AN ACTIVE RULE-BASED SYSTEM FOR XACML 3.0

A Project

Presented to the faculty of the Department of Computer Science
California State University, Sacramento

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

by

Minolini Nithyanandam

FALL 2016
AN ACTIVE RULE-BASED SYSTEM FOR XACML 3.0

A Project

by

Minolini Nithyanandam

Approved by:

__________________________________, Committee Chair
Ying Jin, Ph.D.

__________________________________, Second Reader
Gita Faroughi

__________________________________
Date
Student: Minolini Nithyanandam

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the project.

__________________________, Graduate Coordinator
Ying Jin, Ph.D.

Department of Computer Science
Abstract

of

AN ACTIVE RULE-BASED SYSTEM FOR XACML 3.0

by

Minolini Nithyanandam

Extensible Access Control Markup Language (XACML) is an access control policy language and a processing model which evaluates the access requests according to the rules defined in the policies. XACML is widely used in order to decouple client applications from access decisions. Even though the access decisions and permissions for the resource handling can be hardcoded by the users, these changes are reflected locally. This results in consistency issues when multiple client systems need to use the modified access decisions defined in the policies rather than the original or unchanged policies. In order to handle the resources efficiently and dynamically on the fly, we have provided an extension to XACML in this project.

In this project, we will be enhancing an existing language specification for incorporating active rules for XACML based policies which has already been implemented using XACML 2.0 as part of a previous Master’s project. The enhancement includes the design of an XACML 3.0 based policy system in lieu of an XACML 2.0 based implementation to utilize the multitude of XACML 3.0 benefits. This project also
includes the active rules for temporal events execution and enabling of an upgraded middleware system prototype which acts as a parser and an event handler to facilitate rule retrievals and executions. A system is designed which incorporates active rules in policies that is implemented in XML and a language model which provides a parser which understands XACML 3.0 language specifications and an event handler for handling the rules defined in the policies and process according to it.

The system consists of two interfaces: one for handling active rules and temporal rules, and other interface for handling user queries such as insert, delete and replace operations and scheduling temporal events operations. This XACML 3.0 based policy system will have high precision policy control and provides dynamic integrity constraint management.

_______________________, Committee Chair
Ying Jin, Ph.D.

_______________________
Date
# Table of Contents

List of Figures ........................................................................................................... ix

Chapter

1. INTRODUCTION ...................................................................................................... 1

2. BACKGROUND ........................................................................................................... 4

   2.1 XACML .................................................................................................................. 4

   2.2 BaseX .................................................................................................................... 6

   2.3 XQuery .................................................................................................................. 7

   2.4 ECA Rules or Active Rules .................................................................................. 7

   2.5 Quartz Scheduler .................................................................................................. 8

   2.6 SMTP .................................................................................................................... 8

3. ECA RULES EXAMPLES for MUTATION EVENTS ............................................. 10

   3.1 ECA Rule Structure ............................................................................................. 11

   3.2 ECA Rule 1 .......................................................................................................... 13

   3.3 ECA Rule 2 .......................................................................................................... 15

   3.4 ECA Rule 3 .......................................................................................................... 18

   3.5 ECA Rule 4 .......................................................................................................... 20
4. ECA RULES for TEMPORAL EVENTS (TIMER RULES) ........................................ 22
   4.1 Timer Rule Structure ....................................................................................... 22
   4.2 Timer Rule 1 ..................................................................................................... 23
   4.3 Timer Rule 2 ..................................................................................................... 25
   4.4 Timer Rule 3 ..................................................................................................... 27
   4.5 Timer Rule 4 ..................................................................................................... 28

5. SYSTEM ARCHITECTURE ...................................................................................... 31
   5.1 Architecture for ECA Rule Execution .............................................................. 31
   5.2 Architecture for Timer Rule Execution ............................................................ 33

6. IMPLEMENTATION .................................................................................................. 35
   6.1 Implementation of ECA Rule Execution ......................................................... 35
   6.2 Implementation of Timer Rule Execution ....................................................... 38
   6.3 Implementation of Sending Email .................................................................... 42

7. CONCLUSION AND FUTURE DIRECTION ......................................................... 44

Appendix. XACML 3.0 Policies for Medical Application ........................................... 45

References .................................................................................................................... 68
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figures</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ECA Rule Structure</td>
<td>11</td>
</tr>
<tr>
<td>2. NurseTechnicianIntern_Delete Rule</td>
<td>14</td>
</tr>
<tr>
<td>3. PRDuration_Replace Rule</td>
<td>17</td>
</tr>
<tr>
<td>4. ExpiryMedicinesBefore_Delete Rule</td>
<td>19</td>
</tr>
<tr>
<td>5. PatientBillingInfo Rule</td>
<td>21</td>
</tr>
<tr>
<td>6. Timer Rule Structure</td>
<td>22</td>
</tr>
<tr>
<td>7. Timely_DrugQty_Check Rule</td>
<td>24</td>
</tr>
<tr>
<td>8. ExpiredMedicines_Delete Rule</td>
<td>26</td>
</tr>
<tr>
<td>9. NurseInternAcess_Replace Rule</td>
<td>28</td>
</tr>
<tr>
<td>10. NursePromotion_Replace Rule</td>
<td>29</td>
</tr>
<tr>
<td>11. ECA Rule Execution Model</td>
<td>31</td>
</tr>
<tr>
<td>12. ECA Rule for Temporal Events Execution Model</td>
<td>33</td>
</tr>
<tr>
<td>13. Interface 1 Workflow for ECA rules</td>
<td>35</td>
</tr>
<tr>
<td>14. Rule Repository Collection Structure for Active Rule</td>
<td>36</td>
</tr>
<tr>
<td>15. Interface 2 Workflow for ECA rules</td>
<td>37</td>
</tr>
<tr>
<td>16. Interface 1 Workflow for Timer Rules</td>
<td>39</td>
</tr>
<tr>
<td>17. Interface 2 Workflow for Timer Rules</td>
<td>40</td>
</tr>
<tr>
<td>18. Rule Repository Collection Structure for Timer Rules</td>
<td>41</td>
</tr>
<tr>
<td>19. Sending Email from Java Application</td>
<td>42</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

Extensible Access Control Markup Language (XACML) is an XML language for expressing access control policies [1]. XACML is used to evaluate access requests according to the authorization rules specified in the policies. One drawback of XACML is that it does not support policies or rules to be altered in a cascading manner automatically. Each policy or rule needs to be changed manually in the policies. This results in data inconsistencies issues while handling the resources when the user makes changes to the policy such as updating a policy. To address this problem, an active rule based policy system for XACML 2.0 was implemented as part of a previous Master’s project which handles dynamic changes and integrity checking.

The goal of this project is to provide an enhancement to existing language specification for incorporating active rules for XACML based policies which has already been implemented using XACML 2.0. This project also introduces the active rules for temporal events or timer rules. The enhancement in this project includes the design of an XACML 3.0 based policy system in lieu of an XACML 2.0 based implementation to utilize the multitude of XACML 3.0 benefits. This project also includes the enabling of an upgraded middleware system prototype which acts as a parser and an event handler to facilitate rule retrievals and executions for active rules. In addition, for the execution of timer rules, a parser was developed to parse only the timer rules and store the timer rules
into a rule repository. This in turn facilitates rule retrievals, scheduling and execution of the jobs based upon specific time and frequency interval. A system has been designed which incorporates active rules and timer rules in policies that are implemented in XML and a language model. The language model provides a parser and an event handler. The parser is used to understand XACML 3.0 language specification. The event handler is used for handling the rules defined in the policies and process according to it. Unlike active rules, the timer rules get triggered in the specified timeframe and it does not require the user to initiate the access request.

In this project, a medical application is considered to define XACML3.0 policies in order to demonstrate the system workflow. The medical application consists of different sets of policies for: Administrator, Nurses, Nurse Technician Intern, Physician, Pharmacist and Patient Billing. The attribute subject category defined in the policies is responsible for handling specific resources. The attribute subject category can perform following access operation such as select, insert, delete and replace on the resources. Each of the policy set mentioned above is discussed in detail under ECA rules and timer rules examples. The medical policies are shown in appendix section.

This project report consists of multiple chapters. Each chapter will discuss a particular aspect of the project in great depth. A number of tools and technologies have been used during the development of this system. Chapter 2 provides all the fundamental
information regarding these tools and technologies used. Chapter 3 provides information about the structure of Event Condition Action (ECA) rules and the examples of ECA rules which are defined in this project. Chapter 4 provides information about the structure of timer rules and examples of timer rules which are defined in this project. Chapter 5 leads a discussion about the system architecture of ECA rules and timer rules execution. Chapter 6 discusses about the implementation of ECA rules and timer rules execution. Chapter 7 provides future directions and conclusion for the current system.
2. BACKGROUND

2.1 XACML

XACML is an OASIS standard that describes both policy language and an access control decision request/response language (both written in XML) [2]. XACML also defines a set of functions, which can be used in authorization logic evaluation [3]. There are several conditions, actions, attributes and resources defined in the XACML policy language. Based upon the conditions and other information defined in the policy or rules, the access to the requested resource is either permitted or denied.

A XACML user request to access the resource works in the following manner. The user request is sent to the PEP (Policy Enforcement Point). The PEP forms a query for the PDP (Policy Decision Point) based on the user request. On receiving the query, the PDP evaluates the query with the applicable policy and rules, and interprets the access decisions. The interpreted result is sent to the PEP which can either permit or deny the user request to access the resource.

