InterNational Committee for Information Technology Standards (INCITS)
Secretariat: Information Technology Industry Council (ITI)
1101 K Street NW, Suite 610, Washington, DC 20005
www.INCITS.org

eb-2016-00808-002

Document Date:	2/22/17
To:	INCITS Members
Reply To:	Rachel Porter
Subject:	Public Review and Comments Register for the Approval of:

INCITS 499-201x (revision of INCITS 499-2013), Information technology - Next Generation Access Control - Functional Architecture (NGAC-FA)

Due Date: **The public review is from December 30, 2016 – February 28, 2017**

Action: The InterNational Committee for Information Technology Standards (INCITS) announces that the subject-referenced document(s) is being circulated for a 60-day public review and comment period. Comments received during this period will be considered and answered. Commenters who have objections/suggestions to this document should so indicate and include their reasons.

All comments should be forwarded not later than the date noted above to the following address:

INCITS Secretariat/ITI
1101 K Street NW - Suite 610
Washington DC 20005-3922
Email: comments@standards.incits.org (preferred)

This public review also serves as a call for patents and any other pertinent issues (copyrights, trademarks). Correspondence regarding intellectual property rights may be emailed to the INCITS Secretariat at patents@itic.org.

Comment #1 – J. Sellwood

Comment #2 – J. Sellwood

Working Draft
American National
Standard

DPANS
INCITS 499

Revision 1.00
22 August 2016

Information technology - Next Generation Access Control - Functional Architecture (NGAC-FA)

CS1 Technical Editors:

Roger Cummings
Antesignanus
1001 Heathrow Park Lane, MS 801/3120
Heathrow, FL 32746
USA
Telephone: (407) 357-7257
Email: rjc@computer.org

Wayne Jansen
Bayview Behavioral Consulting
1574 Gulf Rd., #237
Point Roberts, WA 98281
USA
Telephone: (360) 306-5263
Email: jansen@computer.org

Reference number
ANSI INCITS 499-201x

Points of Contact

InterNational Committee for Information Technology Standards (INCITS) CS1 Technical Committee

CS1 Chair
Dan Benigni

Telephone:
Email: drbenigni47@gmail.com

CS1 Vice-Chair
Sal Francomacaro
NIST
Stop 8930
Gaithersburg, MD 20899-8930
USA

Telephone: (301) 975-6414
Email: salvatore.francomacaro@nist.gov

CS1 Web Site: http://cs1.incits.org/

CS1 E-mail reflector: cyber-security@standards.incits.org

INCITS Secretariat
Suite 200
1250 Eye Street, NW
Washington, DC 20005
USA

Telephone: (202) 737-8888
Web site: http://www.incits.org
Email: incits@itic.org

Information Technology Industry Council
Web site: http://www.itic.org

Document Distribution
INCITS Online Store
managed by Techstreet
1327 Jones Drive
Ann Arbor, MI 48105
USA

Web site: http://www.techstreet.com/incits.html
Telephone: (734) 302-7801 or (800) 699-9277

Revision Information

Version 1.00 (22 August 2016)

Draft of proposed revisions to INCITS 499-2013.

Draft

American National Standards
for Information Systems

Next Generation Access Control - Functional Architecture (NGAC-FA)

Secretariat

InterNational Committee for Information Technology Standards

Approved mm.dd.yyyy

American National Standards Institute, Inc.

Abstract

Next Generation Access Control (NGAC) is a fundamental reworking of traditional access control to meet the needs of the modern, distributed, interconnected enterprise. NGAC is based on a flexible infrastructure that can provide access control services for a number of different types of resources, accessed by a number of different types of applications and users. The infrastructure is scalable, able to support policies of different types simultaneously, and remain manageable in the face of changing technology, organizational restructuring, and increasing data volumes. This standard defines the functional architecture that is the basis for all other NGAC standards.

Draft

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

Caution: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard,

No further patent search is conducted by the developer or publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by
American National Standards Institute
11 W. 42nd Street, New York, New York 10036

Printed in the United States of America

Table of Contents

List of Figures

Figure **Page**

Foreword

(This foreword is not part of American National Standard INCITS.***:201x.)

Technical Committee CS1 of Accredited Standards Committee INCITS developed the NGAC Functional Architecture standard during 2011-2012. The standards approval process started in 2012 and completed in 2013. This revision to the 2013 NGAC Functional Architecture standard was developed during 2015-2016. The approval process started in 2016.

Next Generation Access Control (NGAC) is a fundamental reworking of traditional access control into a form suited to the needs of the modern, distributed, interconnected enterprise. NGAC is based on a flexible infrastructure that can provide access control services for a number of different types of resources, accessed by a number of different types of applications and users. The NGAC infrastructure is scalable and able to support policies of different types simultaneously, while remaining manageable in the face of changing technology, organizational restructuring, and increasing data volumes.

Access control is both an administrative and an automated process of defining and restricting which users and their processes can perform which operations on which system resources. The information that provides the basis by which access requests are granted or denied is known as a security policy. A security model is a formal representation of a security policy and its working. A wide variety of policy types and supporting security models have been created to address different situations. Examples of well-known policies are Discretionary Access Control (DAC), Mandatory Access Control (MAC), Role Based Access Control (RBAC) and Chinese Wall.

NGAC diverges from traditional approaches to access control in defining a generic architecture that is separate from any particular policy or type of policy. NGAC is not an extension of, or adaption of, any existing access control model, but instead is a redefinition of access control in terms of a fundamental and reusable set of data abstractions and functions. NGAC provides a unifying framework capable without extension of supporting not only current access control approaches, but also novel types of policy that have been conceived but never implemented due to the lack of a suitable means of expression and enforcement.

NGAC follows an attribute-based construction in which characteristics or properties are used to control access to resources and to describe and manage policy. NGAC accommodates combinations of different policies merely by changes to its control information, and thus it is possible to have several types of policies supported concurrently in a manner that is both deterministic and manageable. NGAC is also suitable for applications in which some information is stored locally and some is stored in a grid or cloud, since different policies can be asserted in each context. Even more generally, NGAC supports a situation where policy determined by a central organization is able to operate concurrently with a local, specific and more ad-hoc policy.

Through its support of access control policies, NGAC is also able to protect data services, such as e-mail, workflow, and records management. Support for data services is effected through the use of NGAC access control information to mediate data service operations.

The family of NGAC standards specifies the architecture, functions, operations, and interfaces at a level of detail necessary to ensure their realization in different types of implementation environments at a range of scalability levels. This standard specifies the functional architecture upon which the family of NGAC standards is based.

This standard contains the following items:

a) illustrations of the functional architecture and the entities that it comprises;

b) descriptions of each entity of the functional architecture;

c) descriptions of the interfaces between entities;

d) definitions of the information flows between entities; and

e) overviews of the other standards in the NGAC standards family.

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, InterNational Committee for Information Technology Standards, Information Technology Institute, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922.

Users of this standard are encouraged to determine if there are standards in development or new versions of this standard that may extend or clarify technical information contained in this standard.

This standard was processed and approved for submittal to ANSI by the InterNational Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, INCITS had the following members:

Organization Represented **Name of Representative**

Editor's Note: <<Insert INCITS member list>>

Technical Committee CS1 on Cyber Security, which reviewed this standard, had the following members:

Dan Benigni, Chair
Sal Francomacaro, Vice-Chair
Eric Hibbard, International Representative

Organization Represented	Name of Representative

CS1 Ad Hoc on Next Generation Access Control, which developed and reviewed this standard, had the following members:

Sal Francomacaro, Chair
Roger Cummings, Wayne Jansen, Editors

Organization Represented	Name of Representative
Antesignanus	Roger Cummings
Data Security Inc.	Hilary Hosmer
Hewlett-Packard Co.	Richard Austin
NIST	David Ferraiolo
	Sal Francomacaro
	Serban Gavrila
	Wayne Jansen
VHA CIO	Mike Davis
	Richard Grow
	Adrianne James
	Diana Proud-Madruga

Introduction

Next Generation Access Control (NGAC) is a fundamental reworking of traditional access controls to meet the needs of the modern, distributed, and interconnected enterprise. NGAC provides a unifying framework capable without extension of supporting current access control policies, as well as novel types of policy, and is also able to support different types of policies simultaneously. NGAC defines access control in terms of a fundamental and reusable set of data abstractions and functions, following an attribute-based access control model in which authorization is defined in terms of attributes. Security-relevant properties of users, processes and objects, such as role, sensitivity, affiliation and class, can be expressed as attributes.

The family of NGAC standards specifies the architecture, functions, operations, and interfaces necessary to enable conforming implementations to interact in an effective manner. This standard, NGAC-FA, defines an attribute-based, access control architecture and supporting framework as the foundation for NGAC.

This standard is divided into the following clauses and annexes:

Clause 1 defines the scope of this standard.

Clause 2 enumerates the normative references that apply to this standard.

Clause 3 defines the definitions, symbols, abbreviations, and notation used in this standard.

Clause 4 describes the functional architecture that underpins the NGAC standards.

Clause 5 defines the entities that comprise the functional architecture, their operation and their interfaces.

Clause 6 provides an overview of the NGAC standards family.

Annex A contains a bibliography of documents that provide background to this standard.

Annex B provides a retrospective background to access control.

Annex C contains a multistep example of the creation and administration of an NGAC configuration.

AMERICAN NATIONAL STANDARD BSR INCITS.***:201x

**American National Standard
for Information Technology -**

Next Generation Access Control - Functional Architecture (NGAC-FA)

1 Scope

NGAC follows an attribute-based construction in which characteristics or properties are used to describe and manage policy and to control access to resources. The family of NGAC standards specifies the architecture, functions, operations, and interfaces necessary to ensure their realization in different types of implementation environments at a range of scalability levels. This standard contains an abstract functional description of the NGAC architecture, and also provides an overview of the other standards within the NGAC family of standards. The description herein is abstract because it excludes irrelevant details, and is functional because it partitions the entities comprising the architecture purely on the basis of their function and excludes all other constraints.

2 Normative References

The following standards contain provisions that, by reference in the text, constitute provisions of this standard. At the time of publication, the editions listed were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the documents may be obtained from ANSI, an ISO member organization:

Approved ANSI standards;
Approved international and regional standards (ISO and IEC); and
Approved foreign standards (including JIS and DIN).

For further information, contact the ANSI Customer Service Department:

Phone: +1 212-642-4900
Fax: +1 212-302-1286
Web: http://www.ansi.org
E-mail: ansionline@ansi.org

or the InterNational Committee for Information Technology Standards (INCITS):

Phone 202-626-5738
Web: http://www.incits.org
E-mail: incits@itic.org

The following are approved references that pertain to this standard:

ACF: ISO/IEC 10181-3:1996, *Open Systems Interconnection - Security frameworks for open systems: Access control framework*

NGAC-GOADS: INCITS 526-2016, Information technology – *Next Generation Access Control - Generic Operations And Data Structures (NGAC-GOADS)*

The following are references under development:

NGAC-IRPAD: INCITS 525-201x, Information technology – *Next Generation Access Control - Implementation Requirements, Protocols and API Definitions (NGAC-IRPAD)*

Note that the status of the referenced American National Standards under development may have changed since the time of publication. For information about the current status of a document, or regarding availability, contact the relevant standards body.

3 Definitions, Symbols, Abbreviations, and Conventions

3.1 Definitions

3.1.1 access control: The prevention of unauthorized behavior, including the use of a resource in an unauthorized manner, which is regulated through a defined policy.

3.1.2 access control entry: A type of derived relation that for a specific policy element, governs which users and which operations those users may exercise on the policy element, in the absence of any relevant restrictions.

3.1.3 access request: Information that a process acting on behalf of a user initiates to accomplish an action affecting either a resource or the basic elements, containers and relations that comprise policy.

3.1.4 access right: A property that enables a user to perform operations either on objects representing resources or on information persisted in the PIP.

3.1.5 access right set: A subset of all access rights, whose members are related for the purposes of access control.

3.1.6 ascendant: A policy element that is contained by another policy element through a chain of assignments originating from the former to the latter.

3.1.7 assignment: A type of configured relation that establishes a correspondence between two policy elements.

3.1.8 association: A type of configured relation that establishes a basis for one or more privileges.

3.1.9 attribute: A type of container that may be either a user attribute or an object attribute.

3.1.10 authorization: The allocation of access rights to users and processes via the association and prohibition relations.

3.1.11 basic elements: A collective term that designates fundamental policy entities, namely users, processes, objects, operations, and access rights.

3.1.12 capability: A type of derived relation that for a specific user, governs the operations the user may perform on policy elements, in the absence of any relevant restrictions.

3.1.13 chain of assignments: A sequence of one or more assignments in which the second item of any assignment, except for the last assignment, is the same as the first item of the next assignment.

3.1.14 configured relation: An instance of information contained in the PIP representing a relationship among basic elements and containers.

3.1.15 container: A collective term that designates either a user attribute, an object attribute, or a policy class.

3.1.16 derived relation: An instance of information determined from one or more configured relations, which represents a relationship.

3.1.17 descendant: A policy element that contains another policy element through a chain of assignments originating from the latter to the former.

3.1.18 direct ascendant: A policy element that is contained by another policy element as a result of a single assignment.

3.1.19 direct descendant: A policy element that contains another policy element as a result of a single assignment between them.

3.1.20 event: An occurrence of the successful completion of an access request.

3.1.21 event context: The information about an event that is passed to the EPP from either the PEP or PDP, which includes the user ID, process ID, operation performed, operands used with the operation, and other pertinent information concerning the object or PIP-resident policy information that was accessed.

3.1.22 event pattern: A component of an obligation that defines when an event response is to be applied to the contents of the PIP.

3.1.23 Event Processing Point (EPP): The functional entity that matches event contexts it receives against event patterns in obligations, and conveys event responses from the matched obligations to the PDP for validation and processing.

3.1.24 event response: A component of an obligation that defines changes that are to be applied to the contents of the PIP when an event context is matched by the EPP to the obligation's event pattern.

