Recognition of Authority in Virtual Organisations

Tuan-Anh Nguyen, David Chadwick, and Bassem Nasser

University of Kent, Canterbury, England

Abstract. A Virtual Organisation (VO) is a temporary alliance of autonomous, diverse, and geographically dispersed organisations, where the participants pool resources, information and knowledge in order to meet common objectives. This requires dynamic security policy management. We propose an authorisation policy management model called *recognition of authority* (ROA) which allows dynamically trusted authorities to adjust the authorisation policies for VO resources. The model supports dynamic delegation of authority, and the expansion and contraction of organizations in a VO, so that the underlying authorisation system is able to use existing user credentials issued by participating organisations to evaluate the user's access rights to VO resources.

1 Introduction

A Virtual Organisation (VO) is a temporary alliance of autonomous, diverse, and geographically dispersed organisations, where the participants pool resources, information and knowledge in order to meet common objectives. The objectives of an alliance can evolve and the relationships between the different parties may change. Therefore virtual organisations are naturally dynamic. Consequently, management, especially security management in such a dynamic environment must be provided with suitable dynamic mechanisms. There are several areas of security under consideration for VOs but in this paper we are concerned with authorisation and access control.

The behaviour of an organisation's authorisation system is normally governed by an authorisation policy, written by the policy officer (or Source of Authority - SoA). In a dynamic environment like a VO, organisations may continually join or leave the collaboration. When joining a VO, an organisation may need to provide access to its protected resources to users from other organisations in the VO. When the organisation leaves the VO, access rights to its protected resources from users outside the organisation have to be removed. In these cases, the authorisation policy of the organisation has to be *dynamically modified and updated* to cater for these dynamic changes. However,

1. in a VO, which is a pan-organisational system, the number of attributes and users can be in the hundreds or thousands. Managing these attributes and users and their relationships is a formidable task that can not realistically be done by one person ([16]).

2. in reality, an authorisation policy is a set of low-level policies derived from high-level ones and the refinement process requires the involvement of many people, within the same or partner organisations ([6], [11]).
3. the exact form of collaboration between an organisation and a partner in the VO is normally not known beforehand, so the permissions to modify the authorisation policy need to be delegated on demand to the people that deal with the collaboration.

Therefore, the permissions to modify and update the policy may need to be *dynamically delegated* from the SoA to other delegates on demand. Consequently, these delegates are allowed to adjust the organisation's policy, in order to accommodate requirements in the collaborations and to give users in partner organisations access rights to the protected resources of the organisation.

In the RBAC model ([5], [17]), an authorisation policy includes a set of role-permission assignments (RPA), a role hierarchy (optional) and a set of rules that regulate the assignments of roles to users (user-role assignments, URA). In order to avoid policy conflicts, especially when the same organisations are simultaneously members of multiple VOs, we require that each collaboration be independent with its own security objectives and requirements. For example, within one collaboration, a Student role may be considered the subordinate role of a Staff role and the later to inherit the permissions of the former, but within another collaboration, the two roles may be independent with no permission inheritance. If the two collaborations are not independent, it is possible that the requirements of one collaboration cannot be fulfilled or they may conflict with those of the other.

On the other hand, in a VO there may be several organisations that support an inter-organisational workflow and these organisations may need to be changed during the workflow's life cycle. Furthermore, the workflow's requirements (or specification) may also need to be changed. The workflow's security infrastructure should not be tied to users or attributes from any of the partner organisations. Otherwise, if one partner is replaced by another then the workflow security infrastructure would have to be modified to account for this change. Additionally, the partner organisations should not tie permissions used by the workflow to their own users or attributes because if the permissions needed for the workflow change, the partner organisation would need to modify the permissions given to its users or attributes to accommodate these changes. Consequently there needs to be a level of indirection between the workflow's security infrastructure and the organisation's security infrastructure. Since each organisation may support several inter-organisational workflows, it is not realistic for each organisation to restructure its organisational level security infrastructure when workflow security infrastructure changes occur and vice versa. Therefore, the workflow security infrastructure needs to be separated from the organisational-level security infrastructure as stated in [8] and [14]. Our model provides this separation through the dynamic on demand specification of organizational level attributes that grant access to a VO's workflow resources. The organizational

level attributes are dynamically mapped into either workflow roles or workflow privileges.