The major components of a policy language model are policy set, policy and rules [4]. A Policy Set is a container that can hold other Policies or Policy Sets and each policy can have one or more rules that defines the conditions necessary to access a set of
resources [4][5]. These rules are composed of targets [5]. The target defines the set of requests to which the rule is intended to apply in the form of a logical expression on attributes in the request [4]. The permission for the resource access is granted (i.e. permit) only when all the conditions in an applicable policy or rules are satisfied. These conditions are explicitly defined within <Match> and <Apply> tags. The <Match> element shall identify a set of entities by matching attribute values in an <Attributes> element of the request context with the embedded attribute value [4]. The attribute “Category” within the <AttributeDesignator> element identifies a category of an attribute such as access-subject, resource and action. The <AnyOf> contains the disjunctive sequence of <AllOf> elements [4]. The <AllOf> element encloses the conjunctive sequence of <Match> elements [4]. Thus, the above XACML tags are used to define the XACML 3.0 policies.

In this project, XACML 3.0 is used instead of XACML 2.0, in order to utilize the benefits of XACML 3.0 policy. One of the major differences between XACML 2.0 and XACML 3.0 is the attribute structure defined in the policy. In XACML 2.0 policy, the attributes are arranged by use of XML element tags such as subject, resource, environment or action categories. That is, the attributes which falls under same category are grouped together under XML element tags that reflect the category [6]. For example, all the attributes which belong to subject category are grouped under following XML element tags <Subjects> and <Subject>. The attributes of same category are combined
using an or (disjunctive) relationship and the top-level category elements are combined using and (conjunctive) relationship [6]. In contrast, in XACML 3.0 policy, the attributes are organized by means of XML attributes rather than XML element tags. XACML 3.0 clearly removes the disjunctive and conjunctive function of the category elements and introduces the AnyOf and AllOf elements [6]. The attribute category information is listed in “Category” node within <AttributeDesignator> element. In addition, XACML 3.0 introduces new functions and datatypes when compared to XACML 2.0. Thus the structural change of XACML 3.0 policy simplifies the processing model.

2.2 BaseX

BaseX is both a light-weight, high-performance and scalable XML Database and XQuery 3.1 Processor with full support for the W3C Update and Full Text extension [7]. In this project, BaseX is used to store, visualize the graphical user interface of the XML documents and query against the XML database engine to retrieve the relevant results. BaseX provides an XQuery API for Java (XQJ) which is used in Java applications to execute XQuery against the BaseX XML database engine [7][8].
2.3 XQuery

XQuery is a querying language. It is used to query the XML database engine in order to extract useful information. XQuery makes use of XPath, a language that describes a way to locate and process items in XML documents [9]. XQuery offers several built-in functions and user-defined functions which are used in manipulation of XML database. When the user request an access to the resource, the user request is specified in an XQuery language and also the condition action part of active rules and timer rules are also written in XQuery language.

2.4 ECA Rules or Active Rules

An active rule consists of an event, a condition, and an action [10]. An ECA rule has the general syntax: on event if conditions do actions [11]. These ECA rules are stored in a database. This type of database where ECA rules are stored is known as an active database. When one of the mutation events such as insert, delete or replace occurs, the system searches for relevant ECA rules from the active database and invokes it. Immediately after invoking the relevant ECA rules, the system checks for the condition part of the rule. Based upon the condition which is either evaluated to be true or false, the respective action part of the rule gets executed. The action is the set of operations that needs to run if the event and conditions were met [5].
The temporal events invoke the triggers during a specified time and frequency duration. For example, checking the medicine inventory during a specific time and frequency duration in order to ensure the medicine quantities are more than the threshold limit.

2.5 Quartz Scheduler

Quartz Scheduler is an open source job scheduling framework for Java application. The JAR files pertaining to this scheduler is integrated with the Java application in order to schedule and execute one or more number of jobs at the defined start time and frequency interval. The start time, frequency and other information are defined in the timer rules. The jobs that need to be executed are defined in the Java application. The ECA rules for temporal events or the timer rules uses this type of framework to schedule and execute the jobs in the provided timeframe and frequency.

2.6 SMTP

The objective of Simple Mail Transfer Protocol (SMTP) is to transfer mail reliably and efficiently [12]. An important feature of SMTP is its capability to relay mail across transport service environments [12]. The e-mails are transmitted over internet from one email server to another. Some of the public companies like Gmail and
Yahoo uses SMTP protocol for exchanging the mails with outside world. In this project, the system uses Gmail SMTP server for sending e-mail notifications to pharmacist when the drug quantity goes lower than the threshold.
3. ECA RULES EXAMPLES for MUTATION EVENTS

In XACML 2.0 policy, the attributes which fall under same category are grouped together under element that reflects that category [6]. To quote an instance, attributes which belong to subject category are grouped under following XML element tags <Subjects> <Subject>, <SubjectMatch>. Under <SubjectMatch> tag, the attribute value and attribute ID information are specified using tags such as <AttributeValue> and <SubjectAttributeDesignator> respectively. Similarly for <Resources>, <Action> and <Environment>. So, it is easy to locate the attributes in the policy since the category type is reflected in the XML element tags. In contrast, in XACML 3.0 policy it removes XML element tags and introduces XML attributes. For instance, attributes which belong to subject, resource, action or environment category are grouped under following XML attribute tags <AnyOf>, <AllOf> and <Match>. Under <Match> tag, the attribute value, attribute ID and category information are specified using <AttributeValue> and <AttributeDesignator> tags respectively. In this case, it becomes hard to locate the attribute information in the XACML 3.0 policy since the XML attribute tags does not reflect attribute category information. To deal with this problem, we have introduced <category> element to our existing ECA rule structure, to uniquely identify and access the attributes in XACML 3.0 policy. In this project, we have created and implemented new ECA rules examples and new set of policies which is defined in XACML 3.0 policy. The following section 3.1 presents ECA Rule structure for XACML 3.0 policy.
3.1 ECA Rule Structure

The ECA rule structure is depicted in figure 1. The tags which are used to write ECA rules are `<ecarules>`, `<ecarule>`, `<ruleid>`, `<doc>`, `<type>`, `<focalpoint>`, `<category>`, `<happen>`, `<priority>` and `<cond_action>`. There can be one or more `<ecarule>` blocks enclosed within the `<ecarules>` tag. The `<ecarule>` marks the beginning of a new ECA rule. The `<ruleid>` tag contains the name of the ECA rule which is a unique identifier. Consider the case where an application consists of several ECA rules. In this case, the value within `<ruleid>` is used to identify the active rules. `<doc>` is the XACML file name denoting the document modified that triggers the execution of the active rule [5]. The value within `<type>` tag indicates the event type such as insert, delete or replace. The value specified within the `<focalpoint>` indicates the point of trigger. The values which the `<focalpoint>` tag can accept are one of the following XACML 3.0

![Figure 1. ECA Rule Structure](image-url)
policy tags: <PolicySet>, <Policy>, <Rule>, <Target>, <AnyOf>, <AllOf>, <Match>, <Condition> or <Apply>. <category> denotes the attribute category. The values such as access-subject, resource and action are allowed for the <category> node.

The <happen> node determines the type of trigger. It can take one of the two values: “before” or “after”. The value “before”, initially executes the trigger defined within the <cond_action> block of the active rule and then executes the user XQuery request. The value “after”, initially executes user XQuery request and then executes the trigger defined within the <cond_action> block of the active rule. If a single mutation event triggers more than one ECA rule, then the rule execution order is determined by the value within the <priority> tag. The value of <priority> ranges from 1 to N, where N is an integer. For example, consider the case of two ECA rules that are triggered by the same event, where the priority of rule 1 is “2” and the rule 2 priority is “50”. In this case, rule 2 gets executed first, since “50” is the highest priority and then rule 1 executes after rule 2 is completed, since “2” is the lowest priority.

The <cond_action> block contains the condition and action part of the ECA rule. The condition and action part is written in the form of XQuery statements. These XQuery statements contain some special variables. These special variables begin with a “#” sign such as #PNODE, #OLD and #NEW and they get substituted with actual values during runtime. The #PNODE variable denotes the parent node and performs one of the
following operations such as “inserting a node”, “deleting a node” or “replacing an old
node with a new node” under its parent node. The #NEW represents the new node that
needs to be inserted or replaced and #OLD denotes the old node that was deleted or
replaced. The following sub-sections present examples of ECA Rules.

3.2 ECA Rule 1

The ECA rule 1 “NurseTechnicianIntern_Delete” is shown in Figure 2. In the
“Nurse Technician Intern” policy, the interns are responsible for selecting and inserting
progress notes of the patient and can select the current medication of a patient. This
policy is shown in appendix page 54.

The objective of the above ECA rule is to delete the “Nurse Technician Intern”
job position in the related policies whenever the hospital administrator removes the
“Nurse Technician Intern” job position from the policy.

