TIMOTHY EDWIN SQUAIR

# SERVIÇO DE CONTROLE DE ACESSO UTILIZANDO MODELO DE PROVISIONAMENTO

**CURITIBA**

**2005**

TIMOTHY EDWIN SQUAIR

# SERVIÇO DE CONTROLE DE ACESSO UTILIZANDO MODELO DE PROVISIONAMENTO

Dissertação de Mestrado apresentado ao Programa de Pós-Graduação em Informática Aplicada da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Ciências.

Área de Concentração:

*Segurança de Sistemas*

Orientador:

Professor Edgard Jamhour, Dr.

**CURITIBA**

**2005**

Squair, Timothy Edwin
Serviço de Controle de Acesso Utilizando Modelo de Provisionamento. Curitiba, 2005. 213 p.

Dissertação – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática Aplicada

1- PBNM, 2- Provisionamento, 3- Controle de Acesso, 4- RBAC

I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática Aplicada

*À minha esposa, Tania e à minha filha Susan*
*pelo apoio e pela compreensão aos inúmeros*
*momentos dedicados à elaboração deste trabalho*

# Agradecimentos

Ao professor Edgard Jamhour, pelo companheirismo, apoio, confiança e, sobretudo, pela excelente orientação para a execução deste trabalho.

À Companhia de Informática do Paraná, CELEPAR, pelo apoio, para a viabilização deste trabalho.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviaturas e Siglas

| Abreviatura | Referência | Tradução |
|---|---|---|
| ACL | *Access Control List* | Lista de Controle de Acesso |
| API | *Application Program Interface* | Interface de Aplicação |
| CIM | *Commom Information Model* | Modelo Comum de Informações |
| CNF | *Conjuntive Normal Form* | Forma Conjuntiva Normal |
| COPS | *Commom Open Policy Service* | Serviço Comum e Aberto para Políticas |
| COPS-PR | *Commom Open Policy Service - Provisioning* | Serviço Comum e Aberto para Políticas – Modo Provisionamento |
| DAP | *Directory Access Protocol* | Protocolo de Acesso a Diretório |
| DEN | *Directory Enabled Networks* | Redes Baseadas em Diretórios |
| DMTF | *Distributed Management Task Force* | Grupo de Trabalho de Gerência Distribuída |
| DN | *Distinguished Name* | Identificação Única |
| DNF | *Disjuntive Normal Form* | Forma Disjuntiva Normal |
| IANA | *Internet Assignment Numbers Authority* | Autoridade Internet de designação de Números |
| IETF | *The Internet Engineering Task Force* | Grupo de Trabalho de Engenharia da Internet |
| IPSEC | *IP Security Protocol* | Protocolo de Segurança IP |

| Abreviatura | Referência | Tradução |
|---|---|---|
| ISO | *International Standards Organization* | Organização Internacional de Padronizações |
| ITU-T | *International Telecomunications Union – Telecomunications Standardization Sector* | União de Telecomunicações Internacionais – Setor de padronizações em Telecomunicações |
| LDAP | *Lightwheight Directory Access Protocol* | Protocolo de Acesso Leve a Diretórios |
| LDIF | *LDAP Data Interchange Format* | Formato de troca de Dados em diretórios LDAP |
| LPDP | *Local Policy Decision Point* | Ponto de Decisão de Políticas Locais |
| MDL | *Model Definition Language* | Linguagem de Definição de Modelo |
| MIB | *Management Information Base* | Base de Gerenciamento de Informações |
| PBN | *Policy Based Networking* | Redes Baseadas em Políticas |
| PCELS | *Policy Core Extensions Schema* | Esquema de representação da Extensão Central de Informação de Política |
| PCIM | *Policy Core Information Model* | Modelo Central de Informação de Política |
| PCIMe | *Policy Core Information Model Extension* | Modelo de Extensão Central de Informação de Política |
| PCLS | *Policy Core LDAP Schema* | Esquema de representação Central de Informação de Política |
| PDP | *Policy Decision Point* | Ponto de Decisão de Política |
| PEP | *Policy Enforcement Point* | Ponto de Aplicação de Política |

| Abreviatura | Referência | Tradução |
|---|---|---|
| PIB | *Policy Information Base* | Base de Informação de Políticas |
| QoS | *Quality of Service* | Qualidade de Serviço |
| QPIM | *Policy QoS Information Model* | Modelo de Informação de Políticas QoS |
| RBAC | *Role Based Access Control* | Controle de Acesso Baseado em Papéis |
| RDN | *Relative Distinguished Name* | Identificação Única Relativa |
| RFC | *Request for Comments* | Chamada para Comentários |
| RSVP | *Resource Reservation Protocol* | Protocolo de Reserva de Recursos |
| SAP | *Service Access Point* | Ponto de Acesso a Serviços |
| SNMP | *Simple Network Management Protocol* | Protocolo Simples para Gerência de Redes |
| SNMP | *Simple Network Management protocol* | Protocolo Simples de Gerenciamento de Rede |
| TLS | *Transport Layer Security* | Camada Segura de Transporte |
| UML | *Unified Modeling Language* | Linguagem de Modelagem Unificada |

# Resumo

Este trabalho tem como função apresentar um modelo capaz de representar, distribuir e aplicar políticas de controle de acesso aplicados a ambientes heterogêneos e distribuídos.

O modelo hora apresentado é baseado na estratégia de provisionamento definido pelo IETF. As políticas RBAC são descritas a partir de modelos baseados no PCIM e distribuídas através da abordagem PEP PDP utilizando o protocolo COPS-PR.

A PIB (*Policy Information Base*) RBAC é definida como forma de armazenar as informações relativas ao controle de acesso provisionadas ao PEP. Com o intuito de explorar a estratégia de distribuição de políticas definidas pelo IETF, cada PEP recebe somente o subconjunto de políticas RBAC a ele destinadas, podendo oferecer suporte às aplicações que representa.

Um conjunto de API's também é definido como forma de suportar a integração entre as aplicações e a PIB RBAC.

**Palavras-Chave**: 1- PBNM, 2- Provisionamento, 3- Controle de Acesso, 4- RBAC.

# Abstract

*This paper presents a framework for representing, distributing and enforcing access control policies in distributed heterogeneous systems.*

*The framework is based on the provisioning strategy definedby IETF. The RBAC-based policies are described using a PCIMbased model, and distributed by a PDP to PEPs using the COPS-PR protocol.*

*A RBAC PIB (Policy Information Base) was defined in order to store the provisioned access control policies in the PEP. By exploring the policy distribution strategy defined by IETF, each PEP receives only the sub-set of RBAC policies which are required to support the applications it represents.*

*An API is also defined in order to support the integration between applications and the RBAC PIB.*

***Keywords**: 1- PBNM, 2- Provisioning, 3- Access Control, 4- RBAC*

# Capítulo 1

# Introdução

## 1.1 Desafio

A grande utilização de recursos computacionais, associada à facilidade de acesso aos mesmos, impôs um crescimento substancial das redes de comunicação. Diferentes serviços e dispositivos foram, e continuam a ser, incorporados às estruturas com a finalidade de prover o necessário suporte às funções requisitadas pelos usuários.

Neste cenário, o controle de acesso a sistemas, aplicações e recursos ganha também elevada importância, pois deve garantir que as informações disponibilizadas pelas corporações sejam acessadas somente por aqueles que, efetivamente, devam fazê-lo.

Tradicionalmente, o controle de acesso tem sido feito diretamente sobre aplicações ou sistemas, seja através de matrizes ou listas de acesso. Os ambientes computacionais modernos, no entanto, por compreenderem uma complexa heterogeneidade, necessitam de um tratamento diferenciado na forma como o controle é feito, exigindo assim que haja compartilhamento de informações entre os diversos sistemas e ambientes com a finalidade de simplificar e unificar a administração.

O desafio nesta área é implementar um método de controle de acesso que atenda às necessidades dos ambientes atuais, seja ágil e capaz de consolidar e unificar as diversas características existentes, possibilitando assim, sua utilização de forma centralizada e compartilhada pelos diversos sistemas, aplicações e recursos das corporações.

## 1.2 Motivação

Segundo [Sandhu 94], métodos de controle de acesso têm como propósito limitar as ações ou operações que um usuário legítimo de um sistema de computação pode exercer. Contudo, nos atuais ambientes computacionais, devido às características de distribuição e diversidade, esse controle tem exigido, cada vez mais, grandes esforços para seu gerenciamento. Sendo assim, há a necessidade de se adequar esses métodos, ou ainda, criar outros novos capazes de prover a devida garantia de segurança exigida pelos ambientes.

Diversas novas tecnologias têm surgido no intuito de simplificar e organizar a administração dos ambientes computacionais heterogêneos e distribuídos. A forma utilizada por essas novas tecnologias para obter o resultado esperado consiste em estimular a utilização e agrupamento de características comuns aos diversos sistemas e ambientes. Padrões e modelos são propostos para reforçar essa idéia, uma vez que possibilitam o incremento das diversas informações comuns aos sistemas.

A possibilidade de utilização destas tecnologias abre um caminho para que métodos de controle de acesso sejam sugeridos e implementados de formas mais ágeis. Uma dessas abordagens, capazes de prover a necessária melhoria, é a utilização de PBN – *Policy Based Networking* –.

De acordo com [Shepard 00], PBN surgiu como um novo paradigma em relação à operação e gerência de redes de computadores, assegurando aos usuários a capacidade de obter qualidade de serviço (QoS), segurança e outros componentes necessários à boa utilização da rede. A forma de assegurar esta utilização racional da rede dá-se através de regras de negócio que efetuam o controle necessário, a fim de priorizar e parametrizar o acesso e utilização dos recursos disponíveis.

O crescente interesse na área fez com que o IETF – *Internet Engineering Task Force* – em conjunto com o DMTF – *Distributed Management Task Force* – estabelecesse um grupo de trabalho chamado *Policy Framework* com a finalidade de desenvolver um modelo de informação extensível, genérico, capaz de representar, gerenciar, compartilhar e reutilizar políticas, independente dos fabricantes, de forma a obter escalabilidade e interoperabilidade. [Flegkas 01].

A facilidade na criação, modificação e exclusão das políticas dinamicamente sem que essas ações interfiram em outras, de acordo com [Kanada 01], é uma outra grande

característica das *Policy-Based Networks* que, a partir da utilização de um sistema baseado em regras, essas políticas podem ser reaproveitadas, diminuindo assim a carga na administração e gerência de redes.

As políticas aqui referenciadas podem ser vistas como um conjunto de regras do tipo Condição-Ação que definem o comportamento de uma entidade, devendo ser determinadas de forma centralizada num modelo organizacional, seguindo um modelo de informação comum, e tendo em seu escopo, mecanismo e ações a serem tomadas bem estabelecidas, além de contar com seus próprios atributos [RFC 3198]. Políticas são, normalmente, criadas pelo administrador do sistema e, exatamente por esta razão, não são diretamente entendidas pelas aplicações ou equipamentos de rede sendo necessária, então, a tradução das mesmas em regras, que transformadas, podem ser aplicadas [Martin 99].

Um método de controle de acesso passível de ser estruturado, dentro da visão utilizada em PBN, e que pode utilizar os padrões sugeridos elo IETF e DMTF é o modelo RBAC – *Role Based Access Control* –.

RBAC [Ferraiolo 95] é um dos mais significantes esforços no sentido de definir um método de controle de acesso que atenda de forma completa as necessidades da descrição de políticas de segurança relacionadas às regras de negócio. Como seu nome sugere, o RBAC define políticas de controle de acesso usando o conceito de papéis, em vez de usuários ou grupos, geralmente empregados pelo modelo discricionário, ou níveis de segurança hierárquicos, usados pelo modelo mandatório. O papel é uma maneira bastante flexível de agrupar objetos que têm atributos em comum. Além disso, o RBAC permite expressar a sobreposição de responsabilidades através da hierarquia entre papéis. Esta característica permite, facilmente, descrever as funções executadas por indivíduos em uma organização e o relacionamento hierárquico entre elas.

Os padrões propostos pelo IETF e DMTF, no entanto, não suportam explicitamente o modelo RBAC, sendo necessária a extensão desses padrões para que o modelo possa acomodar objetos complexos capazes de descreverem políticas inerentes ao controle de acesso.

Neste sentido, [Nabhen 03] propõe uma abordagem para o problema de controle de acesso que utiliza o modelo RBAC proposto em [Sandhu 00] associado a extensões do modelo PCIM – *Policy Core Information Model* – [RFC 3060] proposto pelo IETF,

para representarem este tipo de política, em um repositório LDAP – *Lightweight Directory Access Protocol* .

O modelo PCIM define uma arquitetura para sua implementação, onde estão especificadas duas entidades centrais: o PDP – *Policy Decision Point* – responsável pelas decisões de políticas para si próprio ou para outras entidades que solicitam decisões [RFC 2753]; e o PEP – *Policy Enforcement Point* – responsável pela implementação das decisões de políticas apuradas pelo PDP [RFC 2753]. A associação destas duas entidades, bem como seu inter-relacionamento, é obtido através da utilização do protocolo COPS – *Commom Open Policy Service Protocol* – [RFC 2748]. Este protocolo é responsável pela troca de mensagens entre o PEP e o PDP.

O protocolo COPS dá suporte a dois modelos de controle de políticas:

- Modelo *Outsourcing* – No modelo *outsourcing*, qualquer evento endereçado ao PEP requer uma decisão de política instantânea (em tempo real). O PEP envia uma requisição ao PDP e aguarda a decisão de política [RFC 2749]. A cada requisição vinda do PEP é gerada pelo PDP uma decisão correspondente que deve ser obrigatoriamente aplicada pelo PEP.
- Modelo *Provisioning* – No modelo, também chamado configuração, todos os dados relevantes à implementação das políticas no PEP são transferidos no processo de inicialização, não sendo necessárias novas chamadas ao PDP, a não ser para processos de controle, ou em caso de mudança nas políticas referentes àquele cliente. Este modelo é definido através de RFC específica [RFC 3084] e é identificado pelo termo COPS-PR.

O modelo *provisioning*, COPS-PR, define ainda, para seu funcionamento, uma estrutura capaz de armazenar as diferentes informações de configuração recebidas do servidor de políticas identificando seu tipo e propósito. Chamada de PIB – *Policy Information Base* –, esta estrutura em árvore tem seus ramos definidos como classes de provisionamento (PRC) e folhas representando instâncias destas classes (PRI).

O LDAP é um serviço de diretório aberto [RFC 1777; RFC 2251], capaz de ser implementado em diversos sistemas operacionais e que, pela sua larga utilização, conta com API's – *Application Program Interface* – construídas e disponibilizadas em diversas linguagens de programação como C e Java.

Notadamente, em [Nabhen 03] o modelo utilizado pelo COPS foi o *outsourcing*. Desta forma, a cada necessidade de se efetuar o controle de acesso, a aplicação ou agente, faz uma chamada ao PEP (cliente) que encaminha uma mensagem de requisição de política ao PDP (servidor) que faz a consulta ao repositório. Obtidas as políticas aplicáveis, é criada pelo PDP uma mensagem de decisão que é encaminhada ao PEP para que seja aplicada à solicitação feita pela aplicação ou agente. Exatamente por esta característica, necessidade de troca de mensagens constantes entre PEP e PDP, a abordagem *outsourcing* pode apresentar potenciais problemas, principalmente, em relação a desempenho e escalabilidade. Outra desvantagem desta abordagem é que exige a centralização das políticas em um único host podendo também incidir sobre a performance do sistema.

## 1.3 Proposta

A proposta deste trabalho é apresentar uma abordagem para o controle de acesso em ambientes heterogêneos e distribuídos utilizando RBAC [Sandhu 00] associado aos modelos do IETF PCIM [RFC 3060] e PCIMe – Policy Core Information Model Extensions – [RFC 3460], que utilizem, como repositório para as políticas relacionadas, o LDAP [RFC 1777; RFC 2251], fazendo uso do protocolo COPS-PR para troca de mensagens entre as entidades PEP e PDP, segundo o modo provisionamento [RFC 3084].

Inicialmente, propõe-se a definição do *framework* necessário ao controle de acesso RBAC sob o modelo de provisionamento, no qual está inserido o desenvolvimento das estruturas do cliente PEP e do servidor PDP, capazes de prover o necessário relacionamento entre usuários, repositório, políticas e decisões.

A implementação dos algoritmos necessários ao modelo, seja para funcionamento do PEP ou do PDP, ou para os demais relacionados ao controle de acesso, montagem e adequação das políticas, estão no escopo de desenvolvimento deste trabalho como contribuição por ele gerado.

A constituição da PIB – *Policy Information Base* – elemento necessário ao modelo de provisionamento [RFC 3084] tem elevada importância no trabalho proposto. Para sua composição será utilizada linguagem XML – *Extended Markup Language* – a partir da proposição contida na [RFC 3318].

## 1.4 Organização

O trabalho aqui proposto será estruturado em sete capítulos organizados da seguinte forma:

- **Capítulo 1** – Introdução: Compreende uma apresentação geral do contexto referente ao trabalho
- **Capítulo 2** – Métodos de Controle de Acesso e Trabalhos Relacionados: Mostra um apanhado sobre trabalhos relacionados a controle de acesso, principalmente, em sistemas e ambientes distribuídos.
- **Capítulo 3** – Padrões IETF: Apresenta os diversos padrões sugeridos para a aplicação de métodos de gerenciamento baseados em políticas, contextualizando o protocolo de comunicação COPS-PR como mecanismo para a troca de informações relacionadas à abordagem provisionamento.
- **Capítulo 4** – Modelo de Informação e PIB: Apresenta o modelo proposto com suas características e abordagens ressaltando a característica do seu componente principal, a PIB.
- **Capítulo 5** – Arquitetura de Implementação, Algoritmos e API: Apresenta a arquitetura de implementação sugerida para o modelo, destacando o inter-relacionamento entre PEP e PDP, as mensagens COPS-PR trocadas e os diversos algoritmos e API's disponibilizados.
- **Capítulo 6** – Estudo de Caso e Avaliação da Proposta: Apresenta avaliação do modelo proposto a partir do estudo de caso, validando o mesmo através da implementação em serviço de diretórios e construção de protótipo para aferição do funcionamento.
- **Capítulo 7** – Conclusões e Trabalhos Futuros: Apresenta as conclusões do trabalho, bem como, sugestões de continuidade do mesmo.

# Capítulo 2

# Métodos de Controle de Acesso e Trabalhos Relacionados

## 2.1 Introdução

Os sistemas de controle de acesso têm como função definir quem pode acessar um determinado recurso e qual o grau de permissão envolvido, além de possibilitar a definição de quando o recurso estará disponível.

De forma geral, é importante notar que um sistema de controle de acesso não tem como finalidade ser uma solução completa em segurança para sistemas. Contudo, deve ser visualizado como mais um componente no complexo mecanismo da segurança de sistemas.

Neste capítulo serão apresentados alguns conceitos relacionados aos métodos de controle de acesso, e enfocado, de forma especial, o modelo RBAC – *Role Based Access Control,* o qual será utilizado como base para o desenvolvimento deste trabalho. Na seção Trabalhos Relacionados serão apresentadas algumas linguagens utilizadas para representação de políticas, como o Ponder e o XACML, além de trabalhos relacionados a controle de acesso. Ao final da seção será apresentado, de forma detalhada, o modelo RBPIM, proposto em [Nabhen 03a], que será utilizado na continuidade do trabalho.

## 2.2 Controle de Acesso

Os métodos de controle de acesso, como citado anteriormente, coexistem com diversos outros serviços num sistema de computação, como pode ser visualizado na figura a seguir.

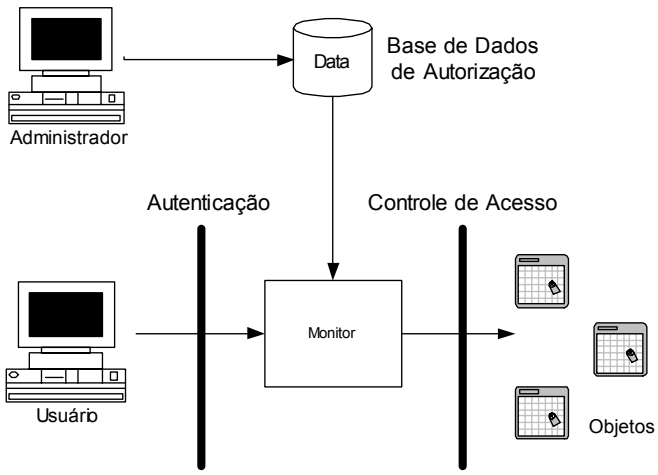


**Figura 2.1 – Sistema de Computação**

É importante, porém, fazer uma distinção entre controle de acesso e autenticação. O processo de autenticação é responsável por estabelecer a identificação correta de um usuário, cabendo ao controle de acesso avaliar se este pode acessar o objeto solicitado, limitando, desta forma, a atividade relacionada a um usuário legítimo do sistema. Esta ação é executada (Figura 2.1) pelo processo monitor, agente responsável por consultar a base de dados de autorização cada vez que um usuário tenta fazer um acesso a um objeto relacionado ao sistema.

De acordo com [Sandhu 94], os sistemas computacionais podem ser analisados, no que se refere a controle de acesso, sob o ponto de vista de três modelos:

- Modelo Discricionário
- Modelo Mandatório
- Modelo Baseado em Papéis

De forma geral, os modelos de controle de acesso não são, necessariamente, aplicados de forma exclusiva e única. Muitos sistemas, na busca por uma proteção mais efetiva, utilizam esses modelos de maneira associada.

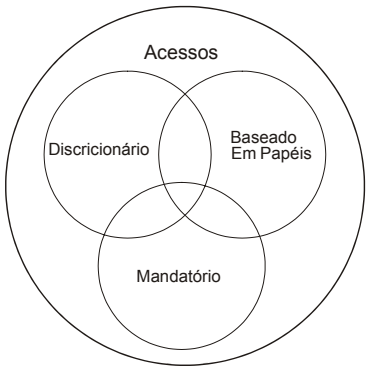


**Figura 2.2 – Métodos de Controle de Acesso**

### 2.2.1 Controle de Acesso Discricionário

Modelo proposto em 1971, tem sua utilização em diversos sistemas operacionais comerciais. Este sistema baseia-se na identidade do usuário, sendo que o acesso é controlado em cada objeto, definindo o modo de acesso de acordo com o usuário, ou grupo de usuários, e o nível de restrição especificado.

Uma falha apresentada por este modelo é a de não garantir o controle de acesso ao fluxo da informação, sendo possível contornar as restrições de acesso aplicadas. Exemplo desta falha, é que um usuário que tenha acesso de leitura a um determinado arquivo pode repassá-lo a outra pessoa sem que o dono do arquivo tome conhecimento.

Como forma de correção ao problema apontado, há ainda, a possibilidade de se negar, explicitamente, o acesso a objetos. Utilizando-se autorizações positivas e negativas pode-se deixar o modelo mais confiável, contudo, mais complicado.

Como exemplo estão os sistemas Unix atuais onde é possível aplicar a um arquivo permissões ao usuário que o criou (*owner*), a um grupo (*group*) de usuários e a outros usuários (*other*).

### 2.2.2 Controle de Acesso Mandatório

Modelo proposto também em 1971, nasceu em meio à comunidade militar e baseia-se na classificação de acordo com a hierarquia, através de *subjects* e *objects* do

sistema. Para cada objeto e usuário do sistema é associado um nível de segurança. A permissão de acesso ao objeto é definida pelo nível de restrição do usuário. Normalmente os níveis de segurança adotados são: *Top Secret* (TS), *Secret* (S), *Confidential* (C) e *Unclassified* (U), onde TS > S > C > U. Cada nível de segurança domina a si próprio e aos níveis inferiores na hierarquia.

Exemplo: Se um objeto possui nível de segurança C e um usuário que possui nível U tenta fazer uso dele, tem seu acesso negado. De outra forma se um usuário que possui nível de segurança TS tenta acessar o mesmo objeto que possui nível de segurança C, tem seu acesso permitido. Este sistema reflete o nível de confiabilidade do sistema, que expõe a informação somente ao usuário que possui autorização para tal.

### 2.2.3 Controle de Acesso Baseado em Papéis – RBAC

Neste modelo, as políticas de controle de acesso são baseadas nas atividades que os usuários desempenham em um determinado sistema. Desta forma, é necessário que cada papel seja definido dentro do escopo de atuação do sistema analisado. Um papel reflete um conjunto de ações ou responsabilidades associadas a uma atividade de trabalho. A utilização de papéis torna o sistema mais simples, pois, ao invés de se tratar cada usuário de forma individualizada, podem-se associar as atividades desenvolvidas a papéis e, então, relacionar usuários aos papéis que o mesmo pode assumir em relação ao modelo estabelecido.

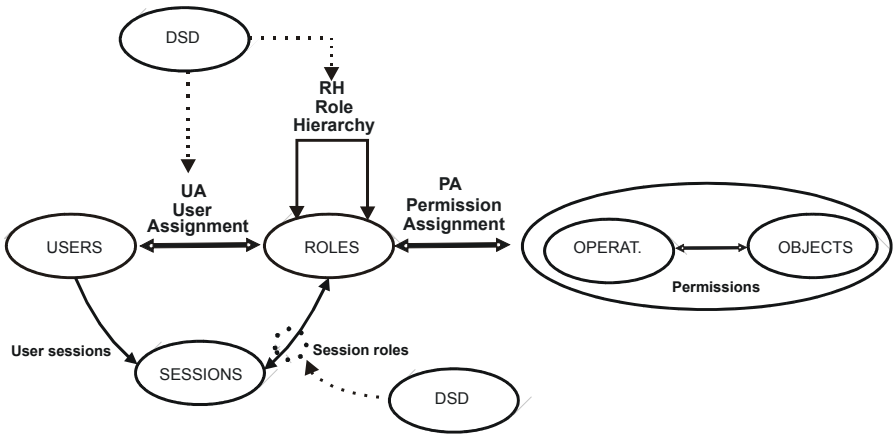


**Figura 2.3 – Modelo RBAC**

O modelo de implementação RBAC [Ferraiolo 01] adotado para desenvolvimento deste trabalho, segue a padronização ANSI INCITS 359-2004 e é baseado em quatro componentes: Core RBAC, *Hierarchical* RBAC, *Static Separation of Duty Relations* e *Dynamic Separation of Duty Relations*. Esta separação em componentes tem como objetivo flexibilizar a implementação do modelo.

**Core RBAC**

O Core RBAC é composto por cinco elementos básicos: usuários (USERS), papéis (ROLES), objetos (OBS), operações (OPS) e permissões (PRMS). A idéia do modelo RBAC é que permissões são associadas a papéis e esses associados aos usuários. A associação definida entre usuários e papéis é de muitos para muitos. Isto é, um usuário pode ser associado a diversos papéis, assim como um papel pode ser associado a diversos usuários. Um papel, para se tornar ativo, deve ser selecionado pelo usuário e ativado sob um contexto de sessão. Isto quer dizer que para um usuário fazer uso de um papel, é necessário que o mesmo selecione o papel desejado e ative-o na sessão, para que assim possa ter acesso às permissões associadas no sistema. Uma sessão é associada a um único usuário e cada usuário pode possuir uma ou mais sessões.

A permissão é definida como uma autorização dada para realizar uma operação determinada (ler, escrever, acessar, gravar) sobre um objeto controlado pelo RBAC, num relacionamento muitos para muitos. Isto é, uma permissão pode ser associada a um ou mais papéis, assim como um papel pode ser associado a uma ou mais permissões.

***Hierarchical* RBAC**

O *Hierarchical* RBAC tem como função adequar o modelo à utilização de hierarquias de papéis como forma de simplificação do processo de gerência. A herança de papéis é extremamente útil, pois permite a composição de papéis e permissões relacionados a um usuário. Desta forma, um usuário que tem relacionado a si um papel A e esse papel herda um papel B, então esse usuário passa também a ter relacionado a si o papel B.

**SSD -** ***Static Separation of Duty***

A separação estática de tarefas representa uma forma forte de estabelecer conflitos entre atividades desempenhadas no sistema. A idéia da utilização de SSD é impossibilitar que um usuário possa ter associado a ele papéis que representem conflito de interesses entre tarefas desempenhadas sobre o sistema. Assim sendo, se um determinado usuário for associado a um papel A e este mesmo papel possuir um relacionamento de SSD com o papel B então, o usuário não poderá ter em sua lista de papéis, simultaneamente, os papéis A e B. Este relacionamento é definido através de dois argumentos, o primeiro indicando um conjunto de papéis, que deve ser formado por dois ou mais componentes, e o segundo informando a cardinalidade do conjunto, ou seja, o número máximo de componentes pertencentes a ele. Para verificação de conflito por SSD, o número de papéis envolvidos no conjunto deve ser sempre menor que a cardinalidade. Assim sendo, para um conjunto SSD composto por dois elementos (papel A e B) sob uma cardinalidade 2, conclui-se que um usuário pode ter associado a ele somente um dos papéis para que não ocorra uma violação por SSD ($1 < 2$).

**DSD –** ***Dynamic Separation of Duty***

A separação dinâmica de tarefas tem como propósito limitar a ocorrência de conflitos na utilização de papéis sob o contexto dinâmico da ativação da sessão. De forma mais fraca que SSD, porém com as mesmas características de formação (contando com o conjunto de papéis envolvidos e a cardinalidade relativa à associação), a DSD tem como função não permitir que, numa mesma sessão, um usuário possa assumir papéis excludentes entre si. Desta forma, mesmo que o usuário possua um conjunto de papéis em sua lista de elegíveis, caso ocorra um conflito por DSD, o mesmo não poderá assumi-los de forma simultânea.

## 2.3 Trabalhos Relacionados

Diversos trabalhos têm sido apresentados na área de gerenciamento baseado em políticas. Sejam relacionados a políticas de segurança, como o modelo RBAC apresentado na sessão anterior, sejam relacionadas a políticas de gerenciamento, ou ainda, relacionados a linguagens capazes de representar essas políticas.

### 2.3.1 Linguagens para representação de políticas

Diversas linguagens têm sido desenvolvidas com a função de representar políticas. Serão apresentadas, a seguir, algumas destas linguagens e suas principais características.

***Role definition Language***

A linguagem RDL (*Role Definition Language*) [Hayton 98] é baseada num conjunto de regras que indicam as condições sob as quais um cliente pode obter um nome ou um papel, onde papel corresponde a um grupo nomeado. As condições associadas a um papel são descritas em forma de credenciais que possibilitem a um cliente definir um papel associado aos parâmetros da sua credencial. A linguagem RDL pode ser caracterizada, por definição, como um controle de acesso baseado em certificados.

Como exemplo, a seguir é apresentada uma regra de autorização RDL que estabelece direito a um cliente destinado a um papel (*SeniorHaematologist*) de invocar o método *append* possuído através do certificado chamado LoggedOn (km, s).

*append (haematology –field,y,x) ← SeniorSaematologist(x) ^ login.LoggedOn(km, s):s in TrustedServers*

Os papéis também são considerados como credenciais e podem ser utilizados para associar clientes a outros papéis como apresentado a seguir:

*SeniorHaematologist (x) ← Haematologist(x) ^ SeniorDoctor(x)*

É possível, ainda, definir delegações de papéis como forma de possibilitar o direito de acesso.

*Examiner (p, e) login.LoggedOn(p, s) ◄ ChiefExaminer : p in Staff*

***Security policy Language***

A SPL é uma linguagem de política dirigida a eventos que suporta políticas de controle de acesso, de histórico e de obrigações [Ribeiro 01]. É implementada como um monitor de eventos, onde, para cada evento gerado é decidido se permite, não permite ou ignora. Eventos, de acordo com SPL, são ligados a chamadas de ação (procedimentos) sobre objetos e podem ser verificados para determinar o assunto, alvo e o próprio evento. SPL suporta dois tipos de conjuntos para agrupar os objetos onde a política se aplica. Grupos definidos por seus elementos e categorias que são conjuntos definidos pela classificação das entidades de acordo com suas propriedades. Os blocos de política em SPL são regras compostas utilizando álgebra tri valorada a partir de três operadores lógicos: *and*, *or* e *not.* Uma regra simples é composta por duas expressões lógicas. Uma que define o domínio sobre o qual é aplicável e outra que decide a aceitabilidade relacionada ao evento. Como exemplo é apresentada, a seguir, uma regra simples e sua composição.

```
// Qualquer evento relacionado a um objeto criado pelo usuário é permitido
OwnerRule: ce.target.owner = ce.author :: true;

// Uma ordem de pagamento não pode ser autorizada por quem a solicitou
DutySep: ce.target.type = "paymentOrder" & ce.action.name = "approve" :: ce.author
!= ce.target.owner;

//Regra implicita de negação
deny: true :: false;

//Conjunção de regras simples com negação padrão
OwnerRule AND DutySep OR Deny;

// Separação tem maior prioridade que regra pessoal
DutySep OR (DutySep AND OwnerRule);
```

A linguagem SPL define, ainda, dois conjuntos abstratos chamados eventos passados (*PastEvents*) e futuros (*FutureEvents*) para especificação de políticas baseadas em histórico e uma forma restrita de políticas ligadas à obrigação. O tipo de obrigação suportada pela SPL é na forma condicional que é acionada por um evento pré-condicional.

Para passado

*Principal_o* ***must do*** *Action_o* ***if*** *Principal_T* ***has done*** *Action_T*

Para futuro

*Principal_o* ***cannot do*** *Action_o* ***if*** *Principal_T* ***will not do*** *Action_T*

***Ponder***

*Ponder* [Damianou 01], constitui-se como uma linguagem declarativa orientada a objetos, capaz de representar políticas de segurança e ou de gerência. As políticas de autorização baseadas em *Ponder*, de acordo com [Corradi 00], podem ser implementadas utilizando diversos mecanismos de controle de acesso para *firewall*, banco de dados, sistemas operacionais e Java. Suporta, ainda, políticas relacionadas a obrigações, que são eventos acionados por regras, compostas por condições e ações, aplicadas sobre políticas de gerenciamento de redes e sistemas distribuídos. Pode ser utilizado também para gerenciamento de atividades de segurança, tais como registro de usuários, registro de atividades, auditoria, a serem utilizados para determinar o acesso a recursos críticos ou violações de segurança. O conceito principal da linguagem inclui domínio, que tem como função agrupar objetos sobre os quais a política se aplica, papéis, que tem como função agrupar políticas relativas à posição na organização, relacionamento, que tem como função definir as interações entre os papéis, e as estruturas gerenciadas para definir a configuração dos papéis e os relacionamentos pertinentes a uma unidade organizacional.

Os domínios, apresentados no *Ponder*, definem uma forma de agrupar objetos sobre os quais a política se aplica e podem ser utilizados para dividir os objetos de forma a possibilitar sua organização, por critérios geográficos, tipo de objeto, responsabilidade, etc. Um domínio pode conter outros subdomínios como forma de possibilitar a organização da informação. Contudo, um subdomínio não é um subconjunto do domínio pai.

As políticas de autorização definem quais atividades podem ser executadas por um membro do domínio sobre um conjunto de objetos. Para permitir essa avaliação são definidas duas possibilidades: Uma autorização de política positiva define que a ação requisitada sobre o objeto é permitida e uma autorização de política negativa define que a ação requisitada foi negada. A linguagem permite, ainda, reutilização de políticas.
Um exemplo de política de autorização *Ponder* será apresentado a seguir:

```
type auth+ PolicyOpsT (subject s, target <policy> t){
        action load(), remove(), enable(), disable() ; }
inst auth+ switchPolicyOps = PoliycyOpsT(/NetworkAdmins, Nregion/switches);
inst auth+ routerPolicyOps = PoliycyOpsT(/QoSAdmins, Nregion/routers);
```

Neste código, as duas instâncias de políticas criadas a partir de *PolicyOpsT* permitem aos membros de /*NetworkAdmins* e /*QoSAdmins* carregar (*load*()), remover (*remove*()) habilitar (*enable*()) e desabilitar (*disable*()) objetos do tipo *PolicyT* sob os domínios /*Nregion/switches* e /*Nregion/routers* , respectivamente.

A linguagem Ponder permite que as políticas sejam compostas. Isto possibilita que sejam agrupadas de acordo com a finalidade, facilitando seu gerenciamento. O conceito de meta-políticas associado à linguagem permite, ainda, que sejam definidas combinações de políticas ampliando ainda mais as possibilidades de utilização e reaproveitamento.

### 2.3.2 PACF – *Policy Based Access Control Framework for Large Networks*

[Haixin 00] apresenta um modelo para gerenciamento e controle aplicados sobre *firewalls,* como forma de possibilitar o controle de acesso baseado em políticas de alto nível relativas a elementos de rede. A idéia principal do trabalho é possibilitar a gerência e configuração de diversos elementos de rede, de forma automática, a partir de políticas.

Para chegar ao objetivo são propostos três níveis de políticas abstratas de controle de acesso: OACP – *Organization Access Control Policy*, GACP – *Global Access Control Policy* e LACP – *Local Access Control Policy*.

1. OACP – *Organization Access Control Policy* – É composto por um conjunto de comandos relacionados a políticas de alto nível que definem quais recursos devem ser protegidos e, ainda, quais atividades devem ser proibidas no escopo da organização.
2. GACP – *Global Access Control Policy* – É responsável por corrigir o módulo OACP, podendo ser implementado de forma automatizada por sistemas de dispositivos de rede, sob o escopo dos domínios administrativos. Sob o ponto de vista de uma rede de trânsito, é composto por um conjunto de regras globais de filtragem que são aplicadas por um conjunto de *firewalls*. Uma regra de política (*policy*) é definida como uma tupla composta por três elementos: Origem, Destino e Ação.
    - Origem – É o conjunto de endereços que deve corresponder ao pacote de origem expresso na forma Endereço IP / Máscara.
    - Destino – Corresponde ao conjunto de endereços de destino do pacote, expresso na forma Endereço IP / Máscara.

- Ação – Determina o que fazer quando um pacote possua critérios idênticos aos definidos na Origem e Destino. Normalmente a Ação é relacionada ao conjunto {permitir, negar}.

3. LACP – *Local Access Control Policy* – É um subconjunto de GACP que é aplicado a cada interface ou dispositivo de forma individual. A aplicação do LACP é feita através de dois conjuntos distintos de regras: Regras de Entrada (*inboundsRS*), responsáveis por filtrar pacotes recebidos e Regras de Saída (*outboundsRS*), responsável por filtrar pacotes enviados.

Uma característica do PACF, é a possibilidade de criar, aplicar e ajustar as diversas políticas dinamicamente. Depois de iniciado, o conjunto global de filtros é dinamicamente adequado de acordo com os resultados obtidos através dos mecanismos de busca e de detecção de intrusão (IDS) efetuados sobre a rede. Também é possível, através dos mecanismos de detecção de intrusão, fazer a verificação das políticas de acesso e prover de informações o mecanismo de gerenciamento para que esse reconfigure o *firewall* ou a topologia da rede.

O conjunto de regras, aplicado a filtros, e as demais configurações dos diversos *firewalls*, são armazenados num repositório LDAP. Como mecanismo de configuração é utilizado o CLI – *Command Line Interface* a partir de sessões *telnet* ou *ssh*.

Esta proposição utiliza, da mesma forma que o desenvolvimento deste trabalho, políticas de alto nível armazenadas em serviço de diretório. Contudo, o foco aqui apresentado, refere-se muito mais ao gerenciamento de redes do que ao controle de acesso propriamente dito.

### 2.3.3 GRBAC – Generalized Role-Based Access Control

[Moyer 01] propõe uma extensão do modelo RBAC, incorporando novos tipos de papéis, classificados sob três abordagens: papéis relacionados a assuntos (*subjects*), a objetos, ou ambientes e, ainda, adotando um novo modelo de transação.

Os papéis relacionados a assuntos correspondem aos tradicionais papéis RBAC. Os relacionados a objetos têm como característica abstrair diversas propriedades referentes aos objetos (texto, *jpeg*, executáveis, etc) tais como tipo, data de criação, tamanho, etc, ou níveis de sensibilidade, que são agrupados em categorias. Papéis referentes a ambientes capturam informações correlatas ao ambiente como um todo, tais como dia, hora do dia, carga do sistema, etc, que podem ser utilizadas como forma intermediária do controle de acesso. O modelo de transação especifica ações particulares que devem ser aplicadas ao sistema.

O modelo de transação apresentado no GRBAC é similar ao modelo RBAC e consiste de uma tupla sob a forma <SRole, ORole, ERole, op> onde SRole especifica um papel relacionado a assunto (*subject*), ORole especifica um papel relacionado a objeto, ERole especifica um papel relacionado ao ambiente e op corresponde à operação (*read*, *write*, execute) que deve ser realizada.

O modelo GRBAC utiliza uma base de políticas compostas de uma lista de transações associada a um bit de permissão, que define se a transação é proibida ou permitida. Essa associação é feita para cada transação, sendo que cada par <*transaction*, *permission bit*> é chamado regra de política.

### 2.3.4 PARBAC – Privacy Enforcement with an Extended RBAC Model

[He 03] propõe combinar o RBAC com a aplicação de políticas de domínio relacionadas à proteção da privacidade dos dados, de acordo com finalidades de negócio. Com o consentimento do cliente, suas preferências são armazenadas como políticas que determinam como utilizá-las. O uso da informação deve ser consentido por seu dono, o que é definido como propósito associado à necessidade de privacidade. O autor define privacidade como um conceito social, moral ou legal que garante ao usuário que suas informações não serão divulgadas sem seu expresso consentimento.

Como forma de prover a capacidade desejada, são propostos dois novos elementos associados ao modelo RBAC tradicional.

- O propósito (*purpose*), define a razão pela qual uma operação é executada. Esta forma de visualização serve para diferenciar propósitos associados a negócios e propósitos especificamente associados a políticas de privacidade consentidas pelos usuários.
- Objetos de política (*object-policy*), que corresponde à utilização dos dados. Cada informação de usuário deve ter sua utilização definida por uma política específica.

Um importante módulo do sistema é o de gerência das preferências sobre informações dos clientes. Este módulo é responsável por coletar as informações e classificá-las de acordo com o consentimento e desejo dos clientes, em relação à política de privacidade que será aplicada. O módulo permite que qualquer informação ou política armazenadas seja alterada. Outra função deste módulo é ligar as preferências aos dados associados.

No modelo, para que as informações de políticas associadas à privacidade sejam acessadas, é necessário que um conjunto de condições sejam satisfeitas, do mesmo modo, que algumas destas políticas necessitam que operações adicionais sejam efetuadas quando o acesso é permitido. Essas operações adicionais são referenciadas como ações obrigatórias que o sistema deve executar para que o acesso seja permitido.

Assim como o desenvolvimento deste trabalho, as abordagens descritas nos itens 2.3.3 e 2.3.4 utilizam-se do modelo RBAC como método de controle de acesso, demonstrando a versatilidade e possibilidades associadas ao mesmo. Desta forma, pode-se notar que a adequação do modelo RBAC é possível e está sendo utilizada como forma de melhoria do método de controle de acesso.

### 2.3.5 XACML – *Extensible Access Control Markup Language*

XACML [Oasis 03] é uma especificação XML para expressar políticas sobre informações de acesso aplicadas sobre a Internet. Similar a outras linguagens relacionadas a políticas, o XACML é utilizado para especificar uma política, nos moldes assunto-alvo-ação-condição sob o contexto de um documento em particular.

[Toktar 04] propõe um modelo a partir do XACML que apresenta diferenças significativas em relação ao modelo de políticas PCIM sugerido pelo IETF. Enquanto o PCIM é a base para a criação de políticas, sob qualquer área de gerenciamento, o XACML tem seu escopo direcionado para o controle de acesso.

O modelo XACML, que utiliza a mesma estratégia do PBN, permite a aplicação de políticas de controle, sobre diversos dispositivos da rede, sem uma infra-estrutura de gerência de grande complexidade. A estrutura consiste de um servidor de políticas denominado PDP *(Policy Decision Point*) e diversos clientes de políticas denominados PEP (*Policy Enforcement Point*) [RFC 3198]. Ao PDP cabe a tarefa de armazenar e distribuir as políticas aos diversos nós da rede. Ao PEP cabe a recepção e aplicação das políticas distribuídas pelo PDP.

No modelo PCIM, para a implementação é necessário definir qual a forma a ser utilizada, se *outsourcing* ou *provisioning*. No modelo XACML esta distinção não existe, assim como não é definido um protocolo específico de comunicação entre o PEP e o PDP.

Outra grande diferença entre as abordagens adotadas pelo IETF e pelo OASIS (Organization *for the Advance of Structured Information's Standards)* é em relação à forma da concepção e armazenamento dos modelos de políticas. O OASIS propõe o XACML como um modelo particular de controle de acesso representado e fornecido na forma de documentos XML, enquanto que o IETF propõe uma forma genérica de representação de políticas independente da sua aplicação.

A linguagem XACML é definida através de dois esquemas XML:

- XACML *Context* – Modelo para representação de mensagens, baseados em perguntas e respostas, trocadas pelo PEP e PDP.
- XACML *Policy* – Utilizado para a descrição das políticas

De acordo com a estratégia adotada pelo OASIS, o PEP envia requisições de acesso ao PDP e recebe regras sob um formato definido. Um componente de software

denominado *context handler* é responsável por transformar as perguntas e respostas em um formato particular padronizado pelo *XACML Context*.

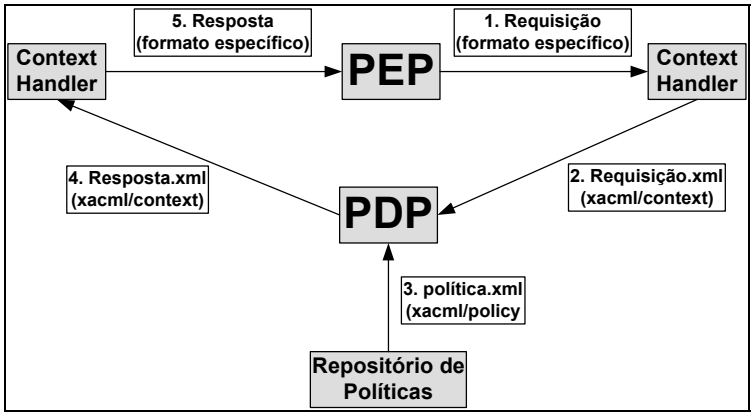


**Figura 2.4 – Elementos Arquitetura XACML**

As políticas são definidas por um conjunto de permissões de acesso através de combinações denominadas *Targets*. Este componente é expresso através da sintaxe "usuários podem realizar ações sobre recursos". Uma política e suas permissões podem ser aplicadas através de um conjunto de regras, onde uma regra pode ser expressa pela sintaxe "Se condição satisfeita então aplique a ação sobre o recurso".