3.1.25 object: An instance of information contained in the PIP which represents a system resource that is subject to access control (e.g., a file, printer, terminal or database record).

3.1.26 object attribute: A container defined by a unique identifier, whose ascendants are objects or other object attributes, and which represents an abstract characteristic of an object.

3.1.27 obligation: A type of configured relation that specifies an event pattern, an event response and a defining user, and allows the policy to be dynamically altered as a result of an event.

3.1.28 operation: A mediated action whose behavior affects either an object or policy information **persisted at the PIP.**

3.1.29 policy: A set of information that is the basis for rendering a decision to permit or deny an access request and for determining whether to initiate a response to an event.

3.1.30 Policy Administration Point (PAP): The functional entity that provides the only means to access and manage PIP-resident policy information.

3.1.31 policy class: A container defined by a unique identifier whose ascendants, other than another policy class, are policy elements that are associated with a particular access control policy.

3.1.32 Policy Decision Point (PDP): The functional entity that evaluates policy pertaining to an access request or an event response, renders an access decision and if favorable, carries out the administration access or the obligation event response respectively.

3.1.33 policy element: A collective term that designates either a user, an object or a container.

3.1.34 Policy Enforcement Point (PEP): The functional entity that performs access control by enforcing an access decision rendered by the PDP regarding an access request.

3.1.35 Policy Information Point (PIP): The functional entity that persists the data structures that represent the basic elements, containers and relations which comprise policy.

3.1.36 privilege: A derived relation that enables access requests to be performed by asserting the authorization of requisite access rights, barring any restrictions to the contrary.

3.1.37 process: A system entity that possesses a reliable identity, operates in a unique memory space and is associated with a single user within a single session.

3.1.38 prohibition: A type of configured relation that establishes a basis for one or more restrictions.

3.1.39 protected resource: A system resource represented by an object within an access control system.

3.1.40 relation: Either a configured relation or a derived relation.

3.1.41 resource: A protected resource.

3.1.42 Resource Access Point (RAP): A functional entity that provides the only method of access to certain protected resources.

3.1.43 restriction: A derived relation that prevents access requests from being performed by nullifying the authorization of requisite access rights.

3.1.44 session: A period during which an interactive information interchange or dialogue occurs between the processes of an authenticated user and a PEP.

3.1.45 unique identifier: A data structure or a component of a data structure that unambiguously distinguishes a particular identity.

3.1.46 user: An instance of information contained in the PIP which represents a human being that has been authenticated and is permitted to access the system.

3.1.47 user attribute: A container defined by a unique identifier, whose ascendants are users or other user attributes, and which represents an abstract characteristic of a user.

NOTE - The term “subject” is not used in NGAC, with both user and process being used as more precise terms in its place.

3.2 Symbols and acronyms

Symbol / Acronym	Meaning
ACE	Access Control Entry
ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programming Interface
DAC	Discretionary Access Control
EPP	Event Processing Point
MAC	Mandatory Access Control
MLS	Multi-Level Security
NGAC	Next Generation Access Control
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
RAP	Resource Access Point
RBAC	Role-Based Access Control
SoD	Separation of Duty

3.3 Keywords

3.3.1 invalid A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

3.3.2 mandatory A keyword indicating an item that is required to be implemented as defined in this standard to claim compliance with this standard.

3.3.3 may A keyword that indicates flexibility of choice with no implied preference.

3.3.4 optional A keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined by this standard is implemented, it shall be implemented as defined in this standard.

3.3.5 reserved A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. Their use and interpretation may be specified by future extensions to this or other standards. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

3.3.6 shall A keyword indicating a mandatory requirement. Designers are required to implement all such requirements to ensure conformance with this standard.

3.3.7 should A keyword indicating flexibility of choice with a preferred alternative; equivalent to the phrase “it is recommended”.

3.4 Conventions

Certain words and terms used in this American National Standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 3 or in the text where they first appear.

Numbers that are not immediately followed by lower-case b or h are decimal values.

Numbers immediately followed by lower-case b (xxb) are binary values.

Numbers immediately followed by lower-case h (xxh) are hexadecimal values.

Hexadecimal digits that are alphabetic characters are uppercase (i.e., ABCDEF, not abcdef).

Hexadecimal numbers may be separated into groups of four digits by spaces. If the number is not a multiple of four digits, the first group may have fewer than four digits (e.g., AB CDEF 1234 5678h)

An alphabetic list of items (e.g., a, b, c or A, B, C) indicates the items in the list are unordered.

A numeric list of items (e.g., 1, 2, 3) indicates the items in the list are ordered (i.e., item 1 shall occur or complete before item 2).

In the event of conflicting information the precedence for requirements defined in this standard is

1) text,
2) figures, then
3) tables.

4 Architecture

4.1 Functional architecture

Next Generation Access Control (NGAC) is reinvention of traditional access control into a form that suits the needs of the modern, distributed, interconnected enterprise. The NGAC framework is designed to be scalable, to support a wide range of access control policies, to enforce different types of policies simultaneously, to provide access control services for different types of resources, and remain manageable in the face change.

NGAC comprises a functional architecture that can be implemented in a number of ways, and a set of functions and data structures that allow a number of different access control schemes to be implemented using a common set of services. The NGAC functional architecture is shown in Figure 1.

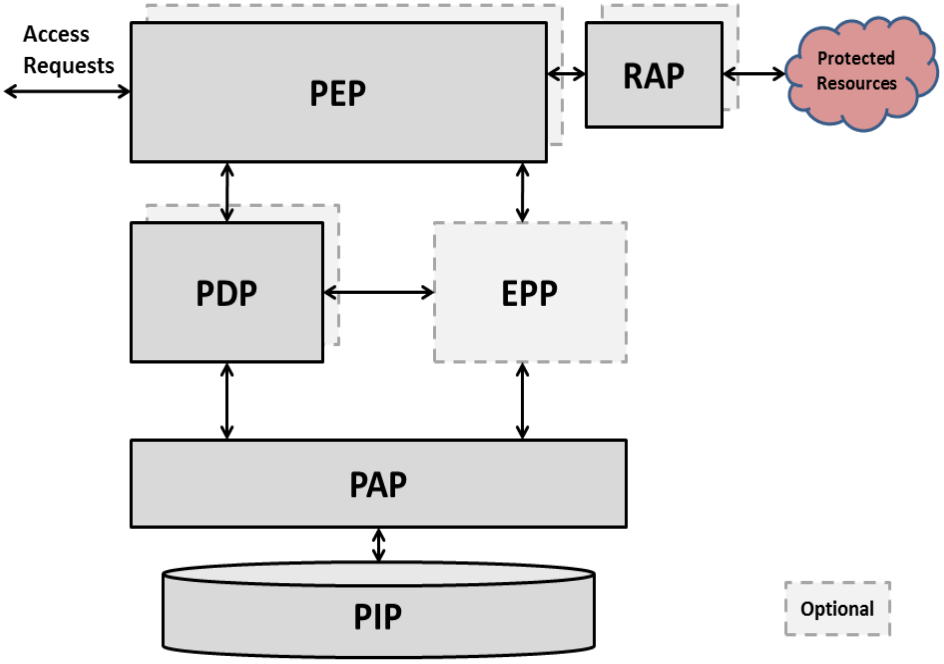

Figure 1: NGAC Functional Architecture

The NGAC Functional Architecture (FA) shall comprise the following functional entities:

a) one or more Policy Enforcement Points (PEPs);
b) one or more Policy Decision Points (PDPs);
c) zero or one Event Processing Point (EPP);
d) one Policy Administration Point (PAP);
e) one Policy Information Point (PIP); and
f) one or more Resource Access Points (RAPs).

4.2 Operational overview

4.2.1 Introduction

NGAC supports two fundamental types of accesses within the functional architecture. They are as follows:

a) resource access (see 4.2.2); and
b) administration access (see 4.2.3).

The descriptions of resource and administration accesses given below assume that an authenticated user has established a session with a PEP, and that all access requests can be securely identified with that session. A user can have only one session active at any time, but may have multiple processes operating on its behalf within that session.

Least privilege is an established administrative practice of assigning users and processes the minimal authorization necessary for the performance of their job function, and no more. NGAC supports the concept of least privilege for both resource and administration access, by allowing different authorizations for a user or its processes to become available at different times for the performance of different tasks. Processes acting for a user within the user's session can be restricted to a subset of the user's authorization, allowing them to be attenuated at a granularity below that of the user.

4.2.2 Resource Access

A resource access is the only way in which users shall gain access to protected resources in NGAC. A resource access begins when a user launches a client application, creating a process that attempts access to a resource via a PEP. Processes run on behalf of a specific user within a single session, and may instantiate other processes. Each process of a user shall use the PEP to which the user has formed a session to request access to a protected resource. This is the only way a process shall be able to gain access to protected resources in NGAC.

A resource access references an object associated with the protected resource targeted. NGAC supports conceptual operations, such as read/write or on/off/reset, against protected resources. The behavior of resource operations is not defined by NGAC. The actual location and required routing to the resource are maintained within the PIP. Protected resources may have different sensitivities and other characteristics designated by attributes, for which policy can be formulated to protect against the leakage of information.

The function of a PEP is to ensure that only those requests for access that meet specific requirements are granted access to the protected resources. The PEP submits individual access requests to a PDP for adjudication. The PDP then obtains additional details necessary to adjudicate an access request by retrieving authorization information related to the request from the PIP, via queries issued to the PAP.

The PDP renders an access decision based on:

a) the existence of all requisite privileges for the access request; and
b) the absence of any restrictions that countermand a requisite privilege.

An access request is granted by the PDP if (a) and (b) are both satisfied. The access request is denied for all other possibilities.

If an access request is granted, the PDP retrieves information for locating the resource identified in the request (i.e., via the identifier of the associated object, translated into a specific system resource at a specified location associated with a specified RAP) and conveys the location information obtained, along with the decision results, to the PEP from which it received the request. The PEP communicates with a specific RAP to perform the operation specified in the access request on the resource identified. Once the operation is complete, the PEP then returns the status information about the operation, and optionally data resulting from it, to the originating process.

If an EPP is present in the system, the PEP generates an event context for each successfully completed resource access, and forwards it to the EPP. The EPP uses the PAP to match the event context against the event patterns of obligations stored in the PIP. For each match, the EPP conveys the event response from the obligation to a PDP for validation and processing, for which the PDP uses the PAP. The resulting changes to the policy information in the PIP can affect the access decisions on future access requests.

4.2.3 Administration Access

An administration access is the only way in which users shall gain access to policy information in NGAC. Similar to a resource access, an administration access begins when a user launches a client application, creating a process that attempts access to policy information by conveying an administrative operation and the required operands to a PEP. A user's processes shall use the PEP to which the user has formed a session to request access to policy information. This is the only way a process shall be able to gain access to policy information in NGAC. NGAC administrative operations are used to create and destroy the basic elements, containers and relations that are maintained by the PIP, and as a consequence, change the policies enforced by the NGAC framework.

Client applications launched by a user are able to administer the contents of the PIP, provided that sufficient authorization is held by the user. Policies governing administration responsibilities can be centralized to a single authority or decentralized to allocate responsibilities selectively among multiple authorities.

For each administrative access attempt, the PEP in turn submits an individual access request to a PDP for adjudication. Administration access requests are treated slightly differently than resource access requests by the NGAC architecture, however. The main difference is the way in which the request is processed by the PDP once the access decision is computed. If a grant decision is rendered, the PDP takes an additional step to carry out the access before returning the results to the PEP. The PDP communicates with the PAP to perform the administration operation given in the access request on the relevant policy information maintained by the PIP. For decisions other than a grant decision, the results are returned to the PEP, similar to that done for a resource access.

If an EPP is present in the system, the PDP generates an event context for each successfully completed administration access, and forwards it to the EPP. The EPP uses the PAP to match the event context against the event patterns contained in obligations stored in the PIP, and processes the event responses using a PDP (not necessarily the same one that generated the event context) in the same way as done for a resource access.

4.3 Information flows

4.3.1 Introduction

There are three sets of major information flows in the NGAC functional architecture:

1) a resource access flow (see 4.3.2);
2) an administration access flow (see 4.3.3); and
3) an optional event context flow (see 4.3.4).

The first two information flows above result from the two types of accesses described earlier (see 4.2.1). The third flow only occurs in a system where an EPP is present, and is initiated as a result of a successful resource or administration access.

All of the information flow descriptions that follow assume that a session is in place between an authenticated user and the PEP, and that all interacting entities in the functional architecture have established communication and authenticated with each other.

4.3.2 Resource Access Information Flow

The resource access information flow within the NGAC architecture is shown in Figure 2.

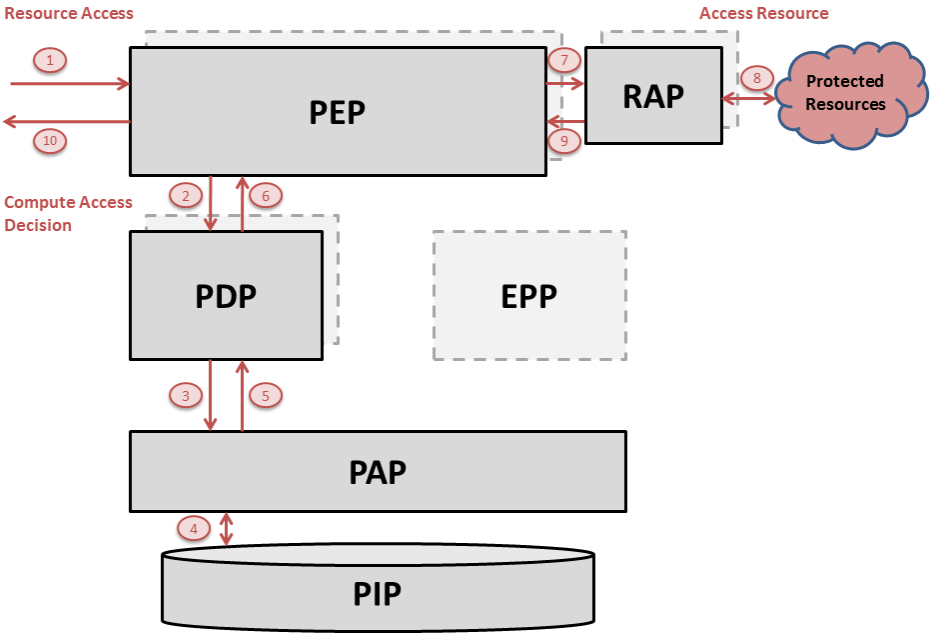

Figure 2: NGAC Resource Access Information Flow

The numeric labels in Figure 2 refer to the step number in the following description.

