldrich, 11] consists of three layers with a clear separation between them:

- **Negotiation Layer**: it provides a protocol and language to negotiate offers and counter-offers. This layer expresses the willingness of both parties to conclude a subsequent agreement.
- **Agreement Layer**: it provides a protocol and language defined in the WS-Agreement Specification to provide the basic functionality to create and monitor agreements.
- **Service Layer**: it defines the services offered by the data provider. The execution of the services of this layer is governed by the Agreement Layer.

WS-Agreement Specification enables XML based exchanges between web service providers and consumers. Two types of XML documents exist: The “Template” pre-filled by the service provider and the “Agreement” upon which both parties have agreed following the negotiation process. An “Agreement” (or a “Template”) is conceptually composed of several sections:

- **Name**: in this section we specify a name and a unique identifier of the agreement.
- **Context**: it contains meta-information about the agreement such as the identifier of the initiator of the agreement and the identifier of the responder of the agreement, the name and identifier of the template which served as the basis for creating the agreement, references to other agreements, the validation period of the agreement, etc.
- **Terms**: the terms of an agreement include service terms (Service Description Terms, Service References and Service Properties) and eventually Guarantee Terms (Service Scope, Service Level Objective and Business Value).

The definition of each of these sections is well detailed in the official documentation [Andrieux, 04]. When implementing our solution, we will come back to the definition of some of these sections, especially for the "Terms" section through which we will extend the SLA-Framework.

4 PolyOrBAC Extension

4.1 PolyOrBAC Access Negotiation Protocol

The WS-Agreement Negotiation Protocol (WSANP) [Wäldrich, 11] is currently the standard used in negotiating SLAs for Web Services [Marino, 18], [Li, 19a]. This protocol defines the services and operations required to negotiate, renegotiate, create and monitor SLAs. Compatible with WSANP, our protocol, illustrated in Figure 1, enables automating the negotiation mechanism of the PolyOrBAC Framework.

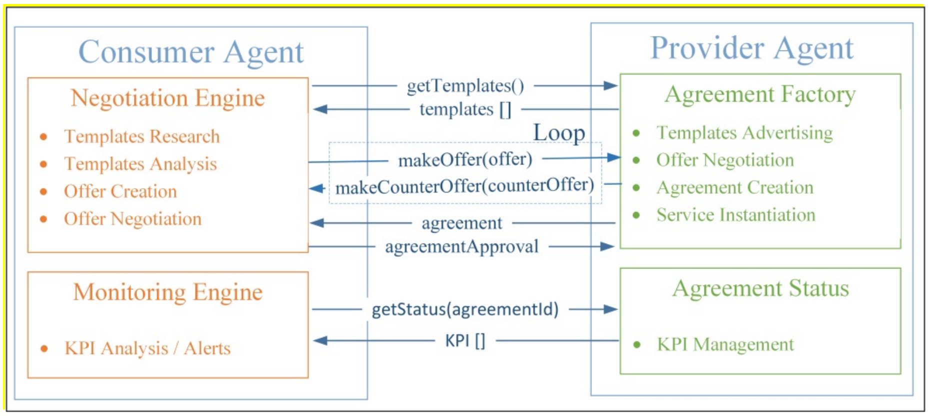


Figure 1: PolyOrBAC Access Negotiation Protocol

A service provider models and publishes templates according to the WS-Agreement Specification in which it describes all its services. Negotiations of access rules and other agreement terms are carried out between agents representing the supplier (*Provider Agent*) and the consumer (*Consumer Agent*) by following these steps:

- **Step 1.** The consumer agent identifies the services whose description matches what interests it and asks the provider agent for the list of templates by calling its public method *getTemplates()*.
- **Step 2.** The consumer agent analyses the list of templates *templates[]* and chooses one from which the final agreement will be created. The template is already initialized by the provider agent, the consumer agent must first fill in the fields for which it is concerned, adapt the access rules for the services it wants and propose the desired values for each SLO. The offer thus constructed is sent to the provider agent by calling upon the *makeOffer(offer)* method.
- **Step 3.** Upon reception of the offer, a validation process is initiated by the provider agent. The validation process consists in verifying the values entered in by the consumer agent by applying the agreement creation constraints predefined in the template and by validating the SLO options desired by the consumer agent. If the offer is valid, the provider agent creates the agreement, signs it and sends it to the consumer agent for approval. Otherwise, it adapts

the values requested by the consumer agent and makes a counter-offer by calling upon the *makeCounterOffer(counterOffer)* method.

- **Step 4.** Upon reception of the counter-offer, the consumer agent can refuse it (end of the negotiation process), adapt it by creating a new offer, or accept it. In these last two cases, the new offer is sent to the provider agent by calling up-on the *makeOffer(offer)* method. Steps 3 and 4 represent a loop of offers and counter-offers between the agents. The number of iterations in the negotiation loop is limited, and if no agreement is reached at the end of the loop, the negotiation process fails. If an offer is accepted by the provider agent, it creates the agreement, signs it and sends it to the consumer agent.
- **Step 5.** Upon reception of the signed agreement, the consumer agent signs its part and sends its approval to the provider agent.
- **Step 6.** Once the agreement has been created and doubly signed, the provider agent updates the service provider's security policy according to the access rules negotiated in the agreement. It also provides the consumer agent with the various KPIs (Key Performance Indicators) necessary for monitoring the agreement.