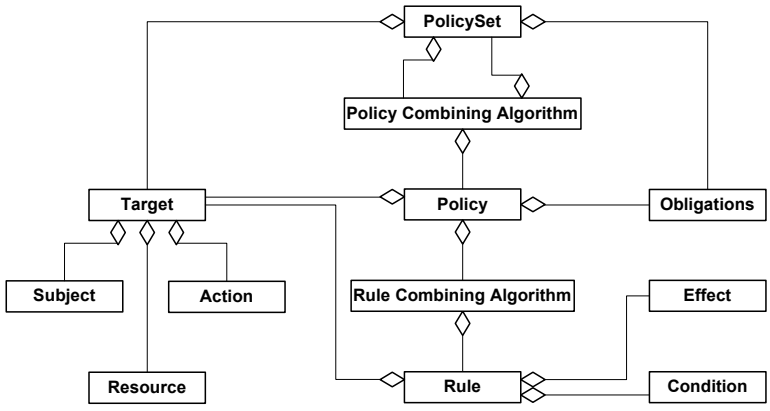


**Figura 2.5 – Linguagem de Política XACML**

Quando um PEP efetua uma requisição ao PDP, fornece os atributos que permitem ao PDP identificar os elementos necessários. O PDP, então, verifica as regras de políticas e determina a existência de *Targets* para os atributos fornecidos e envia de volta o efeito correspondente (negar ou autorizar). Caso falhe a busca por um *Target*

correspondente é retornado o parâmetro "*NotApplicable*", indicando que não há correspondência entre os dados fornecidos e as políticas armazenadas.

As estratégias utilizadas para acessar um conjunto de regras associadas a uma mesma política são definidas no modelo como:

- *Deny-overrides* – Se uma disposição definida como "negar" é satisfeita, então a decisão da política é negar, independente de outras regras
- *Permit-overrides* – Se uma disposição definida como "permitir" é satisfeita, então a decisão da política é permitir, independente das demais regras
- *First-applicable* – Define que a primeira regra satisfeita no conjunto é aplicada.
- *Only-one-applicable* – Determina que somente uma política dentre o conjunto deverá ser aplicada. Porém se mais de uma política puder ser aplicada, então será retornado o parâmetro "*Indeterminate*", determinando que não foi possível aplicar uma regra específica no conjunto.

A principal limitação do modelo de controle de acesso do OASIS refere-se à incapacidade de suporte a métodos de provisionamento, o que é necessário a muitas áreas que utilizam métodos de controle de acesso. Outro grande inconveniente, é que não há um protocolo definido para a troca de informações entre as entidades envolvidas, nem uma definição acerca do local de armazenamento das políticas. Por outro lado, o modelo permite a utilização do RBAC no controle de acesso.

A aplicação de políticas de alto nível associada ao método de controle RBAC além da utilização das entidades PEP e PDP, permite associar a abordagem proposta no modelo do OASIS com o desenvolvimento do trabalho aqui apresentado, sendo possível, assim, mostrar que, apesar de utilizar componente e estruturas similares, as abordagens podem ser completamente diferentes na sua apresentação.

### 2.3.6 RBPIM – Role Based Policy Information Model

[Nabhen 03a] propõe um *framework* completo para a implementação de políticas de controle de acesso, que utiliza como base sistemas padronizados como RBAC, PCIM e COPS aplicado a ambientes heterogêneos e distribuídos.

Fazendo uso de um modelo RBAC flexível, associado a uma implementação *outsourcing* definida no protocolo COPS, aplicada sobre o modelo PCIM, permite especificar o relacionamento entre usuários, papéis, permissões e objetos através de combinações de expressões *booleanas*.

**Modelo PCIM**

O modelo de informação PCIM – *Policy Core Information Model*, definido pelo IETF a partir da extensão do modelo CIM[1], permite a representação de políticas de rede de forma padronizada, independente de fabricante. Para sua implementação, aplicações específicas podem estender o modelo de informação, a fim de representar as políticas necessárias [RFC 3060].

Tipicamente, o PCIM é implementado utilizando componentes PDP e PEP [RFC 2753]. O PDP é o servidor de políticas responsável por repassar as informações de política para os dispositivos de rede e aplicações associadas. O PEP é o cliente, responsável por, efetivamente, aplicar as políticas recebidas do servidor sobre um dispositivo ou aplicação. A comunicação entre o servidor (PDP) e o cliente (PEP) é efetuada através do protocolo COPS [RFC 2748].

O modelo PCIM define duas hierarquias de classes de objetos: Classes estruturais que representam informações e controle de políticas e Classes de associação, que indicam quais instâncias das classes estruturais estão relacionadas. As classes estruturais e associações definidas pelo PCIM são suficientemente genéricas para permitir a representação de políticas relacionadas a qualquer assunto.

A partir do modelo (figura 2.6) pode-se visualizar como são compostas as políticas: A classe *PolicyGroup* tem como função agregar, tanto outros grupos (*PolicyGroup*) como também regras (*PolicyRule*). A classe *PolicyRule* representa a

[1] CIM – Common Information Model é um modelo de informação proposto pelo DMTF – Distributed Management Task Force, que tem como intuito representar recursos de rede.

semântica “se a Condição for satisfeita então a Ação especificada é executada”. É importante notar que um dos critérios de validação da regra, pode estar associado a um período de tempo, a classe *PolicyTimePeriodCondition*. Outro ponto importante são os atributos desta classe, que permitem definir a qual grupo a mesma pertence (*GroupNumber*), se a regra está ativa (*Enabled*), qual seu nível de prioridade (*Priority*), se sua aplicação é exigida (*Mandatory*) e qual a seqüência de ações desejada (*SequencedAction*). A classe *PolicyCondition* representa uma condição, ou um conjunto de condições, a serem avaliadas na composição da regra. A classe *PolicyAction* representa a ação, ou ações, a serem executadas quando as condições forem satisfeitas.

O PCIM também permite a associação de papéis a cada regra de política. Para tanto, especifica uma propriedade para objetos da classe *PolicyRule* chamada de *PolicyRoles*. Esta propriedade possibilita o armazenamento de um conjunto de *strings* que representam os papéis associados a cada regra. Neste esquema, podem ser associados um ou mais papéis a cada recurso da rede e, então, as regras de políticas podem ser associadas a estes papéis. Neste caso, o *framework* de política pode ficar responsável pela configuração de recursos da rede associados a papéis, de tal forma que o recurso passe a ter seu comportamento controlado de acordo com as políticas especificadas para aquele papel.

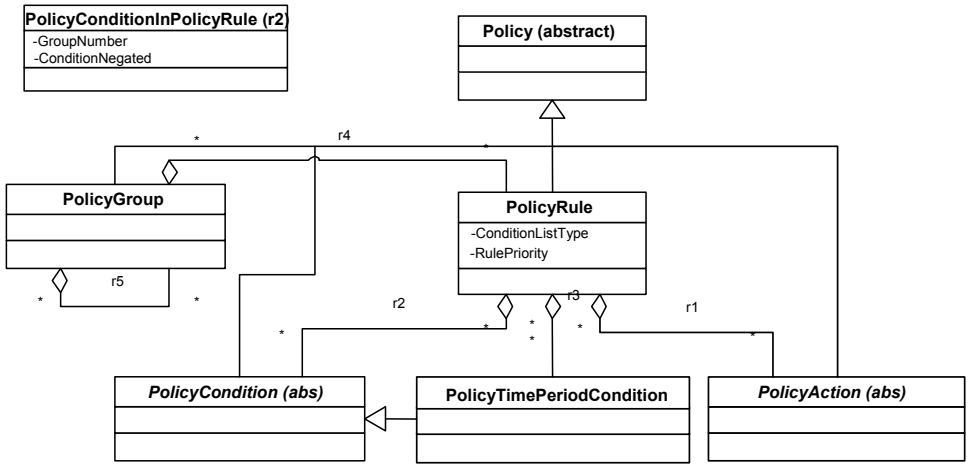


**Figura 2.6 – Modelo PCIM**

Na classe *PolicyRule*, as condições podem ser agrupadas de duas formas: forma conjuntiva (CNF) e forma disjuntiva (DNF) definidas pelo atributo *ConditionListType*. A definição deste atributo permite a composição dos agrupamentos. Na forma CNF, as condições internas a um grupo são associadas por meio do conectivo "OU" (OR), enquanto que a associação entre grupos é feita por meio do conectivo "E" (AND). Na forma DNF as condições internas aos grupos são associadas por meio do conectivo "E" (AND) enquanto que a associação entre grupos é feita por meio do conectivo "OU" (OR). Para exemplificar esta abordagem serão apresentadas, a seguir, cinco condições denominadas C1, C2, C3, C4 e C5 sob o ponto de vista da modelagem PCIM.

- C1: *GroupNumber* = 1; *ConditionNegated* = *False*;
- C2: *GroupNumber* = 1; *ConditionNegated* = *True*;
- C3: *GroupNumber* = 1; *ConditionNegated* = *False*;
- C4: *GroupNumber* = 2; *ConditionNegated* = *False*;
- C5: *GroupNumber* = 2; *ConditionNegated* = *False*;

Na forma disjuntiva (DNF) a expressão lógica gerada pelas condições seria:

DNF = (C1 AND (NOT (C2)) AND C3) OR (C4 AND C5)

Na forma conjuntiva (CNF):

CNF = (C1 OR (NOT (C2)) OR C3) AND (C4 OR C5)

**RBPIM – Modelo de Informação**

O modelo de informação adotado no RBPIM é constituído de uma extensão do modelo PCIM especialmente planejado para representar políticas RBAC. As novas classes introduzidas no modelo são *RBACPermissions* e *RBACRole*, a partir da especialização de *PolicyRule*, DSDRBAC e SSDRBAC, especializações de Policy, *AssignerPermission* e *AssignerOperation*, especializações de *PolicyAction*. A classe *RBACPolicyGroup* é utilizada para agrupar as informações do modelo RBAC.

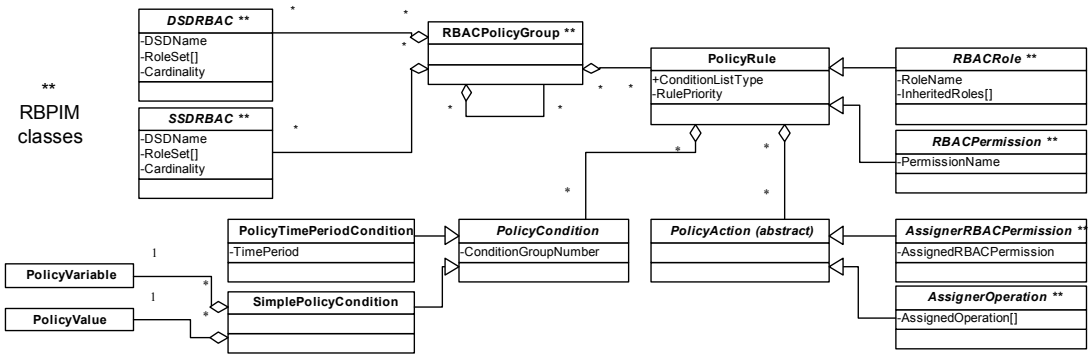


**Figura 2.7 – Modelo de Informação RBPIM**

Para a representação do modelo RBAC, o RBPIM utiliza duas especializações da classe *PolicyRules*: A *RBACRoles*, que representa os papéis RBAC, pode ser associada à lista de instâncias *SimplePolicyCondition*, *AssignerRBACPermission* e *PolicyTimePeriodCondition*, e a *RBACPermission*, que representa as permissões associadas a papéis, pode ser associada à lista de instâncias *SimplePolicyCondition* e *AssignerOperations*. A relação entre usuários e papéis (UA) é obtida através de instâncias *SimplePolicyCondition*. A relação entre permissões associadas a papéis (PA)

é obtida a partir das instâncias *AssignerRBACPermission*. O período em que uma regra deve estar ativa é definido através da instância *PolicyTimePeriodCondition*. As instâncias de *SimplePolicyCondition* são utilizadas para descrever objetos RBAC protegidos e as instâncias de *AssignerOperation* são utilizadas para descrever as operações aprovadas sobre os objetos.

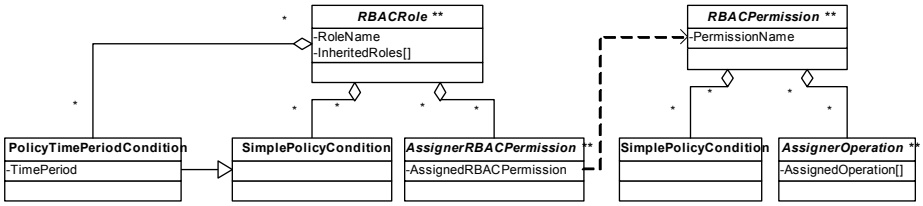


**Figura 2.8 – Associação de Classes RBPIM**

Para que o modelo RBPIM pudesse fazer uso de um sistema de diretórios como repositório padrão, foi necessário o mapeamento de suas classes, bem como daquelas definidas pelo PCIM. A forma de estabelecer este mapeamento seguiu a proposição do IETF definida no PCLS – *Policy Core **Ldap** Schema* [RFC 3703]. Estes mapeamentos não são feitos diretamente na proporção um para um, de forma que uma classe em um modelo não tem, necessariamente, sua correspondente no outro. Sendo assim, são apresentadas as estratégias de mapeamento:

- Classes abstratas dos modelos de informação são mapeadas com classes abstratas no modelo de objetos do LDAP;
- Classes estruturais dos modelos de informação são mapeadas com um conjunto composto de 3 classes LDAP:

- Uma classe abstrata, a qual define a hierarquia de classes LDAP equivalente à hierarquia dos modelos de informação;
- Uma classe estrutural do LDAP.
- Uma classe auxiliar, a qual permite que informações definidas pelos modelos sejam adicionadas a objetos pré-existentes no diretório;

As associações no PCIM efetuam-se através de relacionamentos entre classes, sendo esses passíveis de modelagem de duas formas: Atributos atuando como referências DN e relacionamentos superior-subordinado inerente à DIT do diretório. O RBPIM utiliza as duas formas de modelagem para sua representação.

O *framework* RBPIM, como já citado, adota um modelo baseado nas entidades PDP e PEP sob uma abordagem baseada puramente em *outsourcing*[2]. Desta forma, a maior complexidade reside sobre o PDP ficando sobre o PEP somente as ações mais triviais do modelo. Isto porque o PEP somente recebe os eventos que devem ser tratados, repassa-os ao PDP que processa toda informação recebida, busca no repositório LDAP[3] as políticas aplicáveis, transforma-as de forma que seja entendida pelo PEP e as envia ao PEP para que sejam aplicadas (modelo *outsourcing*). O PDP, no modelo RBPIM, recebe o nome de RBPDP e o PEP recebe o nome de RBPEP. O RBPEP é, basicamente, uma biblioteca de software para as aplicações que necessitem utilizar o método de controle de acesso RBAC definido pelo RBPIM. O RBPIM suporta um conjunto de cinco operações definidas pelas API's:

1. RBPEP_*Open()* – Esta API é a única que não está relacionada ao modelo RBAC, no entanto, é a partir dela que é estabelecida a conexão entre o RBPEP e o RBPDP. A API é utilizada para que uma aplicação solicite ao RBPEP que inicie um serviço RBAC e, somente é executada se não houver uma conexão ativa entre o RBPEP e o RBPDP.
2. RBPEP_*CreateSession(user:string; out session:string, roleset[]:string, usesessions:int)* – Esta API estabelece uma sessão de usuário a partir da sua identificação, retorna um conjunto de papéis, livres de conflitos por separação estática (SSD), que podem ser associados ao usuário, além do número de sessões

[2] No modelo *outsourcing,* praticamente todo o processamento recai sobre o PDP ficando o PEP responsável somente por encaminhar as solicitações recebidas e aplicar as políticas recebidas do PDP
[3] O repositório LDAP utilizado no RBPIM tem como função armazenar tanto objetos que representam informações de rede como objetos que representam políticas.

já abertas por ele (*usesessions*). O parâmetro *session*, é o único gerado pelo RBPEP, que é retornado à aplicação para utilização em futuras solicitações.

3. RBPEP_*SelectRoles(session:string, roleset[]:string; out result:Boolean)* – Esta API tem como função ativar os papéis definidos no parâmetro *roleset*. Através desta API é feita a verificação de separação dinâmica (DSD), para que seja determinado se o conjunto de papéis fornecido pode ser ativado ou não. Se todos os papéis fornecidos pelo parâmetro puderem ser ativados é retornado o valor TRUE no parâmetro *result*.
4. RBPEP_*CheckAccess(session:string, operation:string, objectFilter[]:string; out result:Boolean)* – Esta API tem como função verificar se o usuário possui permissão para executar a operação especificada no parâmetro *operation* , sobre os objetos especificados pelo filtro *objectFilter[] ,* que corresponde a um vetor de expressões do tipo "*PolicyImplicitVariable=PolicyValue*" ou "*PolicyExplicitVariable=PolicyValue*". Se o usuário possuir a permissão para executar a operação sobre o filtro especificado, então é retornado o valor TRUE no parâmetro *result*.
5. RBPEP_*CloseSession(session:string)* – Através desta API é encerrada a sessão do usuário.

A abordagem apresentada no RBPIM utiliza conceitos relacionados ao modelo de controle de acesso associado ao RBAC, políticas de alto nível, conceitos relativos ao PCIM, além de fazer uso do protocolo de comunicação COPS para troca de mensagens entre o servidor de políticas PDP e o cliente de políticas PEP. Muitos destes conceitos e formas de utilização serão adotadas no desenvolvimento do trabalho hora sugerido.

## 2.4 Conclusão

Neste capítulo foram apresentados métodos de controle de acesso, conceitos e padrões utilizados por diversas implementações, além de alguns trabalhos relacionados. Esta apresentação serve como um painel onde se pode verificar as tendências e possibilidades de estudo envolvidas nas áreas relacionadas.

A linguagem Ponder, por ser bastante genérica, pode ser implementada de diferentes formas podendo, inclusive, ser adaptada para gerar uma PIB, adotando também o COPS-PR como protocolo de comunicação.

O XACML mostra-se como uma importante abordagem ao controle de acesso implementando também o modelo RBAC e podendo, ainda, fazer uso da estrutura PEP PDP. No entanto, esta abordagem mostra-se limitada uma vez que permite somente a implementação baseada em outsourcing.

O modelo RBAC foi apresentado como um modelo eficaz para o gerenciamento do controle de acesso. Como visto na proposição RBPIM, a associação deste método a padrões como o PCIM, mostra utilidade e abrangência que o modelo pode alcançar. Dentro desta perspectiva, há um grande espaço para novas proposições. O trabalho aqui apresentado tende a aprimorar mais ainda este modelo, a partir da adaptação do método de controle de acesso sob uma abordagem de provisionamento. Desta forma, o PEP passa a processar informações, diminuindo, assim, a troca de dados entre o servidor de políticas (PDP) e o cliente (PEP). A abordagem de provisionamento tende a ser mais eficiente que a *outsourcing* em algumas aplicações, uma vez que todas as políticas relacionadas a um cliente são a ele repassadas no momento em que é iniciada sua operação, podendo, deste modo, proceder todo o processamento de forma local.

# Capítulo 3

# Padrões IETF

## 3. 1 Introdução

Os padrões, sugeridos pelos diversos órgãos internacionais, têm como intuito desenvolver modelos suficientemente genéricos que sejam capazes de representar suas funções e utilidades, independente de fabricante, como forma de proporcionar escalabilidade e interoperabilidade em ambientes heterogêneos.

Como base para o desenvolvimento deste trabalho, faz-se necessário apresentar alguns padrões, principalmente os propostos pelo IETF (*Internet Engineering Task Force*), destacando suas principais características.

## 3. 2 PBMM - *Policy Based Management Model*

Diversos grupos de trabalho do IETF associados aos grupos de trabalho do DEN – *Directory Enabled Networks* (atualmente absorvidos pelo DMTF), têm definido modelos capazes de identificar entidades envolvidas no processamento de políticas. Um dos modelos de implementação mais utilizados é apresentado na figura a seguir.

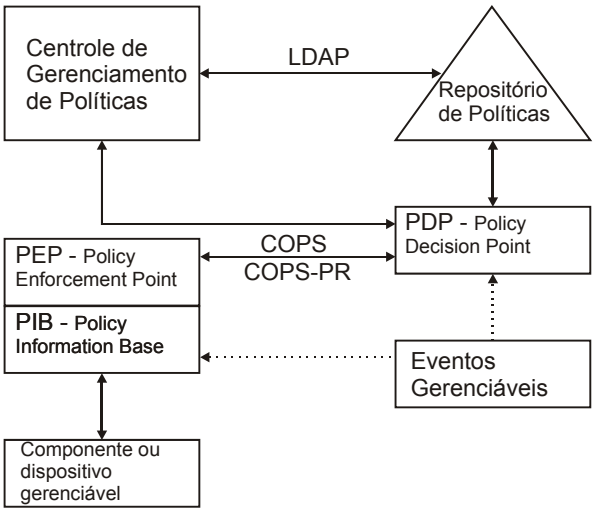


**Figura 3.1 – Modelo PBMM**

No modelo a Console de Gerenciamento tem como função definir e gerenciar as políticas que serão utilizadas. O Repositório armazena todos os dados envolvidos no processo de gerência e administração, gerados pela Ferramenta de Gerência. O PDP é responsável por administrar e interpretar as políticas, armazenadas no repositório, e que serão implementadas pelo PEP, definindo como as mesmas devem ser aplicadas. O PEP é o agente responsável por aplicar efetivamente as políticas; tendo, no modelo provisionamento, também o poder de processamento e decisão através de delegação feita pelo PDP. A PIB – *Policy Information Base* é o elemento encarregado de fazer a tradução das políticas para uma forma que seja entendida pelos dispositivos que farão uso delas. O COPS é um protocolo baseado em estado (*stateful*), simples e extensível, que utiliza um modelo cliente-servidor e tem a sua forma de transporte direcionada ao modelo TCP.

### 3.2.1 Políticas

Políticas podem ser vistas, como um conjunto de regras que especificam como a rede deve atuar para resolver os conflitos gerados em função das interações entre usuários e aplicações disponíveis. Especificam quais recursos e aplicações estarão acessíveis a quais usuários, permitindo a classificação de diferentes aplicações e usuários em categorias [Nabhen 03a].

De acordo com [RFC 3198], a definição de políticas pode ser apresentada de duas formas:

- Como um objetivo, ou método definido de ação, capaz de nortear decisões presentes e futuras. Políticas devem ser implementadas ou executadas sob um contexto definido.
- Como um conjunto de regras para administrar, gerenciar e controlar o acesso a recursos disponíveis em uma rede [RFC 3060].

Cada política inclui uma ou mais condições e uma ou mais ações a serem executadas. As condições definem quando uma regra de política deve ser aplicada, enquanto as ações definem o que fazer e em que ordem [Kosiur 01].

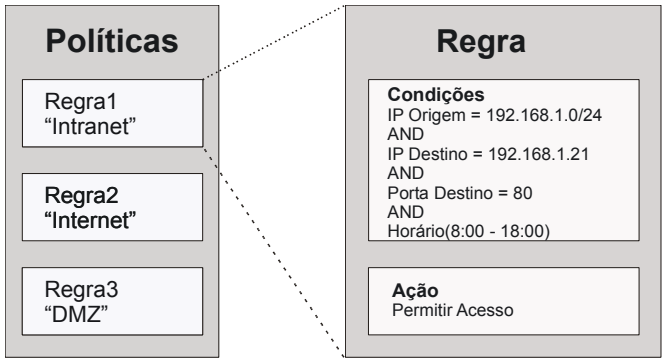


**Figura 3.2 – Elemento de Política**

Como se pode verificar na figura, uma regra pode ser definida de diversas maneiras, através de protocolos, portas, horários, datas, ou ainda, de qualquer parâmetro possível de ser representado e avaliado.

Regras de políticas podem, ainda, ser agrupadas, ou seja, é possível construir uma regra a partir de outras regras mais simples já existentes. Essa possibilidade de aninhar políticas tem como função representar hierarquias, bem como possibilitar a reutilização das mesmas como forma de composição de políticas complexas [RFC 3060].

## 3. 3 PCIMe – *Policy Core Information Model Extensions*

O PCIMe é uma extensão do modelo PCIM [RFC 3060] que traz alterações significativas em relação ao modelo original, através da adição de novos elementos ou substituição de outros que não são mais necessários. A extensão da classe *PolicyCondition*, por exemplo, possibilitou a utilização de variáveis e valores para representação de condições. Essa extensão foi chamada *SimplePolicyCondition*. Outra grande modificação foi a introdução da classe *PolicySet* que permite o agrupamento de regras de políticas. Adiciona, ainda, novas classes que possibilitam a utilização de seqüências ordenadas de ações[RFC 3460].

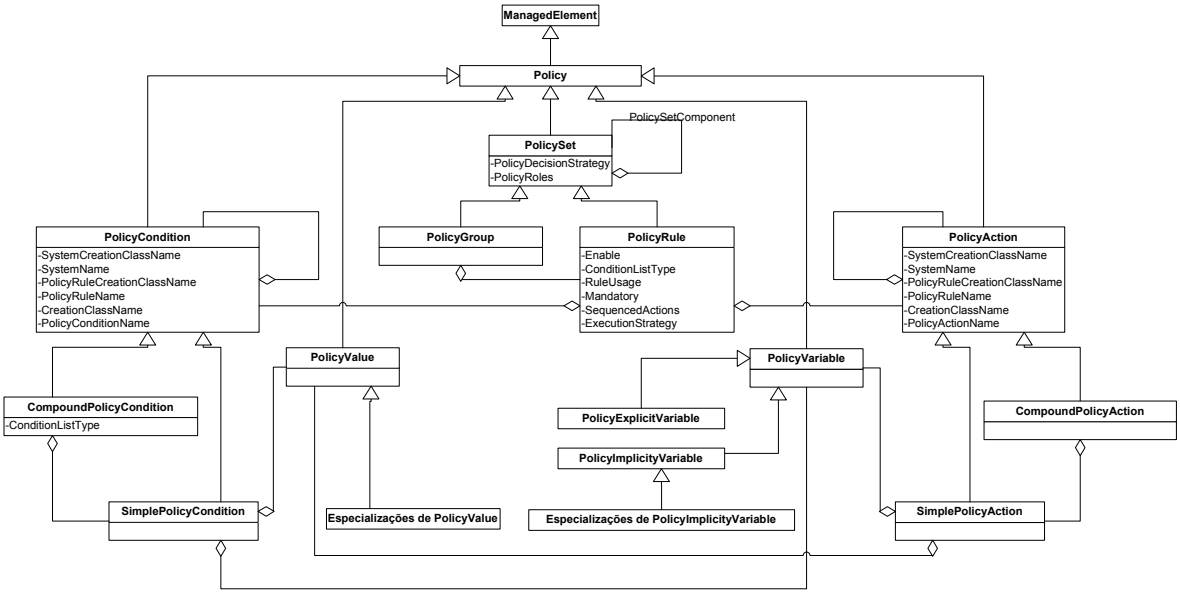


**Figura 3.3 – Representação UML relativa ao Modelo PCIMe**

### 3.3.1 Alterações definidas no PCIMe.

Devido à possibilidade de confusão entre a definição do elemento do *framework* de políticas *Policy Repository* e a classe *PolicyRepository*, essa foi substituída pelo contêiner de políticas reutilizáveis *ReusablePolicyContainer*.

As agregações *PolicyRuleInPolicyRule* e *PolicyGroupInPolicyRule* foram substituídas pela agregação *PolicySetComponent* na classe *PolicySet*.

A propriedade *PolicyDecisionEstrategy* foi adicionada às classes *PolicyGroup* e *PolicyRule* como forma de definir novas estratégias de decisão. Esta propriedade permite ao administrador de política selecionar um dos dois comportamentos relativos à aplicação da regra: *FirstMatching* – Esta estratégia, aplicada a um conjunto de condições, define que somente a primeira ocorrência verdadeira para o conjunto de variáveis e valores será aplicada, sendo as demais descartadas. *AllMatching* – Esta estratégia garante que todas as condições sejam avaliadas, aplicando-se aquelas que forem consideradas verdadeiras em comparação com o conjunto variável valor.

O conceito de papéis de políticas foi adicionado à classe *PolicyGroups*. Isto foi possível através da nova superclasse *PolicySet* que atende tanto ao *PolicyRule* quanto ao *PolicyGroup*.

As estruturas *SimplePolicyCondition*, *PolicyVariable* e *PolicyValue* foram incorporados ao PCIMe, assim como a estrutura correspondente a *SimplePolicyAction*, *PolicyVariable* e *PolicyValue*. A seguir, são apresentadas figuras que reforçam a idéia de formação das condições e ações simples, com suas variáveis e valores.

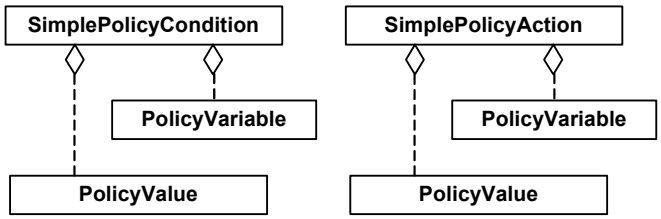


**Figura 3.4 – Estruturas *SimplePolicyCondition* e *SimplePolicyAction***

O conceito de *CompoundPolicyCondition* reforça a idéia de utilizar conjuntos de políticas, que podem existir como elementos nomeados, em *ReusablePolicyContainer*. As classes de composição, e suas associações, incorporam semânticas de condições e ações definidas ao nível da *PolicyRule*. Componentes podem instanciar, respectivamente, *SimplePolicyCondition* e *SimplePolicyAction* , de acordo com a necessidade.

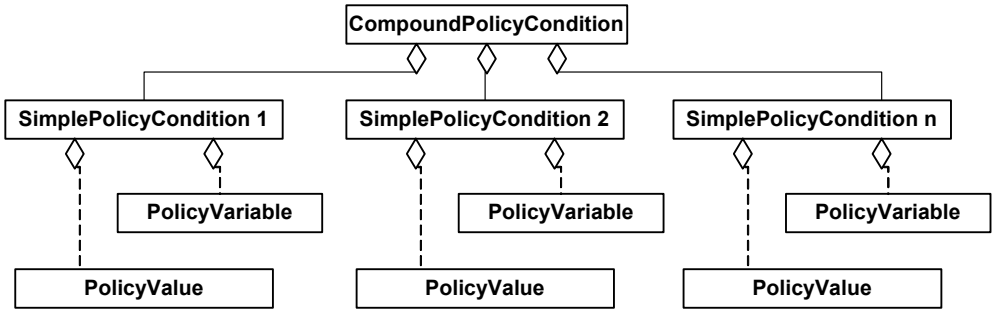


**Figura 3.5 - Exemplo de Composição de Condições Simples**

## 3. 4 PCELS – *Policy Core Extended LDAP Schema*

O modelo de representação definido pelo PCIMe, da mesma forma que o modelo PCIM, não especifica sua forma de implementação, ficando a cargo de cada desenvolvedor optar pela melhor solução. Um grupo de desenvolvimento formado pelo IETF, em conjunto com o DMTF, propôs uma forma de mapear o modelo sob um serviço de diretórios LDAP dando o nome de PCELS – *Policy Core Extended Ldap Schema* [Reyes 04]. Da mesma forma que o PCIMe foi elaborado a partir da extensão do PCIM, o PCELS foi obtido a partir da necessidade de extensão do PCLS a fim de acomodar as funcionalidades exigidas pelo PCIMe.

### 3.4.1 Regras de Composição

O modelo PCIMe [RFC 3460] define duas novas classes que têm como função oferecer ao desenvolvedor a capacidade de criação de condições e ações mais complexas. As classes *CompoundPolicyCondition* e *CompoundPolicyAction* são mapeadas no LDAP como *pcimCompoundConditionAuxClass* e *pcimCompoundActionAuxClass*, subclasses de *pcimConditionAuxClass* e *pcimActionAuxClass*. Essa forma de representar condições/ações estende a capacidade da regra em associar grupos de forma a executar/aplicar condições/ações. Para ilustrar a utilização destas classes, serão apresentadas a seguir, quatro possibilidades de armazenamento de instâncias:

**Caso 1:** Composição específica formada por condições/ações específicas

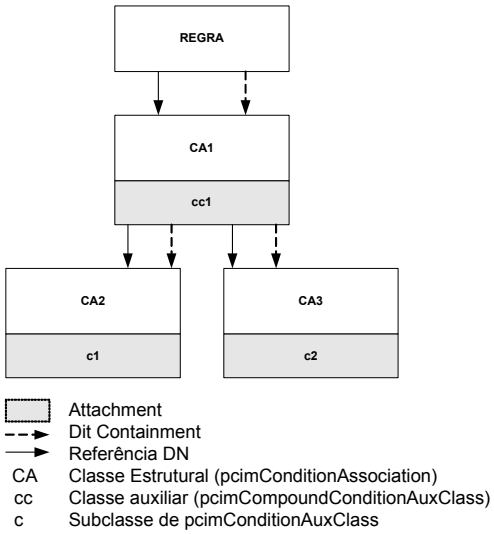


**Figura 3.6 – Composição Específica Formada por Condições/Ações Específicas**

Como as condições/ações pertencem a uma regra específica, devem ser anexadas como classes auxiliares às instâncias de *pcimConditionAssociation* ou *pcimActionAssociation*. Estas classes estruturais representam a associação entre uma regra e as composições de condições/ações. As condições/ações específicas são, portanto, subordinadas (via DIT) à regra.

**Caso 2:** Composição específica formada por condições/ações reutilizáveis

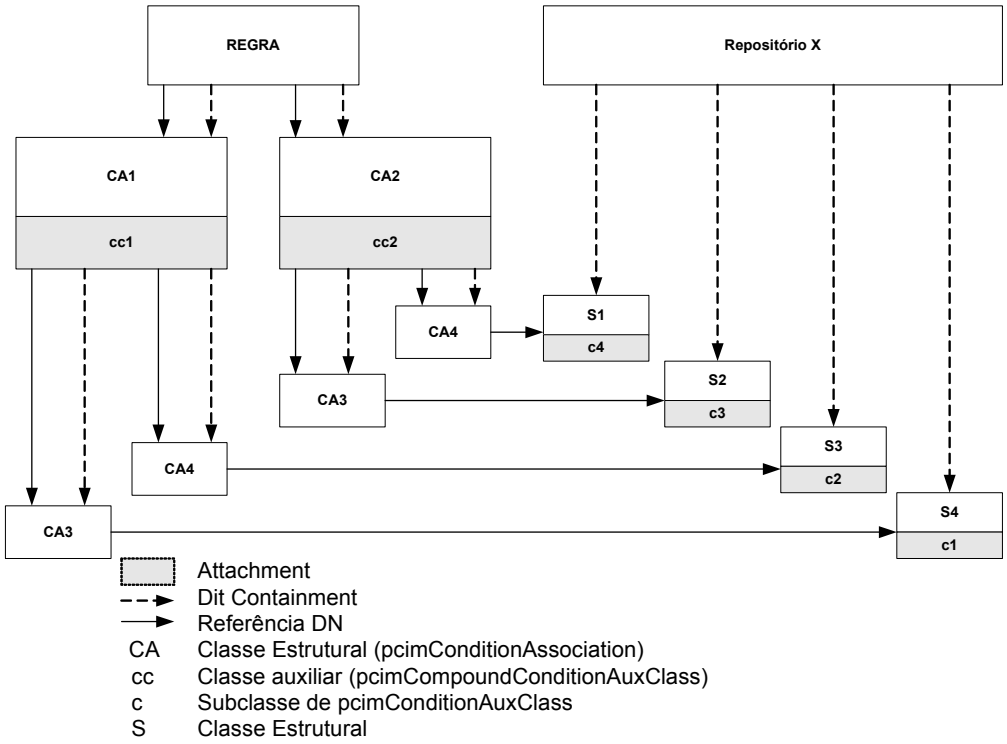


**Figura 3.7 – Composição Específica Formada por Condições/Ações Reutilizáveis**

De forma similar ao primeiro caso, as condições/ações são reutilizáveis, portanto, não podem ser anexadas às classes associativas, sendo anexadas a classes estruturais no container reutilizável. As classes associativas são "amarradas" às condições/ações localizadas no container pela utilização de referências DN.

**Caso 3**: Composição reutilizável formada por condições/ações específicas

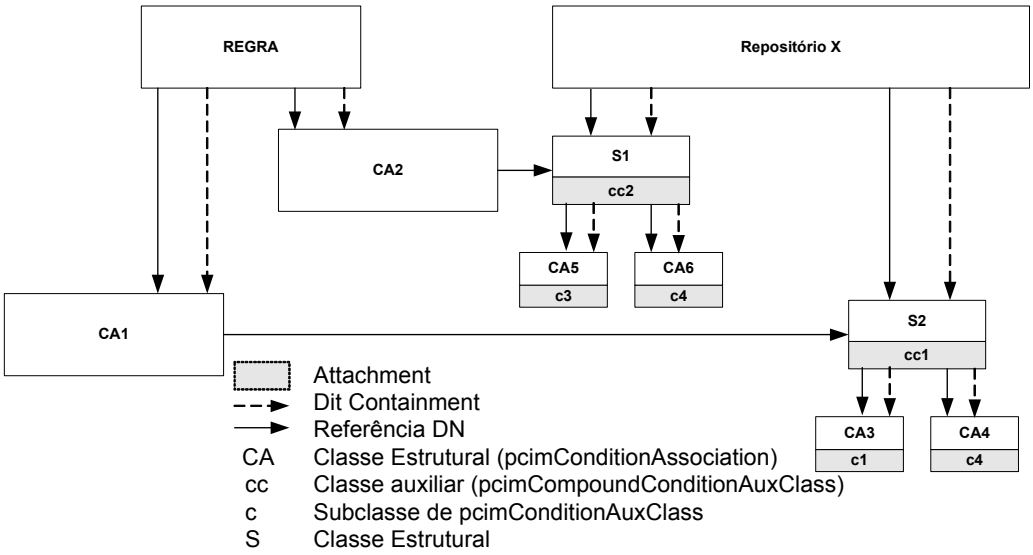


**Figura 3.8 – Composição Reutilizável Formada por Condições/Ações Específicas**

Composições reutilizáveis são anexadas a classes estruturais e armazenadas em um container de políticas reutilizáveis. Essas composições são associadas às regras através de referências DN nas classes de associação. As condições/ações específicas são anexadas às instâncias das classes de associação e subordinadas (via DIT) às composições de condições/ações.

**Caso 4**: Composição reutilizável formada por condições/ações reutilizáveis

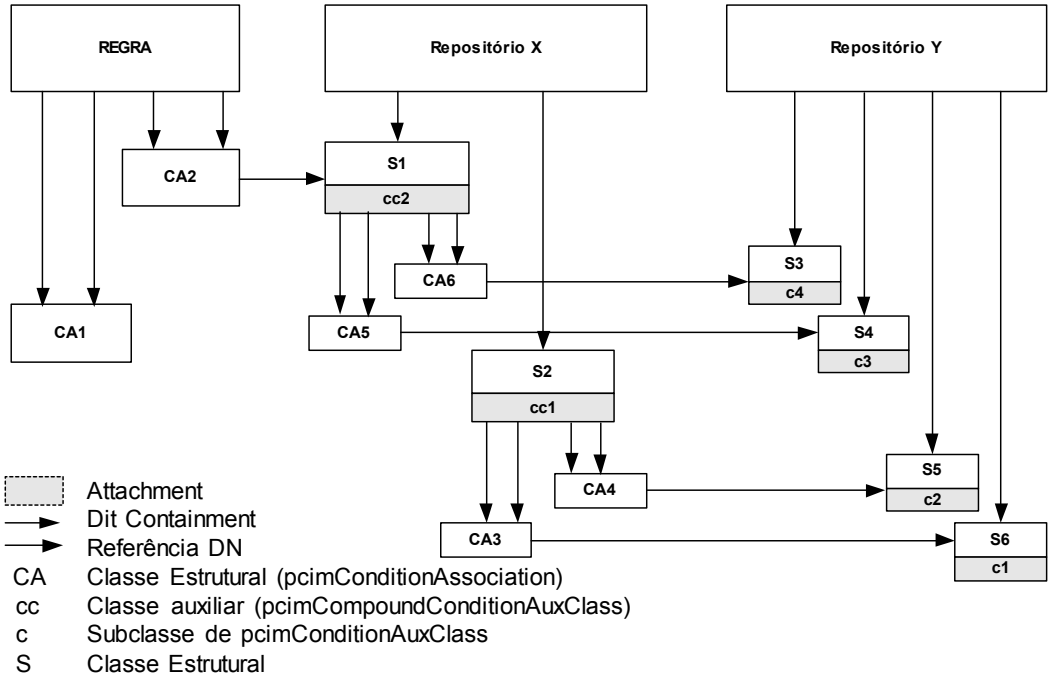


**Figura 3.9 – Composição Reutilizável Formada por Condições/Ações Reutilizáveis**

Todas as condições/ações são reutilizáveis. Portanto, todas são armazenadas em containeres reutilizáveis. A figura 3.9 ilustra dois containeres de políticas reutilizáveis diferentes (Repositório X e Y). O número de containeres do sistema é definido administrativamente. As condições/ações (cc1 e cc2) são reutilizáveis de forma que não podem ser anexadas às classes associativas (CA1 e CA2), assim, são anexadas a classes estruturais (S1 e S2) no container reutilizável (Repositório A). As classes associativas (CA1 e CA2) são "amarradas" às condições/ações localizadas no container (Repositório A) por referência DN. Como as condições/ações (c1, c2, c3 e c4) que formam as composições reutilizáveis (cc1e cc2) também são reutilizáveis e pertencem a outro container (Repositório B), essas condições/ações (c1, c2, c3 e c4) devem ser anexadas a classes estruturais (S1, S2, S3 e S4) do Repositório B para serem referenciadas pelas condições/ações reutilizáveis do Repositório A (CA3, CA4, CA5 e CA6) através de referência DN.

## 3. 5 COPS – *Common Open Policy Service*

O COPS é um protocolo simples de consultas e respostas, que pode ser utilizado para troca de informações de política entre um servidor de políticas (PDP) e seus clientes (PEP) [RFC 2748]. O protocolo utiliza um modelo cliente/servidor, onde o PEP envia requisições ao PDP, que retorna a decisão de volta ao PEP. Utiliza o TCP como seu protocolo de transporte. É extensível, permitindo seu uso generalizado para administração, configuração e aplicação de políticas.

Para possibilitar o funcionamento em modo provisionamento, o COPS foi estendido passando a ser referenciado como COPS-PR [RFC 3084]. Esta extensão possibilitou o controle de políticas relacionadas a eventos não sinalizados. Sua principal utilização é relacionada à configuração de dispositivos de rede, porém, pode ser utilizado, como sugerido neste trabalho, sob outras caracterizações.

### 3.5.1 Modelo Básico

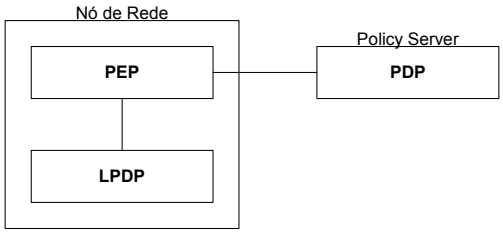


**Figura 3.10 – Modelo Básico COPS**

Na figura, o COPS é usado para comunicar informações de políticas entre o *Policy Enforcement Point* (PEP) e um *Policy Decision Point* (PDP) remoto. O PEP é responsável por iniciar uma conexão TCP persistente com o PDP, e, é através desta conexão que requisições são enviadas e decisões recebidas. A comunicação entre o PEP e um PDP remoto é, normalmente, da forma *stateful* de troca requisição/decisão. Contudo, o PDP pode, ocasionalmente, enviar mensagens de decisões não solicitadas pelo PEP, para que as mesmas sejam aplicadas de forma imediata. O PEP tem a capacidade de reportar ao PDP se a aplicação da política foi executada com sucesso. Esta função é utilizada para monitoramento e contabilidade [RFC 2748].

### 3.5.2 Protocolo COPS

**Cabeçalho**

Cada mensagem COPS consiste de um cabeçalho seguido por um número de objetos tipificados. O cabeçalho comum é visto a seguir:

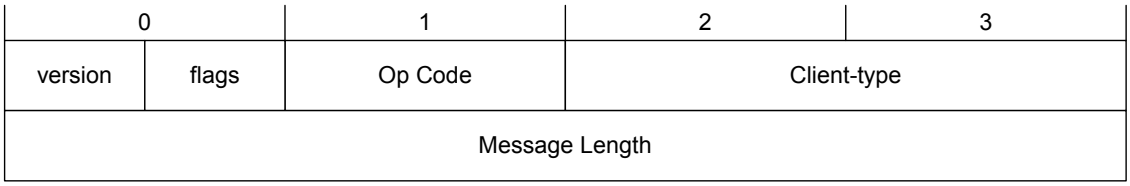


**Figura 3.11 – Cabeçalho COPS**

Os campos contidos no cabeçalho são: Versão – Composto por 4 bits detalham a versão do protocolo utilizado. *Flag* – Composto por 4 bits que podem ser de valor 0x1 para mensagem solicitada ou 0 para as demais mensagens. Este *flag* é setado quando a mensagem é solicitada por outra mensagem COPS. *OpCode* – Composto por 8 bits que indicam a operação COPS. *Client-type* – Composto por 16 bits identifica o tipo do cliente da política. A interpretação de todos os objetos encapsulados tem como base este campo. O bit mais significativo deste campo determina o código específico do dispositivo/fabricante. *Message lenght* – Composto por 32 bits, define o tamanho da mensagem em octetos, incluindo o cabeçalho, alinhados em intervalos de 4 octetos.

As operações comuns suportadas pelo protocolo COPS correspondem a: Requisição (REQ) é a mensagem repassada do PEP ao PDP com a solicitação das políticas correspondentes. Decisão (DEC) é a mensagem repassada do PDP ao PEP com as políticas a serem implementadas. Estado (RPT) é a mensagem trocada entre PEP e PDP informando resultados relacionados às aplicações de políticas. Requisição de exclusão de estado (DRQ) determina que o identificador (*handle*) não é mais necessário. Requisição de sincronização de estado (SSQ) solicita que o PEP reenvie suas informações para que uma sincronização com o PDP seja efetuada. Conexão aberta com cliente (OPN) utilizado pelo PEP para comunicar ao PDP o tipo de cliente suportado. Conexão aceita pelo cliente (CAT) é a mensagem de resposta a OPN que tem como função comunicar o aceite da conexão. Conexão encerrada pelo cliente (CC) responsável por encerrar uma conexão entre o PEP e o PDP. Atividade (KA) é uma mensagem trocada entre as entidades como forma de sinalizar seu estado ativo. Sincronização completa (SSC) enviada do PEP ao PDP após o recebimento da

mensagem SSQ indicando que a sincronização foi finalizada. O protocolo COPS-PR define somente a utilização das mensagens REQ, DEC e RPT como forma de repassar suas informações sobre objetos de dados específicos. Porém, todas as mensagens COPS-PR estão em conformidade com as especificações das mensagens utilizadas no COPS.