The resource access information flow through the NGAC functional architecture is as follows:

1) A process attempts to gain access to a resource via a PEP, conveying the intended operation and the associated operand(s) for the operation, including the object ID designating the resource, and any needed data;
2) The PEP issues an access request to a PDP for adjudication;
3) The PDP queries the PAP for information contained in the PIP, which is needed to compute an access decision;
4) The PAP validates the query and issues one or more corresponding commands to the PIP and receives the responses from the PIP;
5) The PAP returns the queried information to the PDP;
6) The PDP computes a decision (the result may be Yes, No or some error), and if a positive decision is rendered, resolves the resource location and returns the decision and locator to the PEP;
7) The PEP issues a directive to the RAP associated with the resource to carry out the access, which conveys the operation, the resource locator and, if required, data;
8) The RAP initiates the operation on the resource and receives the status information and data (if any) returned;
9) Status information and data (if any) are returned from the RAP to the PEP; and
10) Status information and data (if any) are returned from the PEP to the process.

4.3.3 Administration Access Information Flow

The administration access information flow within the NGAC architecture is shown in Figure 3.

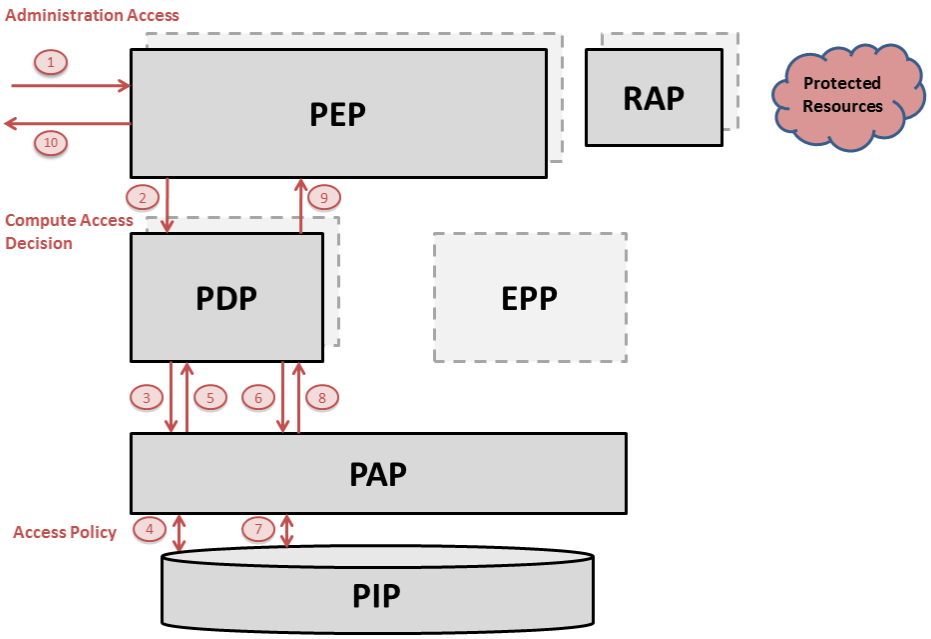

Figure 3: NGAC Administration Access Information Flow

The numeric labels in Figure 3 refer to the step number in the following description.

The administration access information flow through the NGAC functional architecture is as follows:

1) A process attempts to gain access to policy information via a PEP, conveying the intended operation and the operands for that operation;
2) The PEP issues an access request to a PDP for adjudication;
3) The PDP queries the PAP for information contained in the PIP, which is needed to compute an access decision;
4) The PAP validates the query and issues one or more corresponding commands to the PIP and receives the responses returned;
5) The PAP returns the queried information to the PDP, which computes a decision (the result may be Yes, No, or some error) and, if a positive decision is rendered, determines the policy information affected by the request;
6) The PDP issues a directive to the PAP to carry out the administration access, which conveys the operation and operands needed to adjust policy;
7) The PAP executes one or more commands against the PIP, and receives the status information and data (if any) returned;
8) Status information and data (if any) are returned from the PAP to the PDP;
9) Status information and data (if any) in turn are returned from the PDP to the PEP; and
10) Status information and data (if any) are returned from the PEP to the process.

Steps (1) thru (4) in the above flow are identical to the resource access information flow defined in 4.3.2.

4.3.4 Event Context Information Flow

The event context information flow within the NGAC architecture, which occurs only when an EPP functional entity is present, is shown in Figure 4.

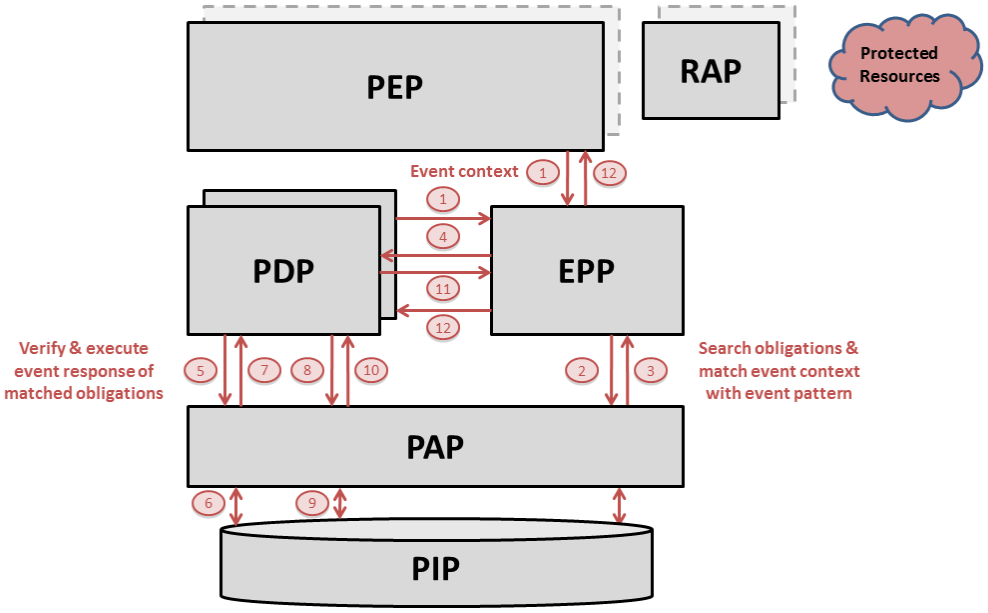

Figure 4: NGAC Event Context Information Flow

The numeric labels in Figure 4 refer to the step number in the following description. Note that unlike the previous information flows, two PDPs are depicted for clarity: one that generates an event context for the EPP, and another that assists the EPP in processing any responses triggered by a matched obligation.

The optional event context information flow through the NGAC functional architecture is as follows:

1) A PEP or a PDP generates an event context that respectively indicates that a resource or an administration access has been successful, and sends the event context to the EPP;
2) The EPP issues one or more directives to the PAP to search all of the obligations stored in the PIP with event patterns that match the event context;
3) The PAP validates the directives(s), issues corresponding commands to the PIP, receives the data and status information (if any) returned by the PIP, which identifies all relevant obligations, and returns this list to the EPP;
4) For each obligation in the list, the EPP issues the obligation's event response to the PDP to be validated and executed;
5) The PDP queries the PAP for information contained in the PIP, which is needed to compute an access decision;
6) The PAP validates the query and issues one or more corresponding commands to the PIP and receives the data and status information (if any) returned;
7) The PAP returns the queried information to the PDP, which computes a decision about the event response;
8) The PDP issues a sequence of directives to the PAP, which convey the operations and operands needed to carry out the event response;
9) The PAP executes one or more commands against the PIP to carry out the directives, and receives the status information and data (if any) returned;
10) Status information and data (if any) are returned from the PAP to the PDP;
11) Status information and data (if any) are returned from the PDP to the EPP; and
12) The EPP returns status information and data (if any) to the PEP or PDP that issued the event context.

The effect of the event context information flow is to modify policy information within the PIP, which is used as the basis for access control decisions. The modifications are dependent on the event response of a matched obligation, and occur dynamically.

The resulting policy changes can affect subsequent decisions about the protected resource or policy information items targeted in the access information flow that triggered the event information flow, as well as decisions about other resources and policy information related to the resource or policy information targeted in the access information flow. Consequentially, an inherent race condition exists in the architecture between the policy changes occurring as a result of the event information flow, and the processing of subsequent or concurrent access requests. This race condition needs to be addressed by an implementation in a manner suitable for its execution environment. The manner in which the race condition is addressed is not prescribed by NGAC.

4.4 Interfaces

The NGAC Functional Architecture is designed to be able to accommodate a number of different situations using a variety of approaches. An implementation of the NGAC Functional Architecture is not required to follow a completely centralized or completely distributed approach. Many types of hybrid configurations are also possible where some of the entities within the architecture reside together within a single system, but others are located in other systems.

The entities within the architecture are described in terms of their functional behavior to allow flexibility for implementation. The interfaces between entities enable implementations to provide the requisite level of cohesion and functionality at the system level. The functional architecture shown in Figure 1 incorporates the following interfaces:

a) A PEP (see 5.2) exposes a set of interfaces for NGAC-aware applications to access resources and policy information;
b) A PDP (see 5.3) exposes a standard set of interfaces for a PEP to request a decision regarding a resource or administration access and receive the decision results, and for the EPP to request a decision regarding the event response of a matched obligation, to have the response carried out if deemed valid, and to receive the decision results;
c) The EPP (see 5.4) exposes a standard set of interfaces for a PEP or a PDP to indicate the occurrence of an event and to communicate its context;
d) The PAP (see 5.5) exposes a standard set of interfaces for a PDP to request information on which to base its decisions, for the EPP, if present, to match an event context with the event pattern of defined obligations, and for a PDP to manage the contents of the PIP;
e) The PIP (see 5.6) exposes a set of interfaces for the PAP to manipulate the basic elements, containers and relations that are persisted by the PIP; and
f) A RAP (see 5.7) exposes a set of interfaces for a PEP to access protected resources and transfer data as necessary to and from those resources.

The NGAC Functional Architecture is defined to be suitable for implementation both within a single computer system that acts as a gateway to a set of resources, and also across a large number of systems interconnected by a variety of networks. In the former case the communication between the functional entities is contained within a computer system, but in the latter case the communication flows across the external network. To optimize the former case, key inter-entity communications interfaces are defined by NGAC in terms of Application Program Interfaces (APIs). To also support the latter case, those same API communications interchanges may be realized in terms of a network protocol. Either the API or the protocol version of each API may be used between entities in an implementation.

See NGAC-IRPAD for more details of these interfaces.

See NGAC-GOADS for more details of the operations and data structures supported by the PIP, which underpin these interfaces.

5 Functional Entities

5.1 Overview

This clause specifies requirements for the operation of each entity contained in the architecture block diagram shown in Figure 1.

5.2 Policy Enforcement Point (PEP)

The PEP shall regulate access attempts from processes to NGAC policy information and protected resources. Each attempt shall convey a process ID, operation, operands of the operation, and any additional data needed. The PEP shall issue an access request to a PDP, which equates to the attempted access.

A PEP shall ensure that only access requests for which a PDP has rendered a grant decision are permitted access to protected resources or policy information. If the PDP replies that a request is granted, the PEP shall take one of two possible courses of action. For a resource access, the PEP shall transmit a command to a RAP, and upon completion of the operation, return the status information and any data resulting from the operation to the originating process. For an administration access, the PEP shall accept the decision rendered by the PDP and the accompanying status information and results from actions taken by the PDP, and return the status information and any data resulting from the operation to the originating process.

A successful resource access request shall cause the PEP to generate an event context characterizing the access and forward it to the EPP, if there is an EPP present in the system.

5.3 Policy Decision Point (PDP)

A PDP shall determine if an access request submitted by a PEP is to be granted. A PDP shall adjudicate the access request by retrieving authorization information pertaining to the request from the PIP via the PAP.

The decision to grant the access request shall be rendered if and only if:

a) privileges exist that match those required for the process to effectuate the access request; and
b) no restriction exists that negates a requisite privilege, for either the process requesting access or the user of the process.

Otherwise, the access request shall be denied.

For a resource access, the PDP shall communicate the results of its access decision to the PEP from which it received the request. If the access request is granted, the PDP shall also obtain the information necessary to locate the resource to which access is being requested, and communicate the locator for the resource to the PEP along with the decision results.

For an administration access, the PDP shall carry out the requested access, if granted, and communicate the results of its access decision, and any subsequent processing taken, to the PEP from which it received the request. A successful administration access request shall cause the PDP to generate an event context characterizing the access and forward it to the EPP, if an EPP is present in the system.

A PDP shall determine if an event response submitted by the EPP is to be processed. The PDP shall obtain authorization information pertaining to the event response from the PIP via the PAP. The PDP shall render an access decision about the event response, and confirm that the authorization held by the

creator of the obligation is sufficient to carry out the response. If the access decision is positive, the PDP shall process the event response. The PDP shall communicate the results of its decision, and any subsequent processing taken, to the EPP.

The event response shall be processed as an atomic transaction by the PDP. Either all of the constituent parts of the response succeed, or nothing succeeds. The result is that either the response is carried out completely, or no part of the response is carried out.

The PDP shall not generate an event context for any of the policy changes that occur in processing the event response of an obligation.

5.4 Event Processing Point (EPP)

The EPP shall respond to event contexts generated by either a PEP or a PDP. An event context received by the EPP shall include information identifying the user, the process, the operation performed, and the operands of the operation. It may also include information about the resource or PIP-resident policy information that was accessed. The EPP shall use this information to match the event context against the event pattern of each obligation stored in the PIP.