1.1 Objectives and Contribution

In the VO environment, there are issuing domains that issue credentials to users and target domains that consume credentials ([3]). The authorization policy of the target domain decides whether an issued credential is to be trusted or not i.e. is valid or not, and whether it provides sufficient permissions or not to the accessed resource. In an attribute (or role) based authorisation policy, the permission-attribute assignments (or RPA) form the access control policy. The URA form the credential validation policy ([3]). Thus, an authorisation policy includes an access control policy and a credential validation policy.

In this paper, we propose a model called **recognition of authority** which provides the following features for authorisation administration in a virtual organisation:

- Administrative roles are defined which grant permission to dynamically update limited parts of the authorisation policy in the target domain, more specifically, to assign organizational level attributes to a subset of the privileges which grant access to the VO's workflow resources.
- Administrators are dynamically created by assigning these administrative roles to them. These roles can be dynamically delegated, and also dynamically revoked, thereby dynamically adding and removing administrators from the system.
- An administrator can dynamically assign a subset of the permissions granted by the administrative role, to any organizational level user attributes (i.e. perform RPA). In addition, the administrator can provide the policy information for validating the user credentials that contain these attributes (i.e. URA validation).
- Collaborations between organisations are independent of each other, since a VO's workflow privileges are independent of those of other VOs.
- Application-level (workflow) security infrastructures are separated from organisational level security infrastructures since workflow permissions are dynamically assigned to organizational level attributes.

By allowing authorization policies to be dynamically updated as above, our model allows the authorisation system of a target domain to dynamically *recognise* trusted administrators, to dynamically *recognise* the new attributes they are trusted to issue, and to dynamically *recognise* new users of the VO. The initial definition of the administrative roles means that the authorization system knows the limit of their administrative authority in assigning permissions to users.

The rest of this paper is structured as follows: section 2 reviews some related research, section 3 compares and contrasts two approaches for assigning permissions to attributes, section 4 presents our recognition of authority management model in detail and the last section provides a conclusion and indicates where future research is still needed.

2 Related Works

In [7], the authorisation policy in a target domain is only modified and updated by the security officer in that domain, so that the model is not appropriate for dynamic and large environments like VOs. The RT model (Role-based Trust-management – [9], [10]) is a very powerful framework for representing policies and credentials in distributed authorisation system. It provides the capability of role mapping i.e. one role issued in one domain is mapped to another role issued in another domain. In this way, permissions in one organisation are assigned to roles issued in another organisation and users from one organisation can access protected resources in another. The disadvantage of the RT model is that it only supports RT formatted credentials, ignoring the fact that users' credentials in VOs are organisation-dependent. In [14], the policy of an organisation is only updated by its administrator and does not have a mechanism to separate collaborations from each other. Furthermore, the policy for role mapping is statically set by the administrator in a system-site. The current PERMIS infrastructure ([1], [2]) supports the dynamic assignment of roles to users in different domains but does not have the capability of dynamically adjusting the authorisation policy. The CAS model ([15]) is used for authorisation in Grid environments but the policy of a CAS server is only modified and updated by its predefined administrators. Furthermore, it can not separate the workflow security infrastructure from the organisation level security infrastructure or explicitly deal with multiple collaborations. If there is a change of participant in the collaboration, the CAS server has to be re-configured with a new set of users and users' permissions. The framework proposed in [6] does not separate inter-organisational workflows from organisation-level changes because if there is a change in participation, the inter-organisational workflow specification which specifies who can have which permissions will have to be changed. Furthermore, the framework has no mechanism to separate collaborations from each other. In [12], the authors proposed the dynamic coalition-based access control (DCBAC) model that facilitates the formation of dynamic coalitions through the use of a registry service, where available services can be advertised by potential coalition members. This model does not consider the decentralised administration of collaborations, so that only the SoA in an organisation can register the organisation's services to coalitions. Furthermore, the workflow security infrastructures are not separated from the organisation level security infrastructures. The major contribution of our paper is bring together in one model the various advantages of the different models above, by allowing the dynamic update of authorization policies by a dynamically changing decentralized pool of administrators, whilst keeping a tight separation between workflow security infrastructure and organisation level security infrastructure and also between one collaboration and another.