**Formato Específico de Objetos COPS**

O cabeçalho do COPS define o tipo da mensagem. Após o cabeçalho existe um grande número de elementos de política que o seguem, e que são utilizados para transferência de informações que dizem respeito a decisões que devem ser tomadas. Os elementos de política são definidos, no COPS, de acordo com a estrutura apresentada abaixo.

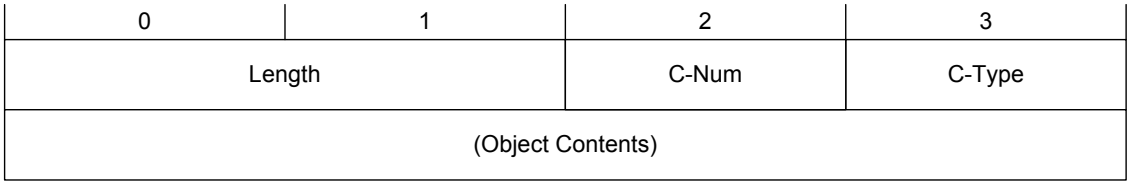


**Figura 3.12 – Formato Específico Objeto COPS**

Todos os objetos seguem o mesmo formato e consistem de uma, ou mais, palavra de 32 bits com um cabeçalho de quatro octetos.

O campo *Lenght* define o tamanho da mensagem e é um valor composto por dois octetos que descrevem o número total de octetos, incluindo o cabeçalho, que compõem o objeto. *C-Num* é composto por 8 bits e, tipicamente, identifica a classe de informações contidas no objeto. *C-Type* é composto por 8 bits e identifica os subtipos ou versões da informação contida no objeto. Cada objeto COPS possui seus respectivos conteúdos. Estes conteúdos possuem comprimentos variáveis que dependem dos campos *C-Num* e *C-Type*. Mensagens entre o PDP e o PEP são construídas a partir destes objetos.

**Formato Específico de Objetos COPS-PR**

O modelo COPS-PR encapsula diversos novos objetos sobre os objetos de informação COPS *Named Client-specific* e *Named Decision Data*. O formato para esses objetos é apresentado a seguir.

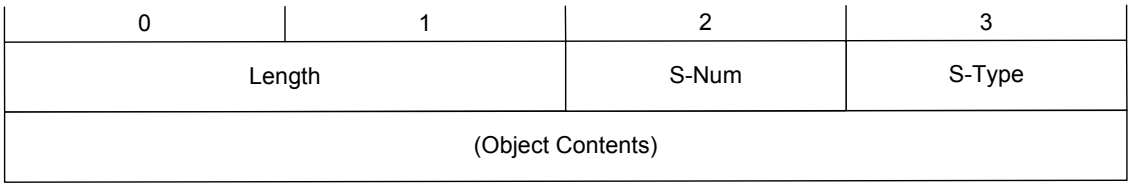


**Figura 3.13 – Formato Específico Objeto COPS-PR**

Os campos *S-Num* e *S-Type* são similares aos campos *C-Num* e *C-Type* definidos sobre o objeto COPS. A diferença é que os campos no COPS-PR são utilizados somente em relação aos objetos *Named ClientSI* ou *Named Decision Data*. O campo *S-Num* identifica o propósito do objeto, enquanto que *S-Type* descreve a codificação específica utilizado pelo objeto.

**Comunicação**

O protocolo COPS utiliza conexão TCP persistente entre o PEP e o PDP. Uma implementação PDP deve utilizar uma porta de conexão TCP conhecida. A porta definida pelo IANA para o COPS é a de número 3288. O PEP é responsável por iniciar a conexão TCP com o PDP.

As principais mensagens COPS, utilizadas neste trabalho, serão apresentadas a seguir:

- **Mensagem OPN -** A mensagem open (OPN) é responsável por estabelecer a conexão entre cliente e servidor.

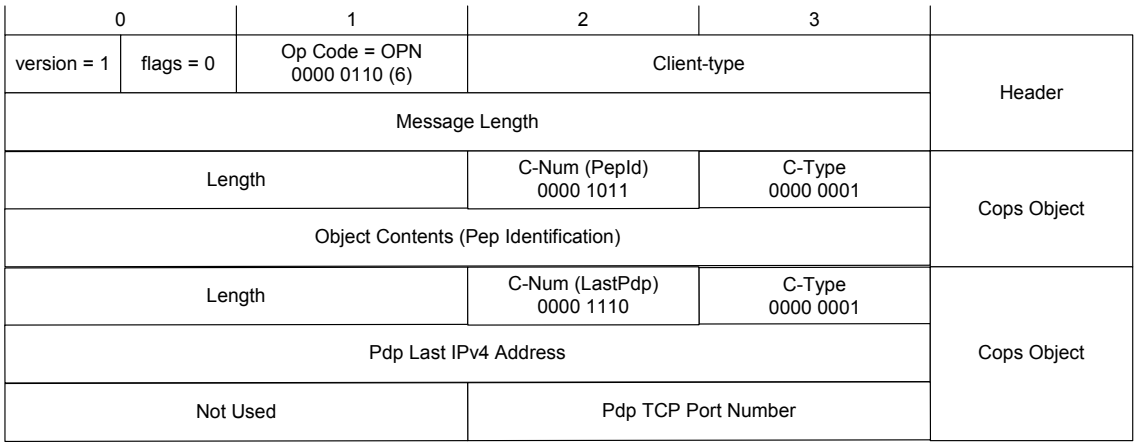


**Figura 3.14 – Mensagem OPN**

Enviada ao PDP no momento da inicialização, esta mensagem é utilizada para especificar o tipo de cliente suportado pelo PEP e sua identificação. Essas informações são relevantes para que a conexão entre cliente e servidor seja estabelecida. De posse destas informações, o PDP verifica se o tipo específico de cliente pode ser atendido e se já não existe uma conexão estabelecida para o mesmo. Caso as condições não sejam satisfeitas é retornada uma mensagem ao PEP informando que a conexão não pode ser estabelecida, encerrando a comunicação, mensagem *Client-Close* (CC). Satisfeitas as condições, é enviada uma mensagem ao PEP informando o aceite da conexão (CAT).

- **Mensagem CC**

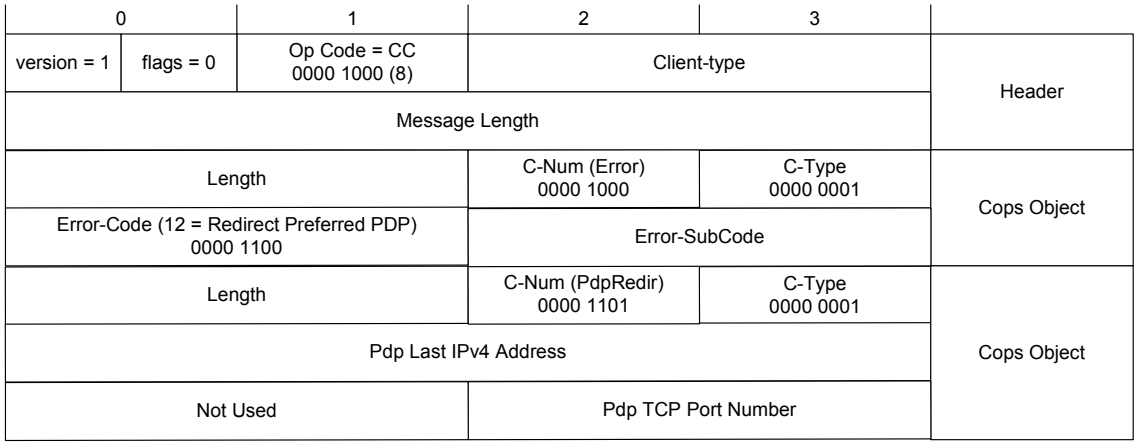


**Figura 3.15 – Mensagem CC**

A mensagem *Client-Close* pode ser enviada tanto do PEP ao PDP quanto em sentido contrário para informar o fim da conexão. Ao ser enviada do PDP para o PEP, em resposta a uma mensagem OPN, indica que o PDP não suporta aquele tipo específico de cliente ou que já existe uma conexão aberta. Na mensagem, é então, anexado um elemento correspondente ao próximo servidor disponível. De posse desta informação o PDP envia uma nova mensagem OPN para esse servidor. Na implementação, como forma de controle, foi incluído um número máximo de mensagens OPN que o PEP pode proceder evitando, assim, a ocorrência de *loop* ou um número elevado de tentativas de conexão. Depois de atingido esse número é gerado uma mensagem de erro no PEP.

- **Mensagem CAT**

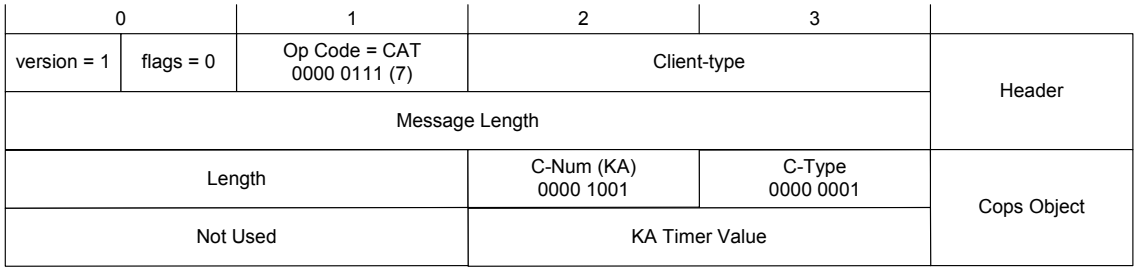


**Figura 3.16 – Mensagem CAT**

A mensagem *Client-Accept* (CAT) é utilizada como resposta positiva a uma solicitação OPN. Nesta mensagem é adicionado um elemento referente ao tempo máximo em que mensagens de verificação de conexão, *Keep-Alive* (KA), devem ser trocadas entre cliente e servidor. Ao receber a mensagem CAT o PEP processa o timer recebido e procede à criação da mensagem de requisição de políticas (REQ) a ser enviada ao servidor.

- **Mensagem REQ**

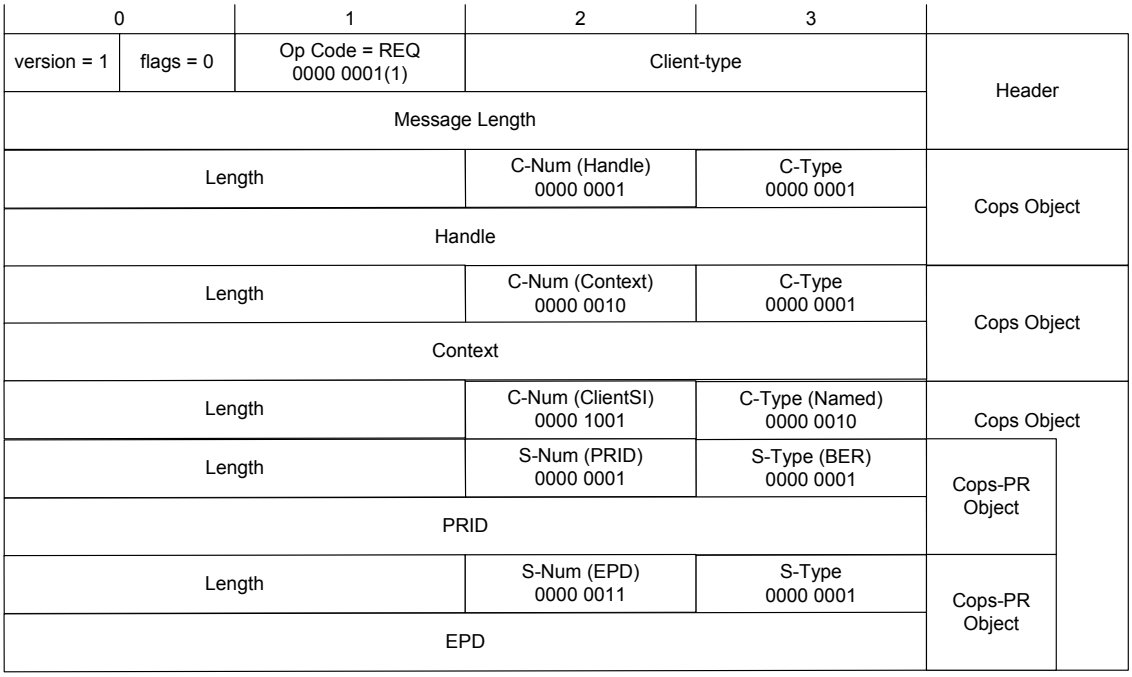


**Figura 3.17 – Mensagem REQ**

A mensagem *Request* (REQ) é enviada ao PDP como solicitação de configuração do PEP e contém todas as informações necessárias ao PDP para que possa associar os dados constantes no repositório ao cliente. O primeiro elemento desta mensagem é o identificador, *Client Handle*, que é utilizado para identificar um estado específico de requisição, e deve ser único para um determinado cliente. O segundo elemento é o *Context*, que tem como função especificar o tipo de evento que está motivando a mensagem. De acordo com o COPS-PR, este objeto é necessário em mensagens de requisição, controle de admissão, alocação de recursos, requisições de repasse (*forwarding*). O objeto *Context* possui o campo conteúdo separado em dois blocos, de dois bytes cada, definidos como *R-Type* e *M-Type*. O campo *R-Type* define o tipo de requisição, no caso proposto 0x08 que define requisição de configuração. O campo *M-Type* define o tipo de mensagem e é composto por 16 bits. Este campo não é definido na documentação do COPS-PR ficando livre a cada implementação definir a forma de utilização, de acordo com a aplicação. Os demais elementos têm como função transportar informações específicas sobre o cliente como capacidades, limitações, tipos de classes e atributos suportados. Estas informações são encapsuladas, como objetos COPS-PR, em objetos COPS do tipo *Named ClientSI*. As informações do tipo PRID

correspondem ao identificador único de uma instância (classe) provisionada e são codificadas seguindo o mesmo conjunto de regras aplicáveis a objetos OID utilizados em SNMP. As informações do tipo EPD correspondem aos atributos compreendidos em uma classe provisionada. Tanto os elementos PRID como os elementos EPD são codificados como séries compostas pelo padrão TLV onde T corresponde ao tipo do dado, L corresponde ao tamanho do conjunto de dados e V corresponde ao valor. Esta codificação segue a especificação BER (*Binary Encoding Rules*) também utilizado para identificação de objetos em SNMP.

A partir da utilização dos conjuntos de PRID e EPD, que identificam de forma única um elemento ou conjunto de elementos, é possível ao COPS-PR aplicar as operações de instalação, remoção e atualização de dados sobre uma estrutura de dados como a PIB[4].

[4] A PIB, *Policy Information Base*, é uma estrutura de dados em árvore capaz de armazenar os elementos de política repassados ao PEP. Esta estrutura será descrita de forma detalhada na seção 3.5.4

- **Mensagem DEC**

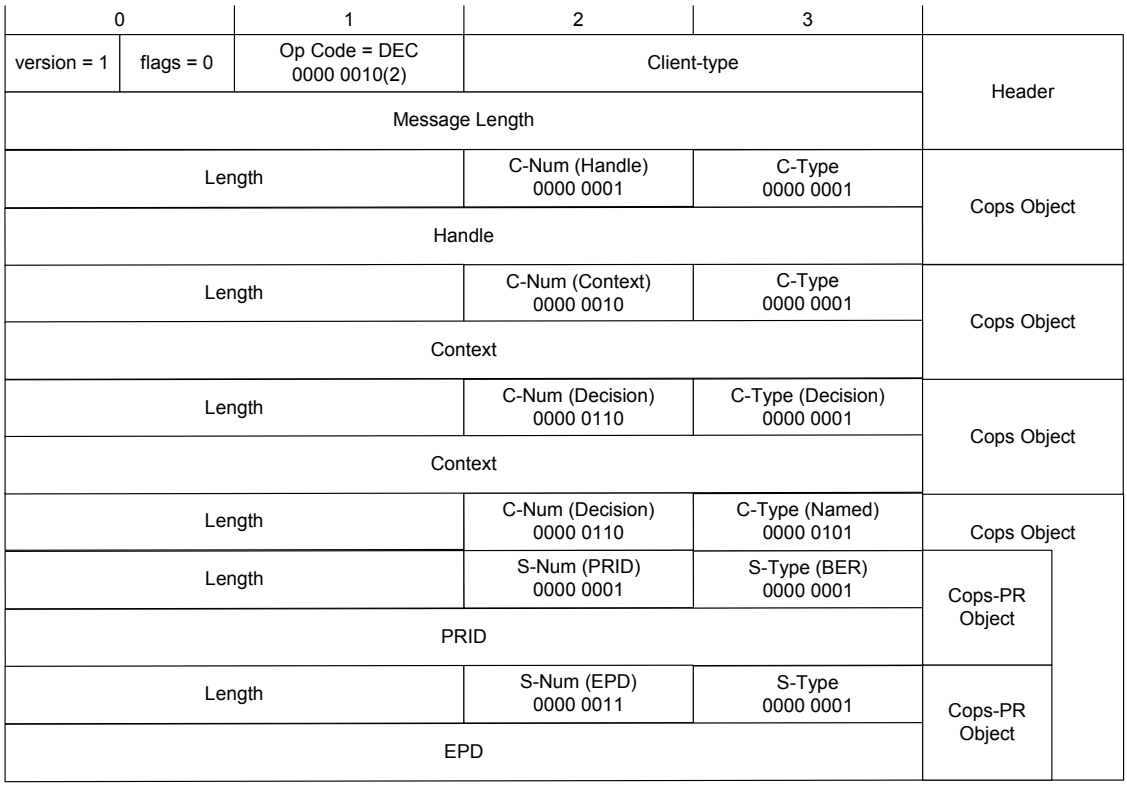


**Figura 3.18 – Mensagem DEC**

A mensagem *Decision* (DEC) é enviada ao PEP como resposta a uma mensagem REQ e corresponde aos seus elementos de configuração. A mensagem deve conter um *client handle* correspondente àquele enviado na requisição, indicando que é uma operação referenciada sob um mesmo domínio. Cada mensagem DEC pode conter diversas decisões de política e configuração. Isto significa que uma mesma mensagem pode instalar algumas políticas como excluir outras já instaladas. Como mecanismo de controle, não sendo possível a implementação da decisão recebida, a mesma é descartada sendo mantida a anteriormente instalada. A componente da mensagem chamada *Decision,* corresponde, efetivamente, à decisão que deve ser implementada na forma de classes e atributos na PIB.

- **Mensagem RPT**

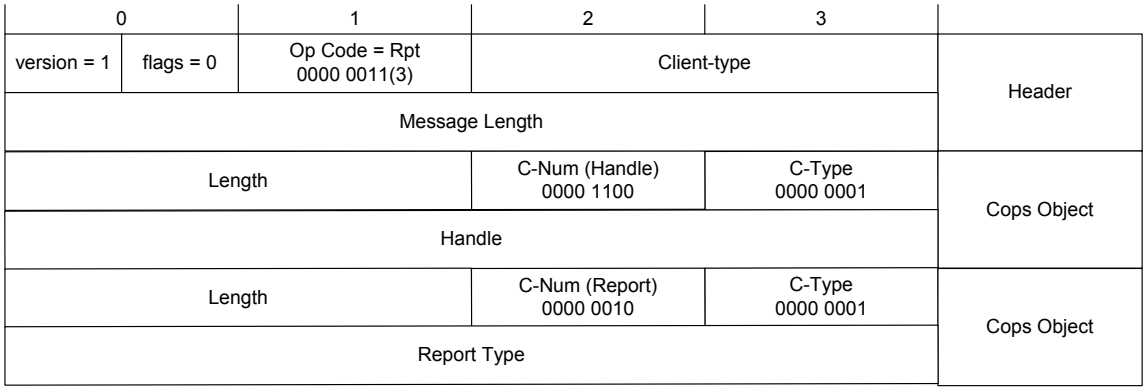


**Figura 3.19 – Mensagem RPT**

As mensagens *Report State* (RPT) são trocadas entre PDP e PEP para indicar o resultado de uma operação solicitada em uma mensagem. Após o recebimento de uma mensagem DEC, o PEP pode enviar uma mensagem RPT ao PDP informando o sucesso ou a falha na aplicação da política de configuração. No caso de sucesso, o PDP recebe a mensagem e atualiza seus dados referentes ao PEP com a mesma informação de configuração de política recebida e aplicada pelo PEP. Em caso de falha o PDP mantém os dados referentes ao PEP exatamente como estavam antes do envio da mensagem DEC descartando a decisão enviada.

### 3.5.3 Modelo *Outsourcing*

O modelo *outsourcing* endereça um tipo de evento ao PEP que requer uma decisão de política instantânea (em tempo real). No cenário de *outsourcing*, o PEP delega a responsabilidade a um servidor de política externo (PDP) que toma as decisões de seu interesse [RFC2748].

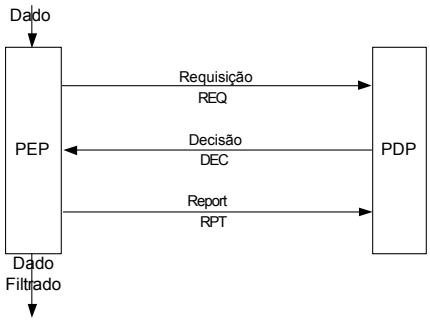


**Figura 3.20 – Modelo *Outsourcing***

Neste modelo, cada evento é recebido pelo PEP e repassado, como uma mensagem de requisição de política (REQ), ao PDP. O PDP verifica qual a política a ser aplicada e devolve ao PEP uma mensagem de decisão. Esta mensagem de decisão é então interpretada e aplicada pelo PEP.

### 3.5.4 Modelo Provisionamento

O modelo de configuração, também chamado provisionamento (*proviosioning*), não pressupõe uma correlação 1:1 entre eventos PEP e decisões PDP. O PDP pode, ativamente, repassar ao PEP decisões que sejam ativadas por eventos externos, eventos PEP, ou qualquer combinação destes numa relação N:M. O modelo pode ser aplicado em conjunto ou em partes.

O modelo *provisioning* pressupõe a carga de um conjunto de políticas no PEP, repassadas pelo PDP, na inicialização, para posteriormente aplicá-las, localmente, de acordo com a ocorrência de eventos [RFC 3084].

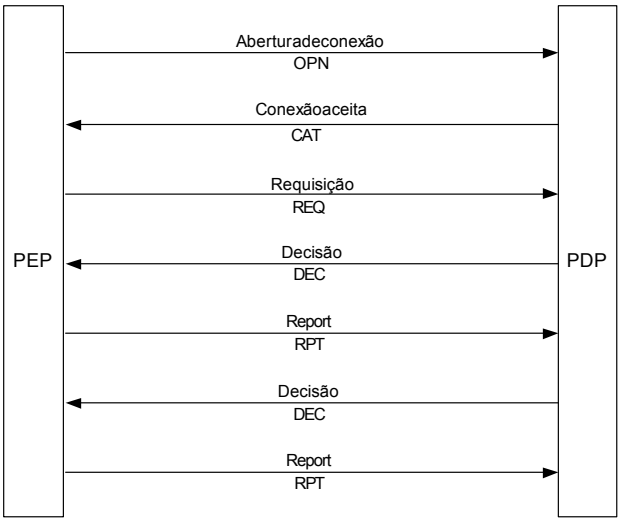


**Figura 3.21 – Modelo Provisionamento**

O PEP, ao iniciar, envia todos os seus dados relevantes ao PDP. De posse destes dados o PDP os processa e obtém todas as informações necessárias à configuração do PEP para que esse possa funcionar de maneira independente. Os dados de configuração são, então, repassados ao PEP que os armazena e, a partir deste momento, passa a processar as informações, recebidas através de eventos externos, localmente. A troca de informações entre o PDP e o PEP a partir deste momento é feita somente como sinalização, ou caso ocorra uma modificação das políticas ou dos dispositivos envolvidos, através de uma solicitação de sincronização entre PEP e PDP.

O COPS pode ser utilizado num modelo *provisioning* por duas razões:

- O COPS-PR permite transporte eficiente de atributos para grandes quantidades de dados, além de relatórios flexíveis e eficazes.
- Mantém uma única conexão entre o cliente e o servidor de acordo com a área de controle de política, identificado como um tipo de cliente COPS. Isto garante que somente um servidor faça atualizações de política de configuração em um determinado tempo.

**PIB – *Policy Information Base***

O protocolo COPS-PR define a estrutura de dados por ele transportado, como um conjunto de políticas. Esta definição é formalizada em uma estrutura de dados denominada *Policy Information Base* (PIB), capaz de identificar o tipo e propósito de uma informação enviada do PDP ao PEP, constituindo um espaço de nome comum, tanto ao PEP como ao PDP. A PIB pode ser descrita como uma estrutura conceitual, em forma de árvore, em que os ramos representam classes de dados de provisionamento (PRC) e as folhas as várias instâncias destas classes (PRI) [RFC 3084].

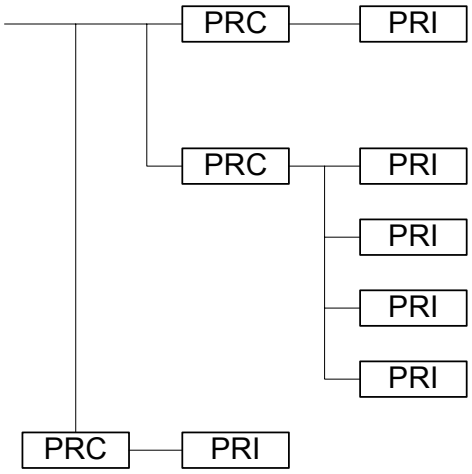


**Figura 3.22 – Representação Básica da PIB**

A PIB permite, ainda, ações que possibilitam sua modificação ou extensão, tornando-a interoperável entre PEP e PDP, como:

1. Novas PRC's podem ser adicionadas ou as já existentes podem ser retiradas.
2. Atributos podem ser adicionados ou retirados da estrutura.
3. Uma PRC existente pode ser entendida através da utilização de novas PRC's definidas em outra PIB.

Uma instância de provisionamento, tipicamente, contém um valor para cada atributo definido na PRC. Cada um desses atributos pode suportar um conjunto de operações:

- Instalação (*Install*) – Esta operação cria ou atualiza uma instância de uma PRC e utiliza dois parâmetros: A identificação da classe (PRID) e o objeto correspondente à instância, com o valor a ser criado ou atualizado (EPD). Este identificador (PRID) deve ser único para uma PRI.
- Remoção (*Remove*) – Esta operação é utilizada para retirar uma determinada instância que esteja carregada. Podem ser removidos atributos de uma classe ou toda a classe. Para a exclusão basta um parâmetro que identifique, unicamente, o dado a ser removido.

Conforme a descrição do COPS-PR, cada cliente suporta um conjunto próprio e independente de módulos PIB. No entanto, existem algumas classes e instâncias comuns a qualquer tipo de cliente [RFC 3318].

Tanto PRC's como PRI's são identificadas através de OID's. Um OID é um identificador único composto por string de números alocados de forma hierárquica. A definição formal dos OID's pode ser verificada na recomendação X 208 do ITU-T. Esta forma de identificação permite definir classes e instâncias pertencentes a esta classe. Assim, se for definido o OID 1.3.6.1.2.2.2.3.2 para a classe *IPFilter* é possível identificar suas instâncias a partir deste OID na forma 1.3.6.1.2.2.2.3.2.**1** para a primeira instância, 1.3.6.1.2.2.2.3.2.**2** para a segunda instância e assim por diante. O prefixo 1.3.6.1.2 refere-se ao identificador (OID), Gerenciamento IETF, definido pelo IANA de acordo com [RFC 1155]. É através deste identificador único que o COPS-PR pode efetuar as operações definidas, instalação ou remoção, sobre uma classe ou instância específica.

**Papéis (*Roles*)**

Os papéis podem ser definidos como uma identificação (String) que é associada a uma interface [RFC 3318].

Essa associação tem como objetivo proporcionar escalabilidade e reutilização de políticas, uma vez que um único papel pode ser atribuído a diversas interfaces que possuem funções semelhantes na estrutura.

Quando o PEP faz uma requisição de políticas ao PDP (REQ), repassa todo o seu conjunto de capacidades, combinações de papéis aceitáveis, interfaces controladas por

políticas e seus relacionamentos com o PDP, para que este possa, com base nestes dados, carregar e enviar as configurações necessárias, através de uma mensagem de decisão (DEC).

## 3. 6 Conclusão

Este capítulo apresentou os principais padrões que serão utilizados na implementação sugerida por este trabalho: a representação de um modelo de controle de acesso baseado em políticas de alto nível, sob a abordagem de provisionamento. A utilização do PCIMe será de grande importância para a representação das políticas associadas ao RBAC. A troca de mensagens entre as entidades PEP e PDP será efetuada através da proposição COPS-PR. A modelagem da PIB será realizada de acordo com a proposição contida no COPS-PR associada ao *Framework Policy Information Base* [RFC 3318].

# Capítulo 4

# Modelo de Informação e PIB

## 4.1 Introdução

Neste capítulo, será apresentado o modelo proposto pelo trabalho, ressaltando suas particularidades, em relação ao modo provisionamento. Primeiramente, será apresentado o modelo de informação sob a ótica do PCIMe ressaltando o conceito de papéis. A seguir, uma visão geral da concepção adotada para o trabalho, e sua aplicabilidade, apresentando, ainda, seus componentes, principalmente PEP e PDP, e suas funções. Encerrando o capítulo é apresentada a PIB, peça chave para o desenvolvimento do trabalho.

## 4.2 Modelo de Informação

O modelo de informação adotado neste trabalho segue a abordagem utilizada pela implementação RBPIM, sugerida em [Nabhen 03]. Constituindo-se como uma extensão do PCIM, tem como propósito descrever políticas de controle de acesso, a partir da extensão do modelo RBAC, proposto pelo NIST e padronizado pela ANSI. As extensões introduzidas no modelo, permitem que o mapeamento entre papéis RBAC e permissões possa ser efetuado de forma mais flexível.

O modelo sugerido pela implementação RBPIM tem como característica a abordagem puramente baseada em *outsoucing*, o que exigiu, para elaboração deste trabalho, uma readequação do mesmo, como forma de possibilitar a nova aplicação sugerida, sob a ótica do provisionamento. A figura a seguir mostra a visão geral do modelo de informação sugerido por este trabalho.

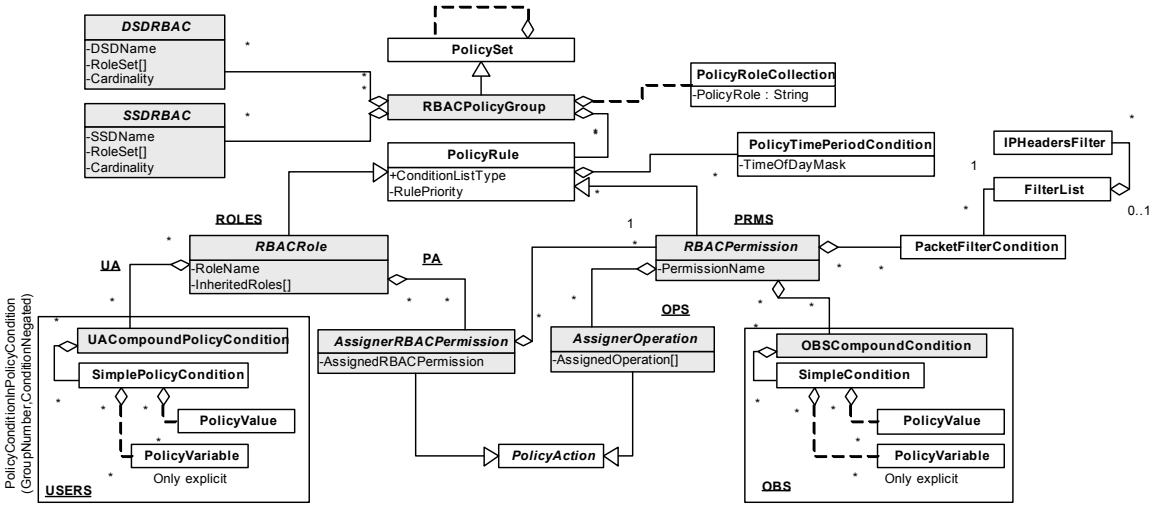


**Figura 4.1 – Visão Geral do Modelo de Informação – RBPIM (revisado)**

Do modelo RBPIM, vieram as classes representadas na cor cinza, sendo as demais originárias dos modelos PCIM [RFC 3060] e PCIMe (RFC 3460). Basicamente, o modelo define duas extensões da classe *PolicyRule*: A *RBACRole* criada como forma de representar os papéis RBAC (ROLES), e a classe *RBACPermission* criada como forma de representar as permissões (PRMS). Usuários (USER) e objetos (OBS) são representados por uma extensão da classe *CompoundPolicyConditions* permitindo, assim, a criação de expressões lógicas na forma CNF ou DNF, a partir de agrupamento de instâncias da classe *SimplePolicyCondition*. Duas extensões da classe *PolicyAction* também são propostas. A classe *AssignerRBACPermissions* permite que seja estabelecida uma associação entre papéis RBAC e permissões, enquanto *AssignerOperation* é utilizada para representar operações (OPS). Os relacionamentos UA (*User Assignment*) e PA (*Permission Assignment*) são representados pelas agregações presentes em RBACRole. As separações estáticas e dinâmicas, descritas no modelo RBAC, são representadas, respectivamente, pelas instâncias DSDRBAC e SSDRBAC.

Diversas alterações foram realizadas sobre o modelo RBPIM sugerido em [Nabhen 03a] sendo que as mais expressivas dizem respeito à utilização de condições de políticas compostas (*CompoundCondition*) a partir de condições simples, a utilização de agregações de *PolicySet* , a utilização da classe *PolicyRoleCollection*, a possibilidade de reaproveitamento de políticas através da utilização de contêineres reutilizáveis, entre

outras. Estas alterações permitiram adequar totalmente o modelo ao PCIMe, bem como prover uma sensível melhoria no funcionamento geral do modelo

A figura a seguir permite um melhor entendimento do modelo a partir de sua instanciação.

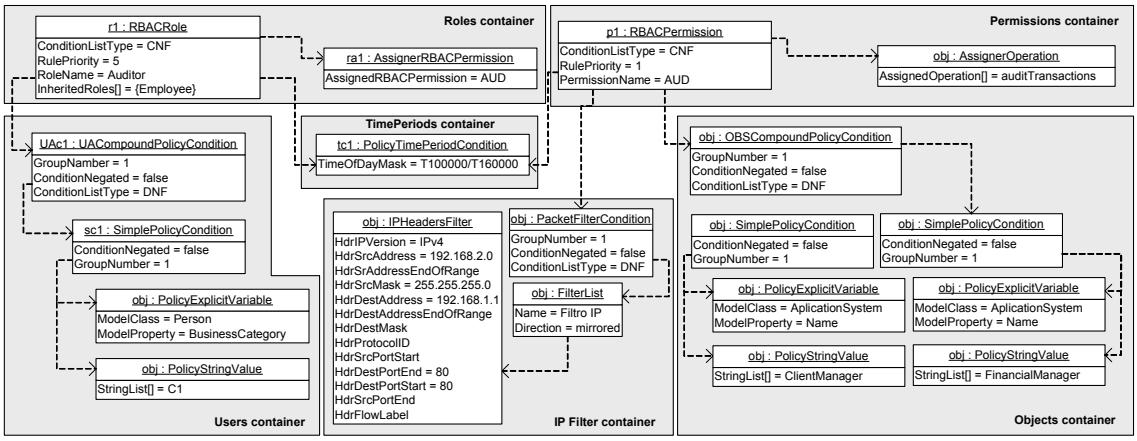


**Figura 4.2 – Exemplo da Aplicação do Modelo**

O papel apresentado, *RBACRole*, foi identificado como "Auditor" através do atributo *RoleName*. O atributo *InheritedRoles* foi utilizado como forma de representar o modelo hierárquico previsto no RBAC (*Hierarquical RBAC*) e corresponde à referência do papel RBAC herdado. Neste caso, o papel RBAC "Auditor" herda as atribuições do papel RBAC "*Employee*". O relacionamento entre usuários e papéis RBAC (UA) – para o papel "*Auditor*" – é obtido a partir da composição de condições simples baseadas na classe *PolicyExplicitVariable*. Esta variável explícita permite criar condições baseadas em objetos do modelo CIM. Neste caso, o papel "Auditor" é associado a todos os usuários que possuam o atributo *BusinessCategory* com valor igual a "C1". Este relacionamento (UA) possui, ainda, uma restrição de tempo, representada pela instância da classe *PolicyTimePeriodCondition*, que define sua atividade entre 10:00 e 16:00. O papel "Auditor" possui um relacionamento entre papéis e permissões (PA) com a permissão "AUD". Esta permissão define que a operação "*auditTransactions*" pode ser executada sempre que as condições, representadas pelas instâncias de *OBSCompoundConditions*, *PacketFilterCondition* e *PolicyTimePeriodCondition*, forem satisfeitas (verdadeiras), característica da associação definida como CNF. Para que *OBSCompoundConditions* seja satisfeita é necessário que uma das duas condições simples a ela associada seja verdadeira (associação DNF). Estas condições simples correspondem a instâncias de objetos CIM relativos a aplicações

(*ApplicationSystem.Name=ClientManager* OR *ApplicationSystem.Name=FinantialManager*). *PacketFilterCondition* refere-se a uma condição associada a um filtro IP e, no caso deve ter como origem a rede 192.168.2.0/24, destino o *host* com endereço 192.168.1.1 e porta 80, enquanto que *PolicyTimePeriodCondition* refere-se a um filtro de tempo e, deve corresponder a um horário entre 10:00 e 16:00.

É importante, a partir do modelo apresentado, fazer distinção entre a classe *RBACRole* e seus atributos relacionados, *RoleName* e *InheritedRoles*, e o atributo *roles* constante da classe *PolicySet*. A primeira é associada diretamente ao modelo RBAC, e faz referência aos papéis que podem ser assumidos pelos usuários em uma determinada sessão. A segunda refere-se ao modelo de políticas e servirá, no escopo deste trabalho, para definir quais dados estarão disponíveis para provisionamento a um determinado cliente. Portanto, neste capítulo, ao ser referenciado o conceito papel (*role*), este estará associado ao modelo de políticas e não ao RBAC.

### 4.2.1 Aplicação do Conceito de Papéis (*Roles*)

Formalmente, um papel pode ser definido como uma característica administrativa específica de um elemento de rede (por exemplo, uma interface). É um seletor para as regras de política que determina a aplicabilidade de uma regra a um elemento de rede em particular [RFC 3198].

Como citado anteriormente, o atributo *Roles* está definido na classe *PolicySet*. Assim sendo, o conceito referente a papéis pode ser estendido a um conjunto de regras (*PolicyRules*) através de herança. Esta característica permite que um grande número de políticas esteja disponível, bastando para tal, que os papéis correspondentes sejam selecionados e/ou combinados. É possível aplicar um conjunto de regras associado a um único papel ou associar papéis como forma de composição da aplicação de regras.

De forma geral, a literatura relacionada a políticas utilizando o conceito de papéis, associa, normalmente, esses papéis a interfaces de diversos dispositivos. Assim, como forma de exemplificar a utilização de papéis, será apresentado a seguir um cenário que retrata exatamente esta situação [RFC 3318].

Por suposição um dispositivo qualquer possui três interfaces associadas a papéis na forma descrita na tabela a seguir.

**Tabela 4.1 – Associação Interface Papel**

| Interface | Papel |
|---|---|
| IF1 | Financeiro |
| IF2 | Financeiro |
| IF3 | Gerência |

Da mesma forma existe um servidor de políticas (PDP) que possui duas políticas definidas.

**Tabela 4.2 – Associação Política Regra Associada**

| Política | Regra Associada |
|---|---|
| P1 | Pacotes de rede com origem no segmento de rede definido para o departamento Financeiro (papel Financeiro) devem ser marcados com DSCP 5. |
| P2 | Pacotes de rede com origem no segmento definido para o departamento Gerência (papel Gerência) devem ser marcados com DSCP 6. |

Assim sendo, o PEP responsável por controlar o dispositivo que possui os dados constantes na tabela 4.1 informa ao PDP estas características e, a partir dos papéis associados, carrega o dispositivo com as regras definidas na tabela 4.2. O resultado desta operação associa a regra definida à interface resultando na tabela a seguir.

**Tabela 4.3 - Associação Interface Regra**

| Interface | Regra |
|---|---|
| IF1 | Marcar todos os pacotes vindos do segmento relativo ao departamento Financeiro com DSCP 5 |
| IF2 | Marcar todos os pacotes vindos do segmento relativo ao departamento Financeiro com DSCP 5 |
| IF3 | Marcar todos os pacotes vindos do segmento relativo ao departamento Gerencia com DSCP 6 |

A possibilidade de associação de papéis permite uma maior flexibilidade do sistema. É necessário, nestes casos, que haja um mecanismo no PDP capaz de tratar os possíveis conflitos permitindo que as políticas aplicadas sejam feitas de forma coerente. Como ilustração desta funcionalidade, considerando o exemplo anterior, à interface IF2 foi adicionado o papel Gerência passando a mesma a corresponder ao conjunto "Financeiro + Gerencia". Desta forma o PDP deve prever um possível conflito entre as duas regras (associadas a cada papel), seja através de algoritmos específicos, seja através de novas regras que correspondam à associação feita. Assim, o PDP pode determinar, por exemplo, que a regra aplicada ao papel Gerência tem precedência sobre o papel Financeiro ou ainda que existe uma terceira política P3 associada aos papéis "Financeiro + Gerência" que especifique que pacotes sobre esta característica devem ser marcados com DSCP 7 .

**Tabela 4.4 - Associação Política Regra Associada**

| Política | Regra Associada |
|---|---|
| P3 | Pacotes de rede com papeis definido como "Financeiro + Gerencia" marque com DSCP 7 |

Tendo como base o modelo de informação proposto, associado à utilização do conceito de papéis é possível definir diversos contextos sobre os quais as políticas serão aplicadas.

## 4.3 Visão Geral

A partir da exposição relacionada à composição e utilização de papéis, pode-se expandir mais ainda sua utilização, relacionando-a não mais a uma interface, mas sim a um servidor ou serviço. Assim sendo, considere um conjunto de políticas, específicas ao modelo de informação RBAC apresentado [Nabhen 03], armazenadas sob um repositório de acordo com a figura a seguir.

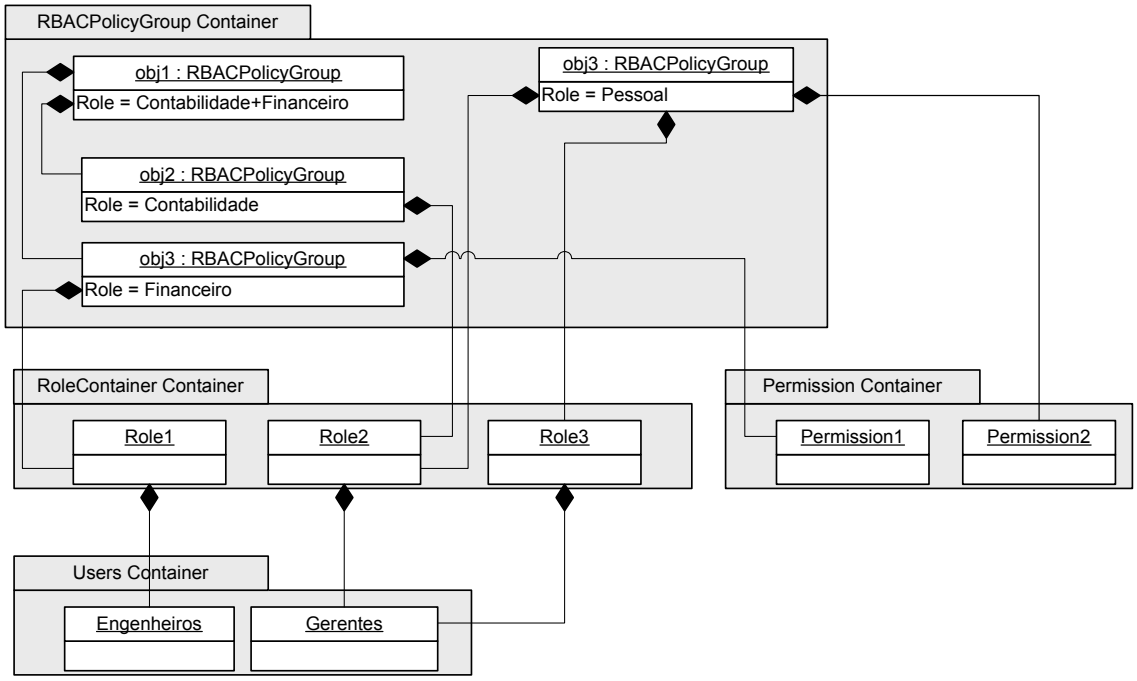


**Figura 4.3 – Exemplo de Agrupamento de Roles e Utilização de Contêineres Reutilizáveis**

A figura apresentada corresponde a uma simplificação do modelo e tem como objetivo mostrar a possibilidade de reutilização de políticas.

A partir do objeto *RBACPolicyGroup,* que possui o atributo *role* definido com o valor "Contabilidade+Financeiro", podemos verificar o agrupamento de dois outros objetos de mesmo tipo (*RBACPolicyGroup),* que tem o atributo *role* definido como "Contabilidade" e "Financeiro", respectivamente. Estes grupos estão conectados a contêineres que correspondem à representação do controle de acesso RBAC e também possuem a característica de serem reutilizáveis. Desta forma, os dois grupos, identificados através do atributo *role*, "Contabilidade" e "Financeiro", compartilham os mesmos contêineres (*Roles container, Users container, TimePeriods container,*

*Permissions container e Objects container*) para composição de políticas RBAC. O grupo, identificado com o atributo *role*, "Pessoal", compartilha os contêineres *Roles container, Users container e TimePeriods container*. Porém, tem seus próprios contêineres *Permissions container e Objects container*. O grupo identificado pelo atributo com valor "Etc" possui todos os contêineres de forma exclusiva. Pela figura, pode-se, então, observar a possibilidade de reutilização dos diversos contêineres envolvidos no modelo, assim como a possibilidade de composição de papéis de política a partir de agrupamento de *RBACPolicyGroup.*

Considere, também, um ambiente de rede composto por servidores de aplicação, responsáveis por diversos aplicativos utilizados em uma corporação, que necessitem de um sistema de controle de acesso eficiente de acordo com a proposição contida no RBAC.

Como forma de composição do cenário, consideremos a existência do Servidor1 responsável por aplicativos específicos da área referente a Pessoal da corporação, e do Servidor2 responsável por aplicativos Financeiros e Contábeis. Da mesma forma que a apresentada na seção 4.2.1, a associação dos servidores às políticas específicas dá-se através dos papéis por eles representados. Assim, ao Servidor 1 está associado o papel "Pessoal" enquanto que ao Servidor 2 estão associados os papéis "Contabilidade + Financeiro".