4.2 Case Study

Thanks to sensors installed in all the streets of a Smart City, an actor (OrgA) manages public parking spaces. The information collected in real time allows this organization to trigger billing or sanctioning processes, perform data analysis, etc. To improve its service, the OrgA organization allows its users (e.g. customers and people with disabilities) to quickly find the nearest free parking space (by calling the *Car_Parking* web service). Concretely, John, a customer of the OrgA organization, can call the *Car_Parking* web service (through the GET method) to quickly park. Bob, a disabled person, can also take advantage of this web service to find places reserved for his situation. In OrBAC, this scenario can be implemented using the following rules:

Permission (OrgA, Customer, Web Services, Find Parking, Parking Management) ∧
Permission (OrgA, Disabled Person, Web Services, Find Parking, Parking Management) ∧
Role (OrgA, John, Customer) ∧
Role (OrgA, Bob, Disabled Person) ∧
View (OrgA, Car_Parking, Web Services) ∧
Activity (OrgA, GET, Find Parking) ∧
Context (OrgA, John, GET, Car_Parking, Parking Management) ∧
Context (OrgA, Bob, GET, Car_Parking, Parking Management) ∧
→ Is_Permitted(John, GET, Car_Parking) ∧
→ Is_Permitted(Bob, GET, Car_Parking)

In addition, as part of a commercial activity, an OrgB organization has purchased the right to use the parking spaces from the OrgA organization. Thanks to the PolyOrBAC Framework, customers of OrgB can also use the *Car_Parking* web service as if they were directly attached to OrgA, provided they pay for the service within 10 minutes. For that purpose, the OrgA and OrgB organizations, via automatic agents, negotiate an access agreement to use the *Car_Parking* web service. Figure 2 illustrates our scenario.

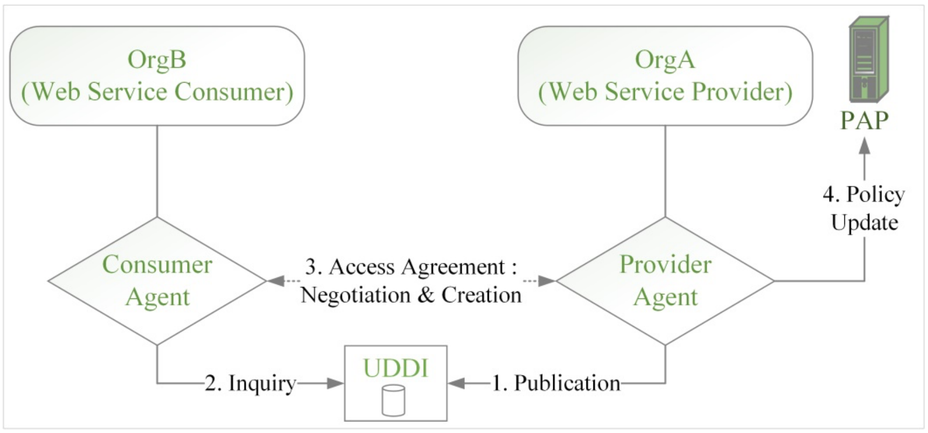


Figure 2: PolyOrBAC – Negotiation of Collaborative Access Agreement

The OrgA organization develops and publishes web services in the UDDI of the collaborative platform through its Provider Agent. The latter is responsible for managing access agreements (publication of templates, negotiation and creation of agreements, etc.). OrgB, on the other hand, has a Consumer Agent that searches the UDDI for web services and initiates the negotiation process for those of interest. After creating and signing the access agreement, the Provider Agent adds the following security rules to its PAP database:

Role (OrgA, Partner_OrgB, Customer) ∧
Context (OrgA, Partner_OrgB, GET, Car_Parking, Parking Management) ∧
→ Is_Permitted(Partner_OrgB, GET, Car_Parking) ∧
→ Is_Obligated(Partner_OrgB, PUT, Pay_Invoice)

This is equivalent to considering the virtual subject *Partner_OrgB* as an OrgA customer with an obligation to pay for the service within 10 minutes (by calling the *Pay_Invoice* web service through the PUT method). To be able to call the *Car_Parking* web service, users of the OrgB organization will have to authenticate with their organization (OrgB) to retrieve a signed and encrypted ticket allowing them the desired access. This ticket, encrypted by the OrgB organization, will be decrypted by the OrgA organization and should contain information such as:

- the organization to which belongs the user (OrgB)
- the virtual subject assigned to the user (Partener_OrgB)
- the identifier of the access agreement (to check its validity)
- the requested web service (Car_Parking)
- the requested action (GET)
- the ticket generation timestamp (to prevent reply attacks)

5 SLA-Framework Extension

The PolyOrBAC Framework specifies security rules through timed automata:

- Permissions, which represent actions authorized by the access agreement clauses, are specified by transitions in timed automata.
- Prohibitions can be implicit or explicit. They are implicit when there is no transition in the automaton that leads to the desired state (i.e. the action requested by the subject). Explicit prohibitions, on the other hand, are specified as a "failed state" that the system will generate if malicious action is detected.
- Obligations are considered to be actions that “must” be carried out. Just like permissions, obligations will be specified by transitions in timed automata. To fulfill the “mandatory” nature of the obligations, each transition is assigned a timeout which is reset to zero if the action is performed before its expiration; otherwise, if the timeout expires, an exception is generated which may result in penalties.

Figure 3 represents a timed automaton specifying the scenario of our case study.

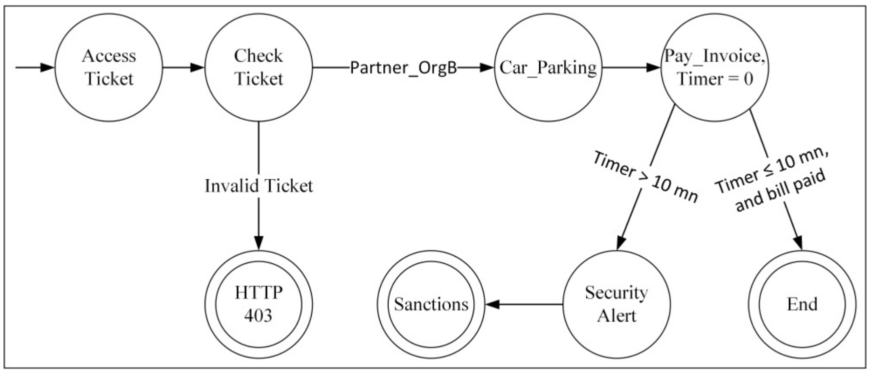


Figure 3: Example of Permission, Prohibition and Obligation specification

In order to extend the WS-Agreement Specification to support the rules of the PolyOrBAC Framework, we will take into account the following definition of an agreement:

```
<wsag:Agreement AgreementId="xs:string">
    <wsag:Name>xs:string</wsag:Name> ?
    <wsag:Context>
        wsag:AgreementContextType
    </wsag:Context>
    <wsag:Terms>
        wsag:TermCompositorType
    </wsag:Terms>
</wsag:Agreement>
```