For each matched obligation, the EPP shall communicate with a PDP to confirm that the authorization held for the obligation's response is sufficient and to process the event response accordingly. The event response is performed against the PIP via the PAP, which may affect the outcome of future access request decisions, because of the alterations made by the response to associated policy information.

The EPP is an optional entity of the functional architecture, needed only if the policy specified utilizes obligations.

5.5 Policy Administration Point (PAP)

Access to administrative information persisted at the PIP shall be performed exclusively via the PAP. The PAP shall regulate access to the PIP from PDPs and the EPP. The PAP shall limit an EPP to read-only access to the PIP, but shall permit a PDP read and other types of access. The PAP shall provide methods that allow administration of the contents of the PIP and also the search and retrieval of information that is needed to render an access decision or carry out the response of an obligation.

The PAP acts as a managed access point through which the information persisted in the PIP is accessed in an administrative access, similar to the way that a RAP acts as a managed access point to protected resources in a resource access.

5.6 Policy Information Point (PIP)

The PIP shall incorporate a data store that contains information which constitutes the access control policy for the system. This information comprises basic elements, containers and relations, and may also include other policy-related entities that facilitate implementation. The data store shall support ACID semantics.

5.7 Resource Access Point (RAP)

A RAP shall provide a managed access point through which a set of protected resources may be accessed by one or more PEPs. Each protected resource shall only be accessible via a RAP, and only via a single RAP at any one time. A RAP shall not provide access to resources to any entity other than a PEP.

6 NGAC Standards Family

6.1 Introduction

The NGAC family of standards provides the architectural, functional and interface definitions necessary to create a full-featured access control system. The family contains a sufficient level of detail to allow equipment realizing the functional entities defined in the previous clause to be implemented separately and procured from separate vendors.

This standard is a functional description of the architecture of the NGAC access control system. It forms the foundation for the remaining standards in the NGAC standards family:

a) Generic Operations and Data Structures; and
b) Implementation Requirements, Protocols and API Definitions.

6.2 NGAC Generic Operations and Data Structures (GOADS)

6.2.1 Overview

The NGAC-GOADS standard defines a number of fundamental constructs that are used to describe the operation of the NGAC functional architecture. The core fundamental constructs are as follows:

a) a set of basic elements (see 6.2.2) that are maintained by the PIP to represent the fundamental entities in which operation of the NGAC framework is described;
b) a set of containers (see 6.2.3) of different types that are maintained by the PIP to represent the characteristics of basic elements and to define basic elements that share those characteristics; and
c) a set of relations (see 6.2.4) that are maintained by the PIP to represent configured relationships among basic elements and containers, which form the basis of the policies enforced by the NGAC functional architecture.

The NGAC-GOADS standard also defines a complete set of administrative commands, which specify the behavior that occurs in the creation, deletion and use of NGAC basic elements, containers and relations, and the effect of the behavior on the authorization state of the policy.

6.2.2 Basic elements

6.2.2.1 Overview

The basic elements in terms of which the operation of the NGAC Functional Architecture is defined are described in this subclause as follows:

a) users (see 6.2.2.2);
b) processes (see 6.2.2.3);
c) objects (see 6.2.2.4);
d) operations (see 6.2.2.5); and
e) access rights (see 6.2.2.6).

6.2.2.2 Users

Users in NGAC are unique entities that represent human beings that have been authenticated by an authentication service. Details of user authentication are outside the scope of NGAC-GOADS. An authenticated user shall establish a session with a PEP prior to submitting access requests to the PEP via a process.

A user is distinguished by a unique identifier (user ID) that identifies it within the scope of an instance of the NGAC functional architecture. Other information about the user, such as references to the attributes to which the user is assigned, the method by which the user is authenticated, etc. may be maintained within the NGAC framework.

6.2.2.3 Processes

Processes in NGAC are system entities that possess a reliable identity and operate in a unique memory space. Authenticated users do not initiate access requests directly, but instead create one or more processes that initiate requests for them. A user may be associated with one or more processes, while a process shall be associated with just one user. Differentiating between users and processes in NGAC allows for the creation of fine-grained access restrictions.

A unique identifier (process ID) identifies a process within the scope of an instance of the NGAC functional architecture. Other information about the process, such as a reference to the user associated with the process, may be maintained within the NGAC framework.

6.2.2.4 Objects

Objects in NGAC represent resources to which access is controlled. These resources can be either external types such as files, messages, database records and devices, or internal types such as policy items within the PIP.

When a process associated with an authenticated user makes a connection with the PEP for the first time, it may execute a query or browse to discover the objects for which the user has authorization to access.

An object is identified by a unique identifier (object ID) within an instance of the NGAC functional architecture. Other information about the object, such as and references to the attributes to which the object is assigned may be maintained within the NGAC framework.

When an object is created, it is regarded as both an object and an object attribute by NGAC. That is, the identifier of the object may not only be treated as an object within NGAC relations, but may also be treated as an object attribute, based on its context within a relation.

6.2.2.5 Operations

Operations in NGAC denote actions performed on elements of policy, which represent either a protected resource or information persisted with the PIP. Resource operations are those used to gain access to and manipulate resources associated with a RAP. Administrative operations are those used to gain access to and manipulate basic elements, containers and relations maintained by the PIP. A single access request containing an operation may involve multiple commands with the targeted entities. A specific operation is represented in NGAC by a unique identifier.

6.2.2.6 Access Rights

Access rights in NGAC enable actions to be performed on elements of policy, which may represent a protected resource or information persisted with the PIP. A single action denoted by an access request may require one or more access rights to be enabled. A specific access right is represented in NGAC by a unique identifier.

6.2.3 Containers

6.2.3.1 Overview

NGAC containers are a part of the information persisted in the PIP that comprise the policy for the NGAC Functional Architecture. NGAC containers denote common characteristics or properties of the items they contain, and are described in this subclause as follows:

a) user attribute (see 6.2.3.2);
b) object attribute (see 6.2.3.3); and
c) policy class (see 6.2.3.4).

Container memberships are established through one or more assignments (see 6.2.4.2) (e.g., the user “Bob” “is in” the “sysadmins” container, if and only if a chain of assignments exists from “Bob” to “sysadmins”).

6.2.3.2 User attribute

A user attribute container defines a membership on the basis of an abstract user characteristic or property that is significant in determining access control. That characteristic may be membership in an organizational entity, a geographic location, performance of a specific role (e.g., bank teller), a clearance level etc.

The members of a user attribute may be users or other user attributes. A user or user attribute “is contained by” or “is in” a user attribute if a chain of assignments exists from that user or user attribute to the user attribute.

A user attribute is identified by a unique identifier.

6.2.3.3 Object attribute

An object attribute container defines a membership on the basis of an abstract object characteristic or property that is significant in determining access control. That characteristic may be related to the relative importance of the contained information (e.g., proprietary), a geographic location, a resource type (e.g., DBMS record, tape volume) etc.

The members of an object attribute may be objects or other object attributes. An object or object attribute “is contained by” or “is in” an object attribute if a chain of assignments exists from that object or object attribute to the object attribute.

An object attribute is identified by a unique identifier.

6.2.3.4 Policy class

A policy class defines membership related to an access control policy, such as RBAC or MLS. The members of a policy class may be users, user attributes, objects, or object attributes. A policy class is identified by a unique identifier.

NGAC supports multiple policy classes simultaneously, with each policy class normally pertaining only to access between a subset of users and objects. A user ”is contained by” or “is in” a policy class if a chain of assignments exists from that user to a user attribute to the policy class. A user attribute “is contained by” or “is in” a policy class if a chain of assignments exists from that user attribute to the policy class. Similarly an object or an object attribute “is contained by” or “is in” a policy class if a chain of assignments exists from that object or object attribute to the policy class.

6.2.4 Relations

6.2.4.1 Overview

NGAC relations are either configured relations or derived relations. NGAC configured relations are a part of the information persisted in the PIP that are managed through administrative operations, and determine the configuration of an NGAC Functional Architecture. NGAC configured relations represent relationships among basic elements and containers, and are described in this subclause as follows:

a) assignment (see 6.2.4.2);
b) association (see 6.2.4.3);
c) prohibition (see 6.2.4.4); and
d) obligation (see 6.2.4.6).

NGAC derived relations are calculated from configured relations for the purpose of rendering an access control decision or conducting an administrative review, and are described in this subclause as follows:

a) privilege (see 6.2.4.3.2);
b) capability (see 6.2.4.3.3);
c) access control entry (see 6.2.4.3.4); and
d) restriction (see 6.2.4.5).

6.2.4.2 Assignment

The assignment relation defines membership within containers. An individual assignment involves a pair of policy elements. Policy elements comprise the set of all defined users, user attributes, objects, object attributes and policy classes.

An assignment between policy elements is restricted to one of the following types:

a) A user may be assigned to a user attribute;
b) A user attribute may be assigned to another user attribute;
c) An object may be assigned to an object attribute;
d) An object attribute may be assigned to another object attribute;
e) A user attribute may be assigned to a policy class; or
f) An object attribute may be assigned to a policy class.

The attributes of users and objects are established through assignments. Each tuple of the assignment relation indicates that the first item mentioned in each case acquires the capabilities or properties of the second item mentioned. The first item is the direct ascendant of the second item, and the second item is the direct descendant of the first item.

Assignments of type (a) in the above list show that a user attribute may be viewed as a “container” of users. Thus a user can be said to be ”contained by” or “in” an attribute, if a chain of assignments consisting of one type (a) and zero or more type (b) assignments exist that links the user with that user attribute.

Assignments of type (c) in the above list shows that an object attribute may be viewed as “container” of objects. Thus an object can be said to be “contained by” or “in” an attribute if a chain of assignments consisting of one type (c) and zero or more (d) assignments exists that links the object with that object attribute.

Assignments of types (b) and (d) in the above list are used to create attribute hierarchies, which is a powerful feature of NGAC that simplifies the management of complex configurations. Those assignment types may only be created if they do not create “loops” (e.g., if item A is assigned to item

B, then an assignment of item B to item A is disallowed) or “cycles” (e.g., if item A is assigned to item B and item B is assigned to item C, then an assignment of item C to item A is disallowed).

The assignments of item A to item B, and item B to item C constitute a valid chain of assignments. Item A is an ascendant of item C, and item C is a descendant of item A.

6.2.4.3 Association

6.2.4.3.1 Definition

The association relation defines authorized modes of access to information, and is the foundation from which other relations used in access decisions are derived. An association consists of a tuple of three parts, as follows:

1) one user attribute;
2) one access right set; and
3) one attribute.

An access right set contains a set of unique identifiers each identifying an access right. Access rights within an access right set are related for the purposes of access control. An access right set is identified by a unique identifier.

The following three relations are derived chiefly from the association relation:

1) privilege;
2) capability; and
3) access control entry (ACE).

6.2.4.3.2 Privilege

The privilege relation is derived from the association and assignment relations, and consists of tuples of three parts, as follows:

1) one user;
2) one access right; and
3) one policy element.

Each privilege tuple indicates that the user has permission to carry out a mode of access denoted by the access right against the policy element, unless a restriction (see 6.2.4.5) exists that prevents access.

The privilege relation is derived as follows: the tuple (user, access right, policy element) is a privilege, if and only if for each policy class that contains the policy element, there exists an association such that:

a) the user is contained by the user attribute of the association;
b) the access right is a member of the access right set of the association;
c) the policy element is contained by the attribute of the association (i.e., the third term); and
d) the attribute of the association is contained by that policy class.

6.2.4.3.3 Capability

The capability relation is derived from the privilege relation, and consists of tuples of two parts, as follows:

a) one access right; and
b) one policy element.

A user holds the capability described by the tuple if and only if a privilege (see 6.2.4.3.2) exists that specifies that user, access right and policy element.

6.2.4.3.4 Access Control Entry

The Access Control Entry (ACE) relation is derived from the privilege relation, and consists of tuples of two parts, as follows:

a) one user; and
b) one access right.

A policy element holds the ACE described by the tuple if and only if a privilege (see 6.2.4.3.2) exists that specifies that user, access right and policy element.

6.2.4.4 Prohibition

Prohibition relations define one or more capabilities that are denied to a user, process or group of users and countermand any corresponding authorization. Three types of prohibition relations exist: user-based, which applies to a specific user and affects all of its associated processes; user attribute-based, which applies to a specific user attribute and affects all processes whose users are ascendants of the attribute; and process-based, which applies to and affects only a specific process. Unlike process-based prohibitions, which are deleted automatically when the associated process terminates, user-based and user attribute-based prohibitions persist indefinitely.

The user-based prohibition relation consists of tuples of four parts, as follows:

1) one user;
2) one access right set;
3) one inclusive attribute set; and
4) one exclusive attribute set.

The user attribute-based prohibition relation consists of tuples of four parts, as follows:

1) one user attribute;
2) one access right set;
3) one inclusive attribute set; and
4) one exclusive attribute set.

The process-based prohibition relation consists of tuples of four parts, as follows:

1) one process;
2) one access right set;
3) one inclusive attribute set; and
4) one exclusive attribute set.

The attribute sets referenced in the above definitions are defined in terms of referent attributes that denote the presence or absence of one or more policy elements for the relation. The inclusive attribute set delineates policy elements that are ascendants of the attributes in the set and subject to the prohibition. Similarly, the exclusive attribute set delineates policy elements that are not ascendants of the attributes in the set and subject to the prohibition.

The existence of a prohibition relation tuple indicates that any process, either initiated for the specified user, initiated for a user contained by the specified user attribute, or having the same identifier as that of the specified process, is prohibited from carrying out certain operations against any policy element designated by the inclusive and exclusive attribute sets, if the operations require an access right in the specified access right set to execute.

Prohibition relations can be either logically disjunctive or conjunctive. The former relations target policy elements that are either contained by a member of the inclusive attribute set or not contained by a member of the exclusive attribute set. The latter target policy elements that are contained by every member of the inclusive attribute set and not contained by every member of the exclusive attribute set.