**Tabela 4.5 - Servidor de Aplicação e Papéis**

| Servidor | Papéis |
|---|---|
| Servidor1 | Pessoal |
| Servidor2 | Contabilidade + Financeiro |

Conceitualmente, somente a definição representada na tabela acima, seria suficiente para determinar a associação entre servidores e papéis relacionados, porém, para implementação, de fato, destes inter-relacionamentos, faz-se necessária a agregação de outros componentes, assim, consideremos o modelo de informação implementado de forma a controlar aplicações específicas disponíveis em servidores de aplicação de acordo com a figura a seguir:

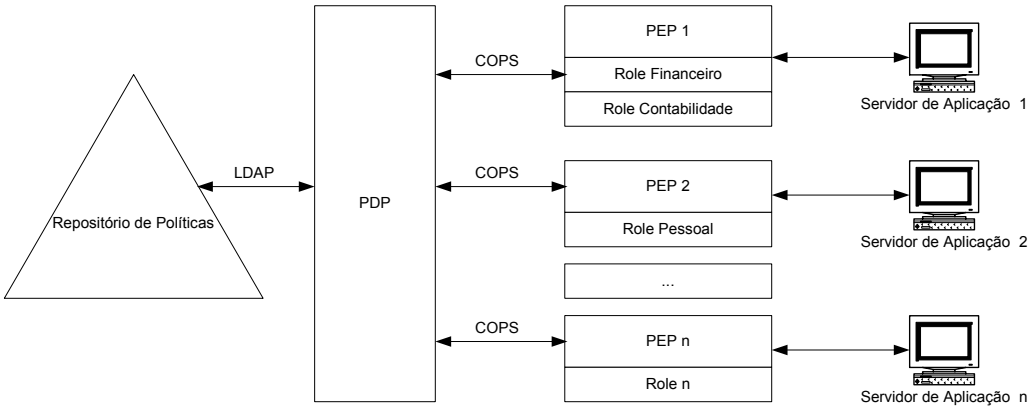


**Figura 4.4 – Modelo de Implementação de Políticas**

Conforme se pode observar, a figura apresenta uma implementação clássica do modelo de políticas. Cada elemento desta implementação será detalhado a seguir.

### 4.3.1 Repositório de Políticas

O repositório de políticas, como o próprio nome indica, tem como função armazenar todos os objetos necessários à representação de políticas relacionadas ao modelo. Neste trabalho adotou-se como repositório padrão o LDAP [RFC 1777; RFC 2251] e sua implementação é feita através dos conceitos contidos tanto no PCLS [RFC 3703] como no PCELS [Reyes 04], já apresentados anteriormente.

A representação das políticas será de acordo com a implementação sugerida pelo RBPIM [Nabhen 03]. Esta composição permite, a partir da adoção de parâmetros equivalentes, a comparação entre as abordagens *outsourcing*, utilizada pelo RBPIM, e *provisioning* sugerida por este trabalho.

### 4.3.2 PDP – Servidor de Políticas

O servidor de políticas PDP é responsável pela decisão sobre a aplicação de políticas no ambiente controlado [RFC 2753]. No modelo, além da decisão relativa à aplicação de políticas, cabe a ele também o controle de estados dos diversos clientes conectados.

Este trabalho definiu o PDP de forma genérica, abordagem essa que permite a sua utilização para controle de uma diversidade de tipos de clientes, bastando para tal, ter acesso a informações relativas àquele tipo específico. Essa característica pôde ser obtida

através da representação dos diversos tipos de cliente a partir de arquivos modelo em XML. Esses arquivos definem totalmente os clientes, servindo como base para que o servidor de políticas possa estabelecer um controle de estados dos clientes conectados.

Ao ser iniciado, o servidor de políticas carrega em memória informações a partir de um arquivo de configuração em XML. A estrutura deste arquivo é apresentada a seguir:

```
<Pdp>
   <Config>
         <PdpId>PdpTeste</PdpId>
         <PdpPort>3288</PdpPort>
         <NextPdpAddr>127.0.0.1</NextPdpAddr>
         <NextPdpPort>3288</NextPdpPort>
         <Ka>10</Ka>
         <SupportedPep>4001 4002 5000 8000</SupportedPep>
         <AuthorizedPep>PepTeste RbacPep Pep2 Pep3 Rbac</AuthorizedPep>
         <LdapServer>localhost</LdapServer>
         <LdapServerPort>389</LdapServerPort>
         <LdapBindDn>cn=Manager</LdapBindDn>
         <LdapPasswd>secret</LdapPasswd>
         <LdapDomain>DC=Domain,DC=com</LdapDomain>
   </Config>
   <ActivePep>
   </ActivePep>
</Pdp>
```

**Figura 4.5 – Arquivo de Configuração XML**

A seção *Config* é responsável pelos dados de configuração do servidor e é composta por entradas que serão descritas a seguir:

- *PdpId* – Corresponde à identificação do servidor de políticas
- *PdpPort* – Define a porta TCP em que o servidor de políticas estará respondendo a requisições. De acordo com a norma COPS [RFC 2748] esta porta deve ser definida como 3288.
- *NextPdpAddr* – Define o endereço do próximo servidor de políticas. Este campo é utilizado quando o servidor recebe uma solicitação e não tem capacidade para prover as políticas necessárias. Dessa forma, encaminha para um próximo servidor que pode responder às políticas solicitadas.
- *NextPdpPort* – Define a porta TCP em que o próximo servidor de políticas estará respondendo.
- *Ka* – Define o intervalo de tempo sob o qual será gerada randomicamente uma mensagem de controle que tem como finalidade informar o status de funcionamento do servidor de políticas.

- *SupportedPep* – Apresenta uma lista com todos os tipos de clientes PEP suportados por este servidor. Clientes que não estejam nesta lista não têm permissão para se conectarem e são encaminhados ao próximo servidor de políticas.
- *AutorizedPep* – Apresenta uma lista com todas as identificações de clientes PEP que têm permissão para se conectar a este servidor de políticas. Da mesma forma que a entrada anterior, clientes que não pertençam a esta lista não têm sua conexão permitida.
- *LdapServer* – Identifica o servidor LDAP responsável por armazenar as políticas necessárias ao PDP.
- *LdapServerPort* – Determina a porta TCP em que o serviço LDAP estará respondendo
- *LdapBindDn* – Determina o *login,* conta com permissão para realizar operações de leitura sobre o diretório, que será utilizado para acesso ao servidor LDAP.
- *LdapPasswd* – Determina a senha de acesso utilizada pela chave constante do campo anterior
- *LdapDomain* – Determina o domínio em que se fará o acesso no servidor LDAP.

A seção *ActivePep* é responsável por armazenar os estados dos diversos PEP's que venham a se conectar ao PDP. Nesta seção estarão referenciados todos os PEP's conectados de acordo com seu tipo e identificação e cada uma dessas referências corresponderão a uma cópia fiel da PIB ativa no cliente. Desta forma, é possível ao PDP estabelecer um controle efetivo sobre cada um dos PEP's, sendo capaz de determinar quais políticas estão aplicadas a quais PEP's. As características da PIB serão descritas em seção própria neste mesmo capítulo.

Após a carga dos dados referentes à configuração, o servidor – a partir deste momento, operacional – passa a aceitar conexões dos diferentes PEP's através de um *socket* específico que atende na porta TCP definida no arquivo de configuração.

Ao conectar-se ao servidor de políticas o cliente solicita uma autorização de conexão. Nesta solicitação informa seu tipo e identificação. De posse destas informações o PDP verifica a existência dos dados em suas tabelas, seleciona o arquivo referente àquele tipo de cliente, adicionando todas as informações constantes do arquivo

modelo ao conjunto de informações específicas na seção *ActivePep*. É, então, retornada uma mensagem ao PEP autorizando sua conexão ao PDP. A cada PEP é permitida uma única conexão ao servidor. Esta diretiva garante que não haja duplicidade de entradas na seção *ActivePep*. A seguir é apresentada a visão do conjunto de informações a partir da conexão de um PEP do tipo 0x4002 (pep4002) identificado como Pep2 ao PDP identificado como PdpTeste.

```
<Pdp>
   <Config>
         <PdpId>PdpTeste</PdpId>
         <PdpPort>3288</PdpPort>
         <NextPdpAddr>127.0.0.1</NextPdpAddr>
         <NextPdpPort>3288</NextPdpPort>
         <Ka>10</Ka>
         <SupportedPep>4001 4002 5000 8000</SupportedPep>
         <AuthorizedPep>PepTeste RbacPep Pep2 Pep3 Rbac</AuthorizedPep>
         <LdapServer>localhost</LdapServer>
         <LdapServerPort>389</LdapServerPort>
         <LdapBindDn>cn=Manager</LdapBindDn>
         <LdapPasswd>secret</LdapPasswd>
         <LdapDomain>DC=Domain,DC=com</LdapDomain>
   </Config>
   <ActivePep>
         <pep4002>
               <pep>id="Pep2"</pep>
         </pep4002>
   </ActivePep>
</Pdp>
```

**Figura 4.6 – Representação de PEP Ativo**

Operacional, o PDP recebe, então, uma requisição de políticas feita pelo PEP. Esta mensagem contém diversas informações necessárias ao PDP que são utilizadas para que possa decidir sobre o conjunto de políticas aplicáveis ao PEP. Uma dessas informações diz respeito aos papéis suportados pelo PEP. Através desta, o PDP prepara uma solicitação de políticas a partir de filtros [RFC 2254] e encaminha ao servidor LDAP. O servidor LDAP devolve então um conjunto de políticas aplicáveis. Essas políticas são formatadas, de acordo com as classes e atributos suportados – informação essa obtida também na mensagem de requisição – passando a compor a mensagem de decisão que será encaminhada ao PEP para que seja aplicada.

### 4.3.3 PEP – Cliente de Políticas

Cada cliente de política PEP é definido através de um valor, utilizado pelo protocolo COPS, que recebe o nome de *Client-Type*. Este valor define as características relativas à aplicação de políticas àquele cliente. Tipos de cliente definidos pelos valores compreendidos entre 0x0001 e 0x3FFF são reservados e registrados no IANA e têm seu comportamento e aplicabilidade descrita em documentos específicos de extensão COPS. Os tipos definidos entre 0x4000 e 0x7FFF são reservados para clientes não padronizados sendo permitido sua utilização para uso privado. Os tipos compreendidos entre 0x8000 e 0xFFFF são para tipos padronizados sem, no entanto, a necessidade de documentos específicos que definam seu funcionamento [RFC 2434]. Considerando o exposto e a não existência de um tipo padronizado de cliente para aplicação em modelos RBAC, este trabalho assumiu o *client-Type* definindo-o como 0x4002.

Da mesma forma que o PDP, este trabalho definiu um modelo de cliente, também genérico, que pode ser configurado possibilitando sua utilização sob outra abordagem. O enfoque deste trabalho é o modelo RBAC, porém outras áreas podem ser atendidas mediante adaptação do modelo.

No modelo *outsourcing*, o PEP funciona apenas como interface, encaminhando os eventos ocorridos ao PDP para que as políticas sejam decididas e só então as aplica. No modelo provisionamento, porém, tem importância fundamental, pois, além de armazenar localmente todas as políticas necessárias ao seu funcionamento, tem capacidade para decidir como aplicá-las, sem a necessidade de troca de informação entre as entidades PEP-PDP.

De forma semelhante ao PDP, o PEP ao ser iniciado, carrega em memória sua configuração a partir de um arquivo definido em XML. O arquivo de configuração do PEP conta com três seções: Configuracao, Pib e ControleSessao e pode ser visualizado a seguir:

```
<Pep>
        <Configuracao>
                <PepId>Pep2</PepId>
                <PdpAddr>localhost</PdpAddr>
                <PdpPorta>3288</PdpPorta>
                <ClientType>4002 </ClientType>
                <MaxHops>3</MaxHops>
        </Configuracao>
        <Pib>
        </Pib>
        <ControleSessao>
        </ControleSessao>
</Pep>
```

**Figura 4.7 – Configuração PEP XML**

A seção Configuracao contém dados que definirão o funcionamento e comportamento do PEP. A seguir serão apresentados os campos e suas funções:

- *PepId* – Corresponde à identificação do PEP.
- *PdpAddr* – Corresponde ao endereço do servidor de políticas ao qual o cliente deve conectar-se para obter os dados necessários ao seu funcionamento.
- *PdpPorta* – Corresponde ao endereço da porta TCP em que o PEP estará respondendo.
- *ClientType* – Corresponde ao tipo definido para o cliente. Neste caso 0x4002.
- *MaxHops* – Este campo tem como função definir o número máximo de buscas por um servidor de políticas, capaz de atender a esse tipo de cliente. Assim, um PDP recebe uma solicitação de conexão de um PEP e caso não possa atendê-la, encaminha esse PEP a um outro servidor de políticas. Um contador interno do PEP registra esses saltos, limitando-os ao máximo previsto no valor definido no campo. Este mecanismo garante que não ocorra um *loop* na busca por um servidor de políticas, reportando erro ao se alcançar o limite definido.

A seção Pib corresponde à base de informações dos dados referentes às políticas destinadas ao PEP. A seção ControleSessao tem como função registrar as sessões ativas para os diversos usuários de acordo com as definições do modelo RBAC. Essas seções serão abordadas ainda neste capítulo.

Após sua inicialização o PEP encaminha ao PDP uma solicitação de conexão informando seu tipo e identificação, dados esses retirados de sua configuração. De posse destas informações o PDP retorna a autorização. O PEP, então, formata – a partir de dados contidos na PIB – uma requisição contendo os papéis desempenhados e suas capacidades, na forma de classes e atributos suportados, informações essas retiradas do arquivo de configuração, que são então enviadas ao PDP, que avalia e obtém as políticas necessárias ao PEP, retornando-as na forma de uma decisão de política. Ao receber essa decisão, o cliente PEP decodifica e carrega na PIB local.

A partir da carga da PIB, o PEP está operacional podendo aplicar as políticas para que foi configurado sem a necessidade de nova busca de informações no PDP. Sob o ponto de vista do controle de acesso baseado em papéis, RBAC, após a carga, a PIB dispõe de dados como usuários, papéis RBAC, separações dinâmicas, permissões de acesso e ações permitidas que possibilitam operacionalizar todas as funções previstas pelo modelo.

Para que, efetivamente, o controle de acesso possa ser aplicado, o PEP disponibiliza uma API através da qual há a interação entre o PEP e as aplicações cliente das políticas. Assim, a aplicação passa um conjunto de informações ao PEP que as processa, aplicando as políticas necessárias e devolve o resultado para que a aplicação possa fazer uso.

### 4.3.4 Servidor de Aplicação

O servidor de aplicação é responsável por disponibilizar as diversas aplicações utilizadas por uma corporação. No contexto deste trabalho, sobre as aplicações será efetuado o método de controle de acesso baseado em papéis (RBAC). Assim, as aplicações interagem com o PEP solicitando o acesso a um determinado usuário que desempenha um papel no sistema para efetuar uma ação específica. O PEP pode, então, autorizar ou negar o acesso requisitado.

Para ilustrar essa interação suponhamos a existência de uma aplicação de nome Financeiro, que possui uma função Transferir Valor, que pode ser efetuada por usuários que possuam o papel Gerente. Os usuários que interagem com a aplicação possuem características de acordo com a tabela a seguir.

**Tabela 4.6 – Associação Usuários Papéis Relacionados e Ações Permitidas**

| Usuário | Papéis Relacionados | Ações Permitidas |
|---|---|---|
| Usuário1 | Usuário, Contador | Efetuar Login, Executar Balanço |
| Usuário2 | Usuário, Supervisor, Gerente | Efetuar Login, Transferir Valor, Gerenciar Contas |

A partir da interação desses usuários para efetuar a função Transferir Valor, a aplicação formata a requisição e encaminha ao PEP através de API específica. Para o usuário Usuário1, a ação é negada, enquanto para o usuário Usuário2, a ação é autorizada.

## 4.4 PIB

A PIB é o elemento principal do modelo apresentado, pois a partir dela é possível representar o modelo de informação na forma de provisionamento e, assim, estabelecer o controle de acesso de forma efetiva no próprio PEP.

Como já apresentado em capítulo anterior, a PIB é uma estrutura de dados em forma de árvore, em que os ramos representam as classes de provisionamento e as folhas representam instâncias destas classes [RFC 3084]. Neste trabalho, a implementação da PIB foi possível a partir de uma estrutura concebida em XML que congrega, tanto conceitos apresentados no *framework* PIB – que correspondem aos elementos comuns, necessários a qualquer tipo de cliente sob uma abordagem de provisionamento [RFC 3318] – como uma nova forma de representação e implementação do método de controle de acesso baseado em papéis, obtido a partir do modelo de informação e da norma relativa ao RBAC.

A seguir, é apresentada uma visualização geral do arquivo de configuração do PEP que representa a PIB nomeada como RBAC-PIB.

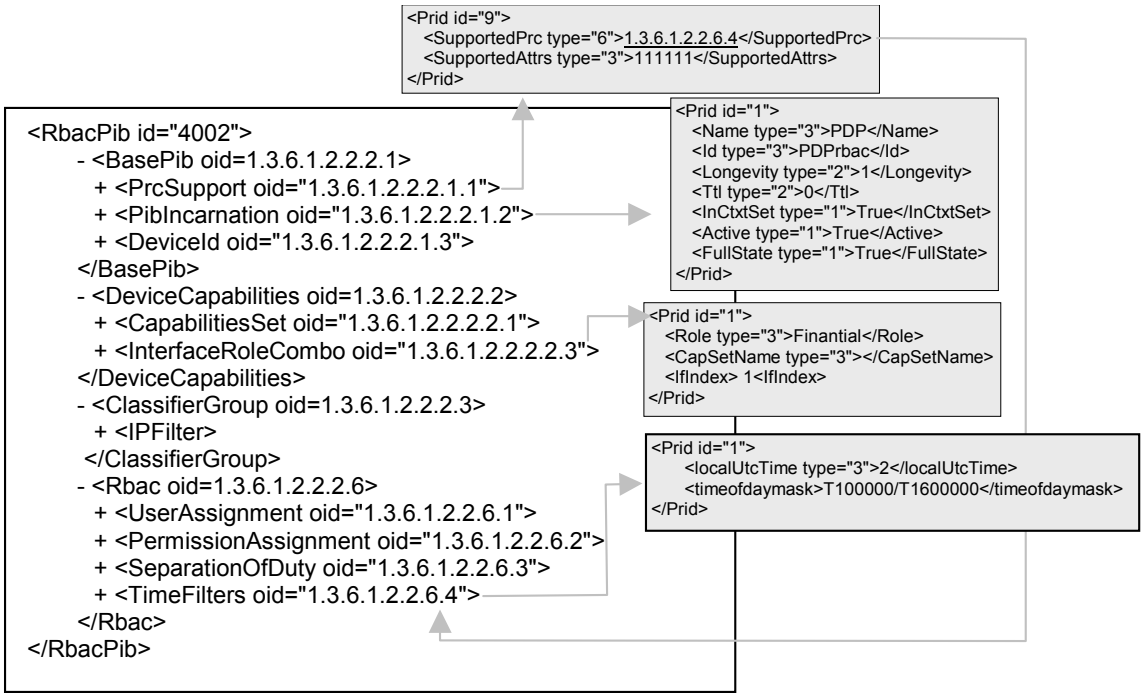


**Figura 4.8 – Representação Estrutura RBAC-PIB em XML**

Nesta visão geral é mostrada a estratégia de mapeamento do modelo RBAC em relação à PIB. Classes PRC são utilizadas a fim de agrupar as informações RBAC, onde, todos os atributos são constituídos a partir de instâncias (PRI) e ponteiros baseados em identificações únicas, *oid* (*object identifiers*), de acordo com as recomendações constantes na ITU-T X.208, estes ponteiros são utilizados para implementar a associação entre as classes. Esta abordagem é necessária para que seja possível a utilização do protocolo COPS-PR para provisionar as informações de política no PEP.

A partir da representação serão descritas, a seguir, os grupos constantes do modelo.

### 4.4.1 Grupo *Base* PIB

Este grupo tem como função descrever as classes e atributos suportados pelo PEP além de definir suas limitações, assim como informar ao PDP as configurações ativas.

```
<BasePib>
        <PrcSupport oid="1.3.6.1.2.2.2.1.1">
        </PrcSupport>
        <PibIncarnation oid="1.3.6.1.2.2.2.1.2">
        </PibIncarnation>
        <DeviceId oid="1.3.6.1.2.2.2.1.3">
        </DeviceId>
</BasePib>
```

**Figura 4.9 – Grupo BasePib**

É composta por três tabelas:

- *PrcSupport* – Esta tabela tem como função descrever as classes e atributos suportados pelo PEP. É composta por três campos. O primeiro definido como *Prid* corresponde ao valor que identifica unicamente a entrada, o segundo definido como *SupportedPrc* armazena a classe suportada na forma de identificadores únicos (OID). Cada classe representada na PIB possui uma identificação única. O terceiro, definido como *SupportedAttrs*, armazena os atributos suportados na forma de uma seqüência de bits (*bitString*). A representação desses atributos é feita utilizando o valor 1 para aqueles suportados e o valor 0 para os não suportados.
- *PibIncarnation* – Esta tabela tem como função determinar ao PDP o contexto que está ativo no PEP naquele momento. É composta por oito campos: *Prid* corresponde ao valor que identifica unicamente a entrada. *Name* correspondente ao nome do PDP que instalou a atual *incarnation*, Id correspondente à identificação da atual *incarnation, Longevity* diz respeito ao tempo de atuação da *incarnation. Ttl* define o tempo em segundos que o PEP ficará ativo em caso de perda da conexão com o PDP. *InCtxtSet* deve ser definido com valor *True* quando utilizado em abordagens provisionamento. *Active* determina se o contexto está ativo ou não. *FullState* quando definido como *True* determina que ao receber uma decisão do PDP a partir de uma requisição de política do PEP

para o PDP esta deve ser implementada na totalidade, descartando os dados já carregados.

- *DeviceId* – Esta tabela contém informações específicas que são utilizadas pelo PDP par facilitar o processo de aplicação de políticas. É composta por quatro campos: *Prid* que define unicamente a instância, *Descr* que corresponde à descrição do PEP, *MaxMsg* que define o tamanho máximo das mensagens COPS-PR suportadas e *MaxContexts* que define o número máximo de contextos permitidos pelo dispositivo.

Devido às características deste trabalho, optou-se por restringir a utilização de múltiplos contextos, estabelecendo como padrão a utilização e definição de um contexto único.

### 4.4.2 Grupo *Device Capabilities*

Este grupo tem como função descrever as características do PEP e as combinações de papéis possíveis para o mesmo.

```
<DeviceCapabilities>
        <CapabilitiesSet oid="1.3.6.1.2.2.2.2.1">
                <Prid id="1">
                        <Name type="3">Politicas RBAC</Name>
                        <Capability type="6">1.3.6.1.2.2.2.2.2</Capability>
                </Prid>
        </CapabilitiesSet>
        <InterfaceRoleCombo oid="1.3.6.1.2.2.2.2.3">
                <Prid id="1">
                        <Roles type="3">PoliticaRBPIM</Roles>
                        <CapSetName type="3">Politicas RBAC</CapSetName>
                        <IfIndex type="3">Unica</IfIndex>
                </Prid>
        </InterfaceRoleCombo>
</DeviceCapabilities>
```

**Figura 4.10 – Grupo DeviceCapabilities**

É composto por duas tabelas:

- *CapabilitiesSet* – Esta tabela descreve o conjunto de capacidades disponíveis nas interfaces do dispositivo. É composta por três campos: *Prid* que identifica unicamente a instância, *Name* que descreve o nome do conjunto de capacidades

devendo ser único e não nulo e *Capability* que corresponde ao identificador da classe e instância (OID) de uma capacidade associada.

- *InterfaceRoleCombo* – Esta tabela relaciona interfaces a papéis desempenhados e é composta por quatro campos: *Prid,* que identifica unicamente a instância, *Roles* que relaciona os papéis associados à interface, *CapSetName* que relaciona a uma capacidade e *IfIndex* que define o índice da interface.

### 4.4.3 Grupo *Classifier*

O grupo *Classifier* tem como função permitir a criação de filtros e classificadores que atuem sobre o protocolo IP.

```
<ClassifierGroup>
        <IPFilter oid="1.3.6.1.2.2.2.3.2">
                <Prid id="1">
                        <Negation type="1">0</Negation>
                        <AddrType type="3">IPV4</AddrType>
                        <DstAddr type="3">192.168.1.0</DstAddr>
                        <DstPrefixLength type="3">24</DstPrefixLength>
                        <SrcAddr type="3"></SrcAddr>
                        <SrcPrefixLength type="3"></SrcPrefixLength>
                        <Dscp type="2"></Dscp>
                        <FlowId type="2">3</FlowId>
                        <Protocol type="3"></Protocol>
                        <DstL4PortMin type="2"></DstL4PortMin>
                        <DstL4PortMax type="2"></DstL4PortMax>
                        <SrcL4PortMin type="2"></SrcL4PortMin>
                        <SrcL4PortMax type="2"></SrcL4PortMax>
                </Prid>
        </IPFilter>
</ClassifierGroup>
```

**Figura 4.11 – Grupo ClassifierGroup**

É composto por uma única tabela:

- *IPFilter* – Define um filtro para pacotes IP. É composto por quatorze campos: *Prid* que define unicamente uma instância, *Negation* que corresponde a um *BaseFilter* e define se o filtro será aplicado ou não, *AddrType* corresponde ao tipo do endereço contido no pacote *IPv4* ou *IPv6*, *DStAddr* corresponde

endereço de destino, *DstPrefixLength* corresponde ao tamanho relativo à máscara do endereço de destino, *SrcAddr* corresponde ao endereço de origem do pacote, *SrcPrefixLength* corresponde ao tamanho relativo à máscara do endereço de origem, *Dscp* corresponde ao valor que o *dscp* deve corresponder no pacote, *FlowId* corresponde ao identificador do fluxo num pacote *IPv6*, *Protocol* corresponde ao nome do protocolo constante do pacote, *DstL4PortMin*, *DstL4PortMax*, *SrcL4PortMin*, *SrcL4PortMax* definem os intervalos para os valores relativos às portas de origem e destino de um pacote camada 4.

### 4.4.4 Grupo RBAC

O grupo RBAC corresponde à representação em XML de todo o modelo correspondente ao controle de acesso baseado em papéis. É composto por tabelas que representam Usuários, Papéis RBAC, Separações Dinâmicas, Permissões associadas a ações e Filtros de tempo. O conceito de separações estáticas de papéis RBAC, descrito no *constrained* RBAC, é avaliado no processo de busca de políticas realizado pelo PDP não sendo necessária sua representação na forma de tabela.

```
<Rbac oid="1.3.6.1.2.2.6">
        <UserAssignment oid="1.3.6.1.2.2.6.1">
                <Users oid="1.3.6.1.2.2.6.1.1">
                </Users>
                <Roles oid="1.3.6.1.2.2.6.1.2">
                </Roles>
                <UserRoles oid="1.3.6.1.2.2.6.1.3">
                </UserRoles>
                <RoleTimeFilters oid="1.3.6.1.2.2.6.1.4">
                </RoleTimeFilters>
        </UserAssignment>
        <PermissionAssignment oid="1.3.6.1.2.2.6.2">
                <Objects oid="1.3.6.1.2.2.6.2.2">
                </Objects>
                <Permissions oid="1.3.6.1.2.2.6.2.1">
                </Permissions>
                <RolePermissions oid="1.3.6.1.2.2.6.2.3">
                </RolePermissions>
                <PermissionIPHeaderFilters oid="1.3.6.1.2.2.6.2.4">
                </PermissionIPHeaderFilters>
                <PermissionTimeFilters oid="1.3.6.1.2.2.6.2.5">
                </PermissionTimeFilters>
        </PermissionAssignment>
        <SeparationOfDuty oid="1.3.6.1.2.2.6.3">
                <DSD oid="1.3.6.1.2.2.6.3.1">
                </DSD>
                <DSDEntries oid="1.3.6.1.2.2.6.3.2">
                </DSDEntries>
        </SeparationOfDuty>
        <TimeFilters oid="1.3.6.1.2.2.6.4">
        </TimeFilters>
</Rbac>
```

**Figura 4.12 – Grupo RBAC**

A seguir, serão apresentados os subgrupos que compõem a representação do grupo RBAC bem como cada uma das tabelas que os compõem, descrevendo seus componentes:

- Subgrupo *UserAssignment* – Tem como função agrupar tabelas responsáveis por representar a associação UA (*User Assignment*), definida no modelo RBAC. Os papéis RBAC associados já estão livres de separação estática, conforme definido no *constrained* RBAC, e relacionados de acordo com a hierarquia de papéis definida no *hierarchical* RBAC. É composto pelas tabelas:
    - *Users* – Esta tabela tem como função armazenar os usuários que utilizam o sistema. É composta pelos seguintes campos: *Prid* que identifica unicamente cada instância, *uid* que corresponde à identificação do usuário
    - *Roles* – Esta tabela tem como função armazenar os papéis RBAC, identificando a prioridade para verificação de precedência de papéis RBAC. É composta dos seguintes campos: *Prid* que identifica unicamente cada instância, *name* que determina o nome do papel RBAC e *priority* que define o grau de precedência em que o papel deve ser aplicado.
    - *UserRoles* – Esta tabela tem como função relacionar usuários e papéis RBAC. É composta pelos campos *Prid* que identifica unicamente cada instância, *uid* que corresponde ao *oid* relativo ao usuário, constante da tabela *Users* e role que corresponde ao *oid* relativo ao papel RBAC, constante da tabela Roles.
    - *RoleTimeFilter* – Esta tabela tem como função associar papéis RBAC a filtros de tempo. É composta pelos campos *Prid* que identifica unicamente cada instância, *timefilter*, que corresponde ao *oid* relativo ao elemento pertencente à tabela *TimeFilters* e pelo campo role que corresponde ao *oid* relativo ao papel RBAC.

A representação deste grupo pode ser visualizada a partir da figura a seguir, bem como o inter-relacionamento das tabelas envolvidas.

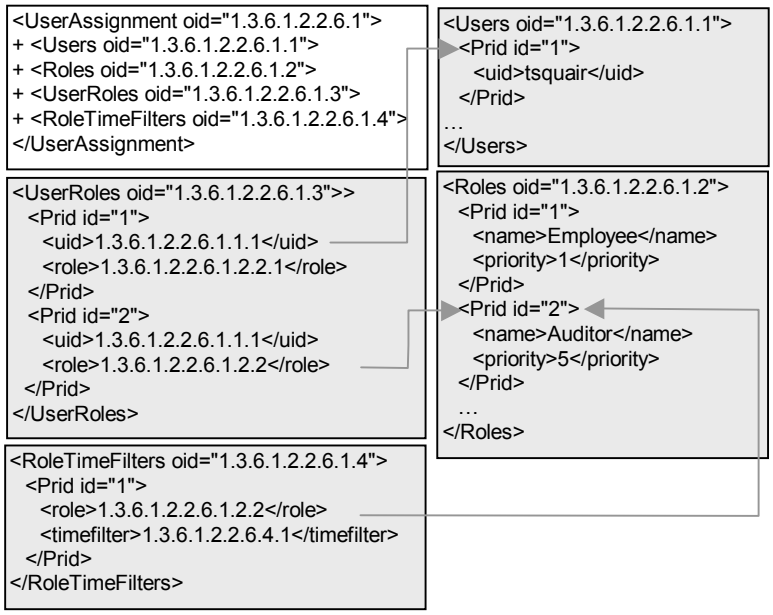


**Figura 4.13 – RBAC-PIB Grupo *UserAssignment***

- Subgrupo *PermissionAssignment* – Este grupo tem como função reunir os elementos necessários para definir o relacionamento PA (*Permission Assignment*), definido no modelo RBAC. É composto pelas tabelas:
    - *Objects* – Esta tabela tem como função representar o objeto sobre o qual será aplicada uma política de controle de acesso. É composta pelos campos *Prid,* que identifica unicamente cada instância, e *expression* que corresponde à expressão que representa um objeto CIM sobre o qual será aplicado o controle.
    - *Permissions* – Esta tabela tem como função representar as permissões associadas a um determinado objeto. É composta pelos campos, *Prid,* que identifica unicamente cada instância, *object* que corresponde ao *oid* relativo ao objeto constante da tabela *Objects* e pelo campo *permission* que corresponde a uma string representando a permissão sobre o objeto.

- *RolePermissions* – Esta tabela corresponde à associação entre papéis RBAC e permissões. É composta pelos campos *Prid* que identifica unicamente cada instância, role que corresponde ao *oid* de um papel RBAC, pertencente à tabela Roles e *permission* que corresponde ao *oid* de uma permissão, pertencente à tabela *Permissions*.
- *PermissionIPHeaderFilters* – Esta tabela tem como função relacionar permissões a filtros IP. É composta pelos campos *Prid* que identifica unicamente cada instância, *permission* que corresponde ao *oid* relativo a uma permissão constante da tabela *Permissions* e *ipfilter* que corresponde ao *oid* relativo ao filtro IP constante da tabela *TimeFilters*.

A representação deste grupo é apresentada na figura a seguir, podendo ser visualizado também o inter-relacionamento entre as tabelas envolvidas.

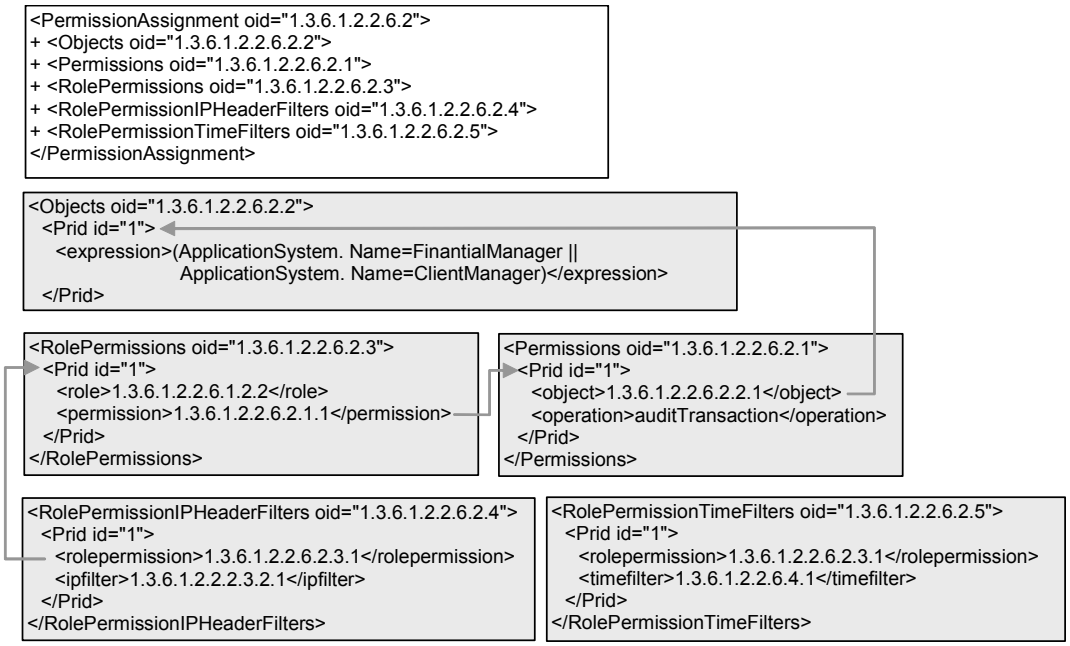


**Figura 4.14 – RBAC-PIB Grupo *PermissionAssignment***

- O subgrupo *SeparationOfDuty* tem como função reunir elementos responsáveis pela separação dinâmica, DSD, definido no modelo RBAC. É composto pelas tabelas:
    - DSD – Responsável por agrupar as diversas entradas relativas ás separações dinâmicas. É composta pelos campos *Prid* que identifica unicamente cada instância e *cardinality* que define a cardinalidade envolvida na separação dinâmica.
    - *DSDEntries* – Tem como função associar papéis RBAC a entradas DSD. É composta pelos campos, *Prid* que identifica unicamente cada instância, *role* que corresponde ao *oid* relativo ao papel RBAC, pertencente à tabela *Roles* e DSD que corresponde ao *oid* relativo a uma entrada de separação dinâmica pertencente à tabela DSD.

A seguir, é apresentada uma figura que corresponde à representação do grupo DSD e também o inter-relacionamento das tabelas constantes no grupo.

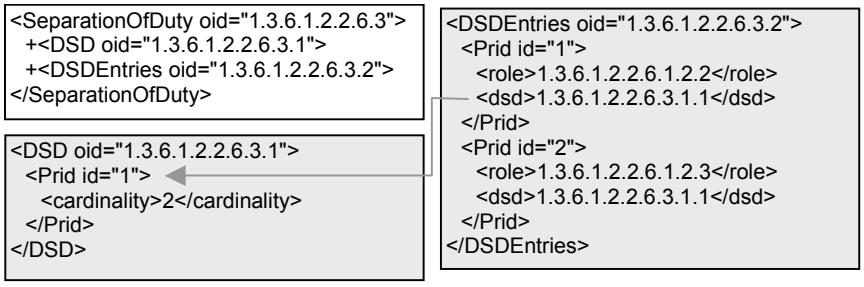


**Figura 4.15 – RBAC-PIB Grupo *SeparationOfDuty***

- A tabela *TimeFilter* não está agrupada sob um item específico e corresponde a um filtro de tempo definido de acordo com a descrição pertencente ao modelo PCIMe. É composta pelos campos, *Prid* que identifica unicamente cada instância, *localutctime* que corresponde à definição de tempo local ou UTC (*universal time clock*), *tpctime* que corresponde a um intervalo de tempo na forma AAAAMMDDTHHMMSS/AAAAMMDDTHHMMSS, *monthofyear* que corresponde ao dia do mês na forma 110011001100 onde 1 representa ativo e 0 inativo correspondendo a cada um dos doze meses do ano, *dayofmonth* que possui a mesma forma definida no campo anterior somente contando com trinta e uma posições relativas a cada um dos dias do mês, *dayofweek* que conta com sete posições correspondentes aos dias da semana e timeofdaymask que corresponde a uma representação na forma THHMMSS/THHMMSS correspondendo a um intervalo de horas do dia.

## 4.5 Conclusão

Este capítulo apresentou o modelo sugerido por este trabalho, que utiliza o conceito de papéis associados a políticas, como forma de possibilitar a aplicação de controle de acesso RBAC, em ambientes distribuídos, de acordo com uma abordagem de provisionamento.

A PIB, parte fundamental para o desenvolvimento deste trabalho, implementada em XML, permite a visualização de forma simples e compreensível, porém completa, do modelo RBAC – *Core*, *Hierarchical, Dynamic Separation of Duty e Static Separation of Duty* – possibilitando ao PEP a aplicação total das políticas relacionadas ao controle de acesso sobre um conjunto de aplicações a ele conectadas.

A forma genérica como foram definidos, tanto o PDP como o PEP, proporcionam grande flexibilidade ao modelo, permitindo sua utilização tanto para provisionamento de políticas de controle de acesso, foco deste trabalho, como para outras abordagens de provisionamento de políticas.

# Capítulo 5

# Arquitetura de Implementação, Algoritmos e API

## 5. 1 Introdução

O capítulo anterior apresentou o modelo de informação com todas as suas características, ressaltando-se a PIB como elemento central do modelo.

Este capítulo tem como função apresentar a arquitetura de implementação para o modelo sugerido no capítulo anterior, destacando os relacionamentos entre as entidades envolvidas, as mensagens trocadas entre as entidades PEP e PDP, os algoritmos e consultas necessárias ao funcionamento do modelo, além da API que faz o relacionamento do modelo com as aplicações externas.

## 5. 2 Arquitetura de Implementação

A arquitetura de implementação proposta neste trabalho, está de acordo com a forma apresentada na figura 4.4, reforçando a utilização dos três componentes básicos do modelo de políticas: o servidor de políticas PDP, o repositório de políticas e o cliente de políticas PEP. Desta forma, a seguir será apresentada a visão geral da arquitetura proposta, ressaltando a abordagem provisionamento, bem como o funcionamento dos seus principais componentes, além da interação existente entre eles.

O servidor de políticas PDP é a entidade responsável por interpretar e distribuir as informações de políticas aos clientes de política PEP. O PEP pode ser considerado como o componente, no cliente de políticas, responsável por estabelecer a comunicação com o PDP e obter as políticas necessárias a seu funcionamento. A forma de comunicação utilizada entre PEP e PDP é implementada através do protocolo COPS-PR, de acordo com a proposição constante na [RFC 3084]. Na figura, pode-se visualizar um único PEP atendido por um único PDP. No entanto, a forma como foi proposta a arquitetura, permite que um mesmo PDP atenda, simultaneamente, a um grande número de PEP's.

O cliente de políticas PEP, de acordo com a figura, pode ser visualizado como um servidor de aplicações responsável por servir a um grande número de clientes. O

protocolo de comunicação utilizado para que as aplicações possam trocar informações, tanto com o PEP quanto com os clientes, não é tratado, definido ou imposto pela arquitetura de implementação, ficando livre sua definição pelos desenvolvedores. A comunicação entre o servidor de aplicações e o modelo de implementação RBAC é obtido a partir de um conjunto de API's disponibilizadas, que seguem as proposições contidas no padrão sugerido pelo NIST. Estas API's serão detalhadas na sessão 5.3.

Como sugerido pelo IETF, tanto informações relativas a política quanto informações relativas ao modelo CIM, podem ser mapeadas sobre um esquema LDAP. Como o repositório LDAP suporta referências externas sobre seu esquema, não há a necessidade que os modelos sejam implementados em um mesmo servidor LDAP. Assim, há a possibilidade da implementação, tanto de políticas como informações do modelo CIM, a partir de outras tecnologias como, por exemplo, XML.

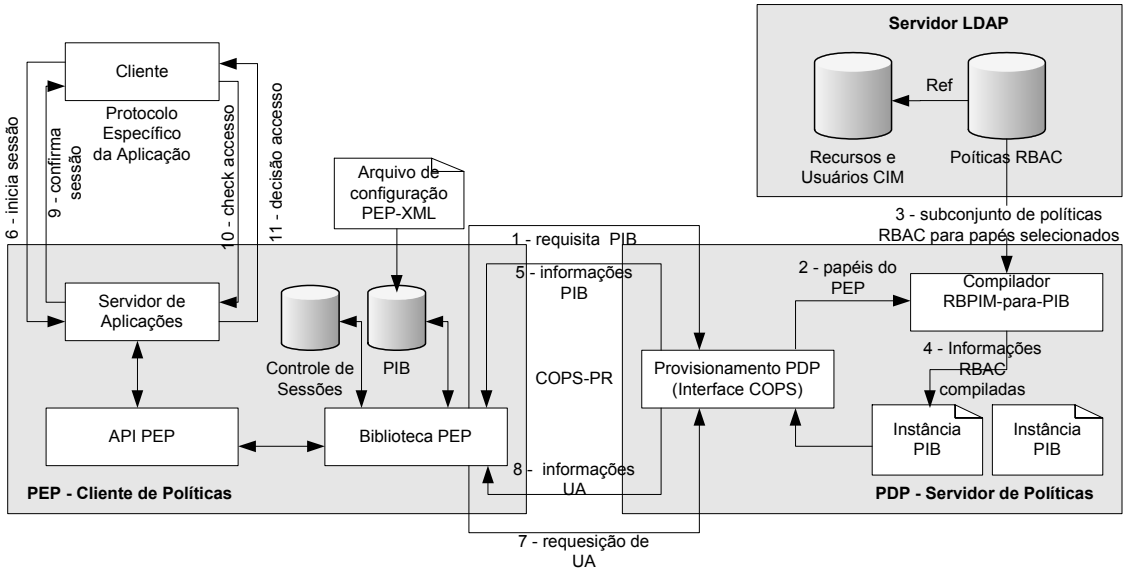


**Figura 5.1- Visão Geral do Modelo de Implementação**

A seqüência típica de eventos relacionados ao provisionamento de política e as decisões efetuadas pelo PEP, são apresentadas na figura e identificadas por números associados a fluxos. Cada um destes eventos será descrito a partir de sua identificação, como forma de explicar o funcionamento da arquitetura proposta.

### 5.2.1 Apresentação da Arquitetura

Após a inicialização, que compreende leitura e carga do arquivo XML de configuração, o PEP estabelece uma conexão COPS-PR com o servidor de políticas e requisita o provisionamento inicial das políticas a ele destinadas. Esta primeira requisição corresponde a uma solicitação de provisionamento total (*full request*), isto é, como resposta a essa requisição, serão devolvidos todos os dados de políticas relevantes ao PEP, exceto o conjunto de usuários (1). Nesta requisição, de acordo com definições do IETF, o PEP fornece uma combinação de papéis a serem utilizados a fim de selecionar um subconjunto de políticas que serão disponibilizados através do PEP às aplicações. Um exemplo desta combinação de papéis pode ser: "Contabilidade" + "Financeiro". Estes papéis são obtidos a partir do arquivo de configuração XML lido na inicialização (2). Ao receber a requisição, o PDP ativa o compilador RBPIM-para-PIB adquirindo o subconjunto de políticas RBAC associadas aos papéis selecionados (3), compilando-as na forma de dados a serem armazenados na PIB (4). O PDP retorna então, os dados a serem armazenados na PIB ao PEP utilizando o protocolo COPS-PR, (5). O PEP, de posse dos dados, interpreta-os, armazenando-os no repositório PIB local. A informação recebida no fluxo (5) não corresponde à totalidade da PIB, uma vez que o mapeamento entre usuários e papéis RBAC (UA – *User Assignment*), não foi provisionado. No entanto, todas as demais informações relevantes ao modelo RBAC tais como definição de papéis RBAC e suas hierarquias, separações dinâmicas (DSD), descrição de permissões e mapeamento entre papéis e permissões (PA – *Permission Assignment*) já foram provisionadas em resposta a esta primeira requisição.

A opção por não provisionar o mapeamento UA na primeira requisição foi uma forma de possibilitar a otimização no desempenho da implementação, pois a ocorrência eventual de um grande número de usuários no repositório poderia implicar em um grande tamanho da PIB a ser provisionada. Assim, optou-se por provisionar o relacionamento UA de forma incremental, a partir da criação de uma nova sessão para um determinado usuário. Ao ser criada uma nova sessão (6) é feita uma consulta à PIB em busca do usuário e caso não o encontre, é ,então, feita uma nova requisição ao PDP, agora, especificamente, um mapeamento UA, utilizando o protocolo COPS-PR (7) A resposta à requisição corresponde a somente um elemento da PIB, correspondendo a um

novo usuário, relativo a um mapeamento UA (8). Após este evento, as requisições de acesso (10) podem ser decididas pelo PEP localmente.

É importante ressaltar que toda a PIB provisionada ao PEP é mantida também em memória no PDP. Isto é necessário devido à característica direcionada a estado (*stateful*) do protocolo COPS-PR e, também, como forma de possibilitar a restauração do PEP em caso de falha ou, ainda, como forma de possibilitar uma atualização incremental nos mesmos moldes do provisionamento do mapeamento UA.