An agreement, encapsulated in the <*wsag:Agreement/*> tag, is identified by a unique identifier and contains a name <*wsag:Name/*>, a context <*wsag: Context/*> and a collection of terms <*wsag:Terms/*>. Among the terms, there is a Service Description Terms (SDT) section that describes the services exposed by the provider. Here is the definition of this section:

```
<wsag:Terms>
...
<wsag:ServiceDescriptionTerm wsag:Name="xs:string" wsag:ServiceName="xs:string">
   <xs:any> ... </xs:any>
</wsag:ServiceDescriptionTerm> +
...
</wsag:Terms>
```

To integrate the security rules, we propose to extend the WS-Agreement Specification and add a new section "Security Term" which may have the following definition:

```
<wsag:Terms>
...
<xs:SecurityTerm wsag:Name="xs:string" wsag:ServiceName="xs:string">
      <xs:Permissions wsag:Name="xs:string" wsag:ServiceName="xs:string">
            <xs:Transition>
                        <xs:Source>xs:string</xs:Source>
                        <xs:Target>xs:string</xs:Target>
                        <xs:Event>xs:string</xs:Event>
                        <xs:Action>xs:string</xs:Action>?
            </xs:Transition>+
      </xs:Permissions>?
      <xs:Obligations wsag:Name="xs:string" wsag:ServiceName="xs:string">
            <xs:Transition>
                        <xs:Source>xs:string</xs:Source>
                        <xs:Target>xs:string</xs:Target>
                        <xs:Event>xs:string</xs:Event>?
                        <xs:Guard>xs:string</xs:Guard>?
                        <xs:Action>xs:string</xs:Action>?
            </xs:Transition>+
      </xs:Obligations >?
      <xs:Prohibitions wsag:Name="xs:string" wsag:ServiceName="xs:string">
            <xs:Transition>
                        <xs:Source>xs:string</xs:Source>
                        <xs:Target>xs:string</xs:Target>
                        <xs:Event>xs:string</xs:Event>
            </xs:Transition>+
      </xs: Prohibitions>?
</xs: SecurityTerm >
...
</wsag:Terms>
```

Basically, we introduce three security terms: *Permissions*, *Obligations* and *Prohibitions*. Each of these terms is made up of a set of *Transitions*. We consider a Transition to be a link from a source state to a target state when an event or a guard occurs in the Timed-Automata. A Transition may possibly trigger an action (for example, resetting a clock). Our implementation is illustrated in Figure 4.

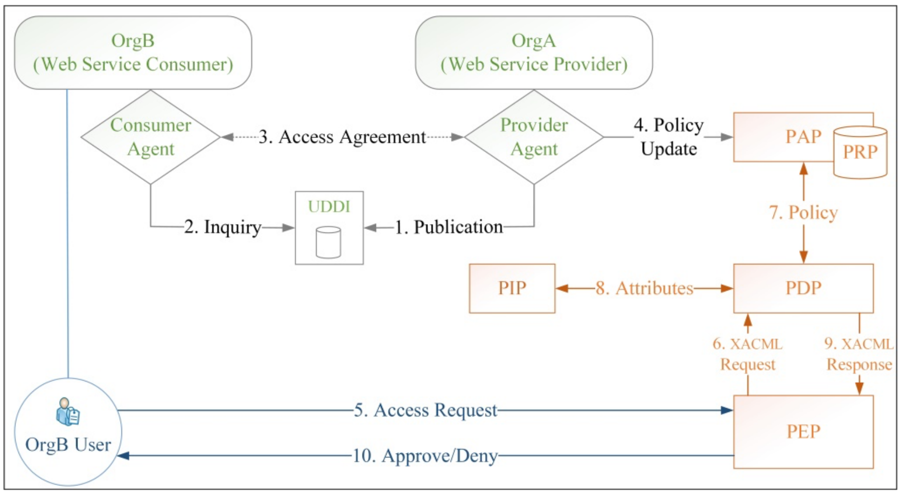


Figure 4: Collaborative Access Framework

Consumer and Provider agents negotiate and create agreements as explained previously in Figures 1 and 2. Once an agreement is created and signed, the Provider Agent updates the provider's PAP database. Often used to ensure the authorization function in SOA architectures, XACML [XACML, 13] is a specification that defines a language for access control, management of security rules and administration of an information system's security policy. In the XACML environment, we mainly have five components:

- **PEP** (Policy Enforcement Point): this is the point of application of the access decision. It is at the PEP level that users request access to a resource.
- **PDP** (Policy Decision Point): this is the engine of the XACML architecture. It is at this level that policies are evaluated and compared against authorization requests.
- **PIP** (Policy Information Point): this is the point where the PDP can connect to external sources of attributes like LDAP or an external database. The idea is that when evaluating a query against a policy, the PDP can, through the PIP, retrieve additional information to make a decision.
- **PRP** (Policy Retrieval Point): this is the storage point for policies. The PRP can be a database or a simple file where policies are stored.
- **PAP** (Policy Administration Point): this is the point of policy administration. This is where access control policies are edited.

The SLA-Framework implements the WS-Agreement Specification according to the official definition [Andrieux, 04]. In order to be able to express the PolyOrBAC security rules, we have extended the SLA-Framework to support the new section SecurityTerm that includes the new rules (Permissions, Obligations and Prohibitions). Each of these classes made up of a list of Transitions. A transition defines different elements (Source, Target, Event, Guard and Action) necessary to add a PolyOrBAC security rule.

Regarding the XACML environment, as OrBAC uses a formal language based on first-order logic, the Prolog Language and the SWI-Prolog Tool seem to meet our needs. Concretely, we have set up four modules (PRP, PAP, PEP and PDP) that we describe in the following.