This ECA rule can be applied to the following situations. One good example for the
application of this rule is when all the nurse interns return to their university at the end of
a summer or fall internship. This rule can also be used during hospital-wide layoffs.
During these circumstances, the administrator removes the “Nurse Technician Intern” job
position from the policy. Therefore, whenever the hospital administrator removes “Nurse
Technician Intern” job position from the policy, then the related policies which have this job position are also deleted.

```xml
<ecarule>
  <ruleid>NurseTechnicianIntern_Delete</ruleid>
  <doc>Policy.xml</doc>
  <type>delete</type>
  <focalpoint>Match</focalpoint>
  <category>access-subject</category>
  <happen>after</happen>
  <priority>1</priority>
  <cond_action>
    let $pos := #OLD/AttributeValue/text()
    return if ($pos="NurseTechnicianIntern")
    then
      for $x in collection("book")//Policy[@PolicyId='NurseTech_Intern']/
          Rule/Target/AnyOf/AllOf/Match[AttributeValue="NurseTechnicianIntern"]/
              AttributeValue/text()
      return delete node $x
    else ()
  </cond_action>
</ecarule>
```

Figure 2. NurseTechnicianIntern_Delete Rule

The event type “delete” listed within the <type> tag triggers this active rule. The <focalpoint> tag contains the value as “Match” where the trigger is set and the attribute category is an “access-subject”. This attribute category is specified within the <category> tag. The node <happen> contains the type of trigger which consists of one of the two values: “before” or “after”. The “after” trigger is applicable in this case. <doc> contains the filename, “Policy.xml”, of the XACML policy on which the trigger was set [5]. The
value within <priority> tag determines the execution order of the condition action part of the active rule in the after queue. In the above example, the priority is set to 1, which is considered to be the least priority.

The condition action part of this rule is enclosed within the node <cond_action>. The condition part of the rule checks whether the deleted node which was performed due to the user request was the “Nurse Technician Intern” node. If the condition is evaluated to be true, then the action part of this rule will be triggered, which includes deleting the “Nurse Technician Intern” node from the related rules/policies. Otherwise, no action should be taken.

3.3 ECA Rule 2

The ECA rule 2 “PRDuration_Replace” is shown in figure 3. In the “Administrator” policies, the administrator is responsible for performing the select, insert and replace operations on reference “patient record duration”. The administrator also manages and modifies a whole bunch of data. Some of this data includes the “patient record duration” of the passive and active patients apart from other information such as physician, pharmacist and also the patient. The policies related to the administrator are found in appendix page 45.
The objective of this active rule is to delete the passive patient records and those records which reside in the medical database for more than a threshold/reference “patient record duration” value which is set by the administrator. Patient Record duration is a parameter which determines “how many years does the patient record exist in the medical database”.

In this example, “passive” means the patient was discharged a long time ago from the hospital, but his/her records are still maintained in the hospital database. In contrast, “active” means the patient is still admitted in the hospital and records are actively reviewed by the physician and nurses. This rule is mainly used in a scenario where the administrator needs to clear old patient records, which are not in use and exist in the medical database for a large number of years. It is mainly used for performance optimization of medical database.

In this ECA rule shown in figure 3, replacing the reference “patient record duration” node triggers this active rule. The <type> tag contains information about the event type. In this case, the value within <type> tag is “replace”. The “Match” is the tag where the trigger is set and attribute category to be replaced is “access-subject”. The node <happen> contains the type of trigger which consists of one of the two values: “before” or “after”. In this case, it is the “after” trigger. In this example, the value within node <doc> is “Policy.xml”. The value within <priority> tag determines the execution order of
the condition action part of the active rule in the after queue. Here, the priority value is “10”. Therefore, it executes the condition action part of this active rule, which resides in the “after” queue only in the end.

```
<ecarule>
   <ruleid>PRDuration_Replace</ruleid>
   <doc>Policy.xml</doc>
   <type>replace</type>
   <focalpoint>Match</focalpoint>
   <category>access-subject</category>
   <happen>after</happen>
   <priority>18</priority>
   <cond_action>
      let $status := collection("book")/Rule[@RuleId='AdministratorR3']/Target/AnyOf/AllOf/Match[AttributeValue="Passive"]/AttributeValue/text()
      return if($status = "Passive")
      then
         let $attr := #NEW
         let $prduration := collection("book")/Rule[@RuleId='AdministratorR3']/Target/AnyOf/AllOf/Match[AttributeDesignator[@AttributeId="prinfo_duration"]]/AttributeValue/text()
         return if($prduration ge $attr)
         then
            delete node collection("book")/Rule[@RuleId='AdministratorR3']/Target/AnyOf/AllOf/Match[AttributeValue="PatientRecord"]
         else ()
      else()
   </cond_action>
</ecarule>
```

**Figure 3. PRDuration_Replace Rule**

The condition action part of this rule is enclosed within the node `<cond_action>`. The condition part of the rule checks whether the attribute “prinfo_duration” is passive or active. If there exists a passive patient record, then it checks whether the patient record duration of the patient is greater than replaced value of reference/threshold “patient
record duration”. If this is true, then the action part of this rule is triggered. The action includes deleting the relevant passive patient records from the database. If the conditions are evaluated to be false, then nothing will be done.

### 3.4 ECA Rule 3

The ECA rule 3 “ExpiredMedicinesBefore_Delete” is shown in Figure 4. In the “Pharmacist” policies, the pharmacist is responsible to manage and modify the information pertaining to drugs/medicines. The pharmacist can also select the prescribed medicines for the patients in certain cases. The policies related to the pharmacist are found in appendix page 60.

The objective of this active rule is to delete the “Adalat5” expired medicine from the medical inventory. This is used in a situation where the pharmacist wants to delete one specific expired medicine from the inventory. If the medicine is not expired, then the delete operation is not allowed. The event type “delete” listed within the <type> tag triggers this active rule. The <focalpoint> tag contains the value as “Match” where the trigger is set and attribute category to be replaced is a “resource” that is specified within <category> tag. The node happen contains the type of trigger which consists of one of the two values: “before” or “after”. In this case, it is the “before” trigger. In this example,
the value within node `<doc>` is “Policy.xml”. The value within `<priority>` tag determines the execution order of the condition action part of the active rule in the before queue.

```xml
<ecarule>
  <ruleid>ExpiredMedicinesBefore_Delete</ruleid>
  <doc>Policy.xml</doc>
  <type>delete</type>
  <focalpoint>Match</focalpoint>
  <category>resource</category>
  <happen>before</happen>
  <priority>70</priority>
  <cond_action>
    let $s := collection('book')/Rule[@RuleId='Pharmacist84']/Target/AnyOf/AllOf/Match[AttributeDesignator[@AttributeId='exp_Date']]/AttributeValue/text()
    return if (xs:date(format-date(current-date(), "[Y0001]-[M01]-[D01]")) gt xs:date($s))
    then db:output(concat('Delete the medicine from inventory since it is expired.',true()))
    else db:output(concat('Delete the medicine from inventory has denied.',false()))
  </cond_action>
</ecarule>
```

**Figure 4. ExpiryMedicinesBefore_Delete Rule**

The condition action part of this active rule does the following. The system first checks whether the current date is greater than the expiration date. If this is true, then a message is displayed saying “Delete the medicine from inventory since it is expired. true” and then the user XQuery request is performed. The user XQuery request will delete the expired medicine from the database. If the condition is evaluated to be “false”, then the message displayed as “Delete the medicine from inventory has been denied. false” and the system will prevent the execution of the user XQuery request, which is deleting the expired medicine.
3.5 ECA Rule 4

Another example of ECA rule “PatientBillingInfo” is shown in Figure 5. In the “PatientBilling” policy, the administrator is responsible for select, replace the medication base amount, number of hours of the patient, billing per hour increase or decrease percentage and patient bill for both general and emergency department. The PatientBilling policy can be found in appendix page 65. The objective of this active rule is to compute Patient Billing information whenever the user updates the “Patient Billing per hour increase or decrease percentage” information.

In this ECA rule, the “replace” event type listed within the <type> tag triggers this active rule. The <focalpoint> tag contains the value as “Match” where the trigger is set and attribute category to be replaced is a “resource” that is specified within the <category> tag. The node <happen> contains the type of trigger which consists of one of the two values: “before” or “after”. In this case, it is the “after” trigger. In this example, the value within node <doc> is “Policy.xml”. The value within <priority> tag determines the execution order of the condition action part of the active rule in the after queue.

The condition action part of this rule is enclosed within the node <cond_action>. The condition part of the rule checks to see whether the user replaced attribute value is not equal to zero. After this, it also checks whether the number of hours for which the patient admitted in the hospital is not equal to zero. If all the conditions are evaluated to be true,
then the action part of the rule will be triggered. If false, then nothing will be done. The action part of the rule calculates the Patient Billing information by obtaining the following information such as number of hours for which the patient was admitted in the hospital, the medication base amount and the updated percentage increase or decrease of the “Patient Billing per hour” information from the policy “GeneralWardBilling”.

```xml
<ecarule>
  <ruleid>PatientBillingInfo</ruleid>
  <doc>Policy.xml</doc>
  <type>replace</type>
  <focalpoint>Watch</focalpoint>
  <category>resource</category>
  <happen>after</happen>
  <priority>80</priority>
  <cond_action>
    let $pb := #PNODE/AttributeValue/text()
    return if ($pb != $0)
    then
      let $n := #PNODE/.../Watch[AttributeDesignator[
        @AttributeId='p_billingNoOfHours']]/AttributeValue/text()
      return if ($n != $0)
      then
        let $bamt := #PNODE/.../Watch[AttributeDesignator[
          @AttributeId='med_baseamt']]/AttributeValue/text()
        return if($bamt != $0)
        then
          replace value of node collection('book')/Rule[@RuleId='GeneralWardBilling']/Target/AnyOf/AllOf/Watch[AttributeDesignator[@AttributeId='p_billingInfo']][@Category='PatientBillingInfo:resource']]/AttributeValue with (($bamt+$bamt*$pb)/$n)
        else()
        else()
      </cond_action>
</ecarule>
```

**Figure 5. PatientBillingInfo Rule**
4. ECA RULES for TEMPORAL EVENTS (TIMER RULES)

4.1 Timer Rule Structure

The timer rule structure is shown in Figure 6. The tags which are used to write timer rules are <ecarules>, <ecarule>, <ruleid>, <doc>, <type>, <starttime>, <frequency> and <cond_action>. There can be one or more <ecarule> blocks enclosed within the <ecarules> tag. The <ecarule> marks the beginning of a new timer rule. The

<ecarules>
  <ecarule>
    <ruleid></ruleid>
    <doc></doc>
    <type></type>
    <starttime></starttime>
    <frequency></frequency>
    <cond_action></cond_action>
  </ecarule>
</ecarules>

Figure 6. Timer Rule Structure

<ruleid> tag contains the name of the timer rule which is unique. <doc> is the XACML file name denoting the document modified that triggers the execution of the timer rule [5]. The value within <type> tag indicates the event type such as temporal etc.