### 5.2.2 Comparação *OutSourcing* / Provisionamento

Como ilustração, é apresentado a seguir um comparativo que mostra, através de um diagrama de seqüência, as interações ocorridas entre PEP e PDP nos modelos *outsourcing* e provisionamento.

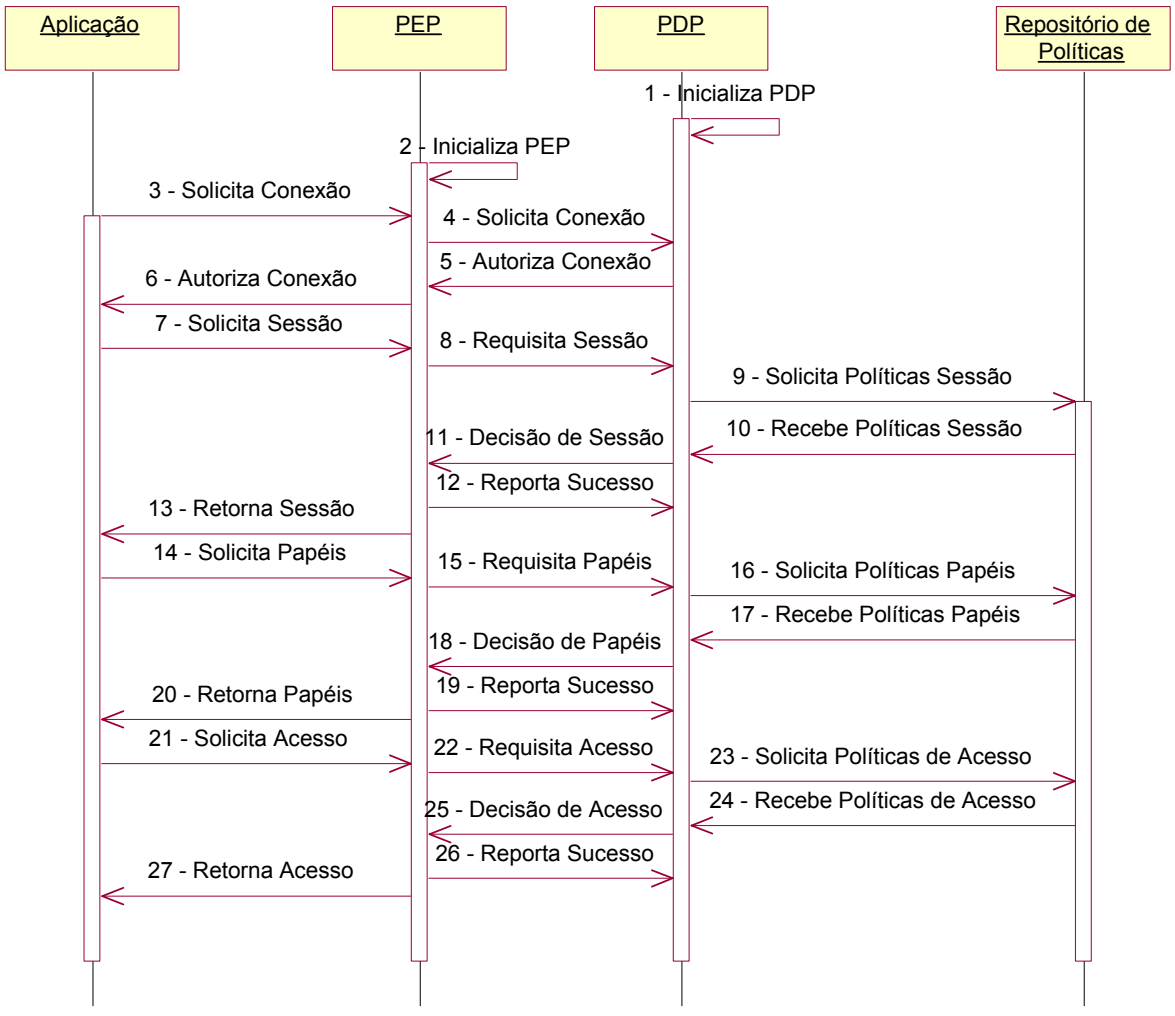


**Figura 5.2 – Diagrama de Seqüência OutSourcing**

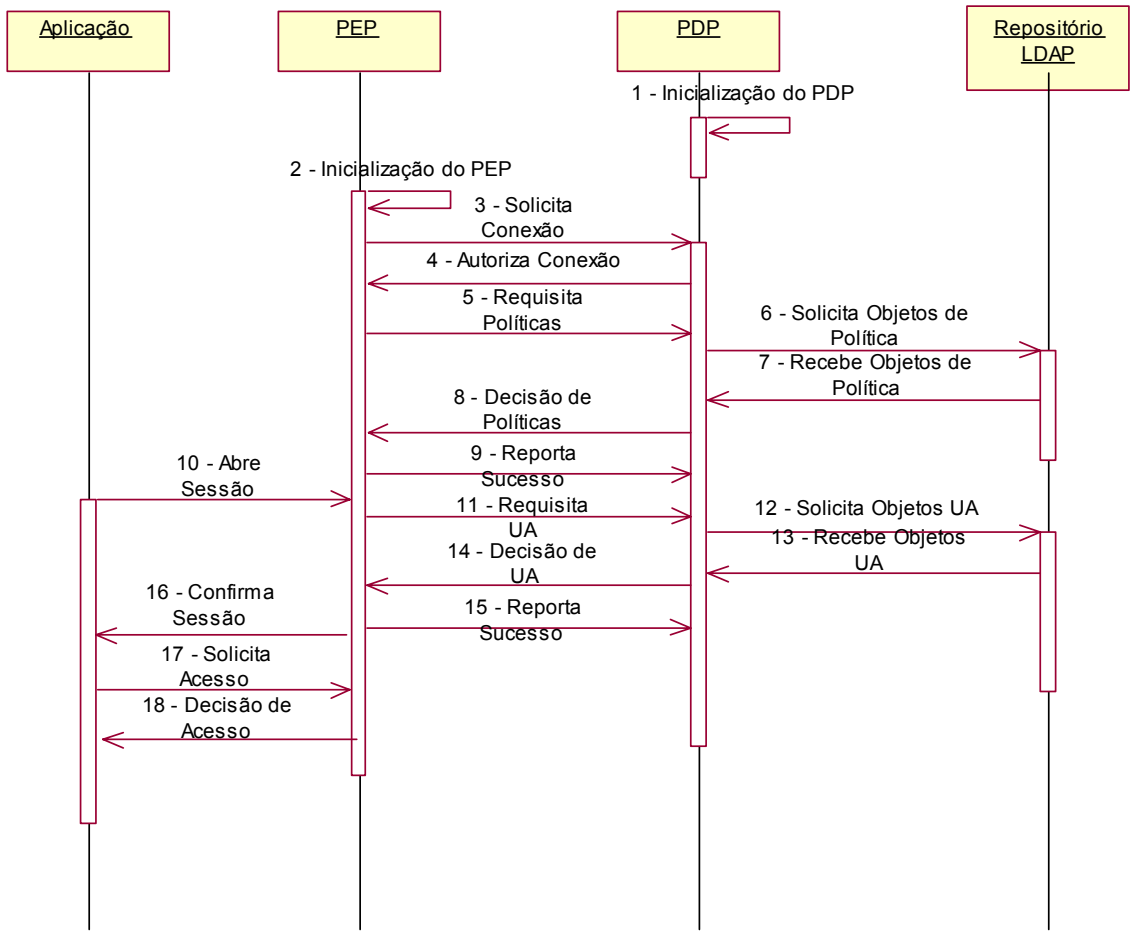


**Figura 5.3 – Diagrama de Seqüência Provisionamento**

Importante observar que no modelo *outsourcing*, os eventos relativos aos fluxos de 7 a 27, repetem-se a cada sessão, necessitando uma maior interação entre os elementos envolvidos Aplicação, PEP, PDP e Repositório de Políticas, enquanto que no modelo provisionamento, após a carga inicial da PIB, só há interação entre todos os elementos no caso da necessidade de provisionamento UA. Assim, se a aplicação solicitar uma nova sessão para um usuário já carregado na PIB, não há a necessidade de troca de informações entre os elementos PEP e PDP e entre PDP e Repositório de políticas, podendo todo o processo ser executado localmente pelo PEP e envolvendo somente este e a Aplicação.

## 5. 3 Implementação PEP e PDP

Nesta sessão serão apresentadas as implementações dos componentes principais da arquitetura, primeiramente o cliente de políticas PEP, e na seqüência, o servidor de políticas PDP.

### 5.3.1 Implementação Cliente de Políticas – PEP

Considerando a característica orientada à aplicação da arquitetura, o PEP é basicamente uma biblioteca de software. Esta característica facilita sua utilização, uma vez que é necessário somente que seja referenciado, sem a preocupação com as características técnicas do modelo RBAC. Desta forma, uma aplicação para utilizar todas as funcionalidades da arquitetura, necessita somente conhecer as API's disponibilizadas, sem a preocupação em relação à interpretação das políticas RBAC contidas na PIB. As API's disponibilizadas na arquitetura, são baseadas na especificação funcional do modelo RBAC definido pelo NIST e constituem-se de cinco funções apresentadas na tabela a seguir.

| API | Atributos de entrada | Atributos de saída | Função da API |
|---|---|---|---|
| *Construtor* | | | Responsável por todo o provisionamento de políticas necessárias ao PEP |
| *RBPEP_CreateSession* | *uid:string* | *session: string roleset[]: string array usessions:int* | Requisitar uma atualização incremental da PIB a partir de uma informação relacionada a um usuário específico. Interpreta a atualização da Pib e retorna à aplicação um conjunto de papéis assinalados ao usuário. |
| *RBPEP_SelectRoles* | *session:string roleset[]:string* | *result:BOOLEAN* | Ativar o conjunto de papéis selecionados para um usuário em uma determinada sessão livres de conflito por separação estática (SSD) e dinâmica (DSD) |
| *RBPEP_CheckAccess* | *session:string operation:string objectfilter:string ipfilter:string* | *result:BOOLEAN* | Verificar se o usuário tem direitos administrativos para executar a operação especificada baseado nos mapeamentos PA, informações e filtros contidos na PIB |
| *RBPEP_CloseSession* | *session:string* | | Encerrar uma sessão do usuário |

**Tabela 5.1 – API's Disponibilizadas pela Implementação**

### *1. Construtor*

Como o protótipo, e em conseqüência a API, foram construídos sob a ótica da orientação a objetos, o construtor é a entidade responsável por toda a inicialização do PEP. Nesta inicialização é estabelecida uma conexão TCP com o PDP e, a partir desta, são requisitadas informações necessárias ao PEP, que serão então provisionadas, utilizando o protocolo COPS-PR.

Grande parte das informações RBAC é transferida ao PEP em decorrência do processamento desta API, contudo, as instâncias (PRI's) correspondentes às classes "*User*" e "*UserAssignement"* não são transferidas nesta primeira fase. Esta característica visa diminuir o tamanho da PIB inicialmente transferida ao PEP, uma vez que nem todos os usuários cadastrados farão uso das aplicações que aquele determinado PEP controla, podendo ser provisionado em tempo futuro, de forma incremental, e de acordo com a necessidade da aplicação.

De forma sintética, esta API pode ser representada pelo algoritmo a seguir:

```
Início
        Solicita abertura de conexão com PDP relacionado na configuração
        Enquanto ((não Autorizado) ou (número de saltos < numero da configuração))
                Solicita abertura de conexão com próximo PDP obtido na mensagem de resposta
        Fim
        Se (número de saltos(hops) = numero da configuração)
                Erro
        Fim
        Recebe Autorização de Conexão
        Solicita ((carga total de Política a partir do papel (role)) ou (conjunto de papéis da
        configuração))
        Recebe Políticas
        Armazena Políticas na PIB
        Se sucesso no armazenar
                Envia report de sucesso
        Senão
                Envia report de falha
        Fim
Fim
```

O número de saltos (*hops*) referenciado no algoritmo, corresponde a uma característica do protocolo COPS-PR que permite o envio do endereço IP do próximo servidor de políticas disponível, caso não possa responder a um conjunto de políticas. Esta informação é passada em uma mensagem *Client-Close* (CC). Desta forma, foi implementado um controle que tem como objetivo evitar que um determinado PEP entre em *loop* no caso de não conseguir estabelecer uma conexão com um servidor PDP.

Como forma de propiciar melhor entendimento, é possível observar cada uma das funções apresentadas no algoritmo anterior sob a forma de um novo algoritmo.

**Algoritmo *createOPN***

Inicialmente é apresentado o algoritmo *createOPN* que tem como função estabelecer a conexão do PEP com o PDP.

**Início**
Obtém a identificação do PEP a partir da tag "Identificação" item "PepId"
Obtém o tipo do PEP a partir da tag "Identificação" item "ClientType"
Cria uma mensagem do tipo OPN solicitando conexão ao PDP informando identificação e tipo
**Fim**

**Algoritmo *readCC***

O algoritmo *readCC* tem como função receber uma mensagem de encerramento de conexão vinda do PDP e obter os dados relativos a outro servidor de políticas ao qual poderá solicitar conexão.

**Início**
Recebe mensagem do PDP
Obtém endereço do próximo servidor de políticas
Obtém porta do próximo servidor de políticas
Executa algoritmo **createOPN** informando o endereço do servidor e porta de conexão
**Fim**

**Algoritmo *readCAT***

O algoritmo *readCAT* tem como função registrar a permissão de conexão emitida pelo PDP. Recebe, ainda, valor relativo ao tempo das mensagens de teste de conexão KA.

**Início**
Obtém valor de KA
Requisita mensagem **createREQ**
**Fim**

**Algoritmo *createREQ***

O algoritmo *createREQ* é responsável por informar ao PDP os papéis, classes e atributos suportados pelo PEP, solicitando o provisionamento total das políticas destinadas ao PEP.

**Início**
Lê documento XML de configuração (PEP.xml)
Obtém os papéis suportados pelo PEP a partir da PRC "InterfaceIndex" item "IfRoles"
Obtém classes suportadas pelo PEP a partir da PRC "PrcSupport" item "SupportedPrc"
Obtém atributos suportados pelo PEP a partir da PRC "PrcSupport" item "SupportedAttrs"
Cria mensagem REQ do tipo total, R-Type = 0, informando os papéis, classes e atributos suportados
**Fim**

**Algoritmo *createREQUSR***

O algoritmo *createREQUSR* tem como função solicitar ao PDP um provisionamento parcial relativo a um determinado usuário.

**Início**
Recebe identificação do usuário
Cria mensagem REQ do tipo usuário, R-Type=0, Atributo *FullState* de *PibIncarnation* = False, informando o usuário
**Fim**

**Algoritmo *readDEC***

O algoritmo *readDEC* recebe as políticas enviadas pelo PDP, decodifica-as, interpreta e, a partir destes dados, compõe a PIB do PEP.

**Início**
Recebe a mensagem de decisão do PDP
Verifica *Handle*
**Se** *Handle* diferente do armazenado
Descarta decisão
**Fim do algoritmo**
Verifica *tipo* da decisão
**Se** *usuário* (R-Type=0 *e FullState=False*)
Decodifica mensagem de decisão
Adiciona usuário recebido (PRC "User" e PRC "UserAssignement") ao documento XML
**Fim**
**Se** *total* (R-Type=0)
Decodifica mensagem de decisão
Monta todo o documento de configuração XML a partir da seção pep + tipo do cliente
**Fim**
Verifica aplicação da decisão
**Se** mensagem de decisão decodificada totalmente
Executa **createRPT** informando sucesso
**Senão**
Descarta decisão
Restaura documento anterior
Executa **createRPT** informando falha
**Fim**
**Fim**

**Algoritmo *createRPT***

Algoritmo *creteRPT* tem como função informar ao PDP o resultado relativo a uma aplicação de política recebida.

***Início***
*Recebe tipo de report (falha ou sucesso)*
*Monta mensagem RPT a partir do tipo de report*
***Fim***

***2. RBPEP_CreateSession***

Esta API estabelece uma sessão de usuário, utilizando a identificação de usuário passada como parâmetro, retornando o conjunto de papéis disponívei, que satisfazem às restrições impostas por separação estática (SSD) (*roleset[]*). Se as informações do usuário não forem encontradas na PIB, a API solicita ao PEP um provisionamento parcial através do algoritmo *createREQUSR* já descrito. Após a carga da PIB, o PEP processa a informação a fim de enviar resposta à aplicação que solicitou a abertura da sessão. A API *RBPEP_CreateSession* difere da função *CreateSession* definida no RBAC pois não ativa nenhum papel ou conjunto de papéis padrão para o usuário. A aplicação deve, explicitamente, definir quais papéis devem ser ativados para um determinado usuário, o que é feito na mensagem subseqüente (*RBPEP_SelectedRoles*). Esta modificação permite que, de acordo com a necessidade do usuário, possa ser removido um papel que não é mais necessário sob uma sessão, como forma de satisfazer as restrições devidas à separação dinâmica (DSD). Para utilizar esta API, a aplicação deve informar o usuário, através da utilização do atributo *uid*, que corresponde a um objeto *Person* do modelo CIM. Objeto esse, que corresponde a um usuário do sistema. A arquitetura sugerida para o PEP não interfere no processo de autenticação, pressupondo que a aplicação já deva tê-lo efetuado e que o usuário já tenha sido avaliado por este processo, tendo a aplicação a identificação de um usuário constante do repositório CIM. Devido à característica relativa à separação dinâmica (DSD) ser aplicada somente em uma sessão, a API *Create_Session* retorna, também, à aplicação o número de sessões atualmente abertas para o usuário. O parâmetro *session* é o único valor gerado pelo RBPEP e retornado à aplicação a fim de ser utilizado em chamadas futuras.

```
Início
        Recebe identificação do usuário (uid)
        Verifica se existe sessão estabelecida pelo usuário através da tag "uid" do controle de sessão
        Se não existe
                Executa chamada ao PEP (createREQUSR) informando a identificação do usuário
        Fim
        Obtém OID do usuário
        Se não existir
                reporta erro
        Fim
        Obtém lista de papéis associados ao usuário
        Cria entrada para usuário
        Cria a sessão para o usuário a partir da entrada relativa ao usuário
        Retorna lista de papéis elegíveis, identificação da sessão e número de sessões ativas
Fim
```

Para que a API *RBPEP_CreateSession* seja implementada é necessária a existência de uma estrutura capaz de armazenar cada uma dessas sessões, relacionando-as aos usuários e papéis ativos. Desta forma, a arquitetura traz, também, esse controle de sessão através de uma estrutura em XML adicionada ao arquivo de configuração do PEP.

Esta tabela, adicionada à estrutura XML, tem como finalidade controlar o estado das sessões RBAC efetuadas pelos usuários do sistema. Assim, é possível verificar quais usuários estão ativos, quais papéis RBAC estão selecionados para uma determinada sessão, quantas sessões um mesmo usuário tem ativas, além de identificar cada sessão unicamente.

### *3. RBPEP_SelectedRoles*

Esta API tem como função ativar um subconjunto de papéis selecionados pela aplicação, de acordo com o conjunto de papéis recebido pelo parâmetro retornado pela API *RBPEP_CreateSession* (*roleset[]).* Esta API avalia as instâncias (PRI's) relativas a "*UserAssignement*", "DSD" e "*DSDEntries*" como forma de determinar quais os componentes do subconjunto informado pela aplicação podem, ou não, ser ativados. Se todos os papéis constantes do subconjunto puderem ser ativados, então a API retorna à aplicação o valor *TRUE*. A API *SelectedRoles*, de forma diferente que a definida no modelo RBAC, pode ser chamada somente na sessão.

```
Início
        Recebe a identificação da sessão e a lista de papéis a serem ativados
        Obtém a sessão a partir da tag “Sessao" atributo “id” de acordo com identificação da sessão
        Se não existir
                Retorna falso
        Fim
        Obtém os papéis já ativos na sessão a partir da tag “CodigoPapel”
        Verifica a existência dos papéis obtidos no passo anterior na lista recebida
        Se não existe na lista recebida
                Adiciona à lista recebida os papéis obtidos a partir da tag
        Fim
        Obtém a lista de DSD contendo papéis atingidos e cardinalidade
        Verifica a ocorrência de DSD entre os papéis listados
        Se ocorrer DSD
                Retorna falso
        Fim
        Para cada papel constante da lista
                Cria um identificador seqüencial (tag “PapelAtivo” item “item”)
                Preenche a tag “Sessao" item “CodigoPapel” com o OID relativo ao papel
        Fim
        Retorna verdadeiro
Fim
```

***4. RBPEP_CheckAccess***

Semelhante à API definida no modelo RBAC proposto pelo NIST, esta API verifica se um usuário tem permissão para executar a operação definida no conjunto de objetos especificados pelo parâmetro *objectfilter*. O parâmetro *objectfilter* é formado por expressões *booleanas* do tipo "*PolicyExplicitVariable=PolicyValue*" agrupadas na forma conjuntiva (CNF) ou disjuntiva (DNF). É importante ressaltar que, na PIB, as instâncias (PRI's) "*Objects*" podem definir um ou mais objetos CIM através da utilização do caractere coringa. Por exemplo, {"*DataFile.Readable=true*" AND "*DataFile.Name*=*.doc"}. A associação entre elementos de objetos CIM, podem ser representados por expressões "AND". Por exemplo: {"*SoftwareApplication.Name =FinantialManager*" AND "SoftwareFeature=AuditModule"}. O PEP avalia a expressão contida no *objectfilter* e contrapõe com os dados armazenados na PIB, de acordo com as permissões assinaladas aos papéis ativos em uma sessão, e assumindo que as variáveis explícitas, ausentes nas expressões, sejam consideradas como falsas. O PEP também avalia as restrições relativas a períodos de tempo e filtros IP. As informações relativas a filtros IP são passadas pela aplicação no parâmetro *ipfilter* e correspondem a uma expressão formada por agrupamentos de campos IP utilizando o conectivo AND. Como exemplo: {"*SrcAddr*=192.169.0.1" AND "*DstAddr*=192.168.0.10"}.

**Início**
Recebe a identificação da sessão, a ação desejada, os argumentos e os dados relativos ao filtro IP na forma ((nome da variável constante do filtro IP) = valor, (nome da variável constante do filtro IP) = valor, ... )
Lê argumentos
Obtém as variáveis explícitas a partir dos argumentos
Obtém o OID relativo à ação desejada a partir da PRI ("Prid") contida na PRC "Permissions" item "operation"
Obtém o OID relativo ao objeto associado à ação a partir da PRC "Permissions" item "object"
**Se** não existir qualquer um dos OID obtidos nos passos anteriores
Retorna falso
**Fim**
Obtém OID da restrição de tempo a partir da PRI ("Prid") contida na PRC "TimeFilterPermission" item "Permission"
**Se** existir
Verifica o filtro de tempo a partir do OID em relação ao tempo registrado no sistema
**Fim**
**Se** os dados relativos ao filtro IP recebidos não for vazio
Compõe o filtro IP recebido de acordo com nomes dos campos recebidos
Obtém OID da restrição IP a partir da PRI ("Prid") contida na PRC "IPFilter" de acordo com os dados compostos no filtro recebido
Valida o filtro IP definido pelo OID com os dados recebido (valores dos campos)
**Se** validação do filtro IP não for possível
Retorna falso
**Fim**
**Fim**
Obtém OID do papel a partir da PRC "PermissionAssignement" item "role" em relação ao OID da permissão contida na PRC "PermissionAssignement" item "permission"
**Se** não existir
Retorna falso
**Fim**
Verifica se o papel está ativo na sessão recebida
**Se** não estiver ativo
Retorna falso
**Fim**
Obtém restrição de tempo associada ao papel PRC "TimeFilterRole" item "timeFilter" a partir do OID do papel contido na PRC "TimeFilterRole" item "role"
**Se** existir restrição de tempo
Verifica validade do filtro definido pelo OID em relação ao tempo atual
**Se** não for verificado
Retorna falso
**Fim**
**Fim**
Obtém a expressão a partir do OID do objeto em relação a PRI ("Prid") contida na PRC "Objects" item "expression"
Verifica expressão com variáveis explicitas recebida
**Se** for validada
Retorna verdadeiro
**Senão**
Retorna falso
**Fim**
**Fim**

***5. RBPEP_CloseSession***

Esta API tem como função encerrar uma sessão aberta por um usuário, permitindo a liberação de recursos de memória alocados para a informação sessão do usuário.

```
Início
        Recebe identificação da sessão
        Verifica se a sessão existe
        Se não existir
                Retorna falso
        Fim
        Se a  sessão for a única relativa ao usuário
                Remove usuário juntamente com a sessão
        Senão
                Remove a sessão
        Fim
Fim
```

### 5.3.2 Implementação Servidor de Políticas – PDP

O PDP implementa os algoritmos necessários para a extração das informações, a partir do modelo RBPIM, necessárias à composição da PIB. O modelo RBPIM, baseado nas definições contidas no PCIM, tem como objetivo possibilitar a reutilização de informações, como forma de simplificação das tarefas envolvidas na especificação de políticas RBAC. De acordo com o número de aplicações e recursos gerenciados pela arquitetura, a quantia de objetos presentes no repositório do PDP pode ser significantemente grande. Contudo, como já apresentado, somente um subconjunto de informações, contidas no RBPIM, são consideradas no processamento da informação requisitada pelo PEP. Devido ao fato do PEP informar a combinação de papéis suportados, já na requisição, somente uma parte dos objetos *RBACPolicyGroup* associado a esses papéis são considerados durante o processo de composição da PIB.

De posse dos papéis suportados, o PDP obtém os objetos *RBACPolicyGroup* relacionados àqueles papéis. O próximo passo é submeter estes grupos de políticas de forma a obter todos os papéis RBAC relacionados. A partir dos papéis RBAC obtidos é possível iniciar o preenchimento da PIB. São preenchidos, então, na tabela *Roles* , o nome do papel RBAC "*rolename*" e sua prioridade "*priority*". Em seguida, a partir do papel, são preenchidos os filtro de tempo em *TimeFilter* e os campos relativos à

associação do papel com o filtro de tempo – tabela *TimeFilterRole* –. A partir da ação constante nos papéis RBAC, pode-se acessar as informações relativas às permissões. Inicialmente é preenchida a tabela *Permission* com os itens relativos à operação "*operation*", em seguida é preenchida a tabela "*Objects*" item "*expression*", com os dados obtidos nas condições retornando então à tabela *Permission* e preenchendo o item "*object*" com o OID do objeto. A tabela a ser preenchida na seqüência é a que faz a ligação entre a permissão e o filtro IP *IPHeaderFilterPermission*. Nesta tabela é preenchido o item "*permission*", com o OID relativo à permissão e, então, é preenchida a tabela *IPFilter*. Após este preenchimento é retornado o OID do filtro e então é possível preencher o item "*ipFilter*" da tabela *IPHeaderFilterPermission.* O próximo passo é associar as permissões às restrições de tempo. Isto é feito preenchendo o item "*PermissionIndex*" da tabela de ligação *TimeFilterPermission* e, em seguida, preenchendo e obtendo o identificador do filtro de tempo OID. De posse do OID, pode ser preenchido o item "*TimeFilterIndex*" da tabela *TimeFilterPermission*. Finalmente, de posse do OID relativo à permissão e do OID relativo ao papel RBAC, é possível associar a permissão ao papel preenchendo os itens "*permission*" e "*role*" da tabela de ligação *PermissionAssignement*.

No preenchimento de todas as tabelas é considerada a preexistência, isto é, caso uma tabela a ser criada possua os mesmos dados constantes de uma instância já preenchida, a instância é somente referenciada, não sendo necessário a sua criação. Este artifício permite uma otimização da PIB, uma vez que não há repetição de instâncias com mesmos valores. A seguir, serão apresentados os algoritmos relativos ao PDP. Cabe reforçar que grande parte destes algoritmos tem como função garantir o funcionamento do protocolo COPS-PR.

**Algoritmo Principal**

```
Início
        Carrega Configuração através do documento XML do arquivo (PDP.xml)
        Repete
                Recebe mensagem do PEP
                Verifica tipo da mensagem
                        Caso OPN : Chama algoritmo ReadOPN
                        Caso REQ: Chama algoritmo ReadREQ
                        Caso RPT: Chama algoritmo ReadRPT
                        Caso CC:   Chama algoritmo ReadCC
                Fim
        Fim
Fim
```

Da mesma forma como descrito para o PEP, o algoritmo principal será detalhado através de algoritmos complementares apresentados a seguir:

**Algoritmo *readOPN***

Este algoritmo recebe uma mensagem do PEP e avalia se pode prover as informações de políticas necessárias ao funcionamento do cliente.

```
Início
        Verifica se o tipo do PEP é suportado, se o PEP identificado é autorizado a se conectar e se
        este mesmo PEP não possui conexões já estabelecidas com o PDP (está ativo).
        Se as três condições forem satisfeitas
                Adiciona a identificação do PEP à lista de PEP's ativos
                Carrega em memória o modelo referente ao tipo do PEP (RBAC.xml)
                Se existir o tipo do PEP no documento
                        Adiciona o modelo ao documento referente ao PDP em memória, na seção
                        clientType referente a partir da identificação
                        Identifica o cliente no Documento de acordo com o PEPId
                Senão
                        Adiciona o modelo ao documento referente ao PDP em memória, na seção
                        ActivePep, de acordo com seu tipo
                        Identifica o cliente no Documento de acordo com o PEPId
                Fim
                Requisita uma mensagem createCAT informando o PepId
        Senão
                Requisita uma mensagem createCC informando o endereço do próximo PDP bem
                como sua porta TCP de conexão.
        Fim
Fim
```

**Algoritmo *readCC***

Este algoritmo tem como função desconectar um PEP, de forma segura possibilitando a limpeza do arquivo de controle de estados.

**Início**
Remove o PEP da lista de clientes conectados de acordo com o PEPId recebido
Obtém o endereço IP do próximo servidor de políticas (tag "NextPdpAddr")
Obtém a porta TCP relativa ao próximo servidor de políticas (tag "NextPdpPort")
Requisita uma mensagem **createCC** informando o endereço do próximo PDP bem como sua porta TCP para conexão
**Fim**

**Algoritmo *readREQ***

Tem como função verificar a solicitação de políticas do cliente em relação aos papéis, classes e atributos suportados determinando, assim, a forma de criação da decisão de políticas a serem enviadas ao PEP.

**Início**
Obtém o *Handle* referente à troca de mensagens entre PEP e PDP
Verifica o *tipo* da requisição de políticas de acordo com R-Type e M-Type da mensagem
**Se** *tipo* = configuração de usuário
Chama algoritmo **decUSR**
**Senão**
Carrega os dados recebidos no documento XML do PDP de acordo com a identificação do cliente e restrições relativas a classes e atributos
Chama algoritmo **decRBAC**
**Fim**
Requisita mensagem **CreateDEC** informando o *handle* e o arquivo referente à decisao obtida em decRBAC ou decUSR
**Fim**

**Algoritmo *decUSR***

Este algoritmo formata uma decisão de política relativa a um usuário. Desta forma, o arquivo formatado conta com um objeto definido pela PRC "*User*" e diversos objetos, quantos forem necessários, definidos pela PRC "*UserAssignement*". Para que seja possível compor o arquivo de decisão, o algoritmo *decUSR* invoca a função *carregaUserAssignement* que, por sua vez, invoca a função *verificaHeranca*. Assim, serão apresentados a seguir, também estes algoritmos.

**Início**
Recebe identificação do usuário
Cria arquivo para provisão de usuário
Executa **carregaUserAssignement** informando identificação do usuário e arquivo para preenchimento
Retorna arquivo relativo à decisão de usuário
**Fim**

### Função *carregaUserAssignement*

**Início**
Recebe arquivo e identificação de usuário
Executa **LDAP_GetUserOu** passando identificação do usuário
Obtém as *Business Category* relativas ao usuário
Executa **LDAP_GetRbpimRole**
Obtém *Lista de Condições*
Executa **LDAP_GetPcimVariable** informando as *condições* e *Business Categories* obtidas anteriormente
Obtém os *papéis* que atendem aos critérios passados (*condições* e *business category*)
Executa **verificaHeranca** informando *papéis*
Obtém *papéis obtidos por herança*
Compara *papéis obtidos por herança* com *papéis* já obtidos
Adiciona *papéis* não existentes na *lista obtidos* a partir da comparação anterior
Verifica ocorrência de **SSD** informando *papéis*
Obtém *lista de papeis livres de ocorrências por SSD* a partir da lista de papéis obtidos nos passos anteriores
Preenche a PRC "User", contida no arquivo, com os dados relativos à instância do usuário recebido
Preenche PRC "UserAssignement" relacionando o usuário com os papéis obtidos
Retorna arquivo preenchido
**Fim**

### Função *verificaHeranca*

**Início**
Recebe *papéis*
Localiza *papeís herdados*
Obtém *OID* dos papeis herdados
Retorna *lista de herança*
**Fim**

**Algoritmo *decRBAC***

Este algoritmo é responsável pela formatação do arquivo de políticas da maioria dos objetos a serem provisionados ao PEP. Invocado na primeira requisição feita pelo PEP, busca todos os elementos, excetuando-se os usuários, de acordo com o conjunto de papéis suportados. Este algoritmo, por ser bastante extenso, será apresentado de forma segmentada através de funções menores, a fim de facilitar o entendimento. Assim, a seguir, será apresentado o algoritmo principal e a seguir as funções correlatas.

**Início**
Cria um novo arquivo identificado como Dec + identificação do cliente (PEPId).xml
Copia o conteúdo referente ao cliente do documento em memória do PDP para o novo arquivo
Obtém os *roles* suportados pelo cliente PRC "InterfaceIndex" a partir do item "ifRoles"
Executa **carregaRoles** informando o documento XML em uso, e os papéis de política suportados
Executa **carregaDSD** informando o documento XML em uso
Grava arquivo de decisão em disco
Retorna arquivo de decisão preenchido
**Fim**

### Função *carregaRoles*

**Início**
Recebe *documento* XML e os *papéis de política suportados*
Executa **LDAP_GetPcimGroup** informando os *papéis de política suportados*
Obtém *OU* relativa às políticas
Executa **LDAP_GetPcimPolicySet** informando os dados obtidos na consulta anterior
Obtém os *papéis RBAC*, suas *prioridades*, *os períodos de validade*, *a lista de ações*,
**Para** cada *papel* obtido
Obtém *o nome do papel*
Preenche PRC "Roles" item "rolename" com o *nome do papel*
Obtém *a prioridade do papel*
Preenche PRC "Roles" item "priority" com a *prioridade*
Obtém o período de *validade do papel*
Executa LDAP_GetPcimTPC informando a *validade*
Obtém o *filtro de tempo* associado ao papel
Executa carregaTimeFilter informando *o filtro de tempo*
Obtém *OID* dos filtros de tempo
Preenche PRC "TimeFilterRole" item "timeFilter" com *OID* do filtro de tempo
Preenche PRC "TimeFilterRole" item "role" com o *OID* do role
Obtém a *lista de ações* associadas ao papel RBAC
**Para** cada *ação* encontrada
Executa **LDAP_GetRbpimPermissionDN** informando a *ação*
Obtém a *DN das ações*
Executa **carregaPermissoes** informando *o documento XML* em uso, *o DN* e *a lista relativa aos filtros de tempo*
Obtém *lista de permissões*
**Para** cada *permissão*
Preenche PRC "PermissionAssignement" item "role" com o *OID do papel*
Preenche PRC "PermissionAssignement" item "permission" com o *OID da permissão*
**Fim**
**Fim**
**Fim**
**Fim**

### Função *carregaTimeFilter*

**Início**
Recebe vetor com os elementos do filtro
Preenche PRC "TimeFilter" item "LocalUtcTime" com o valor relativo à hora local ou UTC
Preenche PRC "TimeFilter" item "TpcTime" com o valor de tempo
Preenche PRC "TimeFilter" item "MesesdoAno" com o valor relativo aos meses do ano
Preenche PRC "TimeFilter" item "DiasdoMes" com o valor relativo aos dias do mês
Preenche PRC "TimeFilter" item "DiasdaSemana" com o valor relativo aos dias da semana
Preenche PRC "TimeFilter" item "HoradoDia" com o valor relativo à hora do dia
**Fim**

### Função *carregaPermissoes*

**Início**
Recebe o documento XML em uso, o DN e a lista relativa aos filtros de tempo
Executa **LDAP_GetRbpimPermission** informando o DN da *permissão*
Obtém lista de *condições*
Obtém a *ação*
Executa a **LDAP_GetPcimAction** informando o *domínio LDAP* e a *ação*
Obtém a *lista de operações*
**Para** cada *operação*
Preenche PRC "Permission" item "operation" com a *operação*
**Fim**
Executa **carregaObj** informando o *documento XML* em uso, *a lista de condições* e *o DN do objeto*
Obtém *OID do objeto*
Preenche PRC "Permission" item "object" com o *OID do objeto*
Executa **LDAP_GetRbpimPermission** informando o *DN da permissão*
Obtém *lista de condições*
**Para** cada *condição*
**Se** *condição* de filtro de pacotes (IP)
**Se** existe filtro
obtém *OID do filtro*
**Senão**
executa **carregaIPFilter** informando o *filtro IP*
**Fim**
Obtém *OID do filtroIP*
Preenche PRC "IPHeaderFilterPermission" item "ipFilter" com o *OID do filtro IP*
Preenche PRC "IPHeaderFilterPermission" item "permission" com o *OID relativo à permissão*
**Fim**
**Se** *condição de filtro de tempo*
Executa **Consulta5** informando a *validade*
Obtém *filtro de tempo*
**Se** existe filtro
obtém *OID do filtro*
**Senão**
executa **carregaTimeFilter** informando o *filtro de tempo*
**Fim**
Obtém *OID do timeFilter*
Preenche PRC "TimeFilterPermission" item "TimeFilterIndex" com *OID do filtro de tempo*
Preenche PRC "TimeFilterPermission" item "permission" com *OID relativo à permissão*
**Fim**
**Fim**
Retorna lista de ações
**Fim**

### Função *carregaObjetos*

**Início**
Recebe *documento XML* em uso, a *lista de condições* e *o DN do objeto*
Executa **LDAP_GetPcimCompoundCondition** informando o *DN do objeto*
Obtém *tipo da lista de condições* (DNF ou CNF)
Define *conector principal* (&& ou ||)
Define *conector secundário*
Obtém *definição* se a condição é negada
Obtém *lista de condições*
**Para** cada *condição*
Obtém o *agrupamento*
Executa **LDAP_GetPcimCondition** informando a *condição*
Obtém *o valor "pcimStringList" relativo à variável explicita*
Obtém *o nome da condição*
Obtém *a classe da variável ("pcimVariableModelClass")*
Obtém *a propriedade relativa à variável ("pcimVariableModelProperty")*
Monta a expressão do grupo a partir dos dados obtidos
**Fim**
Monta a *expressão* relativa a todos os grupos de acordo com os conectivos *principal* e *secundário*
Preenche a PRC "Objets" item "expression" com a *expressão obtida*
Retorna *lista dos OID* relativos aos objetos preenchidos
**Fim**

### Função *carregaIPFilter*

**Início**
Recebe *filtro IP*
Carrega PRC "IPFilter" item "Negation" com valor relativo à negação do filtro
Carrega PRC "IPFilter" item "AddrType" com valor relativo ao tipo de endereço
Carrega PRC "IPFilter" item "DstAddr" com valor relativo ao endereço de destino
Carrega PRC "IPFilter" item "DstPrefixLength" com valor relativo ao tamanho do prefixo de destino
Carrega PRC "IPFilter" item "SrcAddr" com valor relativo ao endereço de origem
Carrega PRC "IPFilter" item "SrcPrefixLength" com valor relativo ao tamanho do prefixo de origem
Carrega PRC "IPFilter" item "Dscp" com valor relativo ao DSCP
Carrega PRC "IPFilter" item "FlowId" com valor relativo à identificação do fluxo
Carrega PRC "IPFilter" item "Protocol" com valor relativo ao protocolo
Carrega PRC "IPFilter" item "DstL4PortMin" com valor relativo à porta de destino mínima
Carrega PRC "IPFilter" item "DstL4PortMax" com valor relativo à porta de destino máxima
Carrega PRC "IPFilter" item "SrcL4PortMin" com valor relativo à porta de origem mínima
Carrega PRC "IPFilter" item "SrcL4PortMax" com valor relativo à porta de origem máxima
**Fim**

**Função *carregaDSD***

**Início**
Recebe *documento XML em uso*
Executa **LDAP_GetRbpimDSD**
Obtém *lista de DSD*
**Para** cada *DSD*
Obtém *cardinalidade da relação*
Preenche PRC "DSD" item "Cardinality" com a *cardinalidade da relação*
Obtém *OID relativa à DSD*
Obtém o *conjunto de papéis envolvidos na relação*
**Para** cada *papel*
Obtém o *OID do papel*
Preenche PRC "DSDEntries" item "Role" com o *OID relativo ao papel*
Preenche PRC "DSDEntries" item "DSDIndex" com o *OID relativo à DSD*
**Fim**
**Fim**
**Fim**

**Algoritmo *readRPT***

Este algoritmo tem como função avaliar se uma determinada política enviada ao PEP foi aplicada com sucesso. Caso a política tenha sido aplicada, a decisão enviada, armazenada em memória, é adicionada ao documento XML correspondente ao controle de estados do PDP de forma a refletir a PIB provisionada ao PEP.

**Início**
Verifica o *tipo do report recebido*
**Se** *report* de sucesso
Copia o conteúdo do documento gerado na criação da decisão CreateDEC para o documento em memória referente ao PDP na seção indicada pela identificação do cliente
**Fim**
**Fim**

Nos diversos algoritmos e funções apresentados nesta sessão são relacionadas algumas chamadas na forma de consultas LDAP. A seguir, serão apresentadas cada uma destas chamadas, especificando os argumentos passados, além da base de pesquisa, Base_DN, os atributos desejados, o escopo da busca e os filtros a serem considerados.

| ***LDAP_GetPcimGroup(pcimrole)*** | |
|---|---|
| Base_DN | dc=Domain,dc=com |
| Atributos | pcimGroupName |
| Escopo | Sub |
| Filtro | (&(objectClass=pcimgroup)(pcimgroupname="+pcimrole+")) |
| Descrição: Obtém o atributo relativo ao nome do grupo a partir do papel recebido | |

| ***LDAP_GetPcimPolicySet(ouUser)*** | |
|---|---|
| Base_DN | ouUser |
| Atributos | { "rbpimRoleName","pcimPriority","pcimConditionListType","pcimConditionList","pcimActionList","pcimRuleValidityPeriodList","rbpimInheritedRoles" } |
| Escopo | Sub |
| Filtro | (&(objectClass=pcimpolicyset)(pcimRuleEnabled=1)) |
| Descrição: Obtém os atributos relativos a uma política que esteja ativa a partir da OU do usuário | |

| ***LDAP_GetPcimTPC(validityPeriodDN)*** | |
|---|---|
| Base_DN | validityPeriodDN |
| Atributos | { "pcimTPCLocalOrUtcTime","pcimTPCTime","pcimTPCMonthOfYearMask","pcimTPCDayOfMonthMask","pcimTPCDayOfWeekMask","pcimTPCTimeOfDayMask" } |
| Escopo | Sub |
| Filtro | (objectClass=pcimconditionauxclass) |
| Descrição: Obtém os atributos relativos ao filtro de tempo a partir do DN do filtro | |

| ***LDAP_GetRbpimDSD()*** | |
|---|---|
| Base_DN | "dc=domain,dc=com" |
| Atributos | { "rbpimDSDName","rbpimRoleSet","rbpimCardinality" } |
| Escopo | Sub |
| Filtro | (objectClass=rbpimDSD) |
| Descrição: Obtém os atributos relativos aos objetos de separação dinâmica | |

| ***LDAP_GetPcimCondition(DN)*** | |
|---|---|
| Base_DN | DN |
| Atributos | { "pcimConditionName","pcimConditionGroupNumber","pcimConditionNegated","rbpimConditionName","pcimStringList","pcimVariableModelClass","pcimVariableModelProperty","pcimExpectedValueTypes","pcimIPv4AddrList","pcimIPv6AddrList","pcimMACAddrList","pcimStringList","pcimBitStringList","pcimIntegerList","pcimBoolean"} |
| Escopo | Sub |
| Filtro | (objectClass=*) |
| Descrição: Obtém os atributos relativos ao objeto de condição simples a partir da DN da condição | |

| ***LDAP_GetRbpimPermissionDN(DN)*** | |
|---|---|
| Base_DN | DN |
| Atributos | { "rbpimPermissionDN" } |
| Escopo | Sub |
| Filtro | (objectClass=*) |
| Descrição: Obtém o atributo relativo à DN de uma permissão rbpim a partir do DN | |

| ***LDAP_GetPcimAction(DN,cond)*** | |
|---|---|
| Base_DN | DN |
| Atributos | { "pcimActionOrder","rbpimOperationList","pcimActionName" } |
| Escopo | Sub |
| Filtro | (&(objectClass=pcimActionAuxClass)("+cond+") |
| Descrição: Obtém os atributos relativos a uma ação a partir da DN e da condição passada | |

| ***LDAP_GetRbpimPermission(DN)***<br>***Consulta25(Dn)*** | |
|---|---|
| Base_DN | DN |
| Atributos | { "rbpimPermissionName","pcimConditionListType","pcimConditionList","pcimActionList" } |
| Escopo | Sub |
| Filtro | (&(objectClass=rbpimPermission)) |
| Descrição: Obtém os atributos relativos a uma permissão rbpim a partir do DN da condição | |

| ***LDAP_GetPcimCompoundCondition(DN)***<br>***Consulta26(Dn)*** | |
|---|---|
| Base_DN | DN |
| Atributos | { "pcimConditionName","pcimConditionListType","pcimConditionList","pcimConditionNegated","pcimConditionGroupNumber" } |
| Escopo | Sub |
| Filtro | (&(objectClass=pcimcompoundconditionauxclass)) |
| Descrição: Obtém os atributos relativos a uma condição composta a partir de um DN | |

| ***LDAP_GetUserOu(userId)***<br>***Consulta28(userId)*** | |
|---|---|
| Base_DN | "dc=domain,dc=com" |
| Atributos | {"ou","businesscategory"} |
| Escopo | Sub |
| Filtro | (&(objectclass=inetorgperson)(cn="+userId+")) |
| Descrição: Obtém a OU e a categoria funcional de um usuário passado como parâmetro | |

| ***LDAP_GetRbpimRole()***<br>***Consulta29()*** | |
|---|---|
| Base_DN | "dc=domain,dc=com" |
| Atributos | { "rbpimRoleName","pcimPriority","pcimConditionListType","pcimConditionList","pcimActionList","pcimRuleValidityPeriodList","rbpimInheritedRoles" } |
| Escopo | Sub |
| Filtro | (&(objectClass=rbpimrole)(pcimRuleEnabled=1)) |
| Descrição: Obtém os atributos relativos aos diversos papéis RBAC ativos | |

| ***LDAP_GetPcimVariable(condDN,categoria)***<br>***Consulta30(condDn, categoria)*** | |
|---|---|
| Base_DN | condDN |
| Atributos | { "pcimvariablemodelproperty","pcimstringlist", "pcimvariablemodelclass" } |
| Escopo | Sub |
| Filtro | (&(objectclass=pcimconditionauxclass)(pcimStringList="+categoria+")) |
| Descrição: Obtém os diversos atributos relativos a uma variável explicita a partir da DN e da categoria da variável | |

## 5. 4 Protocolo COPS-PR

O relacionamento entre as entidades PEP e PDP é feito através do protocolo COPS-PR; deste modo, é necessário definir a forma como as informações são trocadas entre os elementos. Esta sessão tem como finalidade apresentar estas trocas de mensagens, identificando seus conteúdos e, para tanto, fará uso das informações constantes no diagrama de seqüência apresentado na figura 5.3.