3 Direct Permission Assignment vs. Role Mapping

There are two approaches for assigning permissions in a local organisation to users in partner organisations. The first is to directly assign permissions to re-

mote user attributes ([4], [6], [7], [15]) and the second is to map remote user attributes into local user roles by attribute-role mapping ([9], [14]). Both approaches can facilitate collaboration between organisations. In the attribute-role mapping approach, the permission given to a remote attribute is the permission of the local role, which is fixed. Thus, this approach limits the granularity of delegation to that of the pre-defined local roles (and their subordinate roles), whilst direct permission assignment allows each permission to be delegated or assigned separately. On the other hand, by mapping remote user attributes to local roles (used for workflows), the changes of participants in a workflow are confined to the modification of mappings from an organisation's attributes to the local user roles (it does not affect the workflow's specification) and changes to the specification of local roles do not require modifications to the remote user attribute specifications. Thus, this approach supports the separation of workflows from organisational changes ([8], [14]). Since both approaches have their merits, our model is designed to support both approaches. When an administrative role is defined, its administrative permissions are defined as either an ability to assign permissions to user attributes, or an ability to map user attributes into existing local user roles.

4 Recognition of Authority Management Model

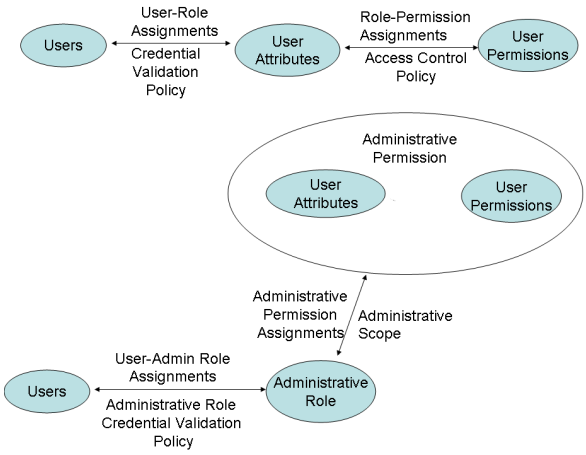

Fig. 1. User Roles and Administrative Roles

We identify two types of permission: a *normal permission* (or *user permission*) and an *administrative permission*. A user permission is a consent (for a user) to perform an action on a particular resource under certain conditions. An

administrative permission is a consent (for an administrator) to perform role permission assignments i.e. to either assign one or more user permissions to a set of (one or more) user attributes, or to perform role mappings between user attributes.

When a set of user permissions is given to an attribute, we say that the attribute is a *user attribute*. When a set of administrative permissions is given to a role we say the role is an *administrative role*. Someone who holds an administrative role is called an administrator. The set of user permissions and user attributes that an administrator can assign or map to new user attributes is called his *administrative scope*.

The recognition of authority management model for facilitating dynamic collaboration between organisations comprises the following steps:

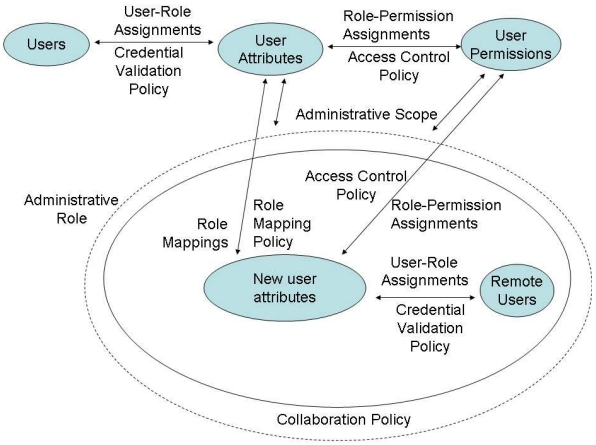

Fig. 2. Collaboration Policy

1. The policy writer (SoA) of the target domain defines a set of administrative roles for the target domain, an administrative role credential validation policy, and the workflow permissions that are attached to these administrative roles (i.e. the administrative scope).
2. The SoA dynamically delegates these administrative roles to trusted people in remote domains with whom there is to be a collaboration, by issuing administrative role credentials to them.
3. To establish a collaboration, one of these administrators must update the SoA's authorisation policy by writing a collaboration policy. The collaboration policy includes an access control policy and/or a role mapping policy, and a user credential validation policy. The latter specifies validation rules for user credentials containing newly defined (organizational level) user attributes, whilst the former specifies either role permission assignments or

role mappings for the newly defined user attributes. In this way, users who hold credentials containing these new attributes will gain access to the appropriate target resources.