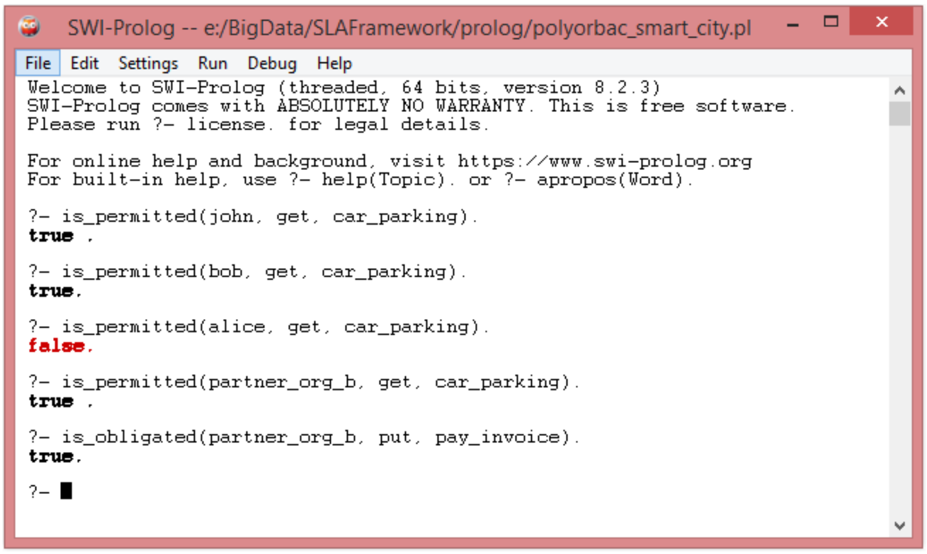
The PRP is the knowledge base initialized with the following facts and rules:

```
permission(org_a, customer, web_services, find_parking, parking_management).
permission(org_a, disabled_person, web_services, find_parking, parking_management).

role(org_a, john, customer).
role(org_a, bob, disabled_person).

view(org_a, car_parking, web_services).

activity(org_a, get, find_parking).

context(org_a, john, get, car_parking, parking_management).
context(org_a, bob, get, car_parking, parking_management).

is_permitted(X, get, car_parking):-
    permission(org_a, customer, web_services, find_parking, parking_management),
    role(org_a, X, customer),
    activity(org_a, get, find_parking),
    view(org_a, car_parking, web_services),
    context(org_a, X, get, car_parking, parking_management).

is_permitted(Y, get, car_parking):-
   permission(org_a, disabled_person, web_services, find_parking, parking_management),
   role(org_a, Y, disabled_person),
   activity(org_a, get, find_parking),
   view(org_a, car_parking, web_services),
   context(org_a, Y, get, car_parking, parking_management).
```

The PAP is a REST web service used for editing the PRP knowledge base. For our case study, when an agreement is created and signed, the provider agent will add the following facts:

```
role(org_a, partner_org_b, customer).
context(org_a, partner_org_b, get, car_parking, parking_management).

is_obligated(Z, put, pay_invoice):-
   obligation(org_a, customer, web_services, find_parking, parking_management),
   role(org_a, Z, customer),
   activity(org_a, get, find_parking),
   view(org_a, car_parking, web_services),
   context(org_a, Z, get, car_parking, parking_management).
```

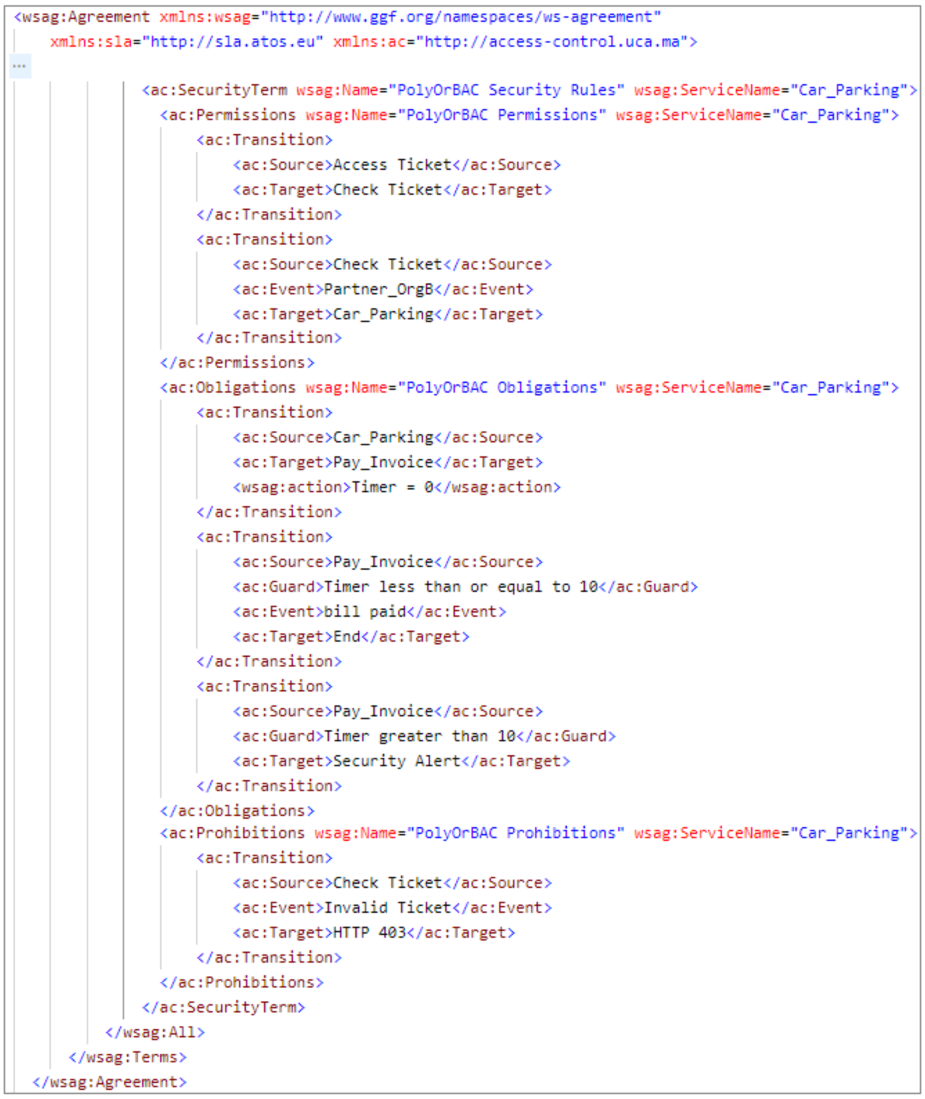
The PEP is also a REST web service used to process user access requests. Upon reception of a request (an encrypted and signed ticket), it decrypts it and proceeds to a first validation step. If the ticket is valid, it retrieves the virtual subject and the desired web service. This information enables it to construct a predicate which it transmits to the PDP. The latter is the inference engine that checks whether the predicate received from the PEP is true or false. For our case study, Figure 5 shows the results of some access requests made by different users (John, Bob and Alice from OrgA) and an external user coming from OrgB with a valid access ticket.