6.2.4.5 Restriction

A restriction relation is derived from conjunctive and disjunctive prohibition relations of the same type: user-based, user attribute-based, or process-based.

The user-based restriction relation consists of a tuple of three parts, as follows:

1) one user;
2) one access right; and
3) one policy element.

A user restriction relation is derived from one or more user prohibition relations as follows: the tuple (user, access right, policy element) is a restriction, if and only if there exists a prohibition such that:

a) the user is the same as that of the prohibition;
b) the access right is a member of the access right set of the prohibition; and;
c) for a disjunctive prohibition, the policy element is contained by one or more members of the inclusive attribute set of the prohibition or it is not contained by any of the members of the exclusive attribute set of the prohibition; or;
   for a conjunctive prohibition, the policy element is contained by all of the members of the inclusive attribute set of the prohibition and it is not contained by any of the members of the exclusive attribute set of the prohibition.

A user attribute-based restriction data relation consists of a tuple of three parts, as follows:

1) one user attribute;
2) one access right; and
3) one policy element.

A user attribute restriction relation is derived from one or more prohibition relations as follows: the tuple (user, access right, policy element) is a restriction, if and only if there exists a prohibition such that:

a) the user attribute is the same as that of the prohibition;
b) the access right is a member of the access right set of the prohibition; and;
c) for a disjunctive prohibition, the policy element is contained by one or more members of the inclusive attribute set of the prohibition or it is not contained by any of the members of the exclusive attribute set of the prohibition; or
   for a conjunctive prohibition, the policy element is contained by all of the members of the inclusive attribute set of the prohibition and it is not contained by any of the members of the exclusive attribute set of the prohibition.

A process-based restriction data relation consists of a tuple of three parts, as follows:

1) one process;
2) one access right; and
3) one policy element.

A process restriction relation is derived from one or more prohibition relations as follows: the tuple (user, access right, policy element) is a restriction, if and only if there exists a prohibition such that:

a) the process has the same identifier as that of a process operating on behalf of a user denoted by the user of the prohibition;
b) the access right is a member of the access right set of the prohibition; and;
c) for a disjunctive prohibition, the policy element is contained by one or more members of the inclusive attribute set of the prohibition or it is not contained by any of the members of the exclusive attribute set of the prohibition; or
for a conjunctive prohibition, the policy element is contained by all of the members of the inclusive attribute set of the prohibition and it is not contained by any of the members of the exclusive attribute set of the prohibition.

The existence of a restriction data relation indicates that any process either initiated for the specified user, initiated for a user contained by the specified user attribute, or possessing the specified process identifier, is prohibited from carrying out certain operation against the policy element, if the operation requires an access right in the specified access right set to execute.

6.2.4.6 Obligation

The obligation relation consists of tuples of three parts:

1) one user (i.e., the creator of the obligation);
2) an event pattern; and
3) an event response.

The event pattern is a statement of conditions related to an event. The event response defines a sequence of administrative actions that specify changes to be effected against the contents of the PIP. A conditional expression may apply to each action to allow flexible execution of the response.

Obligations are triggered by events. The event context contains information about an event. It is produced from details about a process' execution of an operation affecting a protected resource or policy information persisted in the PIP, and includes the following information pertaining to the access:

a) the identifier of the process;
b) the identifier of the user of the process;
c) the operation executed;
d) the identifier(s) of the affected resource or policy information;
e) optionally, direct descendants of the user; and
f) optionally, direct descendants of the operands used to reference the resource or policy information affected by the operation.

Obligations are contingent on the presence of the EPP, which is an optional entity of the functional architecture. An obligation relation defines changes to policy that are triggered by the EPP's receipt of an event context which matches the event pattern of an obligation. Execution of the event response shall occur only if the user that created the obligation holds sufficient authorization to carry out the response. Therefore, it is necessary for users that have created one or more obligations to remain active in the policy configuration for their associated obligations to function.

The event context of a successful access request may match more than one obligation and trigger the execution of multiple responses corresponding to each obligation matched. The order in which obligations are matched and event responses processed in such situations is not prescribed by NGAC. However, all event responses triggered by an event context shall be processed in their entirety before those triggered by another event context.

6.2.5 Commands

An administrative command is performed to carry out a valid administrative access request for a user. An administrative command may also be performed as part of the response for a defined obligation that has been triggered.

Access requests bearing administrative operations can create and destroy basic elements, containers and relations maintained by the PIP. Each administrative operation is carried out within the framework through one or more primitive administrative commands. The NGAC-GOADS standard defines the complete set of administrative commands and their behavior in detail. The definitions specify the preconditions that need to exist for the effect of a command to occur, and the specific effect that the command has on the contents of the PIP.

The following administrative commands are supported by NGAC:

a) create basic elements and containers:
   - user in a user attribute;
   - object in an object attribute;
   - user attribute in a policy class;
   - user attribute in a user attribute;
   - object attribute in a policy class;
   - object attribute in an object attribute;
   - policy class;
   - process;
   - operation; and
   - access right.

b) delete basic elements and containers:
   - user;
   - user attribute;
   - policy class;
   - object;
   - object attribute;
   - association;
   - process;
   - operation; and
   - access right.

c) create and delete relations and their components:
   - access right set;
   - inclusive attribute set;
   - exclusive attribute set;
   - event pattern;
   - event response;
   - assignment;
   - association;
   - user-based prohibition;
   - process-based prohibition;
   - attribute-based prohibition; and
   - obligation.

6.3 NGAC Implementation Requirements, Protocols and API Definitions (IRPAD)

6.3.1 Overview

The NGAC-IRPAD standard contains the information needed to realize the NGAC Functional Architecture and obtain the requisite level of cohesion and functionality to interoperate effectively at the system level. This information is as follows:

a) Protocols and API definitions (see 6.3.2); and
b) Implementation requirements (see 6.3.3).

NGAC does not require a particular style of implementation to be used throughout, and is specifically defined to permit a range of implementation styles from fully centralized through fully distributed designs.

6.3.2 Protocols and API Definitions

The NGAC Functional Architecture incorporates the following set of interfaces between its functional entities:

a) A PEP exposes a set of interfaces for use by NGAC-aware applications to access resources and policy information;
b) A PDP exposes a standard set of interfaces for use by a PEP to request a decision regarding a resource or administration access and receive the decision results, and for use by the EPP to request a decision regarding the event response of a matched obligation, to have the response carried out if deemed valid, and to receive the decision results;
c) The EPP exposes a standard set of interfaces for use by a PEP or a PDP to indicate the occurrence of an event and to communicate its context;
d) The PAP exposes a standard set of interfaces for use by a PDP to request information on which to base its decisions, for use by the EPP to match an event context with the event pattern of defined obligations, and for use by a PDP to manage the contents of the PIP;
e) The PIP exposes a set of interfaces for use by the PAP to manipulate the basic elements, containers and relations that are persisted by the PIP; and
f) A RAP exposes a set of interfaces for use by a PEP to access protected resources and transfer data as necessary to and from those resources.

The interfaces between key functional entities, namely items (b), (c) and (d) described above, are specified in greater detail in NGAC-IRPAD. The interfaces described in (a), (e), and (f) are not elaborated further within the NGAC family of standards, due to variability in the type of resources protected by the policy, the translation of abstract operations performed on objects to concrete operations on the resources they represent, and the data model and services offered by various types of data stores.

The interfaces defined in NGAC-IRPAD may be realized as either a set of APIs or a protocol, depending on the requirements of the implementation. Where two functional entities exist within separate and distinct network-interconnected equipment, a protocol may be used. Where two functional entities exist within a single equipment an API or a protocol may be used. The two approaches are functionally equivalent, although some subsidiary functionality (e.g., message authentication, integrity, and replay protection) may be required in a protocol to compensate for threats that can potentially occur in a network environment.

Interface specifications between key functional entities are defined in NGAC-IRPAD as APIs. Protocols commensurable to theses APIs are not defined in NGAC-IRPAD. To establish such an interface, implementers may utilize an existing standardized network protocol that is appropriate for the networking environment, adequately protects the communications between functional entities, and fully captures the information exchanges described by the API.

6.3.3 Implementation Requirements

The NGAC-IRPAD standard also defines a number of generic requirements that apply to an implementation. They are as follows:

a) Requirements for the functional entities of the NGAC functional architecture; and
b) Requirements for the interfaces between functional entities, including security requirements that pertain to networking protocols.

Requirements are stated as implementation guidelines that describe essential security and operational objectives that need to be satisfied by a conformant implementation, using appropriate security controls and assurances. The security controls and assurances of an implementation may differ for different information systems and may go beyond the minimum security requirements. Similarly, an implementation needs to satisfy the stated operational objectives, but often can do so in a variety of ways. For example, an approach taken to achieve an objective may vary depending on the computational environment, implementation style, and types of protected resources involved.

Annex A
(Informative)

Bibliography

The following documents are not normative but provide important background for understanding this standard. For information on the current status of the listed document(s), or regarding availability, contact the indicated organization.

ACO: ISO/IEC 10181-1:1996, Information technology - Open Systems Interconnection - Security frameworks for open systems: Overview

COPS: IETF Request for Comments (RFC) 2748, The COPS (Common Open Policy Service) Protocol, January 2000

HOS93 H. Hosmer, The Multipolicy Paradigm for Trusted Systems, Proceedings of the 1992-1993 Workshop on New Security Paradigms, August 1993, Little Compton, RI, USA

MOA: ISO/IEC 10164-9-1995, Information technology - Open Systems Interconnection - Systems Management: Objects and attributes for access control

OAC: ISO/IEC 15816-2002 [R2007], Information technology — Security techniques — Security information objects for access control

FER15: D. Ferraiolo, S. Gavrila, W. Jansen, Policy Machine: Features, Architecture, and Specification, NIST Interagency Report (IR) 7987, Rev. 1, October 2015

RBAC: ANSI INCITS 359-2012, Information technology – Role Based Access Control

RBAC-REQ: ANSI INCITS 459-2010, Information technology – Requirements for the Implementation and Interoperability of Role Based Access Control

XACML: *eXtensible Access Control Markup Language (XACML), Version 3.0, Core spec,* January 22, 2013

XACML-RBAC:XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile, Version 1.0*,* Committee Specification 02, October 23, 2014

Annex B
(Informative)

Background to Access Control

Access control is an increasingly important subject in today's internetworked computer systems. A realization of the full value of information, and the importance of its reliability, integrity and continued availability to the functioning of an organization, has led to an increased focus on information assurance and protection. Access control is a fundamental mechanism by which information is protected, in that it restricts which users (and their processes) can perform which operations on which resources. (In this context “resources” may be items of information or facilities which give access to that information).

In access control, one or more policies are associated with each resource. Those policies determine the acceptable access that is allowed to the resource, and the acceptable operations that can be performed on the resource. Users possess attributes that describe their characteristics and needs. An access control mechanism then grants or denies resource access requests based on the application of one or more existing policies selected with reference to the user attributes.

Work on access control began in the mainframe era, and the most significant development during that era was the creation of the well-known Bell-LaPadula mathematical model that formalized military access control rules. The model contains two basic rules - the simple security rule and the *-property - that are often classified as “no read up” and “no write down”. The model also included the notion of categories, related to the concept of “need to know”.

Later the US Department of Defense published the Trusted Computer System Evaluation Criteria, popularly known as the “Orange Book”, that defined two modes known as Discretionary Access Control (DAC), and Mandatory Access Control (MAC). While MAC follows the multi-level security policy defined by the Bell-LaPadula model defined above, DAC is a more-relaxed mode where creators of files assign access rights which assume only a limited granularity amongst the users of the file. Somewhat as a reaction to the above, the commercially derived Clark-Wilson model then added the concepts of a well-formed transaction, and separation of duty (SoD), to access control.

Another significant development in access control has been the standardization of Role-Based Access Control [RBAC]. A role is defined as a set of common characteristics that confer specific privileges on a user. With the standardization of RBAC, sets of roles have been defined that are appropriate for specific industry verticals, of which the work of Health Level Seven International is a prime example.

The notion of a multi-policy machine capable of enforcing multiple, possibly contradictory security policies expressed in terms of user and object attributes was first envisioned in the early 90’s [Hos93]. Domains could be mutually exclusive or overlap each other, allowing subjects and objects to belong to more than one domain and fall under more than one policy, necessitating the resolution of policy conflicts that might arise.

Thus the trend in access control can be seen as one towards specialization and fragmentation, with multiple access control schemes being extant at any time, and each scheme having its own unique sphere of absolute control, to which it applies specific policies. This fragmentation introduces significant privilege and identity management challenges, and can also undermine policy enforcement objectives. For instance, although a file management system may narrowly restrict user access to a specific file, chances are the content of that file can be copied to an attachment or a message and mailed to anyone in the organization, or for that matter, the world, by the e-mail application.

This situation represents two significant problem areas. Firstly, the fragmentation leads to significant deployment and maintenance challenges, to complexity resulting in high management and control overheads, and to significant difficulties for users. Secondly, the policies defined by each system tend to be expressed and enforced differently, making it difficult for systems to interoperate without taking additional measures.

There is, therefore, an urgent need to reinvent access control in a form that suits the needs of the modern distributed interconnected enterprise. A flexible infrastructure is required that can provide access control services for a number of different types of resources, and that can enforce a number of different policies when those resources are accessed by a number of different types of applications and users. Ideally, the infrastructure should be scalable, able to support policies of different types simultaneously, and remain manageable in the face of almost constant change. NGAC is intended to meet these needs.

Annex C
(Informative)

Example NGAC Policy Configuration

C.1 Introduction

This annex contains a simple example of the process of creation and administration of an NGAC configuration. The example describes a hypothetical access control situation in a military medical facility. It demonstrates policies for role-based access control (RBAC), multi-level security (MLS) and separation of duty (SoD) in a single implementation.