4. In order to ensure that no administrator can overstep his delegated authority, the authorisation system has to validate that the collaboration policy lies within the the administrative scope specified in Figure 1 above. If it does, it is accepted, and its policy rules become dynamically incorporated into the SoA's policy. If it does not, it is rejected, and its policy rules will be ignored.
5. When a user from a collaborating domain wants to access a protected resource in the target domain, assuming the collaboration policy has been accepted, the authorisation system retrieves and validates the user's credentials/attributes against the now enlarged credential validation policy. Only valid attributes will then be used by the access control system to make access control decisions for the user's request against the now enlarged access control policy ([3]).
6. An administrator may dynamically delegate his administrative role to another person, providing the delegate falls within the scope of the administrative role credential validation policy set by the resource SoA (see Figure 1). In [13] we have proposed a delegation of authority model that has the capability to further constrain the authority of administrators so that they may not only delegate their administrative roles, but also a subset of them. However, this refined delegation of administrative roles is not considered further in the scope of this paper. We will assume for now that administrators may delegate their (unconstrained) roles to other administrators.

4.1 Administrative Roles

The SoA of a target domain is the person who is fully trusted by the authorisation system to set its authorisation policy. The SoA's administrative scope is all the user permissions that are under his control in the target domain. In our model, we propose that the SoA defines a set of administrative roles which each control either a subset of the user permissions or mappings to subsets of local user attributes. The SoA may then delegate these administrative roles to other people on demand as the need arises, so that the delegates can control subsets of user permissions or role mappings. We express an administrative role as:

- Either a finite set of user permissions $p_i$which can be assigned to (new or existing) user attributes: $aRole = \{p_1, p_2, \ldots, p_n\}$. The holder of this kind of administrative role is trusted to assign any of the user permissions that comprise the definition of the administrative role, to any set of user attributes provided that the assignments satisfy the restrictions placed on the administrative role.
- Or a set of existing user attributes to which new user attributes can be mapped: $aRole = \{uR_1, uR_2, \ldots, uR_n\}$. The holder of this kind of administrative role is trusted to map any set of new user attributes into any set of

existing local user attributes that comprise the definition of the administrative role, provided that the mappings satisfy the restrictions placed on the administrative role.

Note that the remote administrator who deals with a collaboration needs to know either the existing permissions or user attributes in the target domain in order to perform either role permission assignments or role mappings. A DTD for role-permission assignments and attribute-role mappings is provided in the Appendix.

4.2 Validation of an Administrator's Administrative Roles

Validating an administrator's administrative roles is no different to validating a user's credential. The authorisation system validates credentials based on its credential validation policy. We have proposed a model for validating users' credentials, called the Credential Validation Service (CVS) in [3]. In general, the CVS is provided with a trust model that tells it which attribute issuers to trust (roots of trust), and a credential validation policy that provides the rules to control which delegates are allowed to receive which delegated roles.

The formal representation of the CVS's credential validation policy is as follows:

1. a set of attributes $ATTRIBUTES$,
2. a set of attribute hierarchies $S_{RH} = \{RH\}$, RH is a attribute hierarchy,
3. a set of delegation rights $RIGHTS = \{d\}$,
4. a set of trusted root credential issuers or AAs $AAS = \{AA\}$,
5. $HAS \subset AAS \times RIGHTS$ is a AA – Delegation Rights table, which says which trusted credential issuers have which delegation rights.

We formulate a delegation right (or the right to delegate or assign an attribute) as $d = d(attr, Q, n, DT)$ where Q is a restriction of the delegation right – the holder of the delegation right can only delegate or assign *attr* to a user (delegate) that satisfies the restriction Q. Restrictions will be presented shortly. $n > 0$ is the maximum delegation depth of a delegation chain that can be made by the holder. DT is the *maximum validity period* of the delegations that can be made by the holder.

The CVS is able to retrieve (in pull mode) or obtain (in push mode) user credentials, find the delegation chain(s) from a trusted credential issuer to a user's credential and validate the credentials in the delegation chain(s). Trusted credential issuers are only allowed to delegate (or assign) attributes to users who satisfy the restrictions placed on their delegation rights.

For collaborations between organisations, the CVS is able to validate administrative role credentials as well as user attribute credentials. The administrative role credential validation policy provides the rules used to control the validation of administrative role credentials according to the same trust model as user credentials. In this case, the SoA is the only trusted root credential issuer for the delegation of administrative roles.