SWI-Prolog -- e:/BigData/SLAFramework/prolog/polyorbac_smart_city.pl
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.3)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- is_permitted(john, get, car_parking).
true .

?- is_permitted(bob, get, car_parking).
true.

?- is_permitted(alice, get, car_parking).
false.

?- is_permitted(partner_org_b, get, car_parking).
true .

?- is_obligated(partner_org_b, put, pay_invoice).
true.

?-
```

Figure 5: PolyOrBAC – Permissions, Prohibitions and Obligations Control

For our case study, Figure 6 shows an extract of the access agreement created by the extended SLA-Framework (according to the automaton of Figure 3).



```
<wsag:Agreement xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
    xmlns:sla="http://sla.atos.eu" xmlns:ac="http://access-control.uca.ma">
...
            <ac:SecurityTerm wsag:Name="PolyOrBAC Security Rules" wsag:ServiceName="Car_Parking">
              <ac:Permissions wsag:Name="PolyOrBAC Permissions" wsag:ServiceName="Car_Parking">
                  <ac:Transition>
                      <ac:Source>Access Ticket</ac:Source>
                      <ac:Target>Check Ticket</ac:Target>
                  </ac:Transition>
                  <ac:Transition>
                      <ac:Source>Check Ticket</ac:Source>
                      <ac:Event>Partner_OrgB</ac:Event>
                      <ac:Target>Car_Parking</ac:Target>
                  </ac:Transition>
              </ac:Permissions>
              <ac:Obligations wsag:Name="PolyOrBAC Obligations" wsag:ServiceName="Car_Parking">
                  <ac:Transition>
                      <ac:Source>Car_Parking</ac:Source>
                      <ac:Target>Pay_Invoice</ac:Target>
                      <wsag:action>Timer = 0</wsag:action>
                  </ac:Transition>
                  <ac:Transition>
                      <ac:Source>Pay_Invoice</ac:Source>
                      <ac:Guard>Timer less than or equal to 10</ac:Guard>
                      <ac:Event>bill paid</ac:Event>
                      <ac:Target>End</ac:Target>
                  </ac:Transition>
                  <ac:Transition>
                      <ac:Source>Pay_Invoice</ac:Source>
                      <ac:Guard>Timer greater than 10</ac:Guard>
                      <ac:Target>Security Alert</ac:Target>
                  </ac:Transition>
              </ac:Obligations>
              <ac:Prohibitions wsag:Name="PolyOrBAC Prohibitions" wsag:ServiceName="Car_Parking">
                  <ac:Transition>
                      <ac:Source>Check Ticket</ac:Source>
                      <ac:Event>Invalid Ticket</ac:Event>
                      <ac:Target>HTTP 403</ac:Target>
                  </ac:Transition>
              </ac:Prohibitions>
            </ac:SecurityTerm>
        </wsag:All>
    </wsag:Terms>
</wsag:Agreement>
```

Figure 6: PolyOrBAC Agreement generated by the extended SLA-Framework

6 A Secure Data Accuracy Assessment Framework

Our framework resulting from the extension and the combination of PolyOrBAC and SLA-Framework makes it possible to automate the management of access agreements and to dynamically control access to collaborative platforms. As part of our research work on Quality and Security of data in the Big Data context, we have proposed in [Talha, 20] a process to assess the accuracy of the data hosted in a data lake. For this, our solution consists in collecting data from different sources. To better explain our approach, we will take up some notions that we have detailed in this previous work:

- **Data Accuracy**: accuracy, one of the main dimensions of Data Quality, reflects the degree to which the data in an information system represents the real world. More formally, let v be the value of a datum in an information system and ν' the corresponding reference value considered as correct; the accuracy of ν represents the degree of similarity between ν and ν'.
- **Reference Data**: measuring the accuracy of data should refer to the real world it models (other data that the system considers as correct, human knowledge where applicable, etc.). Obtaining this reference data is a very complex process, if not impossible in some cases. Arbitrary considerations can often arise such as the opinion of domain experts, trust in information providers and their reputations, etc.
- **Accuracy Criteria**: in practice, data that needs to be evaluated is compared with data collected from a reference source considered sufficiently reliable. Accuracy criteria bring together a set of measurable criteria which makes it possible to assess the reliability of data sources. Some of these criteria correspond to the providers of the information (such as trust and reputation) and others correspond to the data itself (for example, consistency, completeness, timeliness, etc.). Each organization, depending on its needs and before initiating the data quality assessment process, should determine the appropriate accuracy criteria for measuring the reliability of a data source.

The process we proposed consists in five steps:

- **Master Data Set**: this first step consists in collecting data from an organization's information sources which can be internal or external. These data are then stored in their raw state to constitute what we call the Master Data Set.
- **Golden Data Set**: each set of data in the Master Data Set is assigned, in the form of metadata, a level of reliability calculated according to its original source and the accuracy criteria predetermined by the organization. This process is re-executed whenever new data are collected or whenever the accuracy criteria are changed.