The example describes a process of 9 steps through which the access control policies are configured and used. Each step is described in terms of the configuration of basic elements and relations that confer authorizations for specific users. The sequence is defined for ease of description, and is not intended to define a sequence that a real-world administrator would be likely to follow. However the final configuration, while somewhat simplistic, is likely to be quite realistic in terms of the capabilities and prohibitions that affect users.

Each subclause that defines one of the 9 steps contains both a text and a graphical description of the activities that take place during the step and the access privileges that are conferred as a result of both the preceding steps and the current one. The graphical representation consists of a diagram with separate areas for names of users and objects, user attributes, object attributes, policy classes and obligations, as shown in Figure C.1.

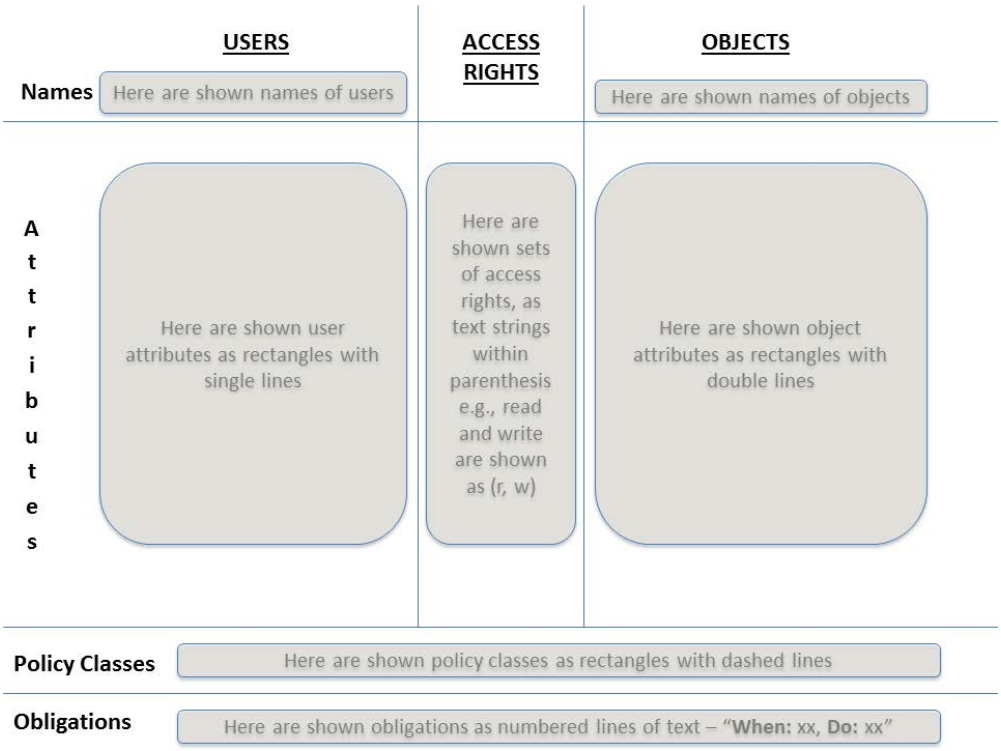

Figure C.1: Example diagram areas

Within the diagrams areas of Figure C.1, user attributes are represented by rectangles with single lines, object attributes are represented by rectangles with double lines and policy classes as rectangles with

dashed lines. In addition, assignments are represented by solid lines with arrowheads, associations by lines consisting of dots and dashes, and access rights sets are represented by text strings surrounded by braces. Figure C.2 depicts these conventions.

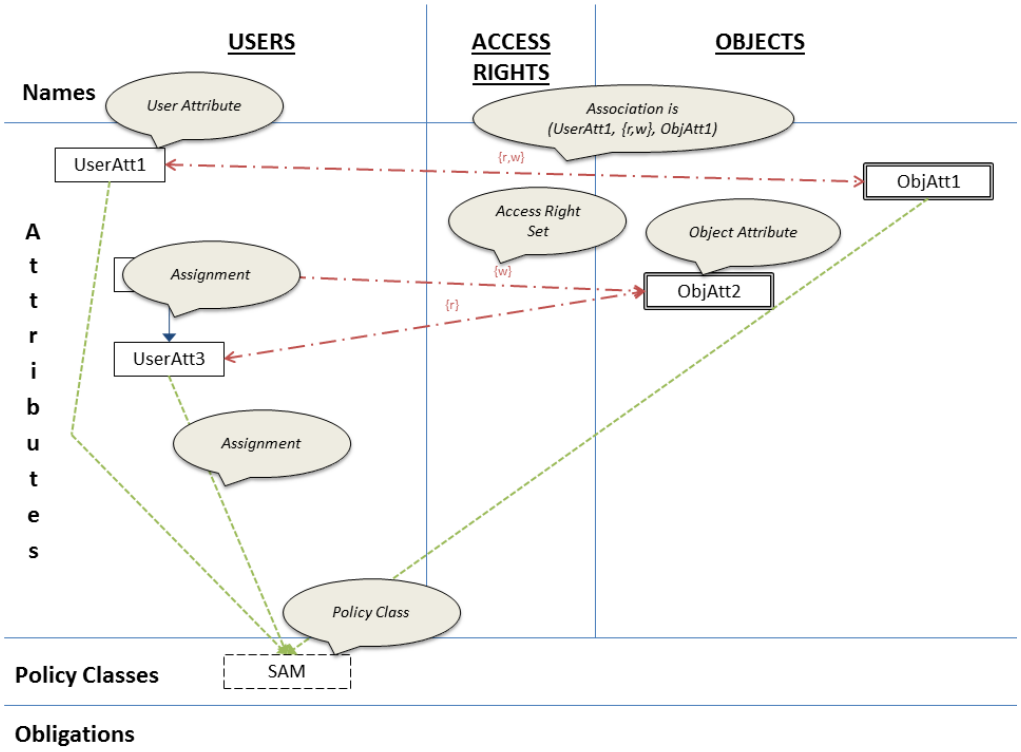

Figure C.2: Example diagram conventions

At each step, the new items that have been added to the graphical representation are shown by thicker arrow lines and rectangle borders and bold (and in some cases larger) text.

C.2 RBAC

C.2.1 Step 1 — initial policy class

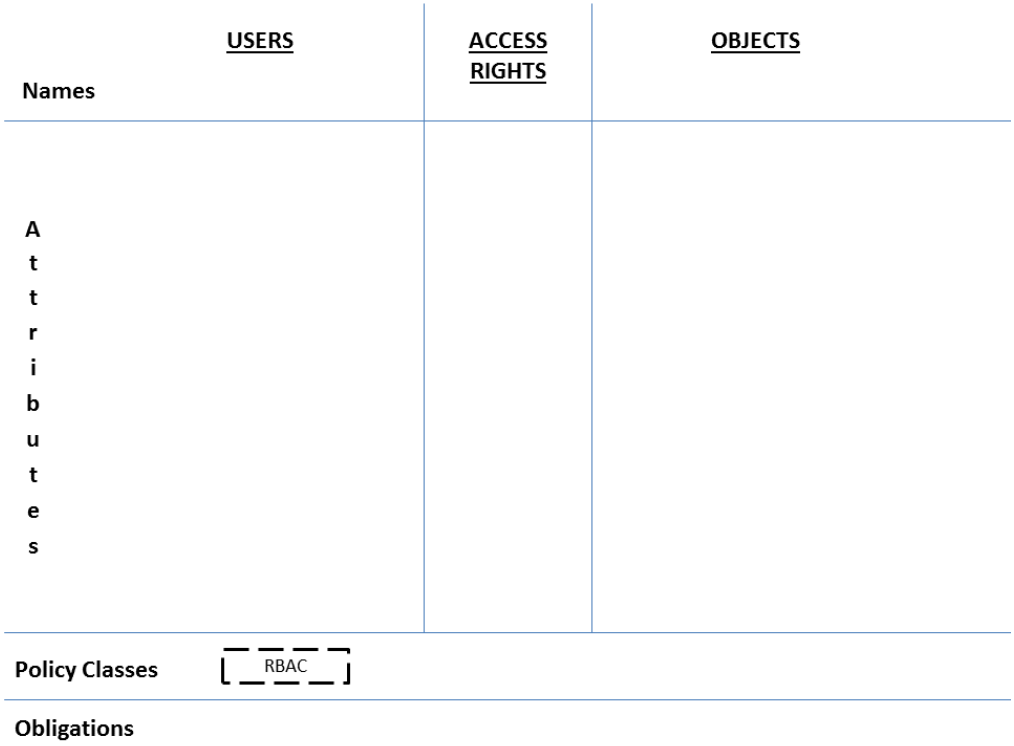

Figure C.3: Step 1

Step 1 involves the creation of an initial policy class. Because the primary users of the system will be medical personnel with existing defined job functions, it makes sense for the administrator to base the access control policies on roles, and to name the policy class “RBAC”. This policy class provides a basis for subsequent creation of user attributes and object attributes, and assignments.

C.2.2 Step 2 — initial relations

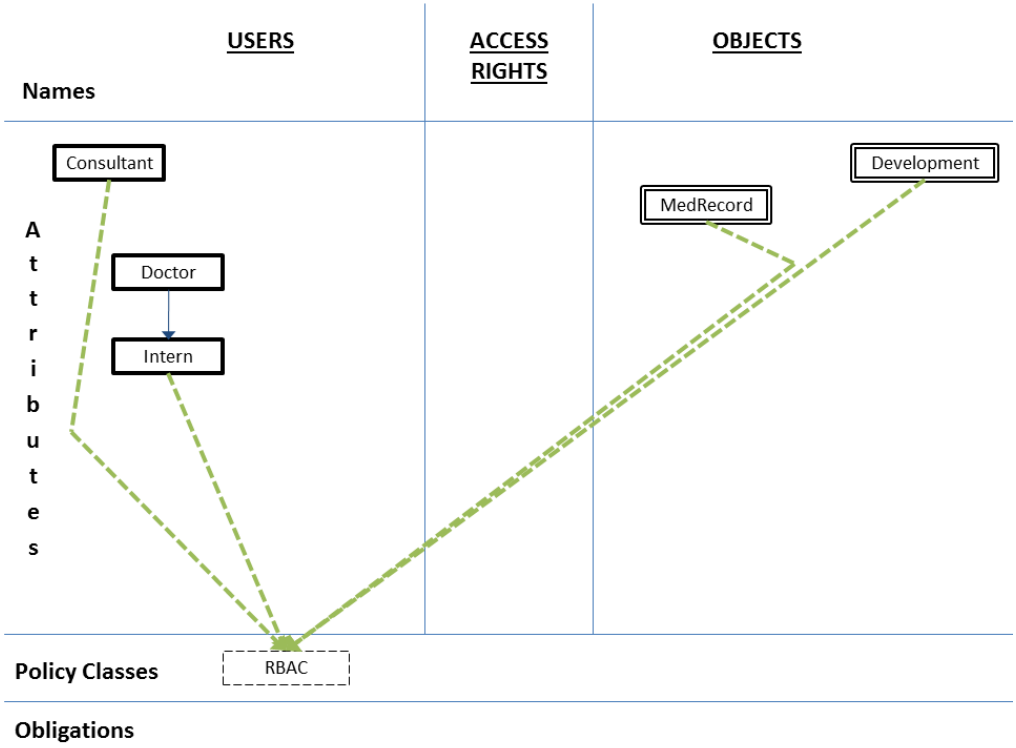

Figure C.4: Step 2

Step 2 involves the creation of an initial set of user and object attributes, attribute to attribute assignments, and attribute to policy class assignments. Because the primary users of the system will be medical personnel - doctors and interns - it makes sense for the administrator to create the initial user attributes as “doctor” and “intern”. (Note that in NGAC an attribute is referenced by a simple Unicode string.)

However the system will be under continuous development, and developers will need to access the system for testing purposes, so the administrator creates a single additional user attribute - “consultant” to support those activities.

On the resource side, the major type of information to which the system will be controlling access will be medical records, and thus it's straightforward for the administrator to create a “MedRecord” attribute. Again though, the developers will need to be able to create special record for testing purposes, so an additional object attribute “Development” is created with which to identify those records.

In addition, Step 2 creates an assignment relation from the “Doctor” user attribute to the “Intern” user attribute. An assignment defines membership within a container. This means that any users that are members of (or in) the user attribute “Doctor” (currently none) are also members of the user attribute “Intern”. As will be shown in later steps, an assignment between user attributes also implies inheritance of user capabilities.

Note that assignments are unidirectional; i.e., the assignment relation between “Doctor” and “Intern” described above does not mean that any users that are members of the user attribute “Intern” are also members of the user attribute “Doctor”.

Finally, all user and object attributes are linked through an assignment, or a chain of assignments, to the “RBAC” policy class. The attributes are thus defined to be members of the policy class.