The <starttime> tag indicates the start time to trigger the timer rule. The value which the <starttime> can take ranges from “0-23 hours and 0-59 minutes”. The value
within `<frequency>` tag denotes the frequency with which the timer rule needs to be executed. The `<cond_action>` block contains the condition and action part of the ECA rule for temporal events. The condition action part is written in the form of XQuery statements. If the condition/conditions inside the `<cond_action>` tag is/are satisfied, then the action part of the timer rule will be triggered. The following sub-sections present examples of timer rules.

4.2 Timer Rule 1

The timer rule 1 “Timely_DrugQty_Check” is shown in Figure 7. The execution of this timer rule is based on the “Pharmacist” policy set. In the “Pharmacist” policy set, the pharmacist is responsible to select and modify the information pertaining to drugs/medicines. The pharmacist can also select the prescribed medicines of the patient. The policies related to pharmacist are found in appendix page 60.

The objective of this timer rule is to check the stock quantity of all the medicines defined in the policies. In this timer rule example, the value within the `<type>` tag is “temporal”. The start time and frequency information which are listed within the corresponding tags such as `<starttime>` and `<frequency>` triggers this timer rule. In this case, the timer rule is triggered daily at 23:00 PM. `<doc>` contains the XACML policy filename, “Policy.xml”, in which the trigger was set [5].
The condition action part is enclosed within the <cond_action> tag. In this timer rule 1, the following steps take place for all the medicines defined in the policies. The system first retrieves the threshold and checks for the quantity of a medicine that is in stock. If the quantity of the medicine is less than its threshold, then the system displays the relevant medicine name with the following message “Drug quantity is less than threshold. Please restock the medicine inventory” and sends an e-mail notification to the pharmacist. From the “Pharmacist” policy set, we know that the pharmacist is responsible
for restocking the medicines in the inventory. Therefore, an email notification is sent to the pharmacist in order to restock the medicines. If the medicine quantity in stock is greater than threshold, then do nothing.

4.3 Timer Rule 2

The timer rule 2 “ExpiredMedicines_Delete” is shown in Figure 8. The execution of this timer rule is also based on the “Pharmacist” policy set. In the “Pharmacist” policy set, the pharmacist is responsible for selecting and modifying the information pertaining to drugs/medicines. The pharmacist can also select the prescribed medicines for the patients. The policies related to pharmacist are found in appendix page 60.

The objective of this timer rule 2 is to delete all the expired medicines from the inventory. This rule is similar to ECA rule 3, except that this rule incorporates automatic execution based on time and frequency specified in the rule and also the checks the expiry date for all medicines and takes the corresponding actions when necessary.

In this timer rule example, the value within the <type> tag is “temporal” and <doc> tag is “Policy.xml”. The start time and frequency information which are listed within the corresponding tags such as <starttime> and <frequency> triggers this timer rule. In this case, the timer rule is triggered daily at 1:00 AM. In this case, the value within <doc> tag is “Policy.xml”.
The condition action part is enclosed within `<cond_action>` tag. In the timer rule 2, the following steps take place for all the medicines defined in the policies. Firstly, the system checks whether the current system date is greater than the expiration date of a medicine. If the condition is evaluated to be true, then the rule will trigger the removal of the respective medicine from the medicine inventory. Otherwise, nothing will be triggered.
4.4 Timer Rule 3

The timer rule 3 “NurseInternAccess.Replace” is shown in Figure 9. The execution of this timer rule is also based on the “NurseTechIntern” policy set. In the “Nurse Technician Intern” policy, the interns are responsible for selecting and inserting progress notes of the patient and can select the current medication of a patient. This policy is shown in appendix page 54.

The objective of this timer rule 3 is to change the access permission from "permit" to "deny" every day at 6pm for all nurse intern policies. Since the nurse interns work for day-shift, the nurse interns are allowed to access all the resource from morning till 6pm. After 6pm, the permission to access all the resources are denied.

In this timer rule example, the value within the <type> tag is “temporal” and <doc> tag is “Policy.xml”. The start time and frequency information which are listed within the corresponding tags such as <starttime> and <frequency> triggers this timer rule. In this case, the timer rule is triggered daily at 18:00 PM.

The condition action part is enclosed within <cond_action> tag. In the timer rule 3, the system first checks whether the rule effect is “Permit” for the NurseTechIntern policies. If the condition is evaluated to be true, then the rule will trigger by changing the
access permission to “deny”. Otherwise, nothing will be triggered. A similar rule can be written to change the permission to “permit” everyday morning when the day-shift starts.

```xml
<ecarule>
  <ruleid>NurseInternAccess_Replace</ruleid>
  <doc>Policy.xml</doc>
  <type>temporal</type>
  <starttime>18:00</starttime>
  <frequency>daily</frequency>
  <cond_action>
    let $r := collection("book")//PolicySet[
      @PolicySetId="NurseTechInternPS"]/Policy/Rule/@RuleId/string()
    for $x in $r
      let $z := collection("book")//PolicySet[
        @PolicySetId="NurseTechInternPS"]/Policy/Rule/@Effect/string()
      return if($z = "Permit")
      then
        replace value of node collection('book')//PolicySet[
          @PolicySetId="NurseTechInternPS"]/Policy/Rule[@RuleId=$x]/@Effect with 'Deny'
      else()
      </cond_action>
</ecarule>
```

**Figure 9. NurseInternAccess_Replace Rule**

### 4.5 Timer Rule 4

The timer rule 4 “NursePromotion_Replace” is shown in Figure 10. The execution of this timer rule is also based on the “NurseEmployees” policy set. In the “NurseEmployees” policy, the nurse employee has access to select their role and work start date. This policy is shown in appendix page 52.
The objective of this timer rule 4 is to calculate the experience every day for each and every nurse employee and promote the nurse to senior level in the hierarchy, when the experience criterion for promotion is met.

In this timer rule example, the value within the <type> tag is “temporal” and <doc> tag is “Policy.xml”. The start time and frequency information which are listed within the corresponding tags such as <starttime> and <frequency> triggers this timer rule. In this case, the timer rule is triggered daily at 23:00 PM.

```xml
<carule>
  <ruleid>NursePromotion_Replace</ruleid>
  <doc>Policy.xml</doc>
  <type>temporal</type>
  <starttime>23:00</starttime>
  <frequency>daily</frequency>
  <cond_action>
    let $r := collection('book')/PolicySet[@PolicySetId="NurseEmployeesPS"]/Policy/Rule/@RuleId|string()
    for $x in $r
      let $ws_date := collection('book')/PolicySet[@PolicySetId="NurseEmployeesPS']/Policy/Rule[@RuleId=$x]/Target/AnyOf/AllOf/Match/AttributeDesignator[@AttributeId='workstartdate']/AttributeValue/text()
      let $diff_date := xs:dateTime(xs:dayTimeDuration('P0Y0M-[M01]-[D01]')) - $ws_date
      let $exp_yrs := days-from-duration($diff_date) div 365
      return if($exp_yrs ge 5)
      then
        replace value of node collection('book')/PolicySet[@PolicySetId='NurseEmployeesPS']/Policy/Rule[@RuleId=$x]/Target/AnyOf/AllOf/Match/AttributeDesignator[@AttributeId='nurse_pos']/AttributeValue/text() with 'ChargeNurse'
      else()
  </cond_action>
</carule>
```

**Figure 10. NursePromotion_Replace Rule**
The condition action part is enclosed within \texttt{<cond_action>} tag. In timer rule 4, the system retrieves the work start date of the nurse employees. Then, calculates the experience every day for all the nurse employee based upon their work start date. The experience increases by 1 day by day. The difference between the current date and work start date yields an experience value in terms of days. This experience value in turn is converted in terms of years. If the years of experience is greater than 5 years, then a staff nurse gets promoted to charge nurse. By doing so, they can have more access right.
5. SYSTEM ARCHITECTURE

5.1 Architecture for ECA Rule Execution

The system architecture for the execution of ECA rules is depicted in Figure 11.

First and foremost, the ECA rule writer creates the ECA rules. Then the ECA rules are
parsed and some of the attributes/information are extracted from the parsed information. The query is formed by using parsed information and it is used to store the ECA rules into a rule repository. The rule repository is composed of three types of collection such as insert, replace and delete. Based on the mutation event type, the ECA rules are stored in their respective collection.

On the other hand, the user provides a request to access the resource in the form of XQuery statements. The user defined XQuery statements are provided as an input to the parser. Then, the parser parses the user request and extracts the required information in order to create an XQuery request.