- **Mapped Data**: each time we want to assess the accuracy of a data set, which therefore constitutes a new entry in the process, we look for its matches in the Golden Data Set. For structured data, this can be done using Schema Matching algorithms. For semi-structured and unstructured data, this step can be very complex and will require special treatment. Regardless of the nature of data, if the input data does not match with data in the Golden Data Set, the assessing process will end in failure.
- **Reference Data**: through a Record Linkage process, reference data are dynamically constructed each time an assessment process is launched. This step consists in eliminating all the data sets, from the Golden Data Set, that do not meet the minimum level of accuracy criteria required by the organization, resolving conflicts (when multiple data sets exist, the most reliable are selected), reducing the volume of data if necessary (applying Big Data Sampling Algorithms), and, finally, identifying the exact records with which the records to be evaluated can be compared.

- **Data Accuracy**: the last step is to calculate the similarity between the data to be evaluated and the reference data. Different methods exist to measure the similarity between data such as the Levenshtein distance and the Jaro-Winkler distance for strings, the difference between values for numbers taking into consideration a threshold from which we can consider two numbers to be similar, dates and geographic coordinates can be converted to numbers, etc.

Our data accuracy assessment process is based on the premise that data can be easily collected from an organization's data sources, which is generally not the case. Thanks to the extended PolyOrBAC and SLA-Framework, it is now possible to automate the aspects of research and collection of data from data providers. Figure 7 illustrates our new approach.

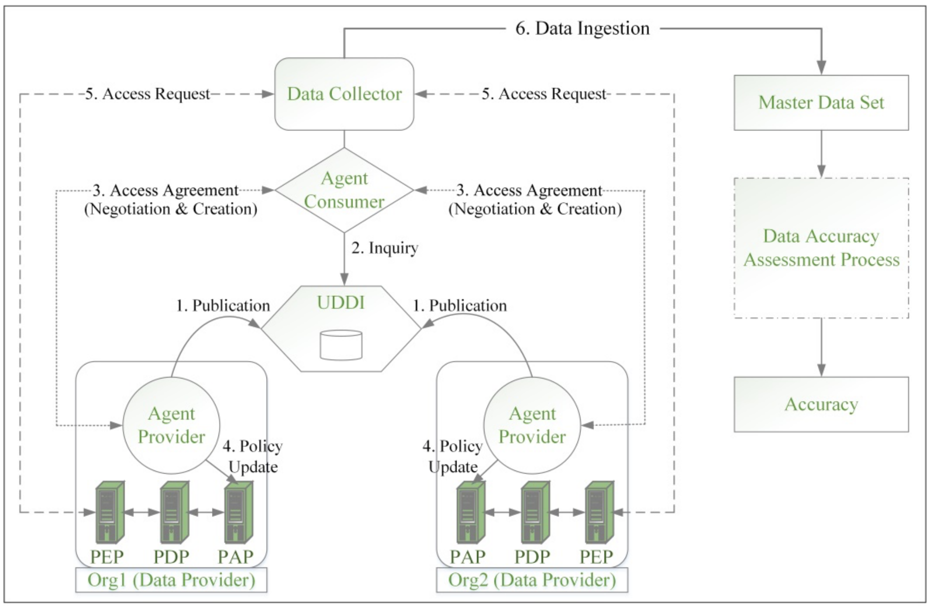


Figure 7: Secure Big Data Accuracy Assessment Framework

Our new approach to securely feed the data lake as part of a collaborative activity is to use the extended PolyOrBAC Framework and SLA-Framework. The idea is that the data providers (Org1, Org2, etc.) publish their web services in the UDDI registry of the collaborative platform via their agents (*Step 1. Publication*). A data collector has a consumer agent that periodically searches the UDDI registry for web services that may be of interest (*Step 2. Inquiry*) and contacts the agent providers to negotiate and create access agreements (*Step 3. Access Agreement*). Each time a new agreement is created and signed, the agent provider updates its organization's security policy based on the XACML architecture (*Step 4. Policy Update*). The data collector can then authenticate with the data providers (*Step 5. Access Request*) to call the web services and thus feed the data lake (*Step 6. Data Ingestion*).

7 Discussion

Today, data is the key element around which many economic and social activities revolve. The Quality and Security of data are two essential processes that promote its use. Big Data environments represent a set of new technologies, models and techniques for collecting, storing, analyzing and extracting values from very large volumes of data. In our research, we look at the conflicts that may exist between Data Quality Management Systems and Data Security Systems. Quality Management is typically done in two phases: assessing data quality (to identify gaps) and then improving data quality (to correct those gaps). Data Quality Assessment may require read access to data collected from different organizations in a collaborative platform (to be able to perform comparison operations) and Data Quality Improvement may require write access (to update obsolete data, delete duplicates, complete missing data, etc.). These two processes can be blocked by security policies and mechanisms. Indeed, the trend today is to centralize data in a single data lake in which all the structures of an organization store their data. This has many advantages for companies (high availability, reduced hosting costs, green IT, etc.) but also poses different challenges, in particular for integrating and merging of security policies. Thanks to the extension of the PolyOrBAC Framework and the SLA-Framework, it has been proven possible to manage heterogeneous data security policies.

However, our solution has certain limitations. First of all, the PolyOrBAC Framework is based on SOAP (Simple Object Access Protocol) which is starting to lose its place to the REST (REpresentational State Transfer) pattern for various reasons such as the exclusive use of XML format. REST, on the other hand, in addition to XML format, is open to various other formats (Plain Text, HTML, and JSON) which are much lighter than XML. It will therefore be necessary to think about reviewing the architecture of the PolyOrBAC Framework to support REST web services. Ideally, it would even be necessary to support other types of communication (remote access, rich notifications, message communications, etc.) and not just remain satisfied with only web services; indeed, it may be possible that recovering data from the same data lake is needed and going through web services will only complicate the processes. In addition to these technical aspects, as we mentioned in the literature review section, the explicit management of trust (through criteria such as Reputation and Recommendations) will further strengthen the security of collaborative platforms and this must imperatively be taken into account in our future work. Regarding the SLA-Framework, as explained above, the current version only supports one-shot negotiation; we have then to think about improving this functionality.