C.2.3 Step 3 — access control information

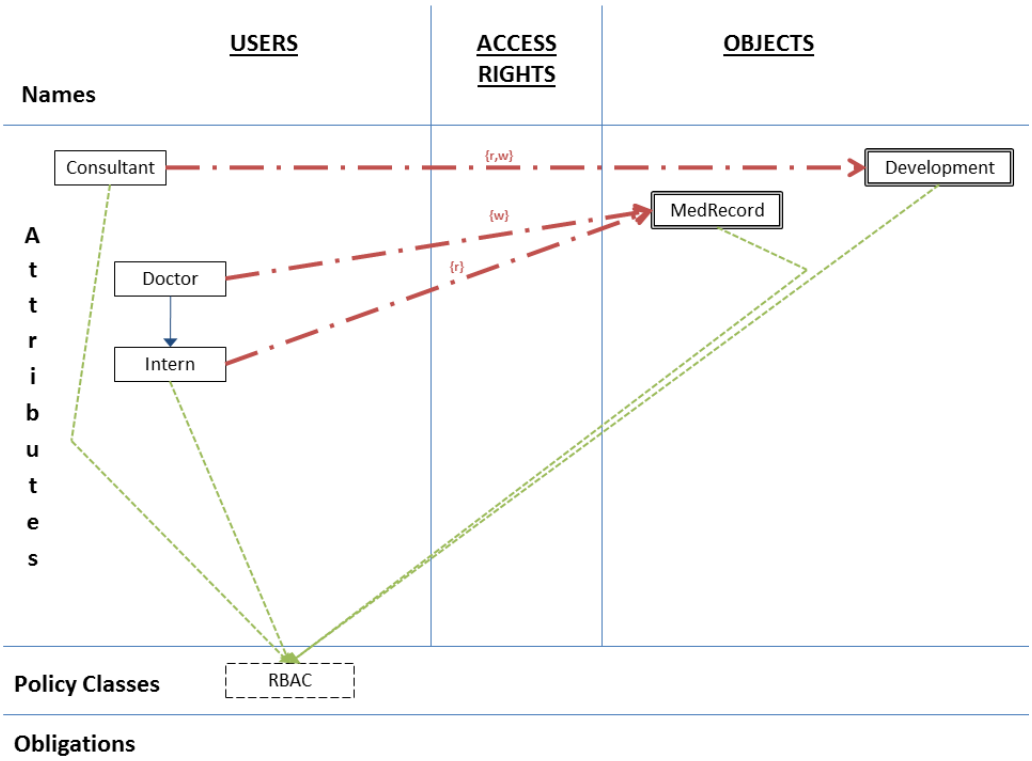

Figure C.5: Step 3

Step 3 addresses access control information for the first time, and to do this the administrator defines three associations. An association is a tuple consisting of a user attribute, access right set and object attribute. In this instance, the three associations are:

1) ("Consultant", {read, write}, "Development");
2) ("Intern", {read}, "MedRecord"); and
3) ("Doctor", {write}, "MedRecord").

In descriptive text, each of these associations respectively defines:

1) Users that are members of the attribute “consultant” (currently none) can read and write objects (resources) that are members of the attribute “Development” (currently none);
2) Users that are members of the attribute “Intern” can read objects (resources) that are members of the attribute “MedRecord”;
3) Users that are members of the attribute “Doctor” can write objects (resources) that are members of the attribute “MedRecord”.

However, because of the assignment that was created in step 2 between “Doctor” and “Intern”, any user that is contained by “Doctor” will also be contained by “Intern”, thus inheriting the capabilities of “Intern”. Thus a user that has a direct assignment to “Doctor” can write to objects that are contained by “MedRecord”, and that user will also be able to read objects that are contained by “MedRecord” (because that user will also possess the capabilities of “Intern”).

Note that at this point the access control system is not yet usable, because users and objects have not yet been assigned to any attributes.

C.2.4 Step 4 — first user and resource creation

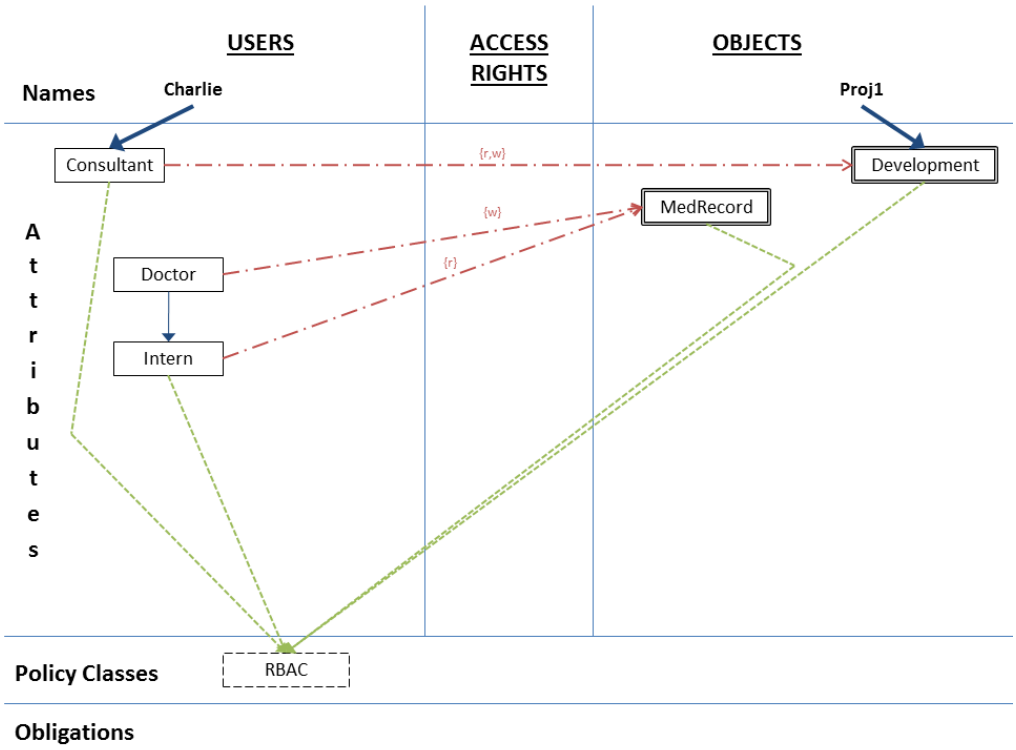

Figure C.6: Step 4

In Step 4, a user and an object are created and assigned to attributes for the first time. The administrator creates the object named “Proj1”, and stores a location for the resource represented by that object. She then creates an assignment of that object to the attribute “Development”. She then creates the user “Charlie” and creates an assignment of that user to the attribute “Consultant”.

At this point the system can do some computations. For example, based on association 1) in Step 3, and the assignments in this step, the system can determine the existence of two privileges:

1) ("Charlie", read, "Proj1"); and
2) ("Charlie", write, "Proj1").

Note that there is no need to actually compute and store privileges, such as these, as entities within the PIP. Instead privileges can be derived from associations, as needed, as part of the computation in rendering an access control decision. A system can also derive and review (on an as needed basis) the capabilities of users and user attributes, and the access control entries for objects and object attributes.

Once this computation is complete, the access control system can be made operational for the first time. User "Charlie" can log in with the system, browse for his capabilities, discover he has access to the object "Proj1", and issue an access request to read the resource represented by that object. In addition to checking for a privilege (see item 1 above), the system will check for the existence of relevant prohibitions, and if none are found, it will issue a command to read "Proj1" on the basis of location information stored when "Proj1" was created.

C.2.5 Step 5 — additional user and resource creation

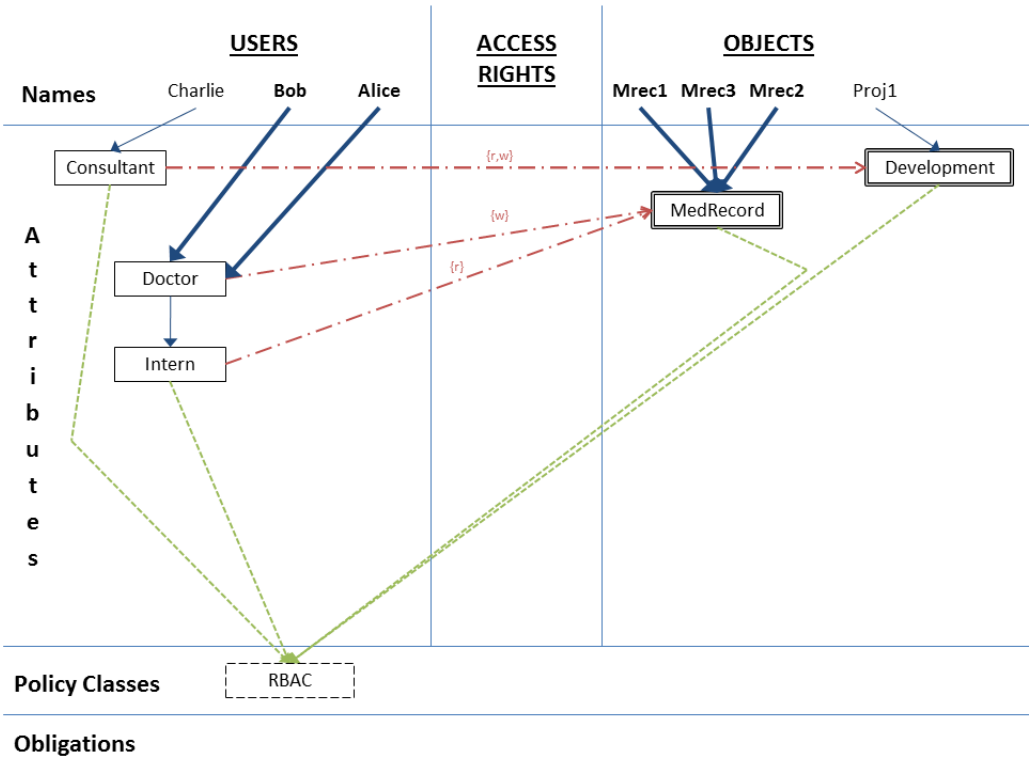

Figure C.7: Step 5

In Step 5, additional objects (“Mrec1”, “Mrec2” and “Mrec3”) and additional users (“Alice” and “Bob”) are created by the administrator. The administrator creates an assignment of each additional object to the attribute “MedRecord”, and an assignment of each additional user to the attribute “Doctor”.

Note that this step can take place while the access control system is in operation, in that "Charlie" can still be accessing "Proj1", as the privileges for "Charlie" and "Proj1" have not changed from Step 4 as follows:

1) ("Charlie", read, "Proj1"); and
2) ("Charlie", write, "Proj1").

However the system can now perform computations to derive the following additional privileges for the users and objects added to the configuration:

1) ("Alice", read, "Mrec1");
2) ("Alice", write, "Mrec1");
3) ("Alice", read, "Mrec2");
4) ("Alice", write, "Mrec2");
5) ("Alice", read, "Mrec3");
6) ("Alice", write, "Mrec3");
7) ("Bob", read, "Mrec1");
8) ("Bob", write, "Mrec1");
9) ("Bob", read, "Mrec2");
10) ("Bob", write, "Mrec2");
11) ("Bob", read, "Mrec3"); and

12) ("Bob", write, "Mrec3").

Note that privilege 4) above is derived from the membership of "Alice" in attribute "Doctor" and the association between "Doctor" and "MedRecord", while privilege 3) is derived from the membership of the attribute "Doctor" in the attribute "Intern" and the association between "Intern" and "MedRecord".

Once this is complete, "Alice" and "Bob" can use the system. They can log in with the system, browse for their capabilities, discover the objects to which they have access, and issue access requests to those objects.

C.3 MLS

C.3.1 Step 6 — new requirements leading to a new policy class

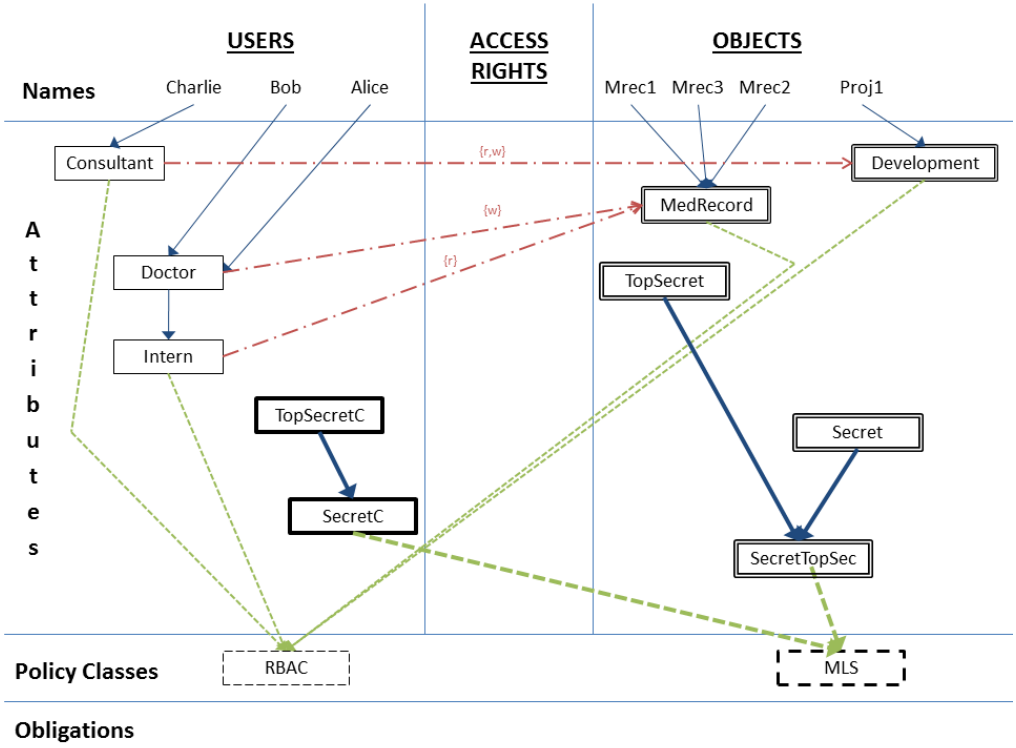

Figure C.8: Step 6

The configuration in Step 5 is logically complete and may exist for a considerable time. Other users and medical records can be added and assigned to the existing attributes, and user access is controlled as intended without the need for any additional associations or adjustments to policy.

However one day a VIP and his aide arrives at the hospital, and the management deem that it's not acceptable for any doctor to be able to access their medical records. Therefore a new approach to access control is needed, and a decision is made to base that approach on the military-style classifications and “need to know” of Multiple Level Security (MLS). Information associated with the VIP will be classified “Top Secret”, and that associate with his aide “Secret”.

Thus the administrator creates a new policy class, and names it “MLS”. She then creates a user attribute “SecretC” (secret clearance), and creates an assignment from it to the "MLS" policy class. The user attribute “TopSecretC” (top secret clearance) follows, and she creates an assignment from it to the "SecretC" attribute.

On the object side, she creates a “SecretTopSec” attribute, and creates an assignment from it to the "MLS" policy class. The object attributes “TopSecret” and “Secret” follow, and assignments are created from each to the "SecretTopSec" attribute.

The "SecretTopSec" attribute is created as a convenience to represent characteristics that are common to all objects to be protected under the "MLS" policy class.