### 5.4.1 Abertura de Conexão

A primeira mensagem trocada entre o PEP e o PDP corresponde à abertura de conexão. Nesta mensagem o PEP informa seu tipo e identificação para que o PDP possa verificar se está apto a prover os dados necessários ao funcionamento do PEP. Este pacote de dados é definido [RFC 2748] – Mensagem OPN, e pode ser visualizado a seguir:

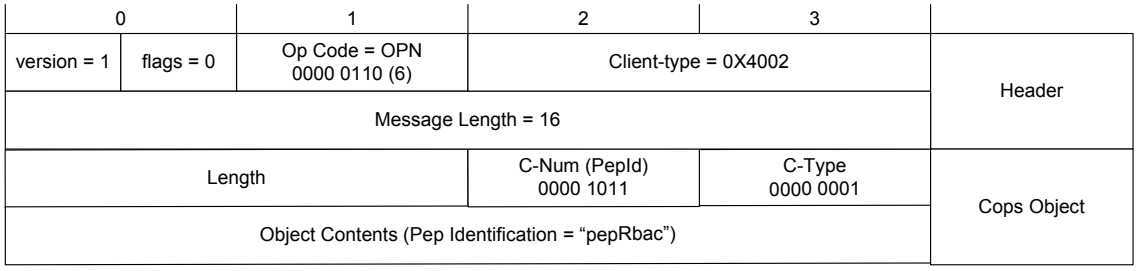


**Figura 5.4 – Mensagem Open (OPN)**

De posse do pacote de solicitação de conexão, o servidor de políticas verifica se em seus dados constam o nome e tipo do cliente que está solicitando a conexão. É verificado, ainda, se este cliente já possui uma conexão estabelecida com o servidor. Avaliadas a existência do tipo e nome, e a não existência de conexões abertas, é criada uma conexão TCP entre o PEP e o PDP.

Caso ocorra algum problema na verificação dos dados que impossibilitem a conexão, o PDP retorna ao PEP uma mensagem que encerra a conexão entre eles. Essa mensagem, [RFC 2748] – mensagem CC, contém o endereço do próximo servidor de políticas ao qual o cliente deve tentar se conectar para obter suas políticas.

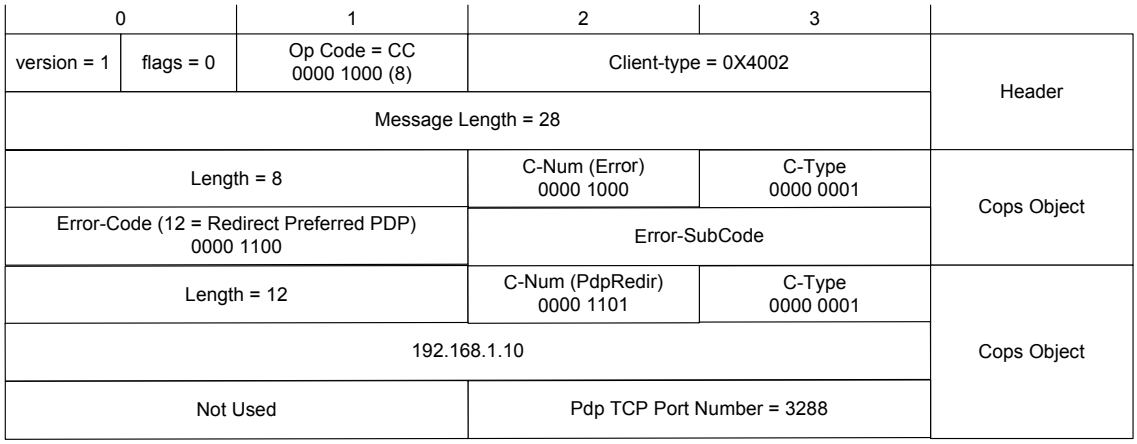


**Figura 5.5 – Mensagem Close (CC)**

O PEP, de posse desta informação, refaz o processo de conexão, agora com o servidor indicado na mensagem, da mesma forma como já descrito.

### 5.4.2 Autorização de Conexão

A resposta do servidor de políticas ao cliente, autorizando a conexão, é uma mensagem CAT definida em [RFC 2748], que contém um elemento *Keep Alive* (KA) correspondente ao tempo de verificação de atividade e conexão entre o PEP e o PDP. Este pacote de dados, também pode ser verificado a seguir:

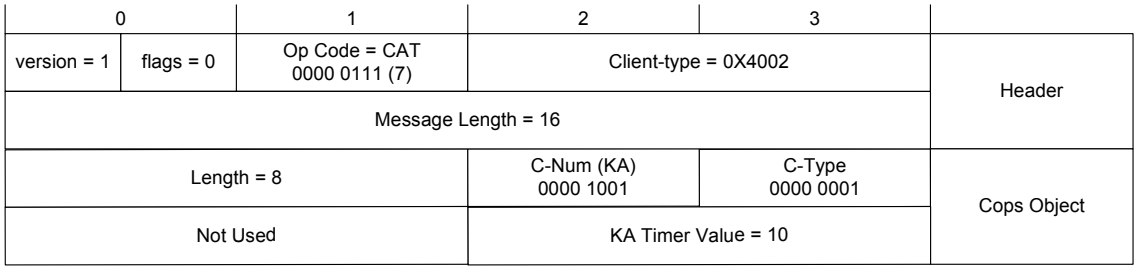


**Figura 5.6 – Mensagem Autorização (CAT)**

### 5.4.3 Requisita Configuração

Autorizada a conexão, o PEP monta um pacote de solicitação de configuração, na forma de requisição [RFC 2748] – mensagem REQ. Os dados componentes deste pacote são: um cabeçalho padrão COPS (*Header*) indicando a operação (REQ) e tipo do cliente, um objeto COPS *Handle* responsável por identificar a conexão PEP / PDP, um objeto COPS *Context* informando o contexto relativo à configuração e objetos COPS *ClientSI*, tantos quantos forem necessários, para representação de todas as classes e atributos constantes da PIB do cliente, excetuando-se a seção Configuracao.

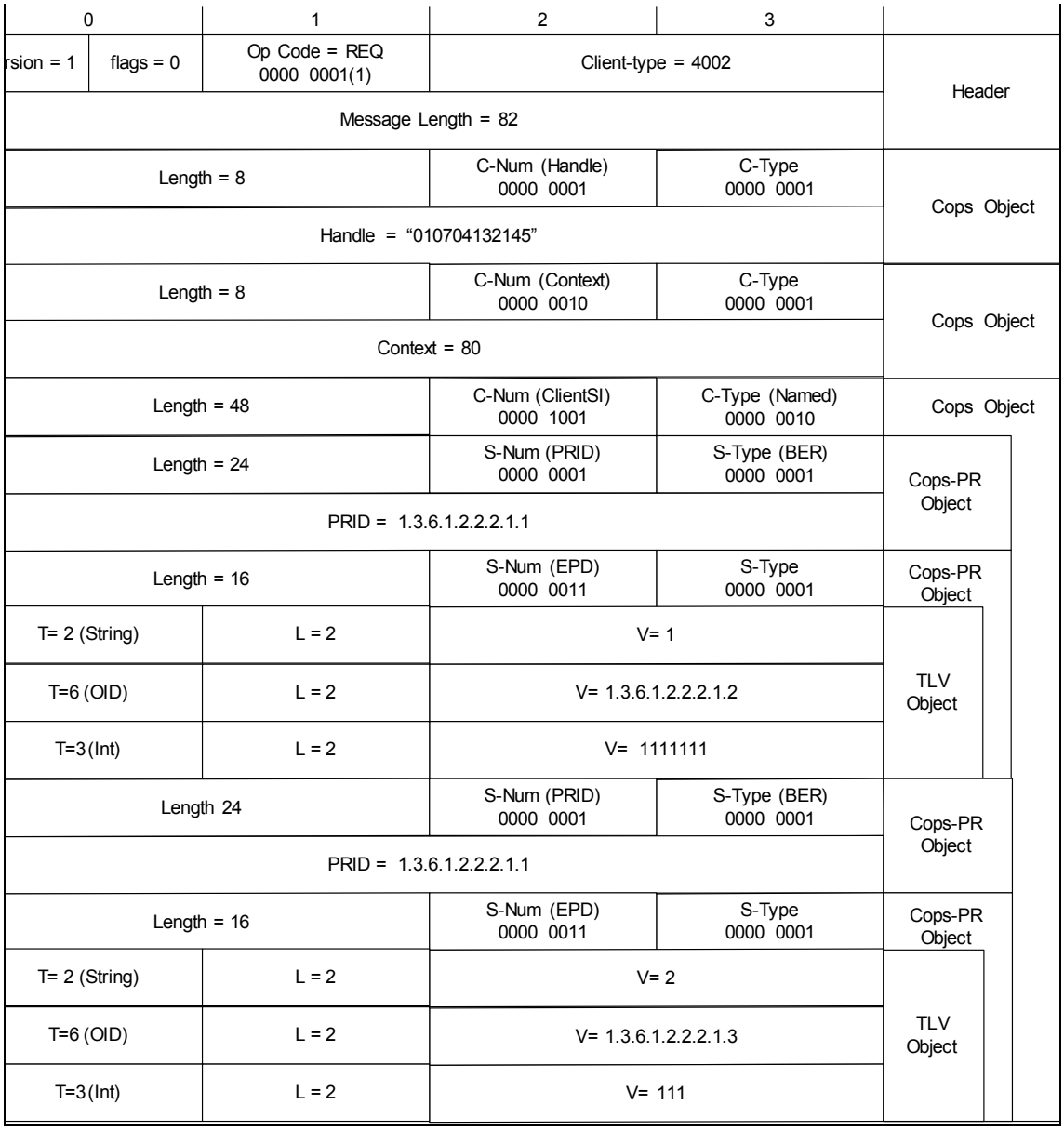


**Figura 5.7 – Mensagem Requisição Mostrando duas Instâncias de PrcSupport (REQ)**

Cada objeto COPS, do tipo *ClientSI,* é composto por objetos COPS-PR na forma PRID (identificação da classe), seguido de EPD (identificação da instância), que contém diversos elementos do tipo TLV (*type*, *length*, *value*), que correspondem aos dados (atributos e valores) relativos a cada instância da classe. Desta forma, é possível encapsular na mensagem COPS todas as informações relativas e necessárias à configuração do PEP.

### 5.4.4 Requisição de Objetos de Políticas

Recebida a requisição vinda do PEP, o PDP interpreta-a, montando, em memória, um novo documento XML relativo àquele cliente. Este novo documento serve de base para a montagem de uma solicitação de políticas ao repositório LDAP e, também, para armazenar os dados recebidos como resposta desta solicitação.

### 5.4.5 Decisão de Objetos de Políticas

A consulta LDAP, gerada a partir dos dados do PEP, busca, primeiramente, por políticas relativas aos papéis, obtendo, a partir desta, os usuários, papéis RBAC, permissões, ações, separações dinâmicas e filtros relativos a horários; sendo, também, os papéis livres de conflito por separação estática. Esta decisão de política é retornada ao PDP para que seja interpretada.

### 5.4.6 Decisão de Políticas

Os dados obtidos como resposta, na decisão de objetos de política, são interpretados e armazenados em memória na forma de documento XML. A partir deste documento, o PDP formata uma mensagem de decisão [RFC 2748] – mensagem DEC, que corresponde ao conteúdo a ser enviado ao PEP como configuração (provisionamento), na forma representada a seguir:

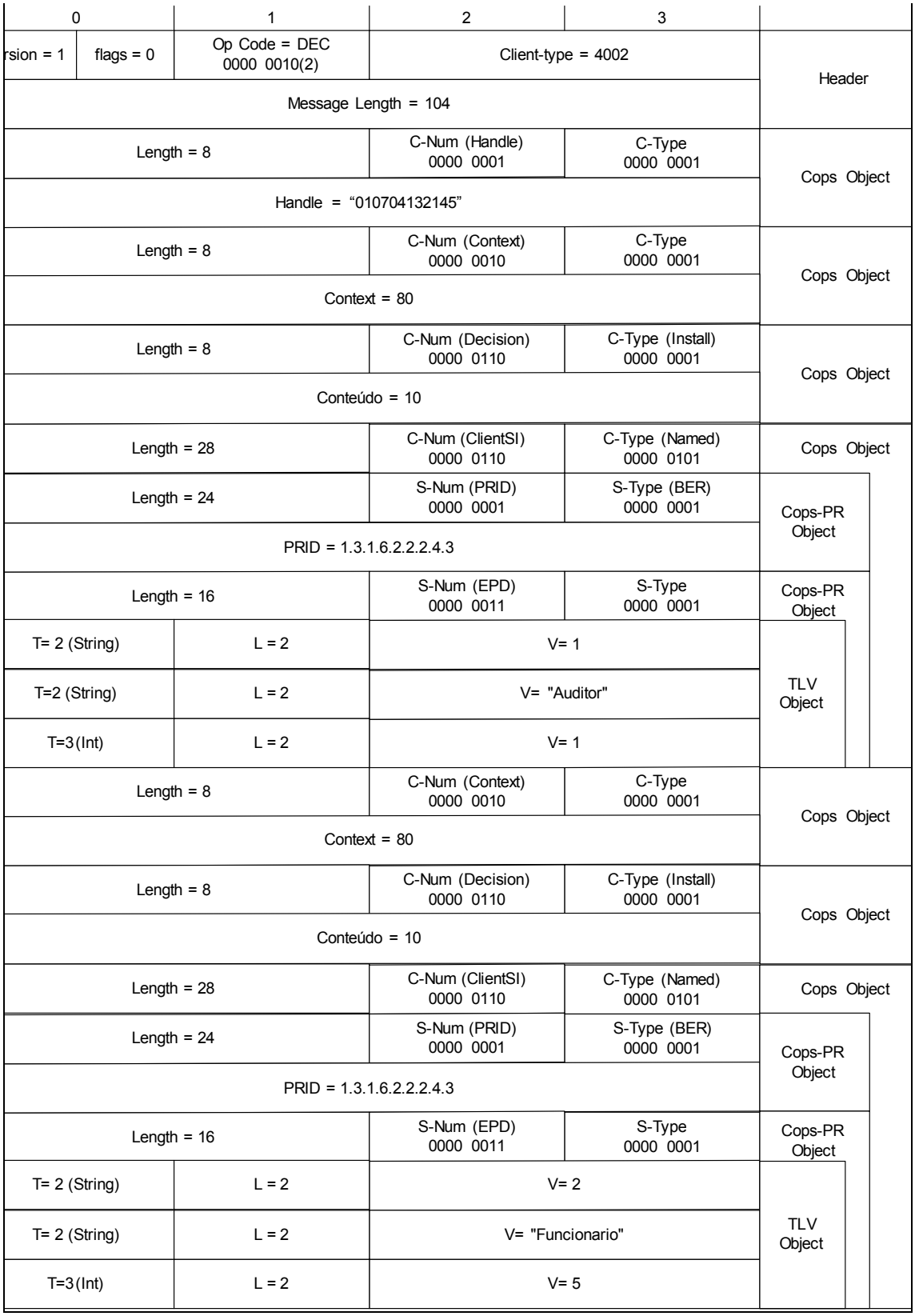


**Figura 5.8 – Mensagem Decisão Completa mostrando duas instâncias de Roles (DEC)**

Como representado, os objetos envolvidos nesta mensagem são: o cabeçalho padrão COPS (*Header*) relativo à decisão de políticas DEC, um objeto COPS *Handle* responsável por identificar a conexão PEP PDP e blocos de objetos COPS que constituem a decisão, tantos quantos forem necessários, para representação de todas as classes e atributos contidos no documento XML. Cada elemento deste bloco é composto por um objeto COPS *Context* que informa o contexto relativo à configuração, um objeto COPS *Decision* que informa a operação de instalação *Install* e um objeto COPS *NamedDecisionData* composto por um objeto COPS-PR na forma PRID (identificação da classe), seguido por um objeto COPS-PR na forma EPD (identificação da instância), que contém diversos elementos do tipo TLV (*type*, *length*, *value*) que correspondem aos dados (atributos e valores) relativos a cada instância da classe. A mensagem de decisão é composta considerando as restrições relativas a classes e atributos suportados pelo PEP.

### 5.4.7 Report

Após receber a mensagem de decisão de políticas (DEC) do PDP, o PEP interpreta-a e cria um documento XML temporário em memória. Caso a política recebida seja interpretada e aplicada de forma completa e correta, é enviada ao PDP uma mensagem do tipo *Report*, informando o sucesso da operação de decisão e, também, é feita a cópia do documento temporário para o repositório de dados do PEP criado na inicialização (seção 5.2.2). Caso contrário, a mensagem é enviada ao PDP, informando erro na aplicação da política, que destrói, também, o arquivo temporário.

A interpretação de uma mensagem de sucesso pelo PDP, implica na cópia do documento XML, referente à decisão, para o repositório de dados criado na inicialização (seção 5.2.1). Após a cópia, o documento referente à decisão é destruído. Ao receber uma mensagem de erro na aplicação da política, o PDP apenas descarta o documento XML referente à decisão.

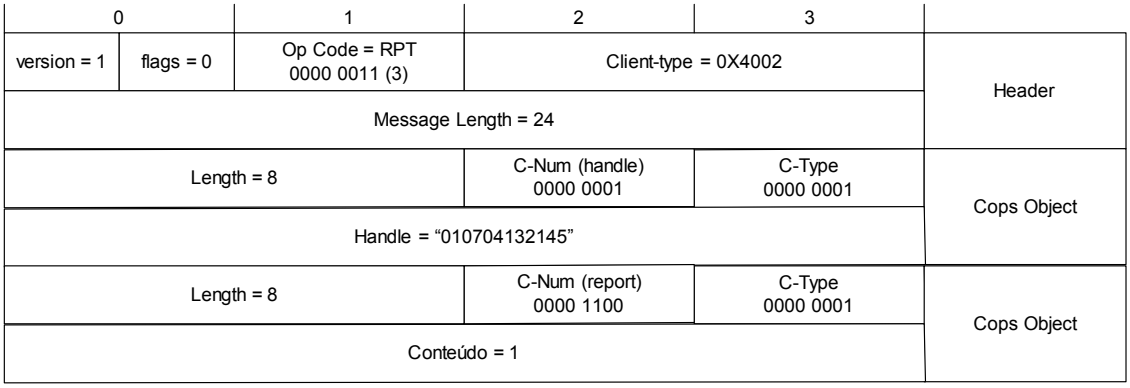


**Figura 5.9 – Mensagem Report (RPT)**

### 5.4.8 Abertura de Sessão

Aplicada a decisão sobre o PEP, o mesmo está apto a responder pelas políticas configuradas, assim, a aplicação, através da API disponibilizada, solicitada a abertura de sessão ao PEP. Esta solicitação contém um identificador único correspondente a um usuário. É importante ressaltar que a aplicação tem, somente, o identificador único do usuário, não importando a forma de aquisição do mesmo. A aquisição do identificador é feita fora do escopo deste trabalho e pode, para tal, utilizar métodos de autenticação dos mais diversos tais como *Kerberos*, *Ldap*, *Nis*, etc.

### 5.4.9 Requisição de UA (User Assignment)

De posse da identificação, é feita uma verificação na base de dados, PIB, do PEP a fim de localizar o usuário. Caso não exista entrada correspondente ao usuário informado é, então, formatada uma nova requisição ao PDP (mensagem REQ), desta vez incremental, contendo o identificador do usuário. Para informar ao PDP que a requisição do PEP corresponde a uma atualização incremental de PIB, é utilizada a PRID relativa à *PIBIncarnation* (oid = 1.3.6.1.2.2.2.1.2), contendo na PRC, o valor do atributo *FullState* = *False*. A mensagem REQ correspondente é apresentada a seguir:

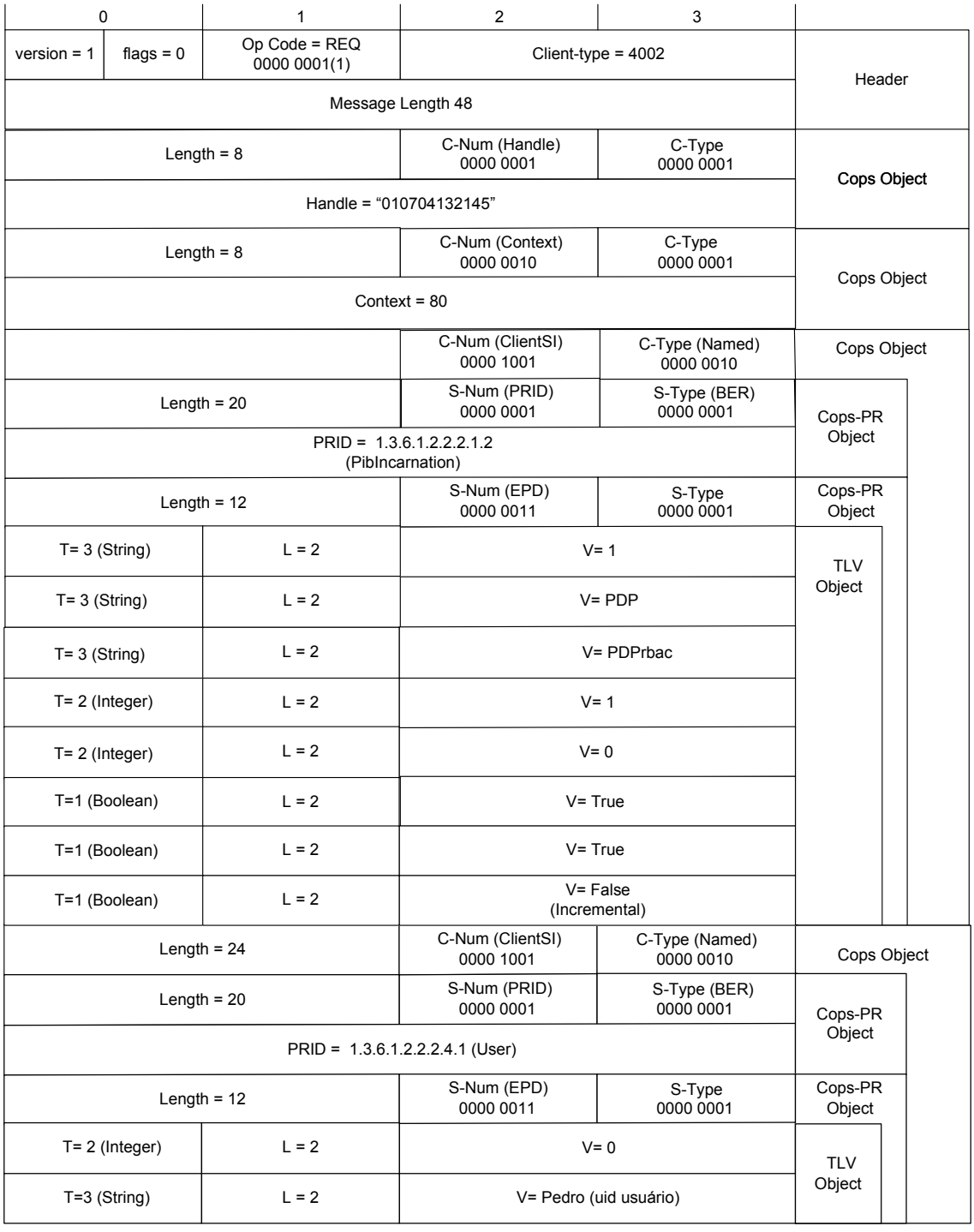


**Figura 5.10 – Mensagem Requisição Usuário Pedro (REQ)**

Esta mensagem é composta, além do objeto identificador *Handle*, por um objeto relativo ao *Context* que, como pode ser verificado na figura 5.10, recebeu o valor 80. Este objeto, como já descrito na seção 3.5.2, é composto por dois blocos, de dois bytes cada, definidos como *R-Type* e *M-Type*. O campo *R-Type* define o tipo de requisição, que para o caso proposto tem o valor 0x08 e que define uma requisição de configuração.

O campo *M-Type* define o tipo de mensagem e é composto por 16 bits e deve ser definido sob o escopo da aplicação. Neste sentido, foi definido no desenvolvimento deste trabalho o valor 0x00 como padrão para o provisionamento ao PEP.

A partir da identificação da classe, através da PRID, e da identificação da instância desta classe, através de um identificador encapsulado em um EPD, é possível requisitar ao PDP o provisionamento incremental de um objeto. Assim, a figura 5.10 apresenta uma requisição de provisionamento da instância com valor 1, relativa a um atributo, que neste caso corresponde à identificação do usuário definido pelo valor Pedro, de um objeto representado pelo OID 1.3.6.1.2.2.2.4.1 (User).

### 5.4.10 Solicitação de Objetos UA (*User Assignment*)

O PDP, ao receber a nova requisição, referente ao usuário, formata uma requisição LDAP e a encaminha ao repositório a fim de obter os dados relativos ao usuário em particular.

### 5.4.11 Decisão de Objetos UA (*User Assignment*)

Ao receber a resposta da requisição encaminhada ao repositório LDAP, o PDP interpreta avaliando quais papéis podem ser associados ao usuário, já livres por separação estática (SSD – definido no modelo RBAC). Esta associação, UA, obtida é, então, armazenada em memória no PDP, na forma de documento XML. Este documento é formatado e encaminhado ao PEP na forma de decisão de UA, conforme mensagem de decisão apresentada a seguir:

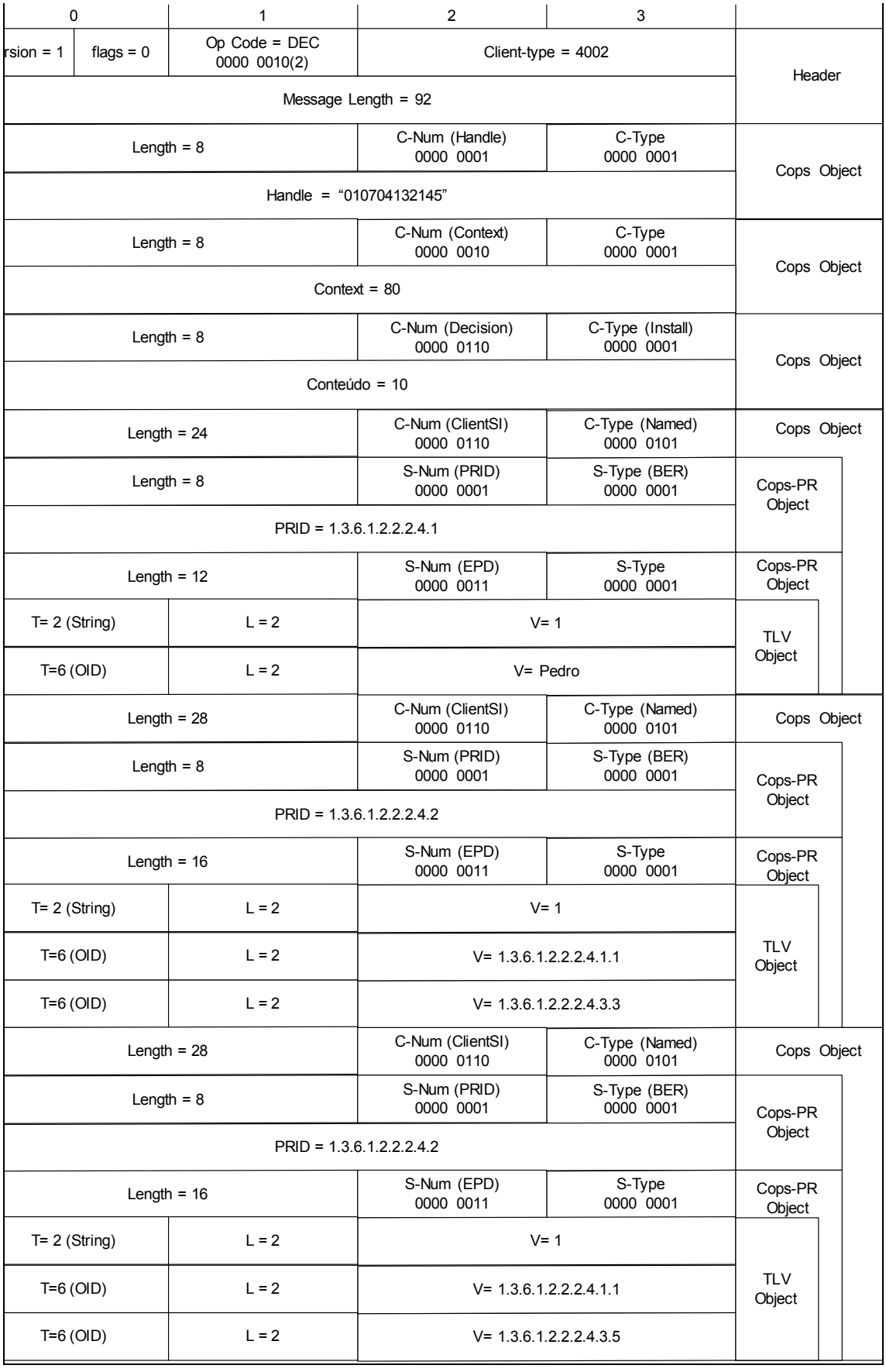


**Figura 5.11 – Mensagem Decisão Usuário Pedro Associado a Dois Papéis (DEC)**

### 5.4.12 Decisão de UA (User Assignment)

A decisão de UA, enviada pelo PDP, é recebida, então, pelo PEP que a interpreta adicionando o usuário à PIB. Desta forma, o usuário estará disponível para novas aberturas de sessão, não sendo mais necessárias trocas de mensagens entre PEP e PDP para requisitar este usuário. Este estado é mantido enquanto o PEP estiver ativo. É enviada ao PDP uma mensagem de *Report*, comunicando o sucesso ou falha da aplicação de política, na mesma forma como apresentado na sessão 5.2.9. Da mesma forma, também, é o procedimento adotado pelo PDP. No caso de sucesso na aplicação da política, o PDP adiciona o documento XML referente à decisão enviada ao documento principal. No caso de falha, o documento é descartado.

### 5.4.13 Retorno de Papéis

No caso do usuário informado não ser encontrado, seja por não existir no repositório, ou, por problemas na aplicação da política referente à associação UA pelo PEP, é retornada à aplicação uma lista de papéis nula. Caso contrário, a API retorna um conjunto de papéis RBAC elegíveis.

### 5.4.14 Seleciona Papéis

A aplicação, a partir do conjunto de papéis RBAC recebidos na sessão anterior, seleciona aqueles necessários, naquele momento, e encaminha-os, novamente, ao PEP, através da API, para que a sessão seja estabelecida.

### 5.4.15 Confirmação de Sessão

O PEP recebe os papéis RBAC selecionados, avalia se estão livres por separação dinâmica (DSD do modelo RBAC), registrando-os na PIB como uma nova sessão associada ao usuário. Após este registro é retornado à aplicação um identificador único relativo à sessão que foi estabelecida com o PEP.

### 5.4.16 Verificação e Decisão de Acesso

Após a ativação da sessão, a aplicação informa a sessão, a operação desejada, e os argumentos necessários para avaliação do controle de acesso. Os dados são, então, avaliados de acordo com as políticas armazenadas na PIB e, de acordo com essa avaliação, é retornada a autorização ou negação para efetuar a ação requisitada.

## 5. 5 Conclusão

Este capítulo apresentou a arquitetura de implementação sugerida para o modelo, reforçando o funcionamento e os papéis dos elementos PEP e PDP.

Em relação ao PEP, foram apresentadas cada uma das API's disponibilizadas, bem como cada um dos algoritmos envolvidos em seus funcionamentos, possibilitando, desta maneira, uma maior compreensão da inter-relação entre o PEP e o PDP e, também, o entendimento do comportamento do PEP em relação às aplicações por ele atendidas.

Em relação ao PDP, foi apresentado seu funcionamento a partir dos algoritmos principais, possibilitando, assim, o entendimento do processo de solicitação, busca, composição e repasse das políticas. Para tanto, foram apresentadas as consultas LDAP mais utilizadas.

Fechando o capítulo foram apresentados fragmentos relativos às mensagens COPS-PR trocadas entre PEP e PDP. Estas mensagens, apresentadas com valores correspondentes ao real, serviram como forma de mostrar as interações descritas nas seções anteriores, possibilitando, assim, uma melhor visualização de todo o processo envolvido na implementação do modelo e reforçando o funcionamento do arquitetura sugerida.

# Capítulo 6

# Estudo de Caso e Validação da Proposta

## 6. 1 Introdução

O capítulo anterior apresentou o modelo de informação com todas as suas características, ressaltando-se a PIB como elemento central do modelo. Este capítulo tem como função apresentar duas visões em relação ao modelo apresentado. Primeiramente, como o modelo é utilizado para representar o RBAC sob o ponto de vista do repositório de informações. Esta visão será feita a partir do estudo de caso para exemplificação do modelo. Num segundo momento, o objetivo será analisar a escalabilidade do modelo em relação aos objetos envolvidos no provisionamento. Estas duas abordagens têm como finalidade validar a proposta apresentada.

## 6. 2 Estudo de Caso para Exemplificação do Modelo

O estudo de caso proposto neste trabalho, foi elaborado sob a ótica de funcionamento de uma instituição bancária que disponibiliza suas aplicações como ferramentas de trabalho utilizadas pelos seus funcionários. Esta instituição será referenciada de forma hipotética sob o nome de Banco ABC.

### 6.2.1 Cenário

A corporação em questão utiliza diversas aplicações, disponibilizadas em sua intranet, como ferramentas de trabalho, para a execução das atividades relacionadas ao seu negócio. Cada uma destas aplicações possui seu próprio sistema de controle de acesso, implementado sob o modelo discricionário, mantendo a lista de usuários permitidos ao sistema, bem como as operações permitidas, associadas a cada usuário.

O sistema, como um todo, sob o ponto de vista da administração e gerenciamento das aplicações, é extremamente complexo, pois qualquer mudança em relação aos direitos de acesso relativos a um usuário ou grupo, deve ser feita de forma específica e pontual em cada uma das aplicações disponibilizadas. Para analisar este

impacto serão apresentadas duas aplicações e as relações das mesmas com as operações realizadas sobre elas.

**Tabela 6.1 – Aplicações e Operações relativas ao Banco X**

| Aplicação | Operações | |
|---|---|---|
| Gerência Financeira | Agendar TED | Autorizar TED |
| | Agendar DOC | Autorizar DOC |
| | Auditar Transações | Efetuar Pagamentos |
| Gerência de Clientes | Abrir Conta Corrente | Conceder Limite |
| | | Auditar Transações |

Os usuários dos sistemas disponibilizados são associados a categorias funcionais de acordo com as atividades exercidas.

**Tabela 6.2 - Usuários dos Sistemas e Categorias Funcionais**

| Usuários | Código da Categoria Funcional |
|---|---|
| Ailton, Ana, Carlos, Joana, Marcos, Rubens | A1 |
| Maria, Silvia, Vivian | A2 |
| Pedro | A1, B1 |
| Matias | B1, C1 |
| Alex, Carla | C1 |

A associação dos códigos relativos às categorias funcionais e as funções exercidas no sistema são apresentadas na tabela a seguir.

**Tabela 6.3 – Associação Código da Categoria Funcional - Função no Sistema**

| Código da Categoria Funcional | Função no Sistema |
|---|---|
| A1 | Atendente |
| A2 | Caixa |
| B1 | Supervisor |
| C1 | Auditor |

Diversas regras são impostas aos funcionários, de acordo com a função que os mesmos possuem. Assim, um usuário que possua a função “Atendente” sobre a aplicação “Gerência Financeira” pode efetuar as operações “Agendar” “TED” (Transferência Eletrônica Disponível) e “DOC” (Documento de Crédito) e sobre a aplicação “Gerência de Clientes” podem efetuar a ação “Abrir Conta Corrente”. Um usuário que possua a função “Caixa” pode realizar operações “Efetuar Pagamentos” sobre a aplicação “Gerência Financeira” e, ainda, exercer todas as funções conferidas a um “Atendente”. Um usuário, com a função “Supervisor”, pode realizar as operações “Autorizar” “DOC” e “TED” sobre as aplicações “Gerência Financeira”, desde que as operações não tenham sido agendadas por este mesmo usuário e, ainda, sobre a aplicação “Gerência de Clientes” pode realizar a operação “Conceder Limite”, desde que a conta não tenha sido aberta por este mesmo usuário. Um usuário que possua a função “Auditor” não poderá efetuar outras funções exceto aquelas descritas e relacionadas a procedimentos de auditoria sobre transações efetuadas. Assim, sobre os aplicativos “Gerência Financeira” e “Gerência de Clientes” é possível aos “Auditores” realizar a operação “Auditar Transações”. Além destas regras impostas, todas as funções somente poderão ser exercidas no período compreendido entre 10:00 e 16:00; existe, ainda, uma restrição associada à operação “Auditar Transações”, que só poderá ser realizada a partir da sub-rede 192.168.1.0/24. Os mecanismos de controle de acesso não devem gerar impacto excessivo sobre os recursos relativos à estrutura de rede, de forma a prejudicar o desempenho dos sistemas voltados ao negócio, devendo, portanto, tais mecanismos serem avaliados, também, sob este aspecto.

De acordo com as particularidades apresentadas, pode-se notar o envolvimento do administrador dos sistemas, pois cabe a ele estabelecer todos os vínculos e restrições de forma a possibilitar o funcionamento de maneira adequada. Contudo, devido à grande massa de dados envolvidos, corre-se o risco de erro no tratamento da informação. Esse erro pode resultar num excesso de privilégios concedidos a um usuário ou, ainda, a falta de privilégios necessários a outro usuário ou função legítima nos sistemas.

### 6.2.2 Implementação do Modelo Proposto Como Solução ao Cenário

A utilização de uma aplicação externa capaz de controlar o acesso aos diversos sistemas, de forma centralizada, que possibilite a reutilização de políticas destinadas a uma determinada aplicação em outra, torna-se uma boa opção para a resolução do problema apresentado no cenário anterior. Desta forma, as aplicações relativas ao modelo de negócio ficam responsáveis somente por executar as funções para as quais foram desenvolvidas, ficando a cargo de um sistema especialista o controle de acesso de forma única e centralizada.

A arquitetura apresentada no capítulo 5 preenche esses requisitos fazendo uso de um repositório LDAP como ponto central de gerenciamento, possibilitando,ainda, o reaproveitamento dos dados envolvidos, como forma de simplificação dos procedimentos das tarefas administrativas.

O modelo RBPIM [Nabhen 03] possibilita a aplicação de conceitos inerentes ao RBAC, capazes de solucionar muitas questões relacionadas no cenário, que justifica a utilização de papéis na forma definida pelo NIST. Cada função tem associada a ela , autoridade, competência e privilégios, podendo ser relacionada diretamente a um papel RBPIM. Este relacionamento compreende também o conceito herança de papéis, apresentado na figura a seguir:

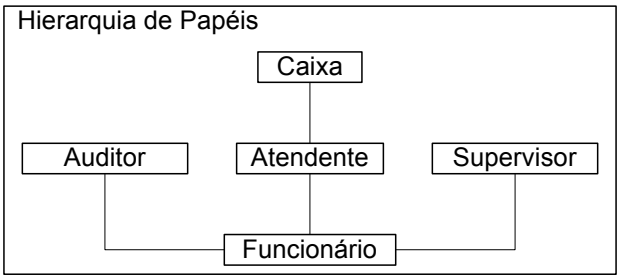


**Figura 6.1 – Hierarquia de Papéis**

De acordo com a figura, todas as funções relativas ao papel “Auditor” herdam as propriedades inerentes ao papel “Funcionário”, o mesmo acontecendo em relação aos papéis “Supervisor” e “Atendente”. Já o papel “Caixa”, herda as características e propriedades relativas ao papel “Atendente” e por conseqüência também as relativas ao papel “Funcionário”. Assim, o modelo apresentaria a tabela a seguir relativa aos papéis associados aos direitos em relação às aplicações:

**Tabela 6.4 – Hierarquia de Papéis e Direitos Associados às Aplicações**

| Papel | Aplicação Gerência Financeira | Aplicação Gerência de Clientes |
|---|---|---|
| Funcionário | | |
| Atendente | Agendar TED<br>Agendar DOC | Abrir Conta Corrente |
| Caixa | Agendar TED<br>Agendar DOC<br>Efetuar Pagamentos | Abrir Conta Corrente |
| Auditor | Auditar Transações | Auditar Transações |
| Supervisor | Autorizar TED<br>Autorizar DOC | Conceder Limite |

Outro requisito atendido pelo modelo, é a separação entre tarefas. Através da implementação de separações estáticas (SSD) é possível evitar que um mesmo usuário possa ter associado a ele papéis conflitantes em relação ao modelo de negócio. É possível, por exemplo, definir que um usuário que possua um papel "Auditor" seja impedido de ter atribuído a ele um papel "Caixa". Desta forma, a regra de negócio impede que eventualmente um auditor possa "Conceder Limite" a ele mesmo. A implementação de separações dinâmicas (DSD) evita que um determinado usuário, sob o escopo de uma sessão, possa ativar papéis conflitantes ao modelo de negócio. Assim, por exemplo, um usuário associado aos papéis "Atendente" e "Supervisor" poderia, eventualmente, ativar, simultaneamente, os dois papéis, sendo permitido, então, "Abrir Conta Corrente" e "Conceder Limite". A forma, definida pelo modelo, que possibilita este controle, define um nome para a separação, seu tipo (estática ou dinâmica) e relaciona, ainda, os papéis envolvidos e a cardinalidade relativa ao conjunto. A cardinalidade define o número máximo de elementos que podem estar contidos no conjunto. Este valor corresponde à cardinalidade menos uma unidade (cardinalidade-1). E pode ser verificada através das tabelas a seguir:

**Tabela 6.5 – Exemplo Aplicação de Separação Estática (SSD)**

| Nome da Separação | Papéis | Cardinalidade |
|---|---|---|
| SSD 01 | Auditor – Atendente | 2 |
| SSD 02 | Auditor – Supervisor | 2 |
| SSD 03 | Auditor – Caixa | 2 |

Nesta tabela foi evidenciada a impossibilidade de associar a um usuário, simultaneamente, os conjuntos de papéis "Auditor" e "Atendente", "Auditor" e "Supervisor" e "Auditor" e "Caixa". Sob o ponto de vista do modelo, esta avaliação leva em consideração a prioridade relativa a cada um dos papéis. Assim, ao serem avaliados os papéis relativos a um usuário, ainda que haja uma atribuição indevida seja pelo administrador do sistema seja por uma herança de papéis não verificada, o algoritmo avalia os papéis conflitantes atribuindo somente aquele de maior prioridade.

**Tabela 6.6 – Exemplo Aplicação de Separação Dinâmica (DSD)**

| Nome da Separação | Papéis | Cardinalidade |
|---|---|---|
| DSD 01 | Supervisor – Atendente | 2 |

Apesar de contar com os mesmos campos que a separação estática, a separação dinâmica é avaliada em relação a uma sessão do usuário, impedindo, neste caso, que sejam ativados os papéis "Supervisor" e "Atendente" em uma mesma sessão. Sob o ponto de vista do modelo, esta chamada é indevida e, por esta razão, a solicitação de ativação do conjunto de papéis é rejeitada.

Filtros de tempo e relacionados a endereços IP também são disponibilizados pelo modelo, permitindo que os papéis sejam ativados somente no intervalo de tempo relacionado. Os filtros de tempo podem, também, compor as condições necessárias para a validação das permissões de acesso da mesma forma que os filtros IP. Assim, é possível compor condições de avaliação das permissões de acesso de acordo com diversos elementos relativos aos pacotes IP, ao tempo, a classes e atributos, na forma de variáveis explícitas. Desta forma a restrição imposta no cenário em relação ao acesso do papel "Auditor" somente ser permitido a partir da rede 192.168.1.0/24 também pode ser satisfeita.

O último requisito a ser atendido por um modelo de controle de acesso, para capacitá-lo a atender ao cenário proposto, diz respeito a critérios de utilização da rede. O modelo permite, também, que este critério seja atendido através da abordagem provisionamento adotada. Através desta abordagem, é possível carregar o servidor de aplicações com as políticas necessárias a seu funcionamento localmente e, assim, evitar a troca de informações entre cliente de políticas PEP e servidor de políticas PDP a cada avaliação do controle de acesso.

### 6.2.3 Representação do Modelo de Informação Sobre o Repositório LDAP

A estratégia de mapeamento do modelo de informação, apresentado no capítulo 4, sobre o repositório de políticas LDAP, segue a estratégia adotada pelo RBPIM [Nabhen 03] associada à definição apresentada no PCELS [Reyes 04], como forma de representação do modelo PCIMe [RFC 3460]. A estratégia relativa às regras de composição das condições seguiu a forma "Composição Reutilizável Formada a Partir de Condições Simples Específicas", apresentada no capítulo 3, sessão 3.4.1.

Os papéis RBAC representados no modelo, são uma extensão da forma prevista pelo NIST, pois são obtidos a partir da composição de condições simples. Uma destas condições é que o atributo *BusinessCategory,* relativo à classe *Person* (CIM), deva ter um valor relativo à categoria funcional definida no cenário (A1, A2, B1, C1). A outra condição avaliada é o período de tempo representando pela classe *PolicyTimePeriodCondition*. As permissões são também obtidas a partir de composição de condições simples, a partir de variáveis explicitas (classe.atributo) relativas aos mais diversos elementos, períodos de tempo e filtros IP.

A figura a seguir apresenta a DIT do diretório utilizado neste estudo de caso. Os principais contêineres são apresentados com seus respectivos atributos RDN. Os objetos que representam usuários do sistema são armazenados no contêiner *People* (ou=*People*, dc=Banco ABC). No contêiner Aplicativos (ou=Aplicativos, dc=Banco ABC) são armazenados aplicativos de acordo com as definições apresentadas no modelo CIM. A partir do contêiner Modelo RBPIM (ou=Modelo RBPIM, dc=Banco ABC), são armazenados cinco outros contêineres, que têm como característica serem reutilizáveis, de acordo com a proposição apresentada no modelo PCIMe [RFC 3460]. Os contêineres *Time Period* (*pcimReusableContainer=TimePeriod Container*, ou=Modelo RBPIM, dc=Banco ABC) e *IP Filter* (*pcimReusableContainer=IPFilter Container*, ou=Modelo

RBPIM, dc=Banco ABC) armazenam, respectivamente, objetos relativos a filtros de tempo e pacotes IP. Os contêineres *Objects* e *Users* armazenam objetos do tipo *CompoundPolicyCondition* compostos por um, ou diversos, objetos de condições simples. No contêiner *PolicyGroup* são agrupadas as políticas de acordo com os papéis (*roles*) de política definidos. Os contêineres *Roles* e *Permissions* agrupam objetos capazes de representar os diversos papéis RBAC e as diversas permissões, relativas a esses papéis RBAC, associadas aos aplicativos controlados.