Moreover, regarding the data quality assessment process, we focused on structured data, while statistics today show that the majority of data is unstructured (nearly 95% of the data produced today is unstructured [Adnan, 19], [Seng, 19]). The search for matches in the lake and the calculation of similarities are very complex processes for unstructured data, which possibly requires a step to classify the data according to their nature and suitable mechanisms to assess the quality of the data for each type. Finally, improving data quality may require write access, a requirement that organizations can be firmer on in order to prevent such processes.

8 Conclusions and Future Work

Data Quality Management and Data Security are two systems essential to the effective and efficient exploitation of data, in particular in the Big Data context. In order to avoid mutual conflicts between these two systems, it is often necessary to conduct careful rationalization. In this paper, we have recalled that read / write access to an organization's data, to manage the quality of data, can be blocked by the security policies. Indeed, large structures are often organized in autonomous sub-organizations, each managing dedicated areas of activity and data sets. Viewing data quality management as a collaborative activity in which all structures participate seems to solve part of this problem. Many models have been proposed in literature to secure collaborative platforms but, to our best knowledge, all have limitations with regards to the Big Data constraints. We have opted for this research work to use the PolyOrBAC Framework which has the advantage of being independent and allows each structure to independently implement its security policy. In order to make up for some of its shortcomings, we extended this framework by integrating the WS-Agreement Specification, through the SLA-Framework, in order to automate the management of access agreements. The goal is to minimize human intervention and automate collaborative activity. Integrating the extended PolyOrBAC Framework into our data accuracy assessment process takes a step forward towards a comprehensive solution that balances data quality and data security systems.

To advance our research work, it will be necessary to work on the security aspect either by further improving the PolyOrBAC Framework or by implementing another solution dedicated to securing collaborative activities on Big Data platforms. Additionally, data heterogeneity will need to be further addressed in the data quality assessment process, especially for unstructured data.

References

[Abou El Kalam, 03] Abou El Kalam, A., Baida, R.E., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., Trouessin, G.: Organization based access control, 2003, 120 - 131, 10.1109/POLICY.2003.1206966

[Abou El Kalam, 07] Abou El Kalam, A., Deswarte, Y., Baina, A., Kaaniche, M.:Access Control for Collaborative Systems: A Web Services Based Approach, 2007, IEEE International Conference on Web Services (ICWS 2007), 10.1109/ICWS.2007.30

[Abou El Kalam, 09] Abou El Kalam, A., Deswarte, Y., Baina, A., Kaaniche, M.: PolyOrBAC: A security framework for Critical Infrastructures, 2009, International Journal of Critical Infrastructure Protection, 2, 154-169, 10.1016/j.ijcip.2009.08.005

[Adnan, 19] Adnan, K., Akbar, R.: An analytical study of information extraction from unstructured and multidimensional big data, 2019, Journal of Big Data, 6, 10.1186/s40537-019-0254-8

[Ait Aali, 15] Ait Aali, N., Baina, A., Echabbi, L.: Trust integration in collaborative access control model for Critical Infrastructures, 2015, 10th International Conference on Intelligent Systems: Theories and Applications (SITA), 1-6, 10.1109/SITA.2015.7358427

[Andrieux, 04] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Nakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M.: Web services agreement specification (WS-Agreement), 2004, Global Grid Forum, 2, http://www.ogf.org/documents/GFD.192.pdf

[Baig, 17] Baig, R., Khan, W., Haq, I., Khan, I.: Agent-Based SLA Negotiation Protocol for Cloud Computing, 2017, IEEE 5th International Conference on Cloud Computing Research and Innovation, 33-37, 10.1109/ICCCRI.2017.13

[Belbergui, 16] Belbergui, C., Elkamoun, N., Rachid, H.: A Dynamic Access Control Model for Cloud Computing Environments, 2016, In Proceedings of the 6th International Conference on Communication and Network Security (ICCNS '16), Association for Computing Machinery, 21–29, 10.1145/3017971.3017979

[Ben Saidi, 12] Ben Saidi, M., Abou Elkalam, A., Marzouk A.: TOrBAC: A Trust Organization Based Access Control Model for Cloud Computing Systems, 2012, International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-2 Issue-4, September 2012

[Ben Saidi, 13] Ben Saidi, M., Marzouk, A.: Multi-Trust_OrBAC: Access Control Model for Multi-Organizational Critical Systems Migrated To the Cloud, International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-3, Issue-2 May 2013

[Bertino, 15] Bertino, N.: Big Data – Security and Privacy, 2015, IEEE International Congress on Big Data, 10.1109/BigDataCongress.2015.126

[Castro, 15] Castro, I., Panda, A., Raghavan, B., Shenker, S., Gorinsky, S.: Route Bazaar: Automatic Interdomain Contract Negotiation, 2015, The 15th Workshop on Hot Topics in Operating Systems (HotOS XV 2015), 18-20 May 2015, Kartause Ittingen, Switzerland

[Coutinho, 16] Coutinho, C., Cretan, A., da Silva, C.F., Ghodous, P., JardimGonçalves, R.: Service-based Negotiation for Advanced Collaboration in Enterprise Networks, 2016, Journal of Intelligent Manufacturing, Springer Verlag (Germany), 201-216, ff10.1007/s10845-013-0857-4ff

[Habib, 11] Habib, S., Ries, S., Mühlhäuser, M.: Towards a Trust Management System for Cloud Computing, 2011, International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11, 10.1109/TrustCom.2011.129