C.3.2 Step 7 — new associations

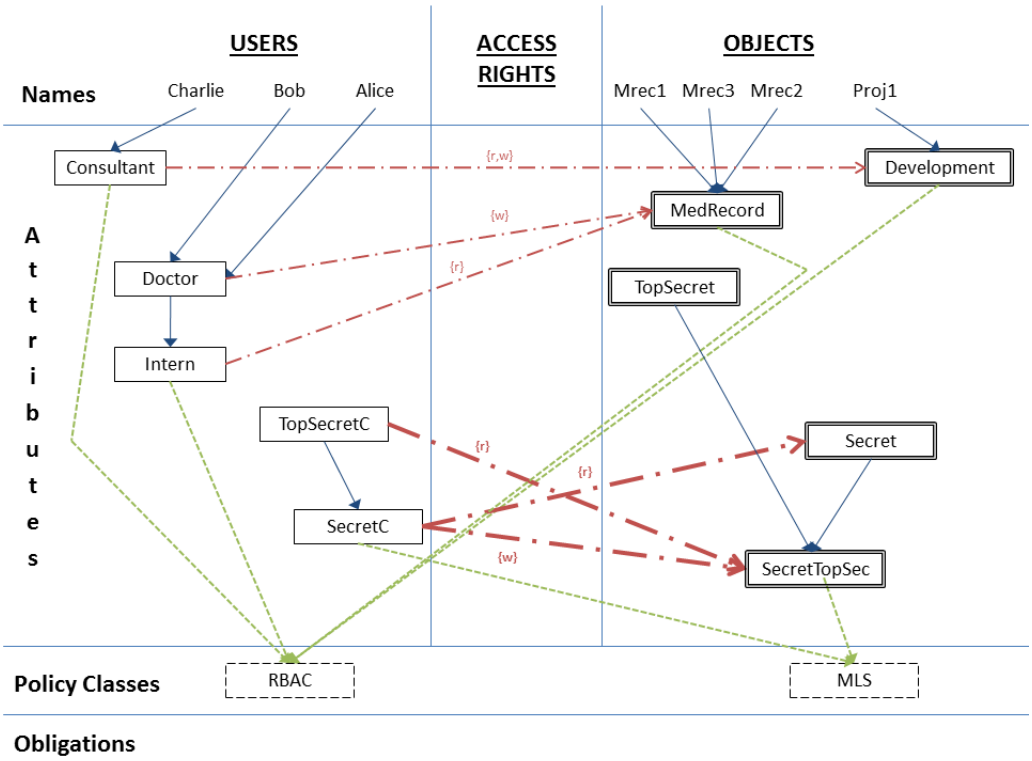

Figure C.9: Step 7

In Step 7, the administrator adds three new associations employing the attributes created in Step 6, namely:

1) (“TopSecretC”, {read}, “SecretTopSec”);
2) (“SecretC”, {read}, “Secret”); and
3) (“SecretC”, {write}, “SecretTopSec”).

Note however that the associations above are not sufficient by themselves to implement Multi-Level Security policies in general and the policy for the Bell-LaPadula model in particular. Although the associations provide support for the simple security rule (“no read up”), because a user that is contained by “SecretC” has no access to objects with “TopSecret”. But they do not support the *-property (“no write down”), for example because a user that is contained by "TopSecretC" is also contained by "SecretC", and thus has write access to objects with "SecretTopSec", i.e., both "TopSecret" and "Secret" objects.

C.3.3 Step 8 — new obligations

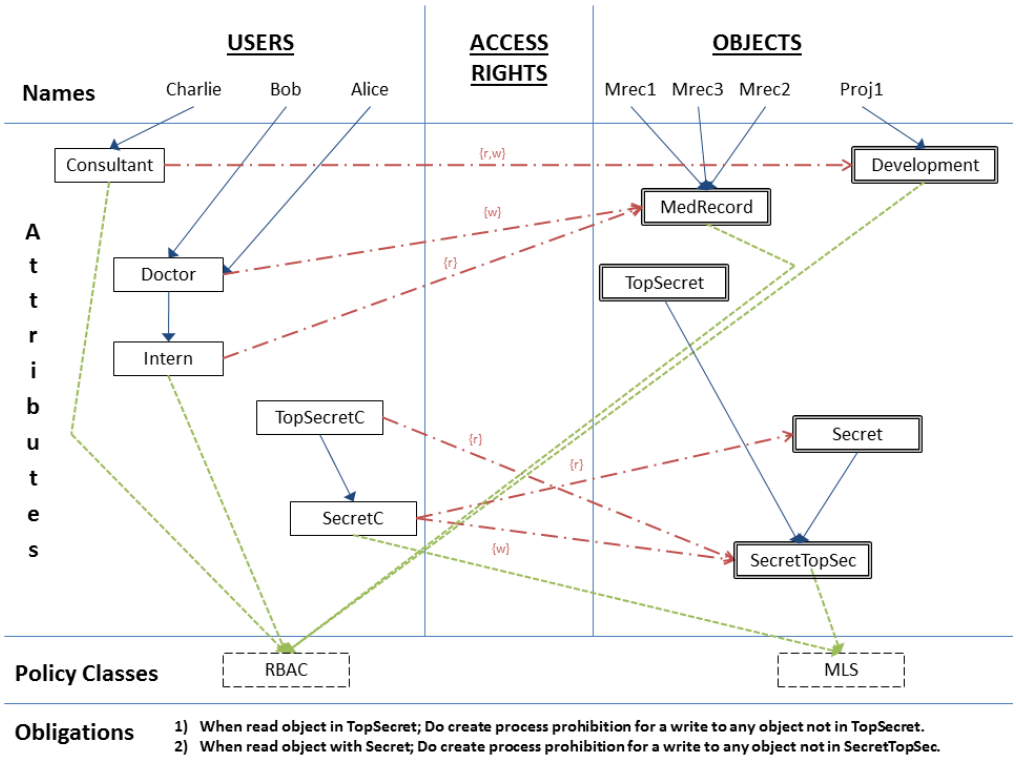

Figure C.10: Step 8

In Step 8, the administrator performs the final step in supporting the new security policy by creating two obligations that in concert with the associations in Step 7 support the *-property of the Bell-Lapadula model.

The obligations are as follows:

1) First obligation:
   - Event Pattern - **When** A process completes an access request containing a read operation to an object that is contained by "TopSecret"; and
   - Event Response - **Do** Create a process prohibition for the process ID that created the access request in the event pattern above to prevent that process from successfully carrying out an access request that attempts a write operation to any object that is not contained by "TopSecret".
2) Second obligation:
   - Event Pattern - **When** A process completes an access request containing a read operation to object that is contained by "Secret"; and
   - Event Response - **Do** Create a process prohibition for the process ID that performed the access request in the event pattern above that to prevent that process from successfully carrying out an access request that attempts a write operation to any object that is not contained by "SecretTopSec".

The use of these obligations will be illustrated more clearly in the following step, but they complete the definition of the MLS policy class, and users and objects may now be safely made members of the attributes that are members of that policy class.

Note that at this step the new definitions necessary to support the new policy class have had no effect on the access of Alice, Bob and Charlie to all three Mrecs and the "Proj1" object, as access is still completely controlled by the RBAC policy class. However that is about to change.

C.3.4 Step 9 — assign the new policy

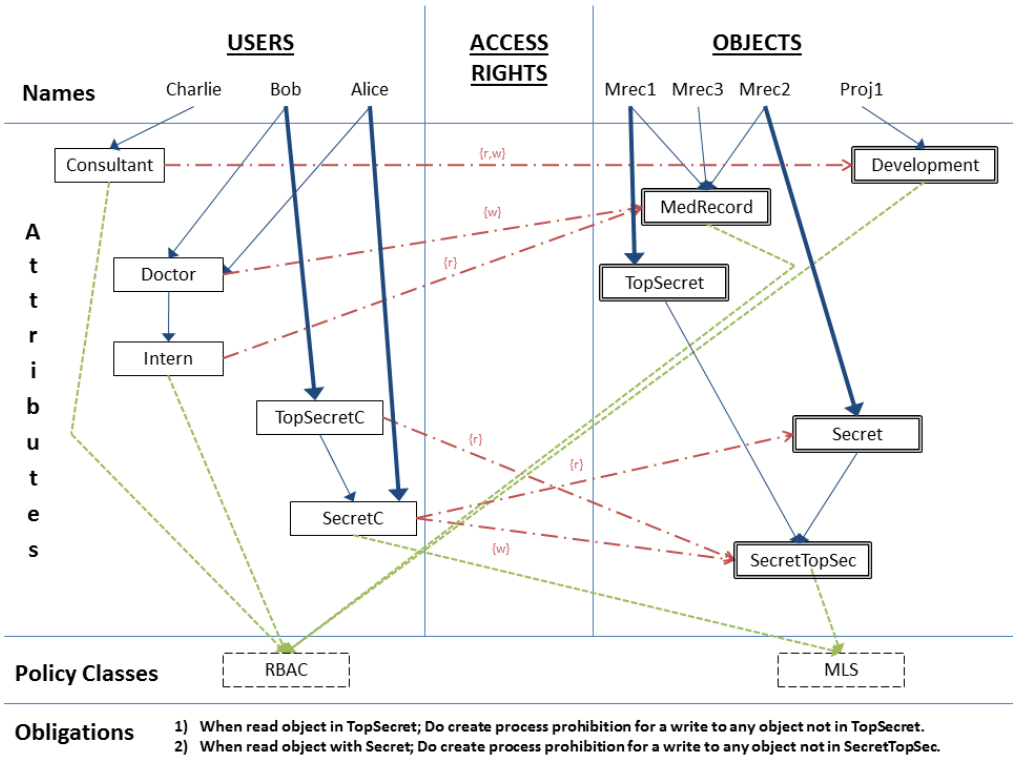

Figure C.11: Step 9

In Step 9, the administrator creates an assignment of "Mrec1" to the attribute “TopSecret”, and an assignment of "Mrec2" to the attribute “Secret”. This means that "Mrec1" and "Mrec2" are now protected under the "MLS" policy class as well as the "RBAC" policy class, and therefore a privilege associated with each policy class will be necessary for an access request to be granted to "Mrec1" and "Mrec2".

The administrator then grants "Bob" Top Secret clearance and "Alice" Secret clearance, by creating the appropriate assignments for each "Doctor" to the respective attributes, "TopSecretC" and "SecretC".

Considering the two policy classes in isolation for the moment, the following is the complete list of the privileges that the system can derive for each policy class:

1) RBAC
   - ("Charlie", read, "Proj1");
   - ("Charlie", write, "Proj1").
   - ("Alice", read, "Mrec1");
   - ("Alice", write, "Mrec1");
   - ("Alice", read, "Mrec2");
   - ("Alice", write, "Mrec2");
   - ("Alice", read, "Mrec3");
   - ("Alice", write, "Mrec3");
   - ("Bob", read, "Mrec1");
   - ("Bob", write, "Mrec1");
   - ("Bob", read, "Mrec2");
   - ("Bob", write, "Mrec2");

- ("Bob", read, "Mrec3"); and
- ("Bob", write, "Mrec3").

2) MLS
   - ("Alice", write, "Mrec1");
   - ("Alice", read, "Mrec2");
   - ("Alice", write, "Mrec2");
   - ("Bob", read, "Mrec1");
   - ("Bob", write, "Mrec1");
   - ("Bob", read, "Mrec2"); and
   - ("Bob", write, "Mrec2").

When considering the two policy classes in combination, privileges are derived using the rule that for each policy class that contains the object of the privilege, an association exists such that the user and object are contained by the attributes of the association, the access right is a member of the access right set of the association, and the object attribute of the association is contained by the policy class. Applying that rule, the following privileges are derived for the policy classes applied in combination:

1) ("Alice", write, "Mrec1");
2) ("Alice", read, "Mrec2");
3) ("Alice", write, "Mrec2");
4) ("Alice", read, "Mrec3");
5) ("Alice", write, "Mrec3");
6) ("Bob", read, "Mrec1");
7) ("Bob", write, "Mrec1");
8) ("Bob", read, "Mrec2");
9) ("Bob", write, "Mrec2");
10) ("Bob", read, "Mrec3");
11) ("Bob", write, "Mrec3");
12) ("Charlie", read, "Proj1"); and
13) ("Charlie", write, "Proj1").

Note that the second privilege results because "Mrec2" is contained by the "RBAC" and "MLS" policy classes and that privilege exists for both policy classes (see the previous list). Also, the penultimate privilege (viz., 12) results because "Proj1" is contained only in the "RBAC" policy class and that privilege exists for that policy class. On the other hand, although the privilege ("Alice", read, "Mrec1") exists for the "RBAC" policy class, no such privilege results when combined, because that same privilege does not exist for the "MLS" policy class.

Although the existence of requisite privileges is necessary to allow an access to succeed, they are not in and of themselves always sufficient in rendering a positive access decision, since prohibitions may negatively affect the outcome.

For example, when considering privileges alone, "Bob" can read "Mrec1" (which is "TopSecret") and either "Bob" or a Trojan horse imbedded in an application executing on behalf of "Bob", can write the content of "Mrec1" (in violation of the *-property) into "Mrec2" (which is "Secret") thus allowing "Alice" (cleared "SecretC") to read "TopSecret" data. Worse yet, it is possible for information to be “leaked” from an object that’s contained by either “Secret” or “TopSecret” to "Mrec3" that is not protected by "MLS" at all.

Now consider the effect of prohibitions in the case where "Bob" reads "Mrec1". In accordance with the first obligation defined in Step 8 (see subclause C.3.3), a prohibition will be in effect prior to the next access request issued from that process. Now when either "Bob" (through that process), or a Trojan horse embedded in that process, attempts to write the "TopSecret" content (in the process memory) to "Mrec2" or "Mrec3", the access request will be denied.

The configuration in Step 9 is again in a logically complete state. Other doctors and medical records can be added and assigned to the existing attributes, and access to data protected without the need for any additional associations or adjustments to policy.