Depending upon the event type, the user XQuery request is fed into their respective collections in the rule repository to check if there are any relevant ECA rules. If the relevant rule exists, then the ECA rule is retrieved. Immediately after ECA rules are retrieved, the user XQuery statements are executed and the retrieved ECA rules are also executed based upon the trigger type. For example, if the trigger type is “before”, the system first executes the ECA rules and then executes the user defined XQuery statements. If the trigger type is “after”, the system first executes the user defined XQuery statements and then executes the ECA rules. Therefore, the XACML policies or the rules within the policy or policy set are modified based upon the user request and the active rules trigger.
5.2 Architecture for Timer Rule Execution

The system architecture for the execution of timer rule is depicted in Figure 12. Firstly, write the timer rules and this is done by the timer rule writer. Then, the timer rules are parsed and stored in the temporal collection of the rule repository.

![Diagram](image-url)

**Figure 12. ECA Rule for Temporal Events Execution Model**
As a result of parsing, the start time and frequency information are extracted and these information are provided to the Cron Trigger in the Quartz scheduler.

The Cron Trigger uses the start time and frequency information, in order to schedule one or more number of jobs at a given time and frequency interval. The job class checks for any corresponding timer rules in the rule repository by providing time and frequency details. If this exists, then the corresponding timer rules are retrieved.

After retrieving the timer rule, the condition action part of the rule is extracted and the condition action is executed at a specific time and frequency interval. As a result, the XACML policies or rules within policy or policy set are modified.
6. IMPLEMENTATION

6.1 Implementation of ECA Rule Execution

The starter class invokes the system calls for Interface 1 and Interface 2. The execution of the ECA rules for Interface 1 and Interface 2 is shown in Figure 13 and Figure 15 respectively.

In Interface 1, the ECA rule writer is responsible for writing the ECA rules. Then the written ECA rules are provided to the parser by invoking the ParseNStoreECA() method which resides within ECAParseNStore class. The ECA rule parser extracts the needed information from the provided active rules. As a result, the extracted information is restructured and then it calls the storeECA() method to store the restructured ECA rules into the rule repository.

![Diagram of Interface 1 Workflow for ECA Rules]

**Figure 13. Interface 1 Workflow for ECA Rules**

The rule repository for active rules consists of three types of collections: insert, delete and replace. The ECA rules are stored into the respective collections of the rule
repository based on their event type. Each collection consists of a filename which is “Policy.xml”, subfocalpoint, focalpoint, ruleid and the parsed string. The parsed string consists of the following information: priority, happen and condition action. The generic structure of rule repository collection for an active rule is shown in Figure 14.

![Rule Repository Collection Structure for Active Rule](image)

**Figure 14. Rule Repository Collection Structure for Active Rule**

The workflow for Interface 2 is shown in Figure 15. In Interface 2, the user provides his/her access request in the form of the XQuery statements. Then the user XQuery statement is sent to the parser by means of invoking the parseXQuery() method. The user XQuery statement is parsed in order to determine the mutation event type. Once the event type is determined, the respective mutation event type method is called. For
example, if the mutation event type is “replace”, then call the replace() method in order to parse information such as Main Tag, Sub Main Tag and Parent node.

Figure 15. Interface 2 Workflow for ECA rules

The parseMainTag() function is used to extract the point of trigger such as focalpoint information. The parseSubMainTag() function is used to extract the category
type information. The parent node is extracted by means of calling the extractPNode() function. Then, the retrieveECA() method is called by providing the above parsed information in order to restructure the XQuery statement. Once the XQuery statement is restructured, executeQuery() method is invoked to determine whether there are any relevant ECA rules in the rule repository. If relevant ECA rules are found, then the corresponding rules are retrieved. Otherwise, nothing is done. After the ECA rules are retrieved, ParseNExecuteECA() method is called to execute the user XQuery and the retrieved ECA rules based upon happen, priority and condition action information. The executeQuery() method is invoked to run the query against the BaseX XML database. The XACML 3.0 policies are dynamically modified depending upon condition action part of the ECA rules.

6.2 Implementation of Timer Rule Execution

The starter class invokes the system calls for Interface 1 and Interface 2 of timer rules execution. The workflow for Interface 1 of timer rules is shown in Figure 16. In Interface 1, the ECA rules for temporal events writer is responsible for writing the timer rules. Then, the written timer rules are fed as an input to the parser by invoking the parseNStoreTemporal() method. Then, the parser extracts the needed information from the timer rules. As a result, the extracted information is restructured and then it calls the
storeTemporal() method to store the restructured timer rules into a temporal collection of rule repository.

**Figure 16. Interface 1 Workflow for Timer rules**

The workflow for Interface 2 of timer rules is shown in Figure 17. In Interface 2, an object for CronJob_Scheduler class has been created. By using this object, the main method within CronJob_Scheduler class is invoked. The main() function in turn invokes run() method which resides within CronJob_Scheduler class, in order to start scheduling and executing the jobs.

In the run() method, an object of “Quartz Std Scheduler factory” is initially created in order to get the Quartz scheduler. Then, the scheduler is started using the start() method. After starting the quartz scheduler, job details are created which holds information such as job name, group name and job class name. Then, a quartz cron trigger is created using the cron expression or the cron schedule builder by using a given time and frequency data. After the above step, the job is scheduled using the created job
Figure 17. Interface 2 Workflow for Timer Rules
and trigger. Whenever the system attains the specified time and frequency, the system executes the code which is defined within the job class. Each job has its own job detail which in turn contains a job class. The jobs interact with the rule repository and the XACML policies to retrieve rules and modify the policy. All the defined jobs get executed as per the start time and frequency which is set in JobDetail. The generic structure of rule repository collection for a timer rule is shown in Figure 18. The timer rules are stored into the temporal collections of the rule repository. The temporal collection consists of a filename which is “Policy.xml”, starttime and frequency information.

In this project, a job class named “Temporal_Job” is created. In the “Temporal_Job” class, the retrieveTemporalECA() method is invoked by providing the start time and frequency data, to retrieve relevant timer rules from the temporal collection.

![Figure 18. Rule Repository Collection Structure for Timer Rules](image-url)
of the rule repository. Then, the parseNExecuteTemporalCA() method is invoked to extract only the condition action part of the relevant timer rules. Finally, the executeQuery() method is called to check the condition and trigger the action part of the timer rule by interacting with XACML 3.0 policies.

6.3 Implementation of Sending Email

The flow of sending an email from the java application is shown in Figure 19. To

![Figure 19. Sending Email from Java Application [14]](image-url)
send an email from a Java application, JavaMail API is used. The JavaMail API provides classes that model a mail system [13]. The JavaMail API supports protocols such as SMTP, POP, IMAP, MIME and others. In this project, the timer rule 1 uses Gmail SMTP server to send an email notification to pharmacist from the Java application.

Firstly, an object of the Properties class is created. Using the created object for properties, the SMTP server settings are set. Then, a session is instantiated and the Gmail username and password of the user are provided for authentication. Secondly, a message object is created. The message object contains information such as: “from email id”, “to email id”, “subject” and “text message”. The above information is set in order to exchange email messages between the users over a network. Finally the email message is sent by using the Transport.send() system call by passing the message objects as an argument. Therefore, the email is transmitted to intended recipient successfully.
7. CONCLUSION AND FUTURE DIRECTION

In this project, we have implemented the active rule manager for XACML 3.0 policy system. To elaborate, we have enhanced and implemented the system from XACML 2.0 which was part of previous Master’s project to XACML 3.0 policy by incorporating active rules and active rules for temporal events in order to utilize the benefits of XACML 3.0 policies. The system consists of two interfaces. Interface 1 is responsible to parse and store the written active rules and timer rules into the respective rule repository. For the execution of active rules, Interface 2 is responsible to parse user XQuery statements, retrieve relevant ECA rules from the repository and execute query in order to modify the information in the policy. For the execution of timer rules, Interface 2 is responsible to schedule, retrieve and execute temporal jobs.

The current system can be enhanced in the future in the following ways. In the current system, a user should be aware of XQuery language, so that the user can query in the form of XQuery statement on interface 2 of ECA rules execution, for accessing the resources. The current system can be made more user friendly by developing a graphical user interface on Interface 2 of the ECA rules execution for this application. The system can also be improved by coding more efficiently and to use efficient scheduler for scheduling temporal based events.
Appendix