[Labidi, 17] Labidi, T., Mtibaa, A., Gaaloul, W., Gargouri, F.: Ontology-Based SLA Negotiation and Re-Negotiation for Cloud Computing, 2017, IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, 36-41, 10.1109/WETICE.2017.24

[Li, 14] Li, Y., Cuppens-Boulahia, N., Crom, J.M., Cuppens, F., Frey, V.: Reaching Agreement in Security Policy Negotiation, 2014, IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, 10.1109/TrustCom.2014.17

[Li, 16] Li, Y., Cuppens-Boulahia, N., Crom, JM., Cuppens, F., Frey, V.: Expression and Enforcement of Security Policy for Virtual Resource Allocation in IaaS Cloud, 2016, In: Hoepman JH., Katzenbeisser S. (eds) ICT Systems Security and Privacy Protection, SEC 2016, IFIP Advances in Information and Communication Technology, vol 471, Springer, Cham, 10.1007/978-3-319-33630-5_8

[Li, 19a] Li, F., Clarke, S.: A Context-Based Strategy for SLA Negotiation in the IoT Environment, 2019, PerIoT'19 - Third International Workshop on Mobile and Pervasive Internet of Things, 10.1109/PERCOMW.2019.8730752

[Li, 19b] Li, F., Cabrera, C., Clarke, S.: A WS-Agreement Based SLA Ontology for IoT Services, 2019, 4th International Conference Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, Proceedings, 10.1007/978-3-030-23357-0_5

[Ludwig, 03] Ludwig, H., Keller, A., Dan, A., King, R., Franck, R.: Web Service Level Agreement (WSLA) Language Specification, 2003, IBM Corporation, 815–824

[Ludwig, 06] Ludwig, H., Nakata, T., Wäldrich, O., Wieder, P., Ziegler, W.: Reliable Orchestration of Resources Using WS-Agreement, 2006, In: Gerndt M., Kranzlmüller D. (eds) High Performance Computing and Communications, HPCC 2006, Lecture Notes in Computer Science, vol 4208, Springer, Berlin, Heidelberg, 10.1007/11847366_78

[Marino, 18] Marino, F., Moiso, C., Petracca, M.: Automatic contract negotiation, service discovery and mutual authentication solutions: A survey on the enabling technologies of the forthcoming IoT ecosystems, 2018, Computer Networks, 148, 10.1016/j.comnet.2018.11.011

[Reinsel, 17] Reinsel, D., Gantz, J., Rydning, J.: Data Age 2025: The Evolution of Data to Life-Critical. Don't Focus on Big Data; Focus on the Data That's Big, 2017, IDC White Paper

[Saleh, 14] Saleh, A., Hamed, E., Hashem, M.: Building trust management model for cloud computing, 2014, 9th International Conference on Informatics and Systems, PDC116-PDC125, 10.1109/INFOS.2014.7036688

[Scoca, 17] Scoca, V., Uriarte, R.B., De Nicola, R.: Smart Contract Negotiation in Cloud Computing, 2017, IEEE 10th International Conference on Cloud Computing, 10.1109/CLOUD.2017.81

[Seng, 19] Seng, K., Ang, L.M.: Multimodal Emotion and Sentiment Modeling From Unstructured Big Data: Challenges, Architecture, & Techniques, 2019, IEEE Access, 10.1109/ACCESS.2019.2926751

[Shojaiemehr, 19] Shojaiemehr, B., Rahmani, A., Qader, N.: A Three-phase Process for SLA Negotiation of Composite Cloud Services, 2019, Computer Standards & Interfaces, 10.1016/j.csi.2019.01.001

[SLA-Framework, 16] SLA Framework, v1.1, 2016, https://github.com/FIWARE/ops.Sla-framework

[Smith, 07] Smith, M., Schmidt, M., Fallenbeck, N., Schridde, C., Freisleben, B.: Optimising Security Configurations with Service Level Agreements, 2007, https://www.semanticscholar.org/paper/Optimising-Security-Configurations-with-Service-Smith-Schmidt/bb18eca8f629edb494cb8dd8c7fab9027cd23dcf

[Stankov, 12] Stankov, I., Datsenka, R., Kurbel, K.: Service Level Agreement as an Instrument to Enhance Trust in Cloud Computing – An Analysis of Infrastructure-as-a-Service Providers, 2012, AMCIS 2012 Proceedings, 12, http://aisel.aisnet.org/amcis2012/proceedings/HCIStudies/12

[Sun, 11] Sun, X., Chang, G., Li, F.: A Trust Management Model to Enhance Security of Cloud Computing Environments, 2011, Proceedings - 2nd International Conference on Networking and Distributed Computing, 244-248, 10.1109/ICNDC.2011.56

[Talha, 19a] Talha, M., Abou El Kalam, A., Elmarzouqi, N.: Big Data: Trade-off between Data Quality and Data Security, 2019, Procedia Computer Science, 151, 916-922, 10.1016/j.procs.2019.04.127

[Talha, 19b] Talha, M., Elmarzouqi, N., Abou El Kalam, A.: Quality and Security in Big Data: Challenges as opportunities to build a powerful wrap-up solution, 2019, Journal of Ubiquitous Systems and Pervasive Networks, 12, 09-15, 10.5383/JUSPN.12.01.002

[Talha, 20] Talha, M., Elmarzouqi, N., Abou El Kalam, A.: Towards a Powerful Solution for Data Accuracy Assessment in the Big Data Context, 2020, International Journal of Advanced Computer Science and Applications (IJACSA), 11, 10.14569/IJACSA.2020.0110254

[Tosic, 02] Tosic, V., Patel, K., Pagurek, B.: WSOL — Web Service Offerings Language, 2002, Information Systems, 30, 564–586, 10.1007/3-540-36189-8_5

[Toumi, 12] Toumi, K., Andrés, C., Cavalli, A.: Trust-orBAC: A Trust Access Control Model in Multi-Organization Environments, 2012, In: Venkatakrishnan V., Goswami D. (eds) Information Systems Security, ICISS 2012, Lecture Notes in Computer Science, vol 7671, Springer, Berlin, Heidelberg, 10.1007/978-3-642-35130-3_7

[Tseng, 16] Tseng, S.S., Chen, H.C., Hu, L.L., Lin, Y.T.: CBR-based negotiation RBAC model for enhancing ubiquitous resources management, 2016, International Journal of Information Management, 37, 10.1016/j.ijinfomgt.2016.05.009

[Wäldrich, 11] Wäldrich, O., Battré, D., Brazier, F., Clark, K., Oey, M., Papaspyrou, A., Wieder, P., Ziegler, W.: WS-Agreement Negotiation Version 1.0, 2011, http://www.ogf.org/documents/GFD.193.pdf

[WSAG4J, 12] WSAG4J framework, V2.0.0, 2012, http://wsag4j.sourceforge.net/site/wsag/overview.html

[XACML, 13] eXtensible Access Control Markup Language (XACML) Version 3.0, 2013, OASIS Standard, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.htm