In our model, Q is an expression of the attributes a user must have in order to become a delegate. Because a user's attributes are the user's properties in his organisation, the expression of user attributes varies between organisations and is application-dependent. User attributes may be the roles of the user in the organisation, the user's age, credit limit, or the domain of the user etc. An example expression of user attributes is (Role = Researcher) $\wedge$ (Age $>$ 35) i.e. i.e. in order to be a delegate, the user must have a "Researcher" role and be aged greater than 35.

4.3 Validation of Collaboration Policies

A collaboration policy made by an administrator includes an access control policy (or role mapping policy) and a credential validation policy. This will control which users are able to access the target resource. The authorisation system in the target domain has to check whether the access control or role mapping policy is within the administrative scope of the administrator, but the credential validation policy does not have any restrictions placed on it, since the administrator is trusted to say which users should have access to the resource. In reality, the VO agreement will state which target resources should be made available to the collaborating organizations, and so the SoA only sets restrictions on which resources can be accessed, via the administrative scope. It is then left up to the various collaborating administrators to decide which of their users should have this access, and to set their credential validation policies accordingly. In this way, the policy that validates the users is delegated to the collaborating administrators, but is enforced by the target resource's PDP.

If we assume that an administrator has a set of valid administrative roles $aRoles = \{aRole^i\}, \quad i = 1..n$, where an administrative role has a set of user permissions $aRole^i = \{p_1^i, p_2^i, \ldots, p_{ii}^i\}$ or user attributes $aRole^i = \{uR_1^i, uR_2^i, \ldots, uR_{ii}^i\}$ then a role-permission assignment $attribute \leftarrow \{p_1, p_2, \ldots, p_k\}$ is valid if and only if $\forall p \in \{p_1, p_2, \ldots, p_k\}$, $\exists l, 1 \leq l \leq n$, $p \in aRole^l$ and an attribute-role mapping $uR \leftarrow attribute$ is valid if and only if $\exists aRole^i, uR \in aRole^i$ or uR is a subordinate of uRR in which $\exists aRole^i, uRR \in aRole^i$.

If the above policies are valid, then the target resource will add these policies to its existing ones. The CVS will add the administrator's credential validation policy to its existing ones, and the PDP will add the role-permission assignments to its existing ones. We believe that role mappings are logically part of the CVS's functionality, and that after validating a user's credentials, the CVS should return the mapped roles to the PEP. In this way the PDP can make an access control decision based on its existing rule set.

5 Conclusion

Dynamically decentralising the administration of an authorisation system for a VO's requirements without loosing central control over broad policy is a challenging goal for system designers and architects. Our work provides a significant and practical advance towards this goal by proposing the recognition of

authority management model. The ROA model allows dynamically assigned administrators to dynamically adjust the authorisation policy of a target domain. Therefore, our model supports decentralised authorisation administration. By separating authorisation policies created for each collaboration, the collaborations remain independent, so that the policies for one collaboration do not affect other collaborations and the policies can be added and removed independently. By supporting attribute-role mapping, our model can separate workflows from organisational changes. By supporting delegated role-permission assignments we maximize the granularity of administrative delegation. Another benefit of delegated role-permission assignments is that administrators can assign target resource permissions to local organizational level user attributes and it facilitates decentralised management of permissions to VO resources.

Currently, an implementation of the recognition of authority model in the PERMIS authorisation infrastructure is under way. We hope that with the implementation, we can evaluate the usability and performance of the model.

References

1. David Chadwick and Sassa Otenko. The permis x.509 role based privilege management infrastructure. In *Proceedings of 7th ACM Symoisium on Access Control Models and Technologies (SACMAT 2002)*.
2. David Chadwick, Gansen Zhao, Sassa Otenko, Romain Laborde, Linying Su, and Tuan Anh Nguyen. Building a modular authorization infrastructure. In *Fifth All Hands Meeting. UK e-science, Achievements, Challenges & New Opportunities*, September 2006.
3. David W Chadwick, Sassa Otenko, and Tuan Anh Nguyen. Adding support to xacml for dynamic delegation of authority in multiple domains. In *10th IFIP Open Conference on Communications and Multimedia Security, Heraklion Crete*, 2006.
4. Marlena Erdos and Scott Cantor. Shibboleth-architecture draft v05. Technical report, Internet2, May 2002.
5. David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.
6. Babak Sadighi Firozabadi, Olle Olsson, and Erik Rissanen. Managing authorisations in dynamic coalitions. In *Swedish Institute of Computer Science*, 2003.
7. Lalana Kagal, Timothy Finin, and Yun Peng. A delegation based model for distributed trust. In *Proceedings of the IJCAI01 Workshop on Autonomy, Delegation and Control: Interacting with Autonomous Agent, Seattle*, pages 73–80, 2001.
8. Myong H. Kang, Joon S. Park, and Judith N. Froscher. Access control mechanisms for inter-organizational workflow. In *The sixth ACM symposium on Access control models and technologies*, pages 66–74, Chantilly, Virginia, United States, 2001. ACM Press.
9. Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE Computer Society Press, 2002.
10. Ninghui Li, John C. Mitchell, and William H. Winsborough. Distributed credential chain discovery in trust management. *Journal of Computer Security*, pages 35–86, 2003.