XACML 3.0 Policies for Medical Application

```xml
<PolicySet>
  <PolicySet PolicySetId="AdministratorPS" PolicyCombiningAlgId="permit-overrides">
    <Description>
      This PolicySet is a collection of all policies related to Administrator
    </Description>
    <!-- --------------------------------------------------------------------- -->
    <Policy PolicyId="Administrator_Generic" RuleCombiningAlgId="permit-overrides">
      <!-- Patient record duration Info access to Administrator -->
      <Rule RuleId="AdministratorR1" Effect="Permit">
        <Target>
          <AnyOf>
            <AllOf>
              <Match MatchId="string-equal">
                <AttributeValue DataType="string">Administrator</AttributeValue>
                <AttributeDesignator AttributeId="admin" DataType="string"/>
              </Match>
              <Category="Administrator:access-subject"/>
            </AllOf>
          </AnyOf>
          <AnyOf>
            <AllOf>
              <Match MatchId="string-equal">
                <AttributeValue DataType="string">3</AttributeValue>
                <AttributeDesignator AttributeId="prinfo_duration_set" DataType="string"/>
              </Match>
              <Category="PrInfo_Duration_set:access-subject"/>
            </AllOf>
          </AnyOf>
          <AnyOf>
            <AllOf>
              <Match MatchId="string-equal">
                <AttributeValue DataType="string">Insert</AttributeValue>
                <AttributeDesignator AttributeId="insert" DataType="string" Category="Insert:action"/>
              </Match>
              <Category="Replace:action"/>
            </AllOf>
          </AnyOf>
        </Target>
      </Rule>
    </PolicySet>
  </PolicySet>
</PolicySet>
```
<Rule Id="AdministratorR2" Effect="Permit">
  <AnyOf>
    <AllOf>
      <Match MatchId="string-equal">
        <AttributeValue DataType="string">Administrator</AttributeValue>
        <AttributeDesignator AttributeId="admin" DataType="string"/>
      </Match>
    </AllOf>
    Category="Administrator:access-subject"/>
  </AnyOf>
  <AnyOf>
    <AllOf>
      <Match MatchId="string-equal">
        <AttributeValue DataType="string">PatientInfo</AttributeValue>
        <AttributeDesignator AttributeId="pinfo" DataType="string"/>
      </Match>
    </AllOf>
    Category="PatientInfo:resource"/>
  </AnyOf>
  <AnyOf>
    <AllOf>
      <Match MatchId="string-equal">
        <AttributeValue DataType="string">PhysicianInfo</AttributeValue>
        <AttributeDesignator AttributeId="phys_info" DataType="string"/>
      </Match>
    </AllOf>
    Category="PhysicianInfo:resource"/>
  </AnyOf>
  <AnyOf>
    <AllOf>
      <Match MatchId="string-equal">
        <AttributeValue DataType="string">PharmacistInfo</AttributeValue>
        <AttributeDesignator AttributeId="pharma_info" DataType="string"/>
      </Match>
    </AllOf>
    Category="PharmacistInfo:resource"/>
  </AnyOf>
  <AnyOf>
    <AllOf>
      <Match MatchId="string-equal">
        <AttributeValue DataType="string">Select</AttributeValue>
        <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
      </Match>
    </AllOf>
    <Match MatchId="string-equal">
      <AttributeValue DataType="string">Insert</AttributeValue>
      <AttributeDesignator AttributeId="insert" DataType="string" Category="Insert:action"/>
    </Match>
    <Match MatchId="string-equal">
      <AttributeValue DataType="string">Delete</AttributeValue>
      <AttributeDesignator AttributeId="delete" DataType="string" Category="Delete:action"/>
    </Match>
  </AnyOf>
</Rule>
<!-- Administrator having access to passive patient record -->

<Rule RuleId="AdministratorR3" Effect="Permit">
<Target>
<AnyOf>
</AnyOf>
</Target>
</Rule>

<!-- Administrator having access to active patient record -->

<Rule RuleId="AdministratorR4" Effect="Permit">
<Target>
<AnyOf>
</AnyOf>
</Target>
</Rule>
<Target>
  <AnyOf>
    <AllOf>
      <Match MatchId="string-equal">
        <AttributeValue DataType="string">Administrator</AttributeValue>
      </Match>
      Category="Administrator:access-subject"/
    </AllOf>
    <Match MatchId="string-equal">
      <AttributeValue DataType="string">PatientRecord</AttributeValue>
      Category="PatientRecordInfo:resource"/
    </Match>
    <Match MatchId="string-equal">
      <AttributeValue DataType="string">2</AttributeValue>
      Category="PrInfo_Duration:access-subject"/
    </Match>
    <Match MatchId="string-equal">
      <AttributeValue DataType="string">Active</AttributeValue>
      Category="PRStatus:resource"/
    </Match>
    <AllOf>
      <Match MatchId="string-equal">
        <AttributeValue DataType="string">Select</AttributeValue>
        Category="Select:action"/
      </Match>
      <Match MatchId="string-equal">
        <AttributeValue DataType="string">Delete</AttributeValue>
        Category="Delete:action"/
      </Match>
    </AllOf>
    <AnyOf>
    </AnyOf>
  </AnyOf>
</Target>
</Rule>
</Policy>
</PolicySet>

<PolicySet PolicySetId="NursePS" PolicyCombiningAlgId="permit-overrides">
  <Description>
    This PolicySet is a collection of all policies related to Nurses
  </Description>
</PolicySet>
<!-- Contains "Nurse" policies that give access to generic tables -->

```
<Policy PolicyId="Nurse_Generic" RuleCombiningAlgId="permitoverrides">

<!-- PatientInfo table access to the attending Nurse of a patient -->
<Rule RuleId="NGenR1" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">ChargeNurse</AttributeValue>
          <AttributeDesignator AttributeId="nurse_pos" DataType="string" Category="Nurse:access-subject"/>
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">PatientInfo</AttributeValue>
          <AttributeDesignator AttributeId="p_info" DataType="string" Category="PatientInfo:resource"/>
        </Match>
      </AllOf>
      <AnyOf>
        <Condition>
          <Apply FunctionId="string-equal">
            <AttributeDesignator AttributeId="Nurse:subject-Id" DataType="string"/>
          </Apply>
          <Apply FunctionId="string-equal">
            <AttributeDesignator AttributeId="**AttendingNurse-Id***" DataType="string"/>
          </Apply>
          <Apply FunctionId="string-equal">
            <AttributeDesignator AttributeId="Patient:subject-id" DataType="string"/>
          </Apply>
          <Apply FunctionId="string-equal">
            <AttributeDesignator AttributeId="**AssignedPatient-id***" DataType="string"/>
          </Apply>
        </Condition>
      </AnyOf>
    </AllOf>
  </Target>
</Rule>

<!-- Diagnosis/LabResults/Medications table access to the attending Nurse of a patient -->
<Rule RuleId="NGenR2" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">ChargeNurse</AttributeValue>
          <AttributeDesignator AttributeId="nurse_pos" DataType="string" Category="Nurse:access-subject"/>
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Diagnosis</AttributeValue>
          <AttributeDesignator AttributeId="dg" DataType="string" Category="Diagnosis:resource"/>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</Rule>
```
<Match MatchId="string-equal">
  <AttributeValue DataType="string">LabResults</AttributeValue>
  <AttributeDesignator AttributeId="lr" DataType="string" Category="LabResults:resource"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">Medications</AttributeValue>
  <AttributeDesignator AttributeId="med" DataType="string" Category="Medications:resource"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">Select</AttributeValue>
  <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
</Match>

<!-- Patient Record table access to Nurses -->
<Rule RuleId="NGenR3" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">ChargeNurse</AttributeValue>
          <AttributeDesignator AttributeId="nurse_pos" DataType="string" Category="Nurse:access-subject"/>
        </Match>
      </AllOf>
    </AnyOf>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">PatientRecords</AttributeValue>
          <AttributeDesignator AttributeId="p_records" DataType="string" Category="PatientRecords:resource"/>
        </Match>
      </AllOf>
    </AnyOf>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Select</AttributeValue>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</Rule>
<AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>
<Rule RuleId="NGenR4" Effect="Permit">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="string-equal">
<AttributeValue DataType="string">StaffNurse</AttributeValue>
</Match>
</AllOf>
<Match MatchId="string-equal">
<AttributeValue DataType="string">PatientInfo</AttributeValue>
</Match>
</AllOf>
<Match MatchId="string-equal">
<AttributeValue DataType="string">PatientRecords</AttributeValue>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>
<Rule RuleId="NGenR5" Effect="Permit">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="string-equal">
<AttributeValue DataType="string">Select</AttributeValue>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>
<Match MatchId="string-equal">
<AttributeValue DataType="string">StaffNurse</AttributeValue>
</Match>
<Match MatchId="string-equal">
<AttributeValue DataType="string">Medications</AttributeValue>
</Match>
Category="Medications:resource"/>
<Match MatchId="string-equal">
  <AttributeValue DataType="string">Select</AttributeValue>
  <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
</Match>
</AllOf>
</AnyOf>
</AnyOf>
</AllOf>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">Select</AttributeValue>
  <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>
</Policy>
</PolicySet>

<!-- ************************************************************************************************************ -->

<PolicySet PolicySetId="NurseEmployeesPS" PolicyCombiningAlgId="permit-overrides">
  <Description>
    This PolicySet is a collection of all policies related to Nurse-Designation
  </Description>
</PolicySet>

<!-- ************************************************************************************************************ -->

<!-- Contains "Nurse Employee" policies that give access to generic tables -->
<Policy PolicyId="NurseEmployees_Generic" RuleCombiningAlgId="permit-overrides">
  <Rule RuleId="NEmpGenR1" Effect="Permit">
    <Target>
      <AnyOf>
        <AllOf>
          <Match MatchId="string-equal"/>
          <AttributeValue DataType="string">Sarah Diana</AttributeValue>
          <AttributeDesignator AttributeId="Name" DataType="string" Category="Name:access-subject"/>
        </AllOf>
        <Match MatchId="string-equal"/>
        <AttributeValue DataType="string">ChargeNurse</AttributeValue>
        <AttributeDesignator AttributeId="nurse_pos" DataType="string" Category="Nurse:access-subject"/>
      </AnyOf>
      <Match MatchId="string-equal"/>
      <AttributeValue DataType="string">2006-01-10</AttributeValue>
      <AttributeDesignator AttributeId="workstartdate" DataType="string" Category="WorkStartDate:resource"/>
    </Target>
  </Rule>
</Policy>
<Match MatchId="string-equal">
  <AttributeValue DataType="string">Select</AttributeValue>
  <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
</Match>
</AnyOf>
</Target>
</Rule>

<Rule RuleId="NEmpGenR2" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Mary Jane</AttributeValue>
          <AttributeDesignator AttributeId="Name" DataType="string" Category="Name:access-subject"/>
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">StaffNurse</AttributeValue>
          <AttributeDesignator AttributeId="nurse_pos" DataType="string" Category="Nurse:access-subject"/>
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">2011-10-10</AttributeValue>
          <AttributeDesignator AttributeId="workstartdate" DataType="string" Category="WorkStartDate:resource"/>
        </Match>
      </AllOf>
    </AnyOf>
    </Target>
</Rule>