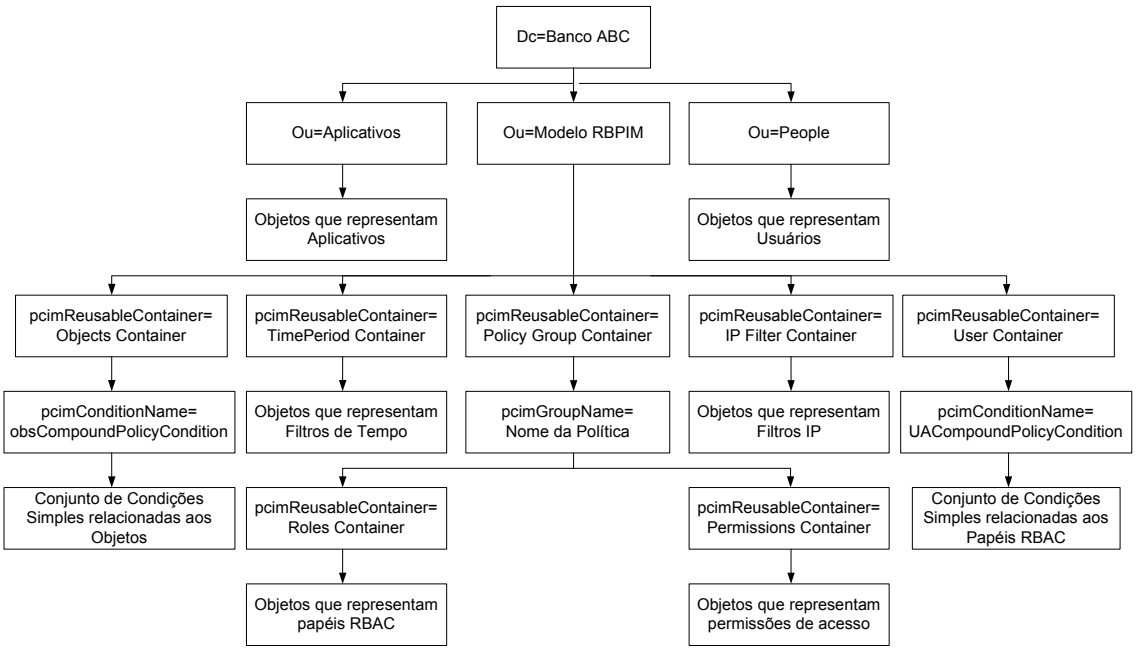


**Figura 6.2 – DIT do diretório Banco ABC**

A implementação da DIT, relativa ao Banco ABC, pode ser verificada a partir da figura a seguir, que mostra a composição de um papel (“Auditor”), bem como as permissões associadas a este papel (“AUD”).

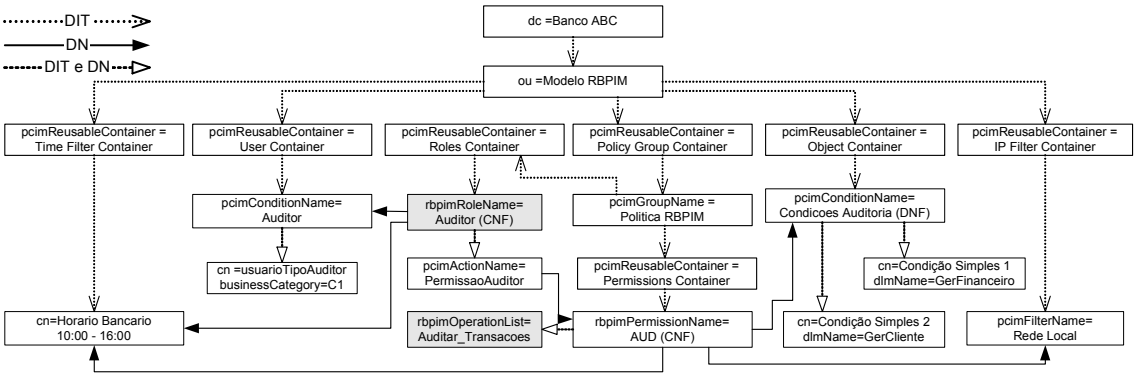


**Figura 6.3 – Composição Papel Auditor e Permissões Relativas a AUD**

De acordo com a figura, para que seja constituído o papel “Auditor”, utiliza-se um conjunto de condições na forma CNF *(rbpimRoleName ”Auditor” = (pcimConditionName=Auditor AND cn=Horário Bancário))* apontadas através de referências DN. Estes relacionamentos podem ser verificados, também, a partir do segmento do arquivo *Ldif* do Banco ABC relativo à composição do papel “Auditor” apresentado a seguir.

```
dn: rbpimRoleName=Auditor, pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM,
        pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimActionList: pcimActionName=PermissaoAuditor, rbpimRoleName=Auditor, pcimreusablecontainername=Roles Container,
        pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimPriority: 1
pcimConditionList: pcimConditionName=Auditor, pcimreusablecontainername=User Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=Horario Comercial, pcimreusablecontainername=Time Perio d Container, ou=Modelo RBPIM, dc=Banco ABC
rbpimRoleName: Auditor
rbpimInheritedRoles: rbpimRoleName=Papel14, ou=Agencia_01, dc=Banco ABC
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicySet
objectClass: pcimPolicySetAssociation
objectClass: pcimPolicyRule
objectClass: rbpimRole
pcimPolicySetDN: rbpimRoleName=Auditor, pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM,
        pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimRuleEnabled: 1
pcimPolicySetName: Auditor
```

**Figura 6.4 – LDIF Banco ABC para papel "Auditor"**

Da mesma forma que o papel, as operações disponíveis são obtidas através de combinação de condições. Assim, no exemplo, a permissão "AUD" é obtida pela composição de condições na forma CNF *(rbpimPermissionName "AUD = (pcimConditionName=Condições Auditoria AND cn=Horário Bancário AND pcimFilterName=Horário Bancario))* também apontadas por referências DN". O fragmento *Ldif* do Banco ABC relativo à permissão "AUD" pode ser visto a seguir.

```
dn: rbpimPermissionName=AUD, pcimreusablecontainername=Permissions Container,pcimGroupName=PoliticaRBPIM,
        pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: pcimConditionName=Condicoes Auditoria, pcimreusablecontainername=Objects Container,
        ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: pcimfiltername=rede local, pcimreusablecontainername=IP Filter Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=Horario Bancario, pcimreusablecontainername=Time Period  Container, ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionName: AUD
pcimConditionListType: 2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicyRule
objectClass: rbpimPermission
pcimActionList: pcimActionName=OperacoesAuditor, rbpimPermissionName=AUD, pcimreusablecontainername=Permissions Container,
        pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
```

**Figura 6.5 – LDIF Banco ABC para Permissão "AUD"**

A possibilidade de reutilização de contêineres permite uma simplificação do modelo, uma vez que se pode, através de referências DN, utilizar um mesmo conjunto de condições a fim de representar diferentes objetos. Esta característica também pode ser observada em relação à condição "Horário Bancário", que foi utilizado, tanto para a composição do papel, como para a composição das permissões.

## 6. 3 Análise de Escalabilidade Implementação do Protótipo

O protótipo apresentado nesta seção tem como função avaliar a proposta apresentada por este trabalho, principalmente, em relação ao provisionamento das políticas, verificando, ainda, a funcionalidade dos algoritmos, API e mensagens COPS-PR.

### 6.3.1 Protótipo

O protótipo segue a estrutura definida na arquitetura de implementação apresentada no capítulo 5, de acordo com a figura a seguir:

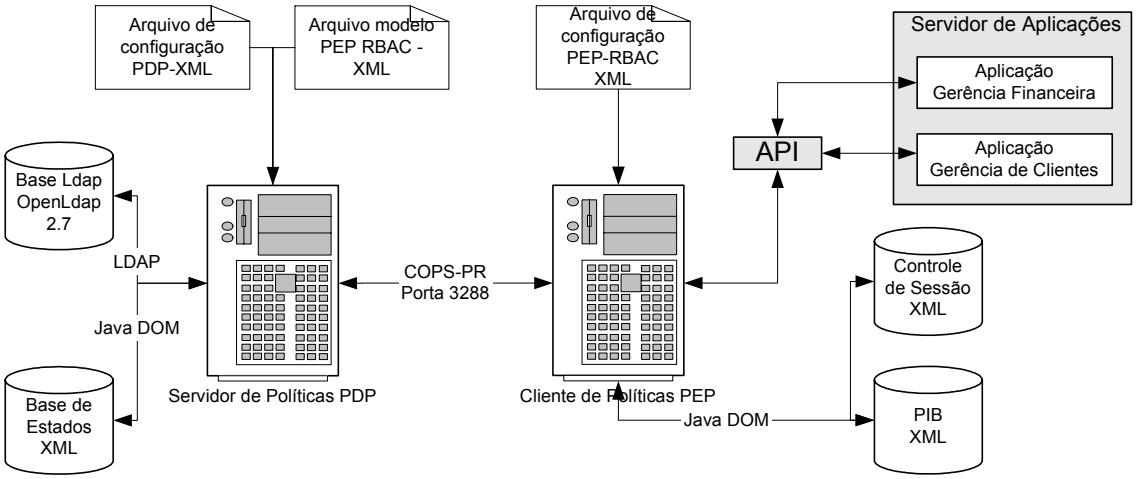


**Figura 6.6 – Arquitetura de Implementação do Protótipo**

O protótipo, totalmente desenvolvido a partir da plataforma *JAVA 2 Standard Edition v.1.4.2*, apresenta no PDP, uma biblioteca de comunicação com o PEP, desenvolvida a partir das definições relativas aos protocolos COPS [RFC 2748] e COPS-PR [RFC 3084], outra duas bibliotecas estão disponíveis no PDP. A primeira estabelece a comunicação entre o PDP e a base de dados LDAP, a partir da implementação *jldap,* e a segunda estabelece comunicação entre o PDP e a base de estados, a partir de implementação Java DOM. No PEP, a biblioteca de comunicação com o PDP foi desenvolvida nos mesmos moldes que no PDP, a partir das definições COPS e COPS-PR. A segunda biblioteca disponibilizada faz a comunicação entre o PEP e o controle de sessão, e entre o PEP e a PIB. Um conjunto de API's é disponibilizado como forma do PEP comunicar-se com as aplicações externas.

### 6.3.2 Processo de Avaliação do Protótipo

O processo de avaliação do protótipo será efetuado em duas etapas:

- Primeiramente, será avaliada a carga da PIB no PEP. A carga será feita variando-se o número dos objetos relativos a papéis RBAC, permissões RBAC, separações estáticas e dinâmicas e objetos de composição de condições. Serão medidas, para cada uma destas variações, o tempo de carga e o tamanho relativo à PIB carregada. Será apresentada, ainda, a medida relativa à carga de um objeto, do tipo usuário, na PIB de forma incremental, relatando o tempo de carga e tamanho da PIB.
- A segunda etapa de avaliação corresponde ao desempenho relativo a API. Nesta fase, serão avaliadas as API's disponibilizadas.

## 6. 4 Resultados Apresentados

A partir da definição dos critérios de medição de resultados apresentados na sessão anterior, foi avaliado o modelo sob o ponto de vista funcional e em relação ao desempenho. No aspecto funcional, foi verificada a exatidão na carga da PIB, a troca de mensagens COPS-PR entre PEP e PDP e os algoritmos envolvidos nas API's disponibilizadas, especificamente, para o controle de acesso sobre as aplicações. Sob o aspecto desempenho, serão verificados os tempos de carga e os tamanhos das PIB's disponibilizadas, tanto de forma integral (PIB inicial completa), quanto de forma incremental (PIB de usuário).

### 6.4.1 Avaliação Funcional

A avaliação funcional do modelo foi realizada verificando-se a consistência das informações obtidas em relação às API's disponibilizadas e, também, aos dados trocados entre o PEP e o PDP, através do protocolo COPS-PR.

Inicialmente, o cliente de políticas PEP, identificado como PepRBAC definido como tipo 0x4002 (cliente RBAC), solicita a abertura de conexão com o PDP através da mensagem COPS-PR – OPN. Como o servidor de políticas foi preparado para responder a políticas RBAC e, como a identificação do cliente foi confirmada, foi concedida a abertura de conexão retornando ao PEP a mensagem COPS-PR – CAT.

O grande teste de funcionalidade, a carga da PIB no PEP, iniciou-se com a solicitação de políticas pelo PEP através da mensagem COPS-PR – REQ na qual informa ao PDP as suas capacidades e papéis suportados. Ao receber a mensagem, o PDP verifica as funcionalidades suportadas e, principalmente, os papéis de política suportados pelo PEP. A partir destas definições é montado todo o pacote de decisão, COPS-PR – DEC, que corresponde à PIB que será provisionada ao PEP. O PEP recebe, então, a mensagem e decodifica-a, preenchendo a PIB. Após estes passos, foi possível avaliar positivamente a carga da PIB, uma vez que todos os dados necessários ao provisionamento foram recebidos de forma adequada, viabilizando a aplicação do modelo.

Após a carga da PIB, foram avaliadas cada uma das API's, principalmente a relacionada à abertura de sessão (*createSession(in:userdId)*), uma vez que, não existindo o usuário na PIB, seria necessária sua carga na forma de um provisionamento incremental. Tanto a verificação da funcionalidade da API em relação à abertura da sessão, como em relação à carga de usuário de forma incremental, foram executadas com exatidão, o mesmo ocorrendo em relação à API de seleção de papéis (*selectedRoles(in:sessionId, roles[])*) . A API que avalia o acesso de um determinado usuário em relação a uma operação sobre uma aplicação (*checkAccess(in:sessionId, action, filter, ipfilter)*) também foi verificada, confirmando positivamente sua funcionalidade. A API de encerramento de sessão *(closeSection(in:sessionId))* correspondeu, também, aos critérios avaliados encerrando e removendo a sessão apresentada.

A avaliação funcional do modelo proposto mostrou-se positiva, uma vez que cada um dos componentes pode ser verificado individualmente, de forma efetiva, de modo a permitir validar toda a proposta apresentada.

### 6.4.2 Avaliação de Desempenho em relação à Carga do PEP

A primeira avaliação apresentada é em relação ao tempo de carga e tamanho da PIB, de acordo com a variação do número de papéis RBAC armazenados no repositório. Nesta avaliação, foi possível verificar o crescimento da PIB em relação ao número de papéis apresentados. Pode-se verificar que, apesar do número de papéis dobrar a cada avaliação, o mesmo não ocorreu com o tempo de carga. Em relação ao tempo de provisionamento de usuário, foi possível verificar a pequena influência deste com a variação do número de papéis.

Os itens da tabela correspondem aos objetos relacionados e identificados no modelo funcional, sessão 4.2 e os dados correspondem à tabela a seguir:

**Tabela 6.7 – Avaliação de Desempenho Variação de Papéis**

| **PRMS** | **OBJS** | **OPS** | **DSD** | **SSD** |
|---|---|---|---|---|
| 6 | 3 | 8 | 1 | 3 |

| **Número de Papéis** | **Média do Tempo de Carga Total (ms)** | **Desvio Padrão** | **Tam PIB (kbytes)** | **Média do Tempo de Carga de Usuário (ms)** | **Desvio Padrão** | **Tam PIB (kbytes)** |
|---|---|---|---|---|---|---|
| 10 | 2178,253 | 462.193 | 16 | 313,081 | 98,812 | 17 |
| 20 | 2661,079 | 403,648 | 22 | 365,791 | 121,220 | 24 |
| 40 | 3904,441 | 367,866 | 34 | 536,204 | 112,119 | 39 |
| 80 | 6713,184 | 584,268 | 58 | 1077,816 | 363,882 | 68 |

Obs. Foram avaliadas 15 (quinze) amostras para cada conjunto de papéis

Para simplificação da análise é apresentado, a seguir, um gráfico que mostra a variação do tempo de provisionamento, tanto total, como de usuário em relação ao número de papéis.

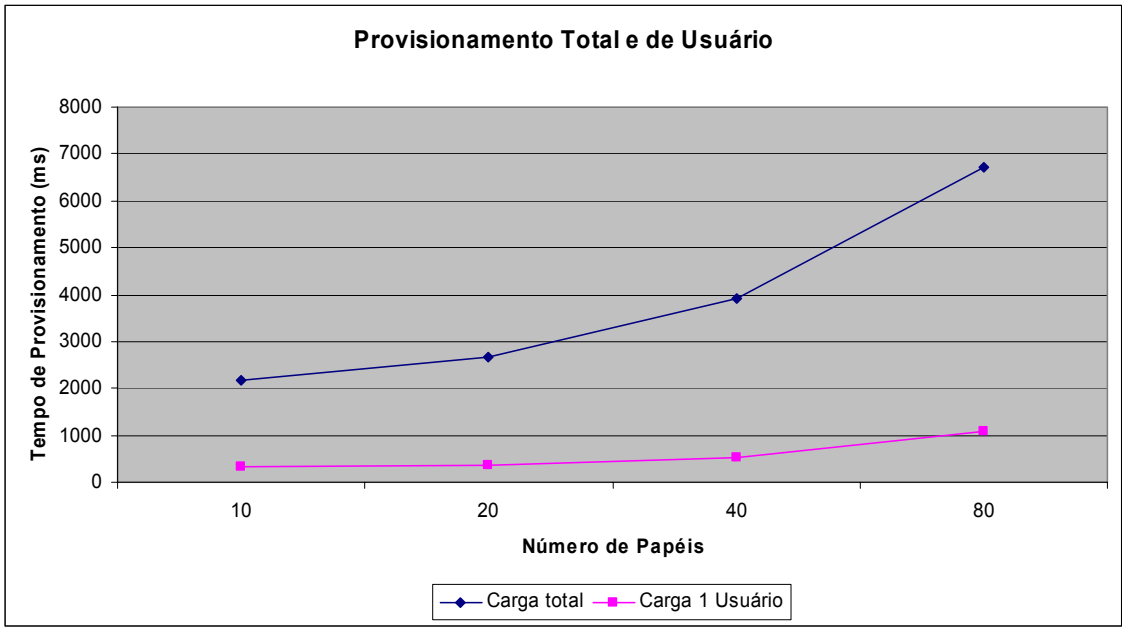


**Figura 6.7 – Variação do Tempo de Provisionamento em Relação ao Número de Papéis**

A tabela 6.7 apresenta, ainda, dados relativos ao crescimento da PIB em relação à variação do número de papéis. Assim, a seguir é apresentado o gráfico correspondente a essa análise.

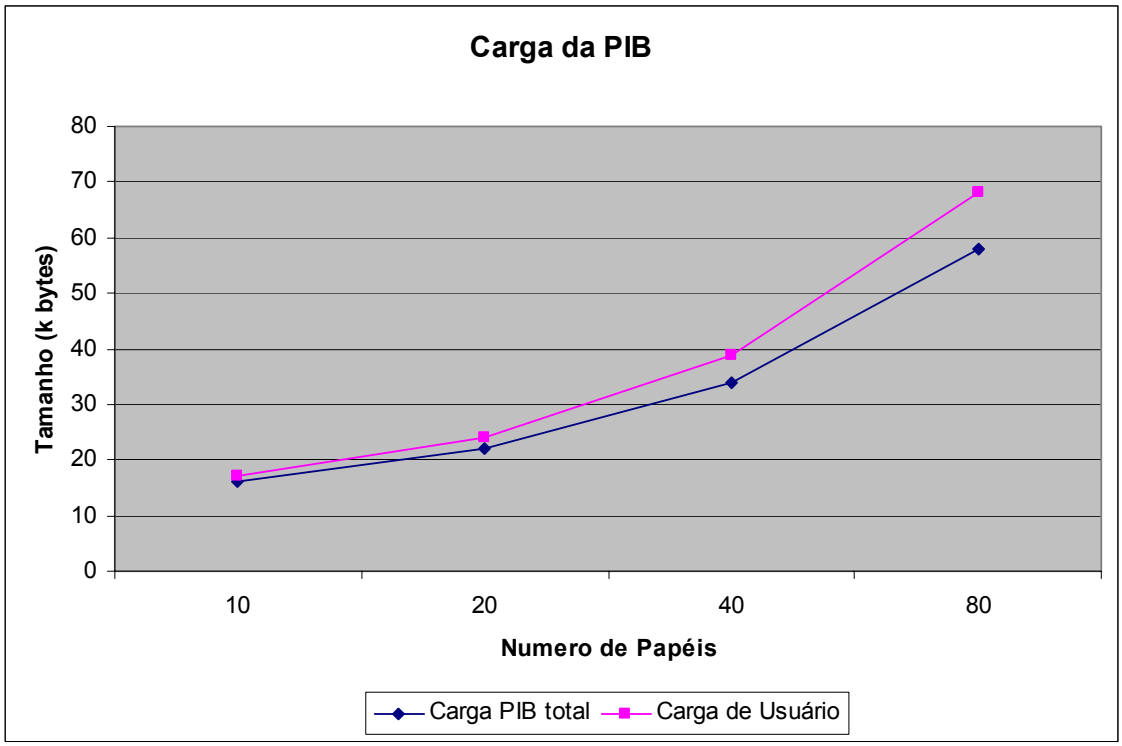


**Figura 6.8 – Carga da PIB em Relação à Variação do Número de Papéis**

A segunda avaliação apresentada é em relação ao tempo de carga e tamanho da PIB, de acordo com a variação do número de objetos armazenados no repositório. Considerando a dependência entre os diversos objetos no modelo, para que essa análise fosse feita, foi necessário variar todos os objetos envolvidos. Assim, a tabela a seguir mostra o comportamento do modelo em relação à variação dos objetos PRMS, OBJS e OPS, mantendo-se os demais objetos fixos.

**Tabela 6.8 – Avaliação de Desempenho Variação de Objetos PRMS, OBJS e OPS**

| **Número de Papéis** | | | **DSD** | | | **SSD** | | |
|---|---|---|---|---|---|---|---|---|
| 20 | | | 1 | | | 3 | | |
| **PRMS** | **OBJS** | **OPS** | **Média do Tempo de Carga Total (ms)** | **Desvio Padrão** | **Tam PIB (kb)** | **Média do Tempo de Carga de Usuário (ms)** | **Desvio Padrão** | **Tam PIB (kb)** |
| 10 | 7 | 12 | 2831,18 | 447,0734 | 24 | 382,294 | 63,0473 | 24 |
| 20 | 17 | 22 | 3068,9 | 550,5357 | 30 | 433,25 | 67,2245 | 30 |
| 40 | 37 | 42 | 4343,43 | 1774,778 | 42 | 462,167 | 93,3671 | 42 |
| 80 | 77 | 82 | 8020,6 | 445,4594 | 67 | 481,9 | 51,7825 | 67 |
| **Obs.** Foram avaliadas 15 (quinze) amostras para cada conjunto de PRMS, OBJS, OPS | | | | | | | | |

Pode-se verificar que, apesar do número de objetos quase dobrar em cada análise, o mesmo não ocorreu em relação ao tempo de provisionamento total. Em relação ao tempo de provisionamento de usuários, foi possível verificar que a variação destes objetos não influiu no tempo de carga. Esse comportamento pode ser visualizado no gráfico a seguir.

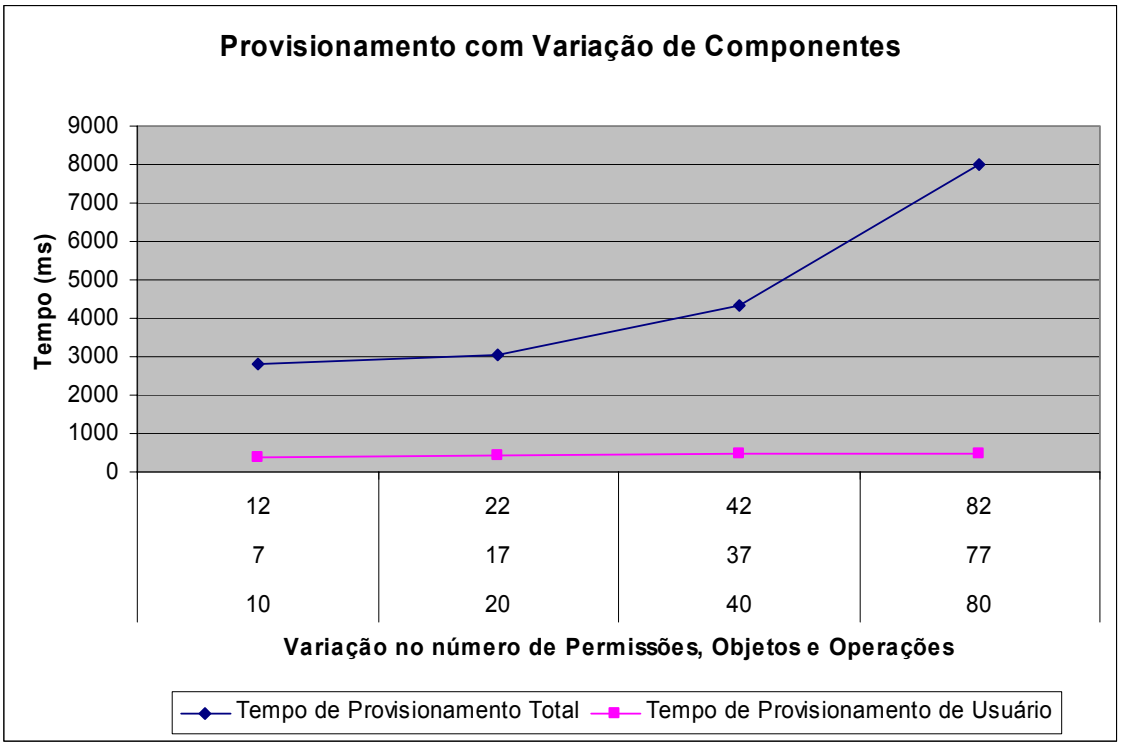


**Figura 6.9 – Variação do Tempo de Carga da PIB em Relação à Variação do Número de Objetos**

A avaliação da carga da PIB em relação à variação no número de objetos PRMS, OBJS e OPS pode, também, ser verificada no gráfico a seguir. Como já citado anteriormente, a forma como a informação é provisionada, fazendo referência aos objetos já existentes, contribui para que o crescimento da PIB não acompanhe, diretamente, o crescimento do número de objetos. Assim, mesmo dobrando o número de objetos, a PIB não dobrará de tamanho. Outra característica importante, aqui observada, é que a carga de um usuário pouco influencia no tamanho final da PIB.

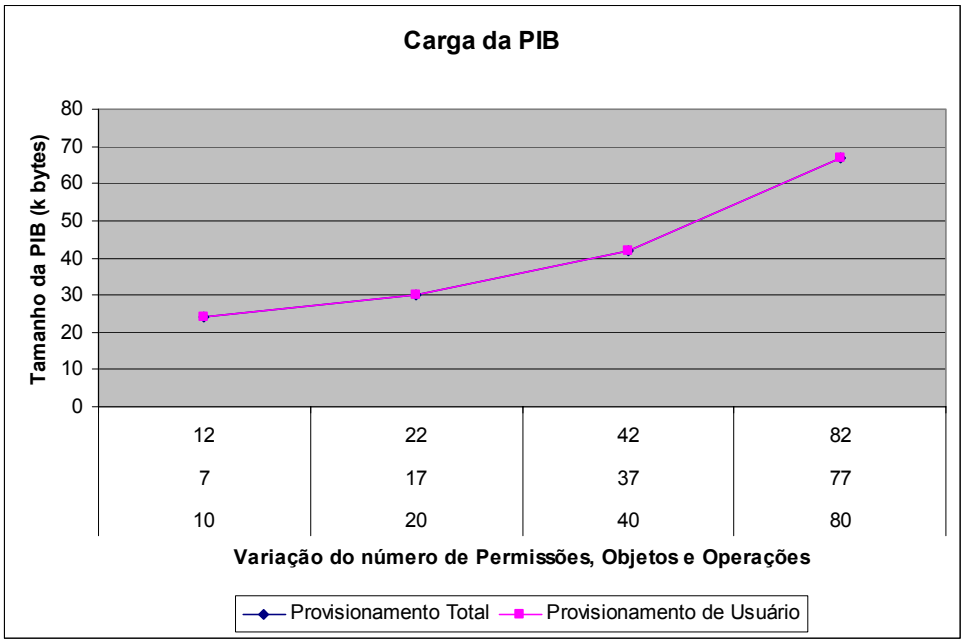


**Figura 6.10 – Carga da PIB em Relação à Variação do Número de Objetos**

A próxima avaliação visa verificar a influência da variação do número de objetos relativos a ocorrências de Separação Dinâmica (DSD), com o tempo de carga total e de usuário e, ainda, qual a variação relativa ao tamanho da PIB. A tabela a seguir apresenta a visão do comportamento do modelo em relação a estas variações.

**Tabela 6.9 – Avaliação de Desempenho para Variação de DSD**

| **Número de Papéis** | **PRMS** | **OBJS** | **OPS** | **SSD** |
|---|---|---|---|---|
| 20 | 6 | 3 | 8 | 3 |

| **DSD** | **Média do Tempo de Carga Total (ms)** | **Desvio Padrão** | **Tam PIB (kbytes)** | **Média do Tempo de Carga de Usuário (ms)** | **Desvio Padrão** | **Tam PIB (kbytes)** |
|---|---|---|---|---|---|---|
| 10 | 2760,467 | 444,745 | 25 | 498,1333 | 236,48 | 25 |
| 20 | 2643,467 | 124,299 | 29 | 449,3333 | 112,96 | 29 |
| 40 | 3190,533 | 459,766 | 36 | 643,4667 | 294,05 | 36 |
| 80 | 3384,8 | 414,236 | 51 | 537,4667 | 180,61 | 51 |
| **Obs.** Foram avaliadas 15 (quinze) amostras para cada conjunto de DSD | | | | | | |

O gráfico que representa o tempo de carga em relação à variação no número de objetos do tipo DSD é apresentado a seguir. Neste gráfico, fica evidente que a variação no número de DSD não influencia no tempo de carga de usuário.

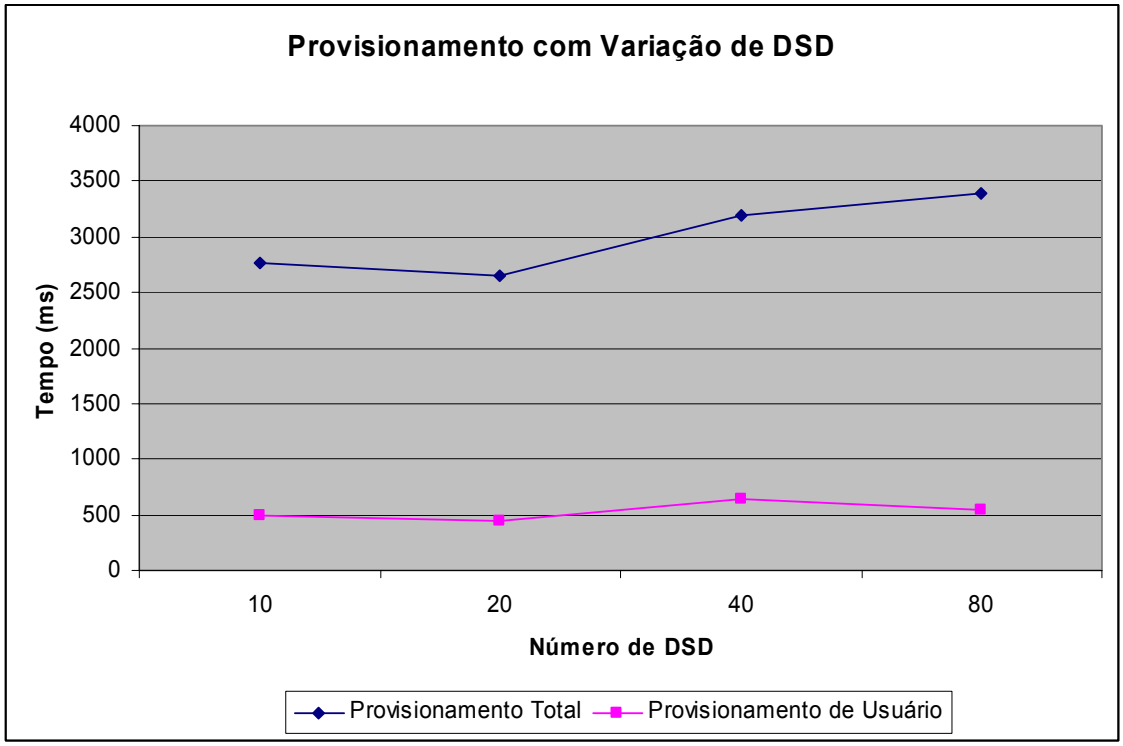


**Figura 6.11 – Tempo de Carga da PIB em Relação à Variação do Número de DSD**

Em relação à carga da PIB, pode-se notar, no gráfico a seguir, a pouca incidência da variação deste objeto sobre a sua carga total. Outra característica importante, é a pouca influência sofrida na carga da PIB em relação ao provisionamento de um usuário simples, possibilitando a representação das duas análises sob uma mesma curva.

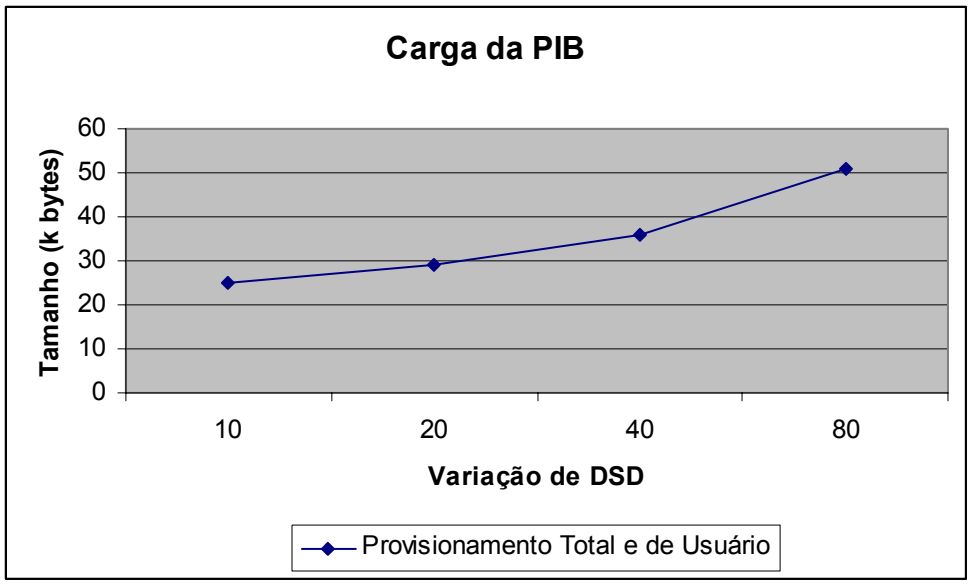


**Figura 6.12 – Carga da PIB em Relação à Variação de DSD**

A última análise efetuada é em relação à variação de objetos do tipo que representam ocorrências de separações estáticas (SSD). Nesta análise, foi verificado o impacto da variação deste objeto em relação ao tempo de provisionamento e em relação ao tamanho da PIB provisionada. A tabela que apresenta o comportamento obtido é apresentada a seguir.

**Tabela 6.10 – Avaliação de Desempenho para Variação de SSD**

| **Número de Papéis** | **PRMS** | **OBJS** | **OPS** | **DSD** |
|---|---|---|---|---|
| 20 | 6 | 3 | 8 | 1 |

| **SSD** | **Média do Tempo de Carga Total (ms)** | **Desvio Padrão** | **Tam PIB (kbytes)** | **Média do Tempo de Carga de Usuário (ms)** | **Desvio Padrão** | **Tam PIB (kbytes)** |
|---|---|---|---|---|---|---|
| 10 | 2970,4 | 493,740 | 22 | 442,1333 | 87,506 | 22 |
| 20 | 3001,533 | 615,464 | 22 | 447,5333 | 88,938 | 22 |
| 40 | 2833,333 | 369,849 | 22 | 449,3333 | 71,004 | 22 |
| 80 | 2631,733 | 327,395 | 22 | 590,6667 | 119,51 | 22 |
| **Obs.** Foram avaliadas 15 (quinze) amostras para cada conjunto de SSD | | | | | | |

Podemos verificar que a ocorrência da variação no número de objetos SSD pouco influencia no tempo total de provisionamento da PIB. Este comportamento é natural se verificarmos que este objeto não é provisionado ao PEP. Este é utilizado no provisionamento de usuário, momento em que são avaliados quais papéis da sua lista relativa sofrem influência por separação estática, removendo aqueles que incidirem e que possuírem maior prioridade. É importante ressaltar que, quanto maior a prioridade definida ao papel, menor sua importância para o modelo. O gráfico a seguir mostra esse comportamento.

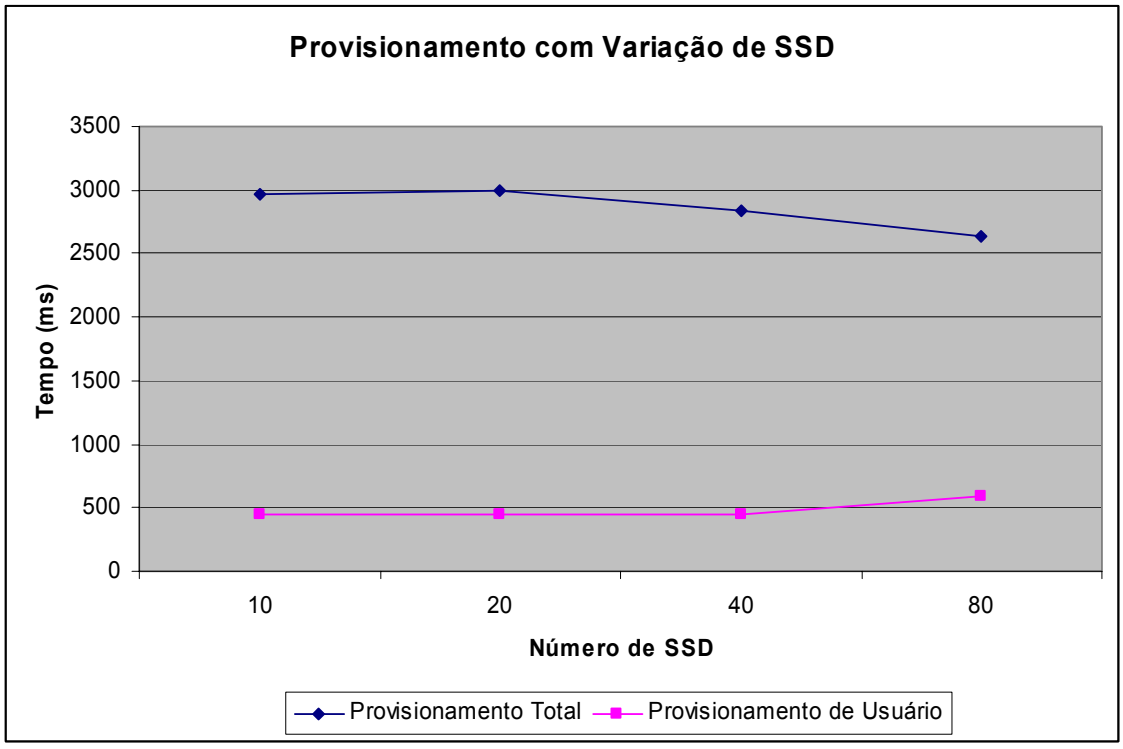


**Figura 6.13 – Tempo de Carga da PIB em Relação à Variação do Número de SSD**

O comportamento do Provisionamento, com Variação de SSD e diminuição no tempo de provisionamento, mesmo com o aumento do número de papéis, é devido ao corte do número de papéis a serem provisionados, característico da análise de separação estática, associado ao algoritmo de otimização utilizado.

A carga total da PIB, devido à informação apresentada anteriormente, não sofre influência dos objetos SSD, sendo desnecessária a apresentação do gráfico relativo à variação de carga.

### 6.4.3 Avaliação de Desempenho em relação a API

Nesta análise, todas as API's disponibilizadas terão seu desempenho verificado de acordo com um conjunto de objetos definido. No primeiro cenário, o modelo é disponibilizado com dez (10) papéis RBAC (Roles), seis (6) permissões (PRMS), três (3) objetos (OBJS), oito (8) operações (OPS), uma (1) ocorrência de separação dinâmica (DSD) e três (3) ocorrências de separação estática. No processo de análise, as API's serão medidas em relação a um usuário que possui, somente, um papel associado e também, com um usuário que possui dois papéis associados, a fim de verificar o impacto das mesmas em relação ao número de papéis. Para efeito de análise, também será considerada a carga da PIB que, no cenário 1, corresponde a um tamanho de 17 Kbytes para a abordagem relativa ao usuário com um papel associado e 24 Kbytes para o usuário com dois papéis associados. A tabela de comportamento relativo a cada uma das API's, de acordo com o primeiro cenário, é apresentada a seguir.

**Tabela 6.11 – Analise de Desempenho API Cenário 1**

| Cenário 1 | | | | | | |
|---|---|---|---|---|---|---|
| Roles | PRMS | OBJS | OPS | DSD | SSD | |
| 10 | 6 | 3 | 8 | 1 | 3 | |
| API | | | Usuário com um papel associado (ms) | | Usuário com dois papéis associados (ms) | |
| | | | Média | Desvio Padrão | Média | Desvio Padrão |
| createSection | Provisionando usuário | | 1008,8 | 4,585692 | 1013,267 | 10,71359 |
| | Usuário Provisionado (t x 100) | | 311,8 | 62,79126 | 358,6667 | 80,84524 |
| selectedRoles (t x 100) | | | 667,3333 | 179,7008 | 707,3333 | 208,2741 |
| checkAccess (t x 100) | | | 1055,533 | 213,1029 | 1091,6 | 144,4456 |
| closeSection (t x 100) | | | 225,0667 | 58,33459 | 227,5333 | 54,18997 |
| **Obs.** Foram avaliadas 15 (quinze) amostras para cada API | | | | | | |

Podemos notar, nos resultados apresentados, a característica do modelo provisionamento. Na primeira criação de sessão, quando o usuário não está provisionado, o tempo é relativamente incidente. Contudo, após o provisionamento, o tempo diminui sensivelmente sendo necessário, inclusive, incrementar o número de

simulações, em cem (100) vezes, para que fosse possível medi-lo de forma confiável. Esta forma de medida, incrementando em cem (100) vezes, foi também efetuado em relação às API's *selectedRoles*, *checkAccess* e *closeSection,* como forma de estabelecer um valor médio mais confiável.

O gráfico de acompanhamento do comportamento relativo as API's é disponibilizado para análise, a seguir.

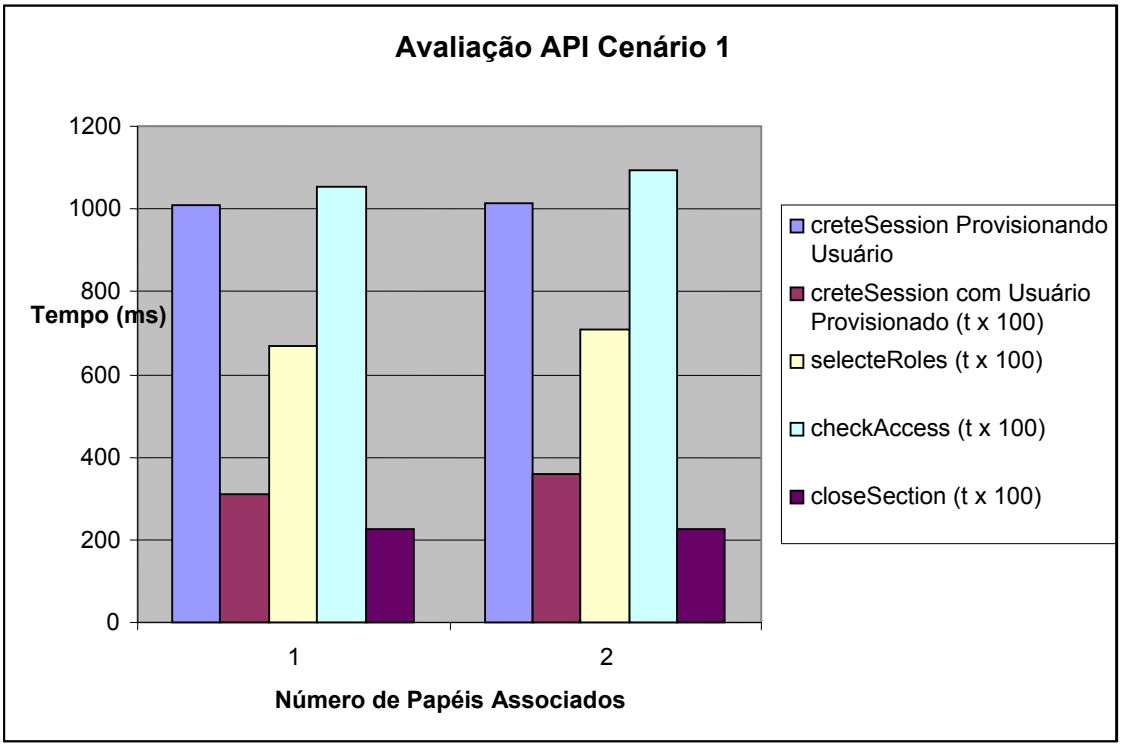


**Figura 6.14 – Gráfico de Análise API's Cenário 1**

O cenário 2 é apresentado na tabela a seguir. Nela podemos notar o incremento substancial dos objetos envolvidos e, também, o aumento do tempo relativo a API *checkAccess* que, exatamente por sua característica, necessita avaliar diversos elementos contidos na PIB. O tamanho da PIB é, também, um dado interessante e para o usuário com um papel associado corresponde a 68 Kbytes e para o usuário com dois papéis associados corresponde a 86Kbytes.

**Tabela 6.12 – Analise de Desempenho API Cenário 2**

| Cenário 2 | | | | | | |
|---|---|---|---|---|---|---|
| Roles | PRMS | OBJS | OPS | DSD | SSD | |
| 20 | 80 | 77 | 82 | 10 | 10 | |
| API | | | Usuário com um papel associado (ms) | | Usuário com dois papéis associados (ms) | |
| | | | Média | Desvio Padrão | Média | Desvio Padrão |
| createSection | Provisionando usuário | | 1115,231 | 963,4016 | 903,6 | 485,5696 |
| | Usuário Provisionado (t x 100) | | 795 | 172,635 | 745,6667 | 28,99918 |
| selectedRoles (t x 100) | | | 1973,077 | 366,4102 | 1816 | 164,0035 |
| checkAccess (t x 100) | | | 13263,31 | 2590,087 | 13734,13 | 2591,359 |
| closeSection (t x 100) | | | 472 | 19,48076 | 480,8 | 45,73401 |
| **Obs.** Foram avaliadas 15 (quinze) amostras para cada API | | | | | | |

O gráfico de desempenho relativo à tabela anterior é apresentado a seguir, evidenciando a API *checkAccess*. Contudo, é necessário reforçar, novamente, que cada API aqui apresentada, exceto *createSession* provisionando o usuário, corresponde a cem (100) aplicações da API. Assim, o tempo de acesso máximo da API *checkAccess* corresponde a um valor próximo a 160 ms.