11. Jonathan D. Moffett and Morris S. Sloman. Policy hierarchies for distributed systems management. *IEEE Journal on Selected Areas in Communications*, 11(9):1404–1414, 1993.
12. Ravi Mukkamala, Vijayalakshmi Atluri, Janice Warner, and Ranjit Abbadasari. A distributed coalition service registry for ad-hoc dynamic coalitions: A service-oriented approach. In *E. Damiani and P. Liu (Eds.): Data and Applications Security 2006, LNCS 4127*, pages 209–223. IFIP, 2006.
13. Tuan-Anh Nguyen, Linying Su, George Inman, and David Chadwick. Flexible and manageable delegation of authority in rbac. In *Proceedings of The IEEE Ubisafe07*, Ontario, Canada, 21-23 May 2007. IEEE Computer Society Press.
14. Joon S. Park, Keith P. Costello, Teresa M. Neven, and Josh A. Diosomito. A composite rbac approach for large, complex organizations. In *ACM SACMAT'04 Yorktown Heights, New York, USA*, 2004.
15. L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. Community authorization service for group collaboration. *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.
16. Ravi Sandhu, Venkata Bhamidipati, and Qamar Munawer. The arbac97 model for role-based administration of roles. *ACM Transactions on Information and System Security*, 2(1):105–135, 1999.
17. Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer Society Press, Loas Alamitos, CA, USA*, 29(2):38–47, 1996.

Appendix

A DTD for Attribute-Permission Assignment and Attribute-Role Mapping

<!ELEMENT AttributeAssignmentPolicy (AttributeAssignment)+ >
<!ELEMENT AttributeAssignment (SubjectDomain, AttributeList, Delegate, TrustedIssuer, Validity) >
<!ELEMENT SubjectDomain EMPTY>
<!ATTLIST SubjectDomain ID IDREF #REQUIRED>
<!ELEMENT AttributeList (Attribute*) >
<!ELEMENT Attribute EMPTY >
<!ATTLIST Attribute Type IDREF #IMPLIED Value IDREF #IMPLIED >
<!ELEMENT TrustedIssuer EMPTY>
<!ATTLIST TrustedIssuer ID IDREF #REQUIRED>
<!ELEMENT Validity (Absolute?, Age?, Maximum?, Minimum?) >
<!ELEMENT Delegate EMPTY >
<!ATTLIST Delegate Depth CDATA #IMPLIED >
<!ELEMENT TargetPolicy (TargetDomainSpec+) >
<!ELEMENT TargetDomainSpec ((Include, Exclude*)+, ObjectClass*) >
<!ATTLIST TargetDomainSpec ID IDREF #REQUIRED>
<!ELEMENT ActionPolicy (Action+) >
<!ELEMENT Action EMPTY>
<!ATTLIST Action Name NMTOKEN #REQUIRED Args NMTOKENS #IMPLIED>
<!ELEMENT TargetAccessPolicy (TargetAccess) >

<!ELEMENT TargetAccess (AttributeList, TargetList, IF?) >
<!ELEMENT TargetList (Target+) >
<!ELEMENT Target (TargetName —TargetDomain) >
<!ATTLIST Target Actions NMTOKENS #IMPLIED >
<!ELEMENT TargetName EMPTY>
<!ATTLIST TargetName LDAPDN CDATA #REQUIRED>
<!ELEMENT TargetDomain EMPTY>
<!ATTLIST TargetDomain ID IDREF #REQUIRED>
<!ELEMENT AttributeMappingPolicy (AttributeMapping) >
<!ELEMENT AttributeMapping (Attribute, LocalRole)+ >
<!ELEMENT LocalRole EMPTY >
<!ATTLIST LocalRole Type IDREF #IMPLIED Value IDREF #IMPLIED >