<Rule RuleId="NEmpGenR3" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Select</AttributeValue>
          <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
        </Match>
      </AllOf>
    </AnyOf>
    </Target>
</Rule>
<Match MatchId="string-equal">
  <AttributeValue DataType="string">StaffNurse</AttributeValue>
  <AttributeDesignator AttributeId="nurse_pos" DataType="string"/>
Category="Nurse:access-subject"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">2014-03-28</AttributeValue>
  <AttributeDesignator AttributeId="workstartdate" DataType="string"/>
Category="WorkStartDate:resource"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">StaffNurse</AttributeValue>
  <AttributeDesignator AttributeId="nurse_pos" DataType="string"/>
Category="Nurse:access-subject"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">Select</AttributeValue>
  <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">NurseTechnicianIntern</AttributeValue>
  <AttributeDesignator AttributeId="NurseIntern_pos" DataType="string"/>
Category="NurseIntern:access-subject"/>
</Match>
<AnyOf>
  <AllOf>
    <Match MatchId="string-equal">
      <AttributeValue DataType="string">ProgressNotes</AttributeValue>
      <AttributeDesignator AttributeId="prg_notes" DataType="string"
        Category="ProgressNotes:resource"/>
    </Match>
  </AllOf>
  <AnyOf>
    <Match MatchId="string-equal">
      <AttributeValue DataType="string">Select</AttributeValue>
      <AttributeDesignator AttributeId="select" DataType="string"
        Category="Select:action"/>
    </Match>
  </AnyOf>
  <Match MatchId="string-equal">
    <AttributeValue DataType="string">Insert</AttributeValue>
    <AttributeDesignator AttributeId="insert" DataType="string"
      Category="Insert:action"/>
  </Match>
</AnyOf>
</AnyOf>
</Target>
</Rule>

<Rule RuleId="NurseInternR2" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">NurseTechnicianIntern</AttributeValue>
          <AttributeDesignator AttributeId="NurseIntern_pos" DataType="string"
            Category="NurseIntern:access-subject"/>
        </Match>
      </AllOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">CurrentMedications</AttributeValue>
            <AttributeDesignator AttributeId="cur_med" DataType="string"
              Category="CurrentMedications:resource"/>
          </Match>
        </AllOf>
        <AnyOf>
          <AllOf>
            <Match MatchId="string-equal">
              <AttributeValue DataType="string">Select</AttributeValue>
              <AttributeDesignator AttributeId="select" DataType="string"
                Category="Select:action"/>
            </Match>
          </AllOf>
        </AnyOf>
      </AnyOf>
    </AnyOf>
  </Target>
</Rule>
<PolicySet PolicySetId="PhysicianPS" PolicyCombiningAlgId="permitoverrides">
  <Description>
    This PolicySet is a collection of all policies related to Physicians
  </Description>
  <Policy PolicyId="Physician_Generic" RuleCombiningAlgId="permitoverrides">
    <!-- PatientHistory/PatientRecord/LabResults/CurrentMedications table access to Physician -->
    <Rule RuleId="PhysicianR1" Effect="Permit">
      <Target>
        <AnyOf>
          <AllOf>
            <Match MatchId="string-equal">
              <AttributeValue DataType="string">Physician</AttributeValue>
              <AttributeDesignator AttributeId="Position" DataType="string" />
            </Match>
          </AllOf>
        </AnyOf>
        <AnyOf>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">PatientHistory</AttributeValue>
            <AttributeDesignator AttributeId="p_history" DataType="string" />
          </Match>
        </AnyOf>
        <AnyOf>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">PatientRecord</AttributeValue>
            <AttributeDesignator AttributeId="p_record" DataType="string" />
          </Match>
        </AnyOf>
        <AnyOf>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">LabResults</AttributeValue>
          </Match>
        </AnyOf>
      </Target>
    </Rule>
  </Policy>
</PolicySet>
<AttributeDesignator AttributeId="lab_res" DataType="string" Category="LabResults:resource"/>
</Match>
</AllOf>
</AnyOf>
<AnyOf>
<AllOf>
<Match MatchId="string-equal">
<AttributeValue DataType="string">CurrentMedications</AttributeValue>
<AttributeDesignator AttributeId="cur_med" DataType="string" Category="CurrentMedications:resource"/>
</Match>
</AllOf>
</AnyOf>
<Condition>
<Apply FunctionId="string-equal">
<AttributeDesignator AttributeId="PrimaryPhysician:subject-Id" DataType="string"/>
<AttributeDesignator AttributeId="**AttendingPhysician-Id**" DataType="string"/>
</Apply>
<Apply FunctionId="string-equal">
<AttributeDesignator AttributeId="Patient:subject-id" DataType="string"/>
<AttributeDesignator AttributeId="**AssignedPatient-id**" DataType="string"/>
</Apply>
</Condition>
</Target>
</Rule>

<!-- Medications table access to Physician -->
<Rule RuleId="PhysicianR2" Effect="Permit">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="string-equal">
<AttributeValue DataType="string">Physician</AttributeValue>
<AttributeDesignator AttributeId="Position" DataType="string" Category="Physician:access-subject"/>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>
<AttributeValue DataType="string">Medications</AttributeValue>
<AttributeDesignator AttributeId="med" DataType="string">
Category="Medications:resource"/>
</Match>
</AllOf>
</AnyOf>
</AnyOf>
</AllOf>

<Match MatchId="string-equal">
<AttributeValue DataType="string">Select</AttributeValue>
<AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
</Match>
</Match>
</MatchId>
<AttributeValue DataType="string">Insert</AttributeValue>
<AttributeDesignator AttributeId="insert" DataType="string" Category="Insert:action"/>
</Match>
</Match>
</AllOf>
</AnyOf>
</Condition>

<Apply FunctionId="string-equal">
<AttributeDesignator AttributeId="PrimaryPhysician:subject-Id" DataType="string"/>
<AttributeDesignator AttributeId="**AttendingPhysician-Id**" DataType="string"/>
</Apply>
</Apply>
</Apply>
</Condition>
</Target>
</Rule>

<!-- Medication table access to Emergency department Physician -->
<Rule RuleId="PhysicianR3" Effect="Permit">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="string-equal">
<AttributeValue DataType="string">Physician</AttributeValue>
<AttributeDesignator AttributeId="Position" DataType="string">
Category="Physician:access-subject"/>
</Match>
</Match>
</MatchId>
<AttributeValue DataType="string">Emergency</AttributeValue>
<AttributeDesignator AttributeId="emergency" DataType="string">
Category="Emergency:access-subject"/>
</Match>
</Match>
</Match>
<Match MatchId="string-equal">
<AttributeValue DataType="string">20</AttributeValue>
<AttributeDesignator AttributeId="exp" DataType="string" Category="Exp:access-subject"/>
<Match>
  <AllOf>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">PatientHistory</AttributeValue>
          <AttributeDesignator AttributeId="p_history" DataType="string" />
        </Match>
      </AllOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">PatientRecord</AttributeValue>
            <AttributeDesignator AttributeId="p_record" DataType="string" />
          </Match>
        </AllOf>
      </AnyOf>
    </AnyOf>
  </AllOf>
  <AnyOf>
    <AllOf>
      <Match MatchId="string-equal">
        <AttributeValue DataType="string">LabResults</AttributeValue>
        <AttributeDesignator AttributeId="lab_res" DataType="string" />
      </Match>
    </AllOf>
  </AnyOf>
</AnyOf>
<AllOf>
  <Match MatchId="string-equal">
    <AttributeValue DataType="string">CurrentMedications</AttributeValue>
    <AttributeDesignator AttributeId="cur_med" DataType="string" />
  </Match>
</AllOf>
<AllOf>
  <Match MatchId="string-equal">
    <AttributeValue DataType="string">Select</AttributeValue>
    <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action" /> 
  </Match>
</AllOf>
<AllOf>
  <Match MatchId="string-equal">
    <AttributeValue DataType="string">Insert</AttributeValue>
    <AttributeDesignator AttributeId="insert" DataType="string" Category="Insert:action" />
  </Match>
</AllOf>
<PolicySet PolicySetId="PharmaPS" PolicyCombiningAlgId="permit-overrides">
  <Description>
    This PolicySet is a collection of all policies related to Pharmacy
  </Description>
  <Policy PolicyId="Pharmacist_Generic" RuleCombiningAlgId="permit-overrides">
    <Rule RuleId="PharmacistR1" Effect="Permit">
      <Target>
        <AnyOf>
        <AllOf>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">Pharmacist</AttributeValue>
            <AttributeDesignator AttributeId="PharmaPosition" DataType="string" Category="Pharmacist:access-subject"/>
          </Match>
        </AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Tylenol</AttributeValue>
          <AttributeDesignator AttributeId="medicine" DataType="string" Category="Medicines:resource"/>
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">600</AttributeValue>
          <AttributeDesignator AttributeId="Med_Qty" DataType="string" Category="MedicinesQty:resource"/>
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="date">2016-09-29</AttributeValue>
          <AttributeDesignator AttributeId="exp_Date" DataType="string" Category="ExpDate:resource"/>
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">800</AttributeValue>
          <AttributeDesignator AttributeId="threshold" DataType="string" Category="Threshold:resource"/>
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Select</AttributeValue>
          <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
        </Match>
      </AnyOf>
    </Target>
  </Policy>
</PolicySet>
<Match MatchId="string-equal">
  <AttributeValue DataType="string">Insert</AttributeValue>
  <AttributeDesignator AttributeId="insert" DataType="string" Category="Insert:action"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">Delete</AttributeValue>
  <AttributeDesignator AttributeId="delete" DataType="string" Category="Delete:action"/>
</Match>

</AllOf>
</AnyOf>
</Target>
</Rule>

<Rule RuleId="PharmacistR2" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Pharmacist</AttributeValue>
          <AttributeDesignator AttributeId="PharmaPosition" DataType="string"
          Category="Pharmacist:access-subject"/>
        </Match>

        <Match MatchId="string-equal">
          <AttributeValue DataType="string">VicksNyQuilCold</AttributeValue>
          <AttributeDesignator AttributeId="medicine" DataType="string"
          Category="Medicines:resource"/>
        </Match>

        <Match MatchId="string-equal">
          <AttributeValue DataType="string">500</AttributeValue>
          <AttributeDesignator AttributeId="Med_Qty" DataType="string"
          Category="MedicinesQty:resource"/>
        </Match>

        <Match MatchId="string-equal">
          <AttributeValue DataType="date">2016-08-28</AttributeValue>
          <AttributeDesignator AttributeId="exp_Date" DataType="string"
          Category="ExpDate:resource"/>
        </Match>

        <Match MatchId="string-equal">
          <AttributeValue DataType="string">800</AttributeValue>
          <AttributeDesignator AttributeId="threshold" DataType="string"
          Category="Threshold:resource"/>
        </Match>

        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Select</AttributeValue>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</Rule>
<AttributeDesignator AttributeId="select" DataType="string"
Category="Select:action"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">Insert</AttributeValue>
  <AttributeDesignator AttributeId="insert" DataType="string"
Category="Insert:action"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">Delete</AttributeValue>
  <AttributeDesignator AttributeId="delete" DataType="string"
Category="Delete:action"/>
</Match>

<AttributeDesignator AttributeId="Pharmacist"
Category="Pharmacist:access-subject"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">Cardicheck-10</AttributeValue>
  <AttributeDesignator AttributeId="medicine" DataType="string"
Category="Medicines:resource"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">1200</AttributeValue>
  <AttributeDesignator AttributeId="Med_Qty" DataType="string"
Category="MedicinesQty:resource"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="date">2016-09-26</AttributeValue>
  <AttributeDesignator AttributeId="exp_Date" DataType="string"
Category="ExpDate:resource"/>
</Match>

<Match MatchId="string-equal">
  <AttributeValue DataType="string">800</AttributeValue>
  <AttributeDesignator AttributeId="threshold" DataType="string"
Category="Threshold:resource"/>
<Match>
  <Match MatchId="string-equal">
    <AttributeValue DataType="string">Select</AttributeValue>
    <AttributeDesignator AttributeId="select" DataType="string" />
  </Match>
  <Match MatchId="string-equal">
    <AttributeValue DataType="string">Insert</AttributeValue>
    <AttributeDesignator AttributeId="insert" DataType="string" Category="Insert:action" />
  </Match>
  <Match MatchId="string-equal">
    <AttributeValue DataType="string">Delete</AttributeValue>
    <AttributeDesignator AttributeId="delete" DataType="string" />
  </Match>
</Match>

<Rule RuleId="PharmacistR4" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Pharmacist</AttributeValue>
          <AttributeDesignator AttributeId="PharmaPosition" DataType="string" />
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Adalat5</AttributeValue>
          <AttributeDesignator AttributeId="medicine" DataType="string" />
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">500</AttributeValue>
          <AttributeDesignator AttributeId="Med_Qty" DataType="string" />
        </Match>
        <Match MatchId="string-equal">
          <AttributeValue DataType="date">2016-08-27</AttributeValue>
          <AttributeDesignator AttributeId="exp_Date" DataType="string" />
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</Rule>
<Rule RuleId="PharmacistR5" Effect="Permit">
  <Target>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Pharmacist</AttributeValue>
          <AttributeDesignator AttributeId="PharmaPosition" DataType="string">
            "Pharmacist:access-subject"
          </AttributeDesignator>
        </Match>
      </AllOf>
    </AnyOf>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">PrescribedMedicines</AttributeValue>
          <AttributeDesignator AttributeId="pres_med" DataType="string">
            "PrescribedMedicines:resource"
          </AttributeDesignator>
        </Match>
      </AllOf>
    </AnyOf>
    <AnyOf>
      <AllOf>
        <Match MatchId="string-equal">
          <AttributeValue DataType="string">Select</AttributeValue>
          <AttributeDesignator AttributeId="select" DataType="string">
            "Select:action"
          </AttributeDesignator>
        </Match>
      </AllOf>
    </AnyOf>
  </Target>
</Rule>
<PolicySet PolicySetId="PatientBilling" PolicyCombiningAlgId="permit-overrides">
  <Description>This PolicySet is a collection of all policies related to patient billing depending upon type of diagnosis</Description>
</PolicySet>

<Policy PolicyId="PatientBilling-Diagnosis" RuleCombiningAlgId="permit-overrides">
  <Rule RuleId="GeneralWardBilling" Effect="Permit">
    <Target>
      <AnyOf>
        <AllOf>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">Administrator</AttributeValue>
            <AttributeDesignator AttributeId="admin_position" DataType="string" />
            Category="Administrator:access-subject" />
          </Match>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">GENERAL</AttributeValue>
            <AttributeDesignator AttributeId="Department_general" DataType="string" />
            Category="GENERAL:access-subject" />
          </Match>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">80</AttributeValue>
            <AttributeDesignator AttributeId="med_baseamt" DataType="string" />
            Category="MedicationsBaseAmt:resource" />
          </Match>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">0.1</AttributeValue>
            <AttributeDesignator AttributeId="p_billingperHourIncDec" DataType="string" />
            Category="PatientBillingPerHourIncDec:resource" />
          </Match>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">8</AttributeValue>
            <AttributeDesignator AttributeId="p_billingNoOfHours" DataType="string" />
            Category="PatientBillingNoOfHours:resource" />
          </Match>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">704</AttributeValue>
            <AttributeDesignator AttributeId="p_billingNoOfHours" DataType="string" />
            Category="PatientBillingNoOfHours:resource" />
          </Match>
          <Match MatchId="string-equal">
            <AttributeValue DataType="string">704</AttributeValue>
            <AttributeDesignator AttributeId="p_billingNoOfHours" DataType="string" />
            Category="PatientBillingNoOfHours:resource" />
          </Match>
        </AllOf>
      </AnyOf>
    </Target>
  </Rule>
</Policy>
<AttributeValue DataType="string">Select</AttributeValue>
Category="Select:action"/>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>

<Rule RuleId="EmergencyWardBilling" Effect="Permit">
<Target>
<AnyOf>
<AllOf>
<Match MatchId="string-equal">
<AttributeValue DataType="string">Administrator</AttributeValue>
<AttributeDesignator AttributeId="admin_position" DataType="string"
Category="Administrator:access-subject"/>
</Match>
</AllOf>
<Match MatchId="string-equal">
<AttributeValue DataType="string">Emergency</AttributeValue>
<AttributeDesignator AttributeId="Department_emer" DataType="string"
Category="Emergency:access-subject"/>
</Match>
</Match>
<Match MatchId="string-equal">
<AttributeValue DataType="string">1000</AttributeValue>
<AttributeDesignator AttributeId="labresult_fee" DataType="string"
Category="LabResultFees:resource"/>
</Match>
<Match MatchId="string-equal">
<AttributeValue DataType="string">500</AttributeValue>
<AttributeDesignator AttributeId="X-RayFee" DataType="string" Category="X-RayFees:resource"/>
</Match>
<Match MatchId="string-equal">
<AttributeValue DataType="string">100</AttributeValue>
<AttributeDesignator AttributeId="med_baseamt" DataType="string"
Category="MedicationsBaseAmt:resource"/>
</Match>
<Match MatchId="string-equal">
<AttributeValue DataType="string">0.3</AttributeValue>
<AttributeDesignator AttributeId="p_billingEmerperHourIncDec" DataType="string"
Category="PatientBillingEmerPerHourIncDec:resource"/>
</Match>
<Match MatchId="string-equal">
<AttributeValue DataType="string">48</AttributeValue>
</Match>
</Target>
</Rule>
<AttributeDesignator AttributeId="p_billingEmerNoOfHours" DataType="string" Category="PatientBillingEmerNoOfHours:resource"/>
</Match>
<Match MatchId="string-equal">
  <AttributeValue DataType="string">704</AttributeValue>
  <AttributeDesignator AttributeId="p_billingInfo" DataType="string" Category="PatientBillingInfo:resource"/>
</Match>
<Match MatchId="string-equal">
  <AttributeValue DataType="string">Select</AttributeValue>
  <AttributeDesignator AttributeId="select" DataType="string" Category="Select:action"/>
</Match>
</AllOf>
</AnyOf>
</Target>
</Rule>
</Policy>
</PolicySet>
References


http://searchsqlserver.techtarget.com/definition/XQuery


[14] Sending email from java application. [Online]. Available: 
http://www.developer.com/java/ent/adding-email-support-to-your-java-applications.html


https://code.ecoinformatics.org/code/wfscheduler/branches/WFSCHEDULER_2_3_0/docs/dev/index.html
[18] XQuery 1.0 and XPath 2.0 Functions and Operators. [Online]. Available:

https://www.w3.org/TR/xpath-functions/#func-hours-from-duration