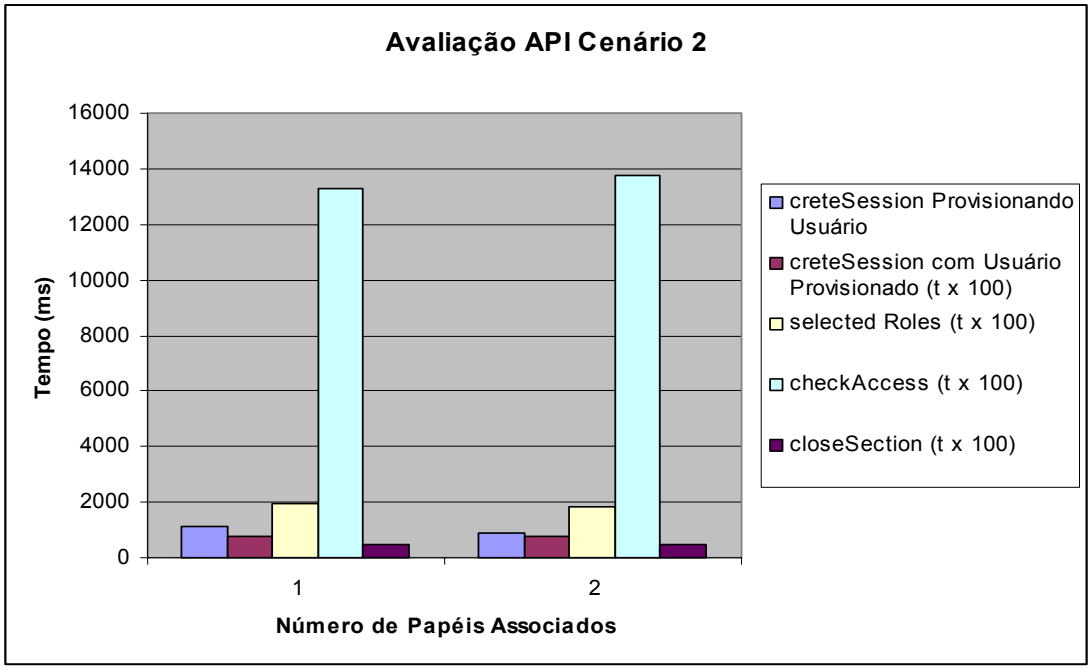


**Figura 6.15 – Gráfico de Análise API's Cenário 2**

### 6.4.4 Comparativo de Desempenho – Modelos Provisionamento e *Outsourcing*

Considerando o exposto neste capítulo, é possível traçar um paralelo entre as abordagens provisionamento e *outsourcing,* uma vez que o trabalho apresentado em [Nabhen 03a] traz estatísticas relativas às API's que possibilitam a comparação.

Em [Nabhen 03a] a avaliação das API's é proposta através de baterias de testes efetuadas com atrasos de tempo definidos. É importante, no entanto, reforçar que todas as API's no modelo *outsourcing* são diretamente dependentes do ambiente de rede, o que não ocorre em relação ao modelo provisionamento, pois uma vez provisionado, o acesso à informação é feito diretamente pela aplicação.

A tabela a seguir apresenta os valores comparativos entre os modelos. Em relação ao modelo provisionamento os valores definidos como "menor tempo", correspondem àqueles presentes na tabela 6.11 referentes a usuário com um papel associado. O maior tempo corresponde ao obtido na tabela 6.12 relativo ao usuário com dois papéis associados. Os valores informados para o modelo *outsourcing* foram obtidos nas tabelas 7.12, 7.13 e 7.14 de [Nabhen 03a].

**Tabela 6.13 – Comparativo Provisionamento x Outsourcing**

| | **Modelo Provisionamento** | | **Modelo *Outsourcing*** | |
|---|---|---|---|---|
| **API** | **Menor Tempo (ms)** | **Maior Tempo (ms)** | **Menor Tempo (ms)** | **Maior Tempo (ms)** |
| ***createSession*** | 2,60 | 30,59 | 71,10 | 1881,03 |
| ***selectedRoles*** | 5,00 | 35,10 | 22,70 | 612,74 |
| ***checkAccess*** | 9,31 | 158,92 | 28,89 | 692,19 |

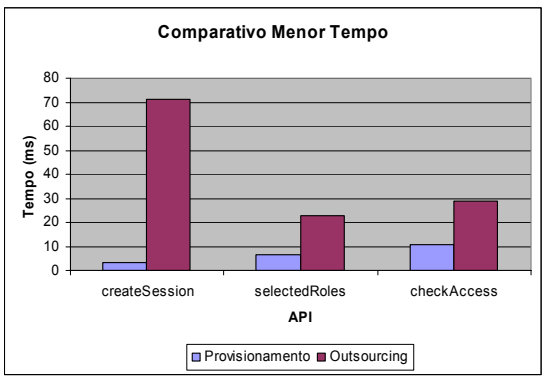


**Figura 6.16 – Comparativo Menor Tempo**

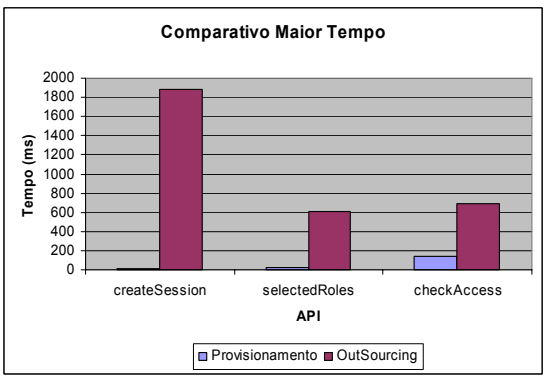


**Figura 6.17 – Comparativo Maior Tempo**

## 6. 5 Conclusão

Este capítulo apresentou a descrição do modelo sob o ponto de vista de sua representação em um diretório e, a partir desta descrição, propôs um estudo de caso onde se pode avaliar a funcionalidade do modelo proposto, bem como seu desempenho em relação a um ambiente de testes. A implementação do modelo, em linguagem Java, permitiu avaliar todos os componentes propostos, desde algoritmos, mensagens COPS-PR, API's e entidades PEP - PDP, permitindo, ainda, verificar a validade do conceito de provisionamento de políticas aplicado a métodos de controle de acesso na forma definida pelo modelo RBAC.

O comparativo entre as abordagens Provisionamento e *Outsourcing* encerrou o capítulo demonstrando a viabilidade funcional do modelo como método de controle de acesso.

# Capítulo 7

# Conclusões e Trabalhos Futuros

Este trabalho apresentou um sistema (*framework*) completo para gerenciamento de políticas de controle de acesso para sistemas distribuídos. O sistema proposto foi desenvolvido de maneira a respeitar um conjunto de requisitos apresentados a seguir:

- Utilizar, na medida do possível, padrões existentes como forma de simplificar e padronizar o desenvolvimento adotando, para tal, linhas de pesquisa existentes e conhecidas, tanto no mercado como na comunidade científica, garantindo assim a aplicabilidade e possível continuidade do trabalho.
- Ser independente de fabricante ou de plataformas específicas, principalmente, devido à heterogeneidade característica dos atuais ambientes computacionais.
- Ser escalável
- Não degradar o desempenho relativo às aplicações gerenciadas
- Adotar RBAC como modelo de controle de acesso, uma vez que esse modelo tem se mostrado um eficiente método de controle de acesso, tendo sua utilização uma forte tendência de crescimento nas redes de computadores, principalmente, por facilitar a descrição de políticas de segurança a partir de regras de negócio
- Permitir a reutilização de elementos de política e das definições de usuários e recursos gerenciados, como forma de simplificar o gerenciamento dos grandes ambientes computacionais, permitindo uma redução substancial no esforço dedicado à administração

Como forma de satisfazer aos requisitos apresentados, o desenvolvimento do sistema proposto adotou elementos fundamentais apresentados a seguir:

- Implementação de acordo com a abordagem PBNM (*Policy Based Network Management*), a partir da utilização de uma arquitetura composta por PEP e PDP. Duas grandes vantagens da abordagem PBNM são a possibilidade de representação das políticas através de linguagens de alto nível e a possibilidade destas mesmas políticas serem interpretadas, ou carregadas, de acordo com a arquitetura de gerenciamento adotada [Linington 04].

- Utilização da abordagem de gerenciamento de políticas “Provisionamento”, associado ao protocolo de distribuição de políticas COPS-PR, como forma de possibilitar escalabilidade e melhoria de desempenho.
- Utilização de um método de controle de acesso, denominado RBPIM, constituído a partir de um *superset* do modelo RBAC proposto pelo NIST que, para satisfazer aos requisitos relativos ao reaproveitamento de informações de gerenciamento, foi definido através de especializações de classes definidas pelos modelos CIM e PCIM.

A partir da constituição do sistema, diversas contribuições foram geradas, destacando-se as apresentadas a seguir:

- Diversos aprimoramentos no modelo RBPIM originalmente proposto por [Nabhen 03]. Estes melhoramentos tiveram como função adaptar o modelo à abordagem provisionamento, e melhorar a reutilização das informações de gerenciamento.
- Elaboração de PIB específica para representar o modelo RBAC estendido, que recebeu o nome de RBAC-PIB
- Construção de um PDP capaz de gerar a RBAC-PIB a partir das definições do modelo RBPIM aprimorado
- Construção de um PEP capaz de interpretar as informações da RBAC-PIB. Este PEP oferece as informações de decisão às aplicações que necessitem do controle de acesso RBAC, a partir de um conjunto de API’s definidas pelo padrão proposto pelo NIST [Ferraiolo 01]

Os requisitos relativos a desempenho e escalabilidade puderam ser avaliados e validados a partir do estudo de caso apresentado no Capítulo 6. Neste capítulo foram apresentadas avaliações relativas à carga do PEP de acordo com a variação de diversos objetos associados, e também, o tempo utilizado para a carga destes componentes. Outro dado importante apresentado, foi o comparativo entre as abordagens *outsourcing* e provisionamento, no qual fica evidente o ganho de desempenho do provisionamento sobre o *outsourcing*.

Em relação à abordagem provisionamento adotada, foi possível avaliá-la como uma opção viável aplicada ao método de controle de acesso em ambientes distribuídos, principalmente em relação ao baixo impacto nas trocas de informações, utilizando o ambiente de rede. O conceito de provisionamento adotado, em que somente um conjunto contendo informações essenciais é transferido no primeiro momento, e que informações adicionais, tais como usuários, podem ser provisionados de forma incremental, permite que a utilização dos recursos de rede possa ser otimizada a ponto de não representar uma grande influência sobre o sistema. Desta forma, apesar de haver a possibilidade de uma alta demanda de informações no primeiro conjunto transferido, este pico pode ser assimilado, se comparado com a taxa média de utilização. Comparando-se com a abordagem *outsourcing*, verifica-se que nesta há uma sensível ocupação da rede, uma vez que, para qualquer interação relativa ao controle de acesso, são trocadas mensagens entre aplicação PEP e PDP, o que não acontece em relação à abordagem provisionamento, pois, após a carga, grande parte das interações acontece, somente, entre a aplicação e o PEP de forma local.

Devido à característica relativa ao provisionamento em que: "toda a informação necessária está disponível ao PEP após a inicialização", é garantido um certo nível de controle de falhas. Caso seja interrompida a comunicação entre o PEP e o PDP, estando a informação de política provisionada, é possível ao PEP continuar seu funcionamento de maneira quase normal, ficando impossibilitado, no entanto, de obter as informações de políticas provisionadas de forma incremental. De forma prática, no caso de já ter sido carregado na PIB, de forma incremental, o conjunto de usuários da aplicação sobre a qual o controle de acesso atuará, mesmo perdendo a conexão com o PDP, o controle de acesso continuará a ser executado como se não houvesse ocorrido falha alguma. Apesar desta característica, é necessário como continuidade do trabalho que sejam definidas formas de atuação e melhoria nos métodos relacionados ao controle de falhas.

Ainda como continuidade do trabalho, devem ser definidas diretivas que possibilitem melhorias e adequações relacionadas à segurança, principalmente em relação às informações trocadas no provisionamento. Métodos como verificação do PEP e PDP através de prova mútua, canais seguros (IpSec) podem ser avaliados. A implementação de mecanismos que possibilitem a substituição de políticas provisionadas, já definido no COPS-PR, é um ponto que, também, deve ser explorado. Outra necessidade, é a implementação de mensagens de controle, como *keep-alive* (KA), utilizadas pelo protocolo COPS-PR.

A partir da apresentação deste trabalho, associado ao conceito de *Capabilities*[5], apresentado pelo *framework* PIB [RFC 3318], que possibilita provisionar um conjunto de políticas de acordo com a capacidade suportada pelo dispositivo, é possível abrir espaço para indagações relativas ao desenvolvimento da utilização do conceito de provisionamento além dos domínios do IETF até agora definidos, sendo possível vislumbrar o controle de acesso, a partir de dispositivos que implementam diferentes tipos de mecanismos, não somente direcionado a aplicações, caso abordado por este trabalho, mas também, por extensão, a serviços de rede, acessos a *filesystem*, composição de perfil de usuário, regras para utilização de *firewall*, entre outros.

[5] O conceito de *Capabilities,* apresentado pelo *framework* PIB [RFC 3318], possibilita provisionar um conjunto de políticas de acordo com a capacidade suportada pelo dispositivo, como por exemplo, interfaces de rede que suportem classificação de entrada ou que possam ser configuradas de forma estática. Sob o escopo deste trabalho o conceito de capabilities foi associado à capacidade suportada pelo PEP.

# Referências Bibliográficas


[Boutaba 01] Andreas Polyrakis, Rauf Boutaba, "COPS-PR with Meta-Policy Support", draft, March-Abril 2002, draft-boutaba-copsprmp-00.txt, October 2001

[Boutaba 02] Andreas Polyrakis, Rauf Boutaba, "The Meta-Policy Information Base", IEEE Network, March-Abril 2002

[Cisco 97] Cisco Systems, Inc., Microsoft and Cisco Collaborate to Establish Directory Services Standard [Acesso em 11/10/2002] <http://www.cisco.com/warp/public/146/pressroom/1997/may97/1870.html

[Cisco 97a] Cisco Systems, Inc., Directory Services Collaboration - Frequently Asked Questions (FAQ) [Acesso em 11/10/2002] <http://www.cisco.com/warp/public/146/pressroom/1997/may97/1871.html

[Cisco 97b] Cisco Systems, Inc., Q & A Directory Enabled Networks---Frequently Asked Questions [Acesso em 11/10/2002] <http://www.cisco.com/warp/public/cc/techno/network/dirserv/den/prodlit/drsrv_qp.htm>

[Damianou 01] Damianou, M., Dulay, N., Et Al, "The Ponder Policy Specification Language", Policy 2001, 2001

[Dinesh 02] Dinesh C. Verma, "Simplifying Network Administration Using Policy-Based Management", IEEE Network, March-Abril 2002

[DMTF 03] DMTF, "Common Information Model (CIM) Standards", <http://www.dmtf.org/standards/cim/>, 2003

[DMTF 03a] DMTF, "CIM Schema: Version 2.8.1 (Final) ", <http://www.dmtf.org/standards/cim/cim_schema_v28>

[DMTF 98] Distributed Management Task Force, Inc., Directory Enabled Networks Initiative Formally Submitted to DMTF [Acesso em 11/10/2002] <http://www.dmtf.org/newsroom/releases/1998_09_28_1.php>

[DMTF 98a] Distributed Management Task Force, Inc., "Directory Enabled Network (DEN) Initiative" [Acesso em 11/10/2002] < http://www.dmtf.org/standards/standard_den.php>

[DMTF 98b] Distributed Management Task Force, Inc., DEN FAQ [Acesso em 11/10/2002] <http://www.dmtf.org/about/faq/den.php>

[DMTF] <http://www.dmtf.org>

[Eyers 03] Eyers, A., Moody, K., "Policy Contexts: Controlling Information Flow in Parameterised RBAC", Proceedings Policy 2003, 2003

[Ferraiolo 01] Ferraiolo, D. F., Sandhu, R .S., Serban, G., "A Proposed Standard for Role-Based Access Control", ACM Transactions on Information System Security, Vol. 4, No. 3, August 2001

[Ferraiolo 95] Ferraiolo, D. F., Cugini, J. A., Khun, D. R., "Role-Based Access Control (RBAC): Features and Motivations", NIST, 11st Annual Computer Security Applications Proceedings, 1995

[Flegkas 01] P.Flegkas, P. Trimintzios, G. Pavlou, I. Andrikopoulos, C.F. Cavalcanti, "On Policy-Based Extensible Hierarchical Network Management in QoS-enabled IP Networks", Proc. of the Workshop on Policies for Distributed Systems and Networks, Springer-Verlag (LNCS series), January 2001

[Haixin 00] Haixin, D., Jianping, W., Xing, L., "Policy-Based Access Control Framework for Large Networks", IEEE, 2000

[Hayton 98] Hayton, R. J., Bacon, M., Et Al, "Access Control in an Open Distributed Enviroment", IEEE Symposium on Secutity and Privacy, 1998

[He 03] He, Q., "Privacy Enforcement with an Extended Role-Based Access Control Model", North Carolina State University , April 2003

[IETF 02] The Internet Engineering Task Force - IETF, "IETF Policy Framework"
[Acesso em 11/10/2002]
<http://www.ietf.org/html.charters/policy-charter.html>

[Iphighway 00] IPHighway, "Introduction To Policy-Based Networking And Quality of Service", January 2000 [Acesso em 01/03/2003]
<http://www.inf.ufrgs.br/granville/QoSManagement/manutencao/iphighway/Policy-based%20Networking%20and%20QoS.pdf>

[Kanada 01] Yasusi Kanada, Brian J. O'Keefe, "Diffserv Policies and Their Combinations in a Policy Server"
[Acesso em 01/03/2003]
<http://www.kanadas.com/activenet/APNOMS01ext.pdf>

[Kosiur 01] Kosiur, D., Herzog, S., "Understanding Policy-based Networking", Wiley Networking Council Books, John Wiley & Sons Inc, February 2001

[Linington 04] Linington, P. F., "Deriving Authority from Security Policy", Final Report Draft 2004,
<http://www.cs.kent.ac.uk/projects/policy/PolicyReportDraft7.pdf>

[Mallenius 00] Mallenius, Seppo, "The COPS (Commom Open Policy Service) Protocol", 08/11/2000 [Acessado em 11/10/2002]
<http://www.cs.helsinki.fi/u/kraatika/Courses/QoS00a/mallenius.pdf>

[Martin 99] Martin, Jean-Cristophe, Policy-Based Networks, Sun BluePrints OnLine - October 1999 [Acesso em 20/10/2002]
<http://www.sun.com/solutions/blueprints/1099/policy.pdf>

[Moody 02] Moody, K., Bacon, J., Yao, M., "A Model of OASIS Role-Based Access Control and its Support for Active Security", ACM Transactions, November 2002

[Moore 02] Moore, B., “Policy Core Information Model Extensions”, <draft-ietf-policy-pcim-ext-08-txt>, 28/05/2002

[Moyer 01] Moyer, M. J., Ahamad, M., “Generalized Role-Based Access Control”, IEEE 2001, 2001

[Nabhen 03] Nabhen, R., Jamhour, E., Maziero, C., “A Policy Based Framework for RBAC”, DSOM 2003, 2003

[Nabhen 03a] Nabhen, R., “RBPIM: Um Modelo de Políticas de Segurança Baseado em Papéis”, Pontifícia Universidade Católica do Paraná, 2003

[Oasis 03] Oasis, “eXtensible Access Control Markup Language (XACML)”, Version 1.03, OASIS Standard, February 2003 <http://www.oasis-open.org>

[Oasis 04] Oasis, “XACML Profile for Role Based Access Control (RBAC)”, draft, February 2004 <http://www.oasis-open.org>

[Ortega 03] Ortega, A., S., “PFIM: Um Modelo de Informação para Gerenciamento de Filtros de Pacotes Baseado em Políticas”, Pontifícia Universidade Católica do Paraná, 2003.

[Reyes 04] Pana, M., Reyes, A., Barba, A., Moron, D., Brunner, M., “Policy Core Extension Lightweight Directory Access Protocol Schema”, draft draft-reyes-policy-core-ext-schema-06.txt, June 2004 <http://www.ietf.org/internet-drafts/draft-reyes-policy-core-ext-schema-06.txt>

[RFC 1558] Howes, T., “A String Representation of LDAP Search Filters”, RFC 1558, December 1993

[RFC 1777] Yeong, W., Howes, T., Killie, S., “LightWeight Directory Access Protocol”, Request for Comments, RFC 1777, March 1995

[RFC 1959] Howes, T., Smith, M., “An LDAP URL Format”, RFC 1959, June 1996

[RFC 2251] Wal, M., Howes, T., Kille, S. “LightWeiht Directory Access Protocol (v3)”, Request for Comments, RFC 2251, December 1997

[RFC 2254] Howes, T., “The String Representation of LDAP Search Filters”, RFC 2254, December 1997

[RFC 2434] Narten, T., Alvestrand, H., “Guidelines for Writing an IANA Considerations Section in RFCs”, RFC 2434, October 1998

[RFC 2748] Boyle, J., Cohen, R., Durham, D., Herzog, S., Raja, R. and A. Sastry, "The COPS (Common Open Policy Service) Protocol", RFC 2748, January 2000

[RFC 2749] Herzog, S., Boyle, J., Cohen, R., Durham, D., Rajan, R., Sastry, A., “COPS usage for RSVP”, RFC 2749, January 2000

[RFC 2753] Yavatar, R. , Pendarakis, D., Guerin, “A Framework for Policy-Based Admission Control”, RFC 2753, January, 2001

[RFC 2753] Yavatkar, R., Pendarakis, D., Guerin, R., “A Framework for Policy-Based Admission Control”, RFC 2753, January 2000

[RFC 2940] Smith, A., Partain, D., Seligson, J., "Definitions of Managed Objects for Common Open Policy Service (COPS) Protocol Client", RFC 2940, October 2000

[RFC 3060] Moore, B., Ellesson, E., Strassner, J., Westerinen, A., "Policy Core Information Model – Version 1 Specification", RFC 3060, February 2001

[RFC 3084] Chan, K. , Seligso, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., Smith, A., "COPS Usage for Policy Provisioning (COPS-PR)", RFC 3084, March 2001

[RFC 3159] McGlorie, K., Fine, M., Seligson, J., Chan, K., Hahn, S., Sahita, R., Reichmeyer, F., "Structure of Policy Provisioning Information (SPPI)", RFC 3159, August 2001

[RFC 3198] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., Waldbusser, S., "Terminology for Policy-Based Management", RFC 3198, November 2001

[RFC 3318] Sahita, R. Hahn, S., Chan, K., McCloghrie, K., "Framework Policy Information Base", RFC 3318, March 2003

[RFC 3460] Moore, B., "Policy Core Information Model Extensions", RFC 3460, January 2003

[RFC 3703] Strassner, J., Ellesson, E., Moore, B., Moats, R., "Policy Core Lightweight Directory Access Protocol (LDAP) Schema", RFC 3703, February 2004

[Sandhu 00] Sandhu, R. S., "A Proposed Standard for Role-Based Access Control", Nist, December 2000

[Sandhu 94] Sandhu, R. S., Samarati, P. S., "Access Control: Principles and Pratice", IEEE Comunications, September 1994

[Sarolahti 00] Sarolahti, P., "COPS Usage", 31/12/2000 [Acessado em 11/10/2002] http://www.cs.helsinki.fi/u/kraatika/Courses/QoS00a/sarolahti.pdf

[Shepard 00] Susan J. Shepard, "Policy Base Networks: Hype and Hope", IT Pro, January – February 2000

[Sloman 02] Sloman, M., Lupu, E., " Security and Management Policy Information", IEEE Network, March 2002

[Strassner 98] Judd, Steven & Strassner, John, "Directory Enabled Networks - Information Model and Base Schema (Version 3.0c)", 21.7.1998 [Acesso em 11/10/2002] <http://murchiso.com/den/specifications/directory-enabled-networks-v3-lastcall.pdf

[Toktar 04] Toktar, E., Jamhour, E., Maziero, C., "RSVP Policy Control Using XACML", IEEE Policy 2004, June 2004

[Xinzhong 98] Xinzhong Yu, "Directory Enabled Networks", December 1998 [Acesso em 11/10/2002] <http://www.tml.hut.fi/Studies/Tik-110.300/1998/Essays/den.html>

[Ylitalo 00] Ylitalo, Katri, Policy Core Information Model, November 2000 [Acesso em 11/10/2002] <http://www.tml.hut.fi/Studies/Tik-110.300/1998/Essays/den.html>

# Anexo A – Arquivos LDIF Estudo de Caso Banco ABC

O formato de arquivos LDIF (*Ldap Data Interchange Format*) corresponde à definição de entradas de diretório a partir de arquivos texto, e são compostos das seguintes informações:

- DN – *Distinguished Name* – Corresponde a uma entrada que define, unicamente, o objeto no diretório. Isto é, não é possível haver duas entradas no diretórioo com o mesmo DN.
- Classes do Esquema (*object Class*) – Cada classe definida no esquema corresponde à composição, uso e atributos obrigatórios e opcionais que podem ser utilizados para representar uma entrada no diretório.
- Atributos – Correspondem, efetivamente, aos valores definidos aos objetos contidos no diretório e especificam uma determinada informação sobre uma entrada.

A listagem, apresentada a seguir, corresponde ao arquivo BancoABC.ldif e contém entradas utilizadas no estudo de caso e validação do protótipo, apresentados no trabalho.

---

```
dn: dc=Banco ABC
dc: Banco ABC
objectClass: dcobject
objectClass: top

dn: cn=Manager, dc=Banco ABC
objectClass: organizationalrole
cn: Manager

dn: ou=People, dc=Banco ABC
ou: People
objectClass: top
objectClass: organizationalunit
```

```
dn: cn=Carlos,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: A1
sn: Machado
cn: Carlos

dn: cn=Ana,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: A1
sn: Fernandes
cn: Ana

dn: cn=Joana,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: A1
sn: Rodrigues
cn: Joana

dn: cn=Rubens,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: A1
sn: Silva
cn: Rubens

dn: cn=Marcos,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: A1
sn: Melo
cn: Marcos

dn: cn=Ailton,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: A1
sn: Freire
cn: Ailton
dn: cn=Matias,ou=People, dc=Banco ABC
```

ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: B1
businessCategory: C1
sn: Oliveira
cn: Matias

dn: cn=Maria,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: A2
sn: Oleg
cn: Maria

dn: cn=Silvia,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: A2
sn: Karsten
cn: Silvia

dn: cn=Vivian,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: A2
sn: Fagundes
cn: Vivian

dn: cn=Pedro,ou=People, dc=Banco ABC
ou: ou=People, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: B1
businessCategory: A1
sn: Batista
cn: Pedro

dn: cn=Carla,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: C1
sn: Kiriale
cn: Carla

dn: cn=Alex,ou=People, dc=Banco ABC
ou: ou=Agencia_01, dc=Banco ABC
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
businessCategory: C1
sn: Lopes
cn: Alex

dn: ou=Aplicativos, dc=Banco ABC
ou: Aplicativos
objectClass: top
objectClass: organizationalunit

dn: dlmName=GerFinanceiro,ou=Aplicativos, dc=Banco ABC
dlmDescription: Aplicativo responsavel pela realizacao das operacoes financeiras do banco
objectClass: top
objectClass: dlm1ManagedElement
objectClass: dlm1ManagedSystemElement
objectClass: dlm1LogicalElement
objectClass: dlm1System
objectClass: dlm1ApplicationSystem
dlmName: GerFinanceiro

dn: dlmName=GerCliente,ou=Aplicativos, dc=Banco ABC
dlmDescription: Aplicativo responsavel pela realizacao das operacoes envolvendo o cadastro de clientes do banco
objectClass: top
objectClass: dlm1ManagedElement
objectClass: dlm1ManagedSystemElement
objectClass: dlm1LogicalElement
objectClass: dlm1System
objectClass: dlm1ApplicationSystem
dlmName: GerCliente

dn: ou=Modelo RBPIM, dc=Banco ABC
ou: Modelo RBPIM
objectClass: top
objectClass: organizationalUnit

dn: pcimreusablecontainername=User Container,ou=Modelo RBPIM, dc=Banco ABC
pcimReusableContainerName: User Container
objectClass: top
objectClass: dlm1managedElement
objectClass: dlm1managedSystemElement
objectClass: dlm1logicalElement
objectClass: dlm1system
objectClass: dlm1admindomain
objectClass: pcimreusablecontainer
objectClass: pcimreusablecontainerauxclass

dn: pcimreusablecontainername=Policy Group Container,ou=Modelo RBPIM, dc=Banco ABC
pcimReusableContainerName: Policy Group Container
objectClass: top
objectClass: dlm1managedElement
objectClass: dlm1managedSystemElement
objectClass: dlm1logicalElement
objectClass: dlm1system
objectClass: dlm1admindomain
objectClass: pcimreusablecontainer
objectClass: pcimreusablecontainerauxclass

dn: pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container,ou=Modelo RBPIM, dc=Banco ABC
objectClass: top
objectClass: pcimpolicy
objectClass: pcimrolecollection
objectClass: pcimpolicyset
objectClass: pcimGroup
objectClass: dlm1managedElement
objectClass: dlm1managedSystemElement
objectClass: dlm1logicalElement
objectClass: dlm1system
objectClass: dlm1admindomain
objectClass: pcimreusablecontainer
objectClass: pcimreusablecontainerauxclass
pcimGroupName: PoliticaRBPIM
pcimRole: politicaRBPIM

dn: pcimreusablecontainername=Roles Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container,ou=Modelo RBPIM, dc=Banco ABC
pcimReusableContainerName: Roles Container
objectClass: top
objectClass: dlm1managedElement
objectClass: dlm1managedSystemElement
objectClass: dlm1logicalElement
objectClass: dlm1system
objectClass: dlm1admindomain
objectClass: pcimreusablecontainer
objectClass: pcimreusablecontainerauxclass

dn: pcimreusablecontainername=Permissions
Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
pcimReusableContainerName: Permissions Container
objectClass: top
objectClass: dlm1managedElement
objectClass: dlm1managedSystemElement
objectClass: dlm1logicalElement
objectClass: dlm1system
objectClass: dlm1admindomain
objectClass: pcimreusablecontainer
objectClass: pcimreusablecontainerauxclass

dn: rbpimPermissionName=AUD,pcimreusablecontainername=Permissions
Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: pcimConditionName=Condições Auditoria,pcimreusablecontainername=Objects
Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: pcimfiltername=rede local, pcimreusablecontainername=IP Filter Container,
ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=Horario Comercial, pcimreusablecontainername=Time Period Container,
ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionName: AUD
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicyRule
objectClass: rbpimPermission
pcimActionList: pcimActionName=OperacoesAuditor, rbpimPermissionName=AUD,
pcimreusablecontainername=Permissions Container, pcimGroupName=PoliticaRBPIM,
pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC

dn:pcimActionName=OperacoesAuditor,rbpimPermissionName=AUD,pcimreusablecontainername=Per
missions Container, pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimOperationList: Auditar_Transacoes
pcimActionName: OperacoesAuditor
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionOrder: 1

dn: pcimreusablecontainername=Time Period Container,ou=Modelo RBPIM, dc=Banco ABC
pcimReusableContainerName: Time Period Container
objectClass: top
objectClass: dlm1managedElement
objectClass: dlm1managedSystemElement
objectClass: dlm1logicalElement
objectClass: dlm1system
objectClass: dlm1admindomain
objectClass: pcimreusablecontainer
objectClass: pcimreusablecontainerauxclass

dn: pcimreusablecontainername=IP Filter Container,ou=Modelo RBPIM, dc=Banco ABC
pcimReusableContainerName: IP Filter Container
objectClass: top
objectClass: dlm1managedElement
objectClass: dlm1managedSystemElement
objectClass: dlm1logicalElement
objectClass: dlm1system
objectClass: dlm1admindomain
objectClass: pcimreusablecontainer
objectClass: pcimreusablecontainerauxclass

dn: pcimfiltername=rede local,pcimreusablecontainername=IP Filter Container,ou=Modelo RBPIM, dc=Banco ABC
pcimFilterName: rede local
pcimIPHdrDestMask: 255.255.255.0
pcimIPHdrDestPortStart: 80
pcimIPHdrSourceMask: 255.255.255.0
pcimIPHdrDestAddress: 192.168.1.10
objectClass: top
objectClass: pcimpolicy
objectClass: pcimfilterentry
objectClass: pcimIPHeaders
pcimIPHdrSourceAddress: 192.168.1.0
cn: rede local
pcimFilterIsNegated: FALSE
pcimIPHdrDestPortEnd: 80

dn: pcimfiltername=Internet,pcimreusablecontainername=IP Filter Container,ou=Modelo RBPIM, dc=Banco ABC
pcimIPHdrSourceMask: 255.255.255.0
objectClass: top
objectClass: pcimpolicy
objectClass: pcimfilterentry
objectClass: pcimIPHeaders
pcimFilterName: Internet
pcimIPHdrDestAddress: ANY
pcimIPHdrSourceAddress: 192.168.1.0
pcimFilterIsNegated: FALSE
cn: internet

dn: cn=filtro internet,pcimreusablecontainername=IP Filter Container,ou=Modelo RBPIM, dc=Banco ABC
pcimFilterEntryList: pcimfiltername=Internet, pcimreusablecontainername=IP Filter Container, ou=Modelo RBPIM, dc=Banco ABC
objectClass: top
objectClass: pcimpolicy
objectClass: pcimconditionauxclass
objectClass: pcimFilterListAuxClass
cn: internet
pcimFilterListName: Filtro internet
pcimFilterDirection: 2

dn: pcimreusablecontainername=Objects Container,ou=Modelo RBPIM, dc=Banco ABC
pcimReusableContainerName: Objects Container
objectClass: top
objectClass: dlm1managedElement
objectClass: dlm1managedSystemElement
objectClass: dlm1logicalElement
objectClass: dlm1system
objectClass: dlm1admindomain
objectClass: pcimreusablecontainer

dn: pcimConditionName=Auditor,pcimreusablecontainername=User Container,ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=usuarioTipoAuditor, pcimConditionName=Auditor, pcimreusablecontainername=User Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionGroupNumber: 1
pcimConditionListType: 2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: pcimcompoundconditionauxclass
pcimConditionName: obsCompoundPolicyCondition
pcimConditionNegated: FALSE

dn: rbpimRoleName=Auditor,pcimreusablecontainername=Roles Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy GroupContainer,ou=Modelo RBPIM, dc=Banco ABC
pcimActionList: pcimActionName=PermissaoAuditor, rbpimRoleName=Auditor, pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimPriority: 1
pcimConditionList: pcimConditionName=Auditor, pcimreusablecontainername=User Container, ou=Modelo RBPIM, dc=Banco ABC
rbpimRoleName: Auditor
rbpimInheritedRoles: rbpimRoleName=Funcionario, pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy  Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicySet
objectClass: pcimPolicySetAssociation
objectClass: pcimPolicyRule
objectClass: rbpimRole
pcimPolicySetDN: rbpimRoleName=Auditor, pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimRuleEnabled: 1
pcimPolicySetName: Auditor
pcimRuleValidityPeriodList: cn=Horario Comercial, pcimreusablecontainername=Time Period Container, ou=Modelo RBPIM, dc=Banco ABC

dn: pcimActionName=PermissaoAuditor,rbpimRoleName=Auditor,pcimreusablecontainername=Roles Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionDN: rbpimPermissionName=AUD, pcimreusablecontainername=Permiss ions Container, pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimActionName: PermissaoAuditor
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionOrder: 1

dn: pcimConditionName=Caixa,pcimreusablecontainername=User Container,ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=usuarioTipoCaixa, pcimConditionName=Caixa, pcimreusablecontainername=User Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionGroupNumber: 1
pcimConditionListType: 2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: pcimcompoundconditionauxclass
pcimConditionNegated: FALSE
pcimConditionName: obsCompoundPolicyCondition

dn: cn=usuarioTipoCaixa,pcimConditionName=Caixa,pcimreusablecontainername=User Container,ou=Modelo RBPIM, dc=Banco ABC
pcimVariableModelProperty: BusinessCategory
pcimStringList: A2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimsimpleconditionauxclass
objectClass: pcimVariable
objectClass: pcimExplicitVariableAuxClass
objectClass: pcimValueAuxClass
objectClass: pcimStringValueAuxClass
objectClass: pcimconditionauxclass
pcimVariableModelClass: inetorgperson
cn: usuarioTipoCaixa

dn: cn=usuarioTipoAuditor,pcimConditionName=Auditor,pcimreusablecontainername =User Container,ou=Modelo RBPIM, dc=Banco ABC
pcimVariableModelProperty: BusinessCategory
pcimStringList: C1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimsimpleconditionauxclass
objectClass: pcimVariable
objectClass: pcimExplicitVariableAuxClass
objectClass: pcimValueAuxClass
objectClass: pcimStringValueAuxClass
objectClass: pcimconditionauxclass
pcimVariableModelClass: inetorgperson
cn: usuarioTipoAuditor

dn: pcimConditionName=Atendente,pcimreusablecontainername=User Container,ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=usuarioTipoAtendente, pcimConditionName=Atendente, pcimreusablecontainername=User Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionGroupNumber: 1
pcimConditionListType: 2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: pcimcompoundconditionauxclass
pcimConditionName: obsCompoundPolicyCondition
pcimConditionNegated: FALSE

dn: cn=usuarioTipoAtendente,pcimConditionName=Atendente,pcimreusablecontainername=User Container,ou=Modelo RBPIM, dc=Banco ABC
pcimVariableModelProperty: BusinessCategory
pcimStringList: A1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimsimpleconditionauxclass
objectClass: pcimVariable
objectClass: pcimExplicitVariableAuxClass
objectClass: pcimValueAuxClass
objectClass: pcimStringValueAuxClass
objectClass: pcimconditionauxclass
pcimVariableModelClass: inetorgperson
cn: usuarioTipoAtendente

dn: pcimConditionName=Supervisor,pcimreusablecontainername=User Container,ou= Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=usuarioTipoSupervisor, pcimConditionName=Supervisor, pcimreusablecontainername=User Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionGroupNumber: 1
pcimConditionListType: 2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: pcimcompoundconditionauxclass
pcimConditionName: Supervisor
pcimConditionNegated: FALSE

dn: cn=usuarioTipoSupervisor,pcimConditionName=Supervisor,pcimreusablecontainername=User Container,ou=Modelo RBPIM, dc=Banco ABC
pcimVariableModelProperty: BusinessCategory
pcimStringList: B1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimsimpleconditionauxclass
objectClass: pcimVariable
objectClass: pcimExplicitVariableAuxClass
objectClass: pcimValueAuxClass
objectClass: pcimStringValueAuxClass
objectClass: pcimconditionauxclass
pcimVariableModelClass: inetOrgPerson
cn: cn=usuarioTipoSupervisor

dn: rbpimRoleName=Funcionario,pcimreusablecontainername=Roles Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container,ou=Modelo RBPIM, dc=Banco ABC
pcimPriority: 5
pcimRuleEnabled: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicySet
objectClass: pcimPolicySetAssociation
objectClass: pcimPolicyRule
objectClass: rbpimRole
rbpimRoleName: Funcionario

dn: rbpimRoleName=Atendente,pcimreusablecontainername=Roles
Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy
Groupontainer,ou=Modelo RBPIM, dc=Banco ABC
pcimActionList: pcimActionName=PermissaoAtendente1, rbpimRoleName=Atendente,
pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM, pcim
reusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimActionList: pcimActionName=PermissaoAtendente2, rbpimRoleName=Atendente,
pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM,
pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimPriority: 4
pcimConditionList: pcimConditionName=Atendente, pcimreusablecontainername=User Container,
ou=Modelo RBPIM, dc=Banco ABC
rbpimRoleName: Atendente
rbpimInheritedRoles: rbpimRoleName=Funcionario, pcimreusablecontainername=Roles Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo
RBPIM, dc=Banco ABC
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicySet
objectClass: pcimPolicySetAssociation
objectClass: pcimPolicyRule
objectClass: rbpimRole
pcimPolicySetDN: rbpimRoleName=Atendente, pcimreusablecontainername=Roles Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo
RBPIM, dc=Banco ABC
pcimPolicySetName: Atendente
pcimRuleEnabled: 1
pcimRuleValidityPeriodList: cn=Horario Bancario, pcimreusablecontainername=Time Period Container,
ou=Modelo RBPIM, dc=Banco ABC

dn:pcimActionName=PermissaoAtendente1,rbpimRoleName=Atendente,pcimreusablecontainername=Ro
les Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionDN: rbpimPermissionName=GF1, pcimreusablecontainername=Permissions Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo
RBPIM, dc=Banco ABC
pcimActionName: PermissaoAtendente1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionOrder: 1

dn: rbpimRoleName=Caixa,pcimreusablecontainername=Roles
Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo
RBPIM, dc=Banco ABC
pcimActionList: pcimActionName=PermissaoCaixa, rbpimRoleName=Caixa,
pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM,
pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimPriority: 3
pcimConditionList: pcimConditionName=Caixa, pcimreusablecontainername=User Container,
ou=Modelo RBPIM, dc=Banco ABC
rbpimRoleName: Caixa
rbpimInheritedRoles: rbpimRoleName=Atendente, pcimreusablecontainername=Roles Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo
RBPIM, dc=Banco ABC
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicySet
objectClass: pcimPolicySetAssociation
objectClass: pcimPolicyRule
objectClass: rbpimRole
pcimPolicySetDN: rbpimRoleName=Caixa, pcimreusablecontainername=Roles Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo
RBPIM, dc=Banco ABC
pcimPolicySetName: Caixa
pcimRuleEnabled: 1
pcimRuleValidityPeriodList: cn=Horario Bancario, pcimreusablecontainername=Time Period Container,
ou=Modelo RBPIM, dc=Banco ABC

dn: pcimActionName=PermissaoCaixa,rbpimRoleName=Caixa,pcimreusablecontainername=Roles
Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionDN: rbpimPermissionName=GF3, pcimreusablecontainername=Permissions Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo
RBPIM, dc=Banco ABC
pcimActionName: PermissaoCaixa
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionOrder: 1

dn: rbpimRoleName=Supervisor,pcimreusablecontainername=Roles
Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy
GroupContainer,ou=Modelo RBPIM, dc=Banco ABC
pcimActionList: pcimActionName=PermissaoSupervisor1, rbpimRoleName=Supervisor ,
pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM,
pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimActionList: pcimActionName=PermissaoSupervisor2, rbpimRoleName=Supervisor ,
pcimreusablecontainername=Roles Container, pcimGroupName=PoliticaRBPIM,
pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimPriority: 2
pcimConditionList: pcimConditionName=Supervisor, pcimreusablecontainername=User Container,
ou=Modelo RBPIM, dc=Banco ABC
rbpimRoleName: Supervisor
rbpimInheritedRoles: rbpimRoleName=Funcionario, pcimreusablecontainername=Roles Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy
Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicySet
objectClass: pcimPolicySetAssociation
objectClass: pcimPolicyRule
objectClass: rbpimRole
pcimPolicySetDN: rbpimRoleName=Supervisor, pcimreusablecontainername=Roles Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo
RBPIM, dc=Banco ABC
pcimPolicySetName: Supervisor
pcimRuleEnabled: 1
pcimRuleValidityPeriodList: cn=Horario Bancario, pcimreusablecontainername=Time Period Container,
ou=Modelo RBPIM, dc=Banco ABC

dn:pcimActionName=PermissaoSupervisor1,rbpimRoleName=Supervisor,pcimreusablecontainername=
Roles Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy
GroupContainer,ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionDN: rbpimPermissionName=GF2, pcimreusablecontainername=Permissions Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo
RBPIM, dc=Banco ABC
pcimActionName: PermissaoSupervisor1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionOrder: 1

dn: rbpimPermissionName=GF1,pcimreusablecontainername=Permissions
Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container,
ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: pcimConditionName=Condicoes Financeiro, pcimreusablecontainername=Objects
Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=Horario Bancario, pcimreusablecontainername=Time Period Container,
ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionName: GF1
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicyRule
objectClass: rbpimPermission
pcimActionList: pcimActionName=OperacoesFinanceiro1, rbpimPermissionName=GF1,
pcimreusablecontainername=Permissions Container, pcimGroupName=PoliticaRBPIM,
pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC

dn: rbpimPermissionName=GF2,pcimreusablecontainername=Permissions
Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container,
ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: pcimConditionName=Condicoes Financeiro, pcimreusablecontainername=Objects
Container,ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=Horario Bancario, pcimreusablecontainername=Time Period Container,
ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionName: GF2
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicyRule
objectClass: rbpimPermission
pcimActionList: pcimActionName=OperacoesFinanceiro2, rbpimPermissionName=GF2,
pcimreusablecontainername=Permissions Container, pcimGroupName=PoliticaRBPIM,
pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC

dn:pcimActionName=OperacoesFinanceiro2,rbpimPermissionName=GF2,pcimreusablecontainername=
Permissions Container,pcimGroupName=PoliticaRBPIM,pcimreusablec ontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimOperationList: AutorizarDOC
rbpimOperationList: AutorizarTED
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionName: OperacoesFinanceiro2
pcimActionOrder: 1

dn: rbpimPermissionName=GC1,pcimreusablecontainername=Permissions Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: pcimConditionName=Condicoes Contabil, pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=Horario Bancario, pcimreusablecontainername=Time Period Container, ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionName: GC1
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicyRule
objectClass: rbpimPermission
pcimActionList: pcimActionName=OperacoesContabil1, rbpimPermissionName=GC1, pcimreusablecontainername=Permissions Container, pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC

dn: rbpimPermissionName=GC2,pcimreusablecontainername=Permissions Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: pcimConditionName=Condicoes Contabil, pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=Horario Bancario, pcimreusablecontainername=Time Period Container, ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionName: GC2
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicyRule
objectClass: rbpimPermission
pcimActionList: pcimActionName=OperacoesContabil2, rbpimPermissionName=GC2, pcimreusablecontainername=Permissions Container, pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC

dn:pcimActionName=OperacoesContabil2,rbpimPermissionName=GC2,pcimreusablecontainername= Permisions Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimOperationList: Conceder Limite
pcimActionName: OperacoesContabil2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionOrder: 1

dn: rbpimPermissionName=GF3,pcimreusablecontainername=Permissions
Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container,
ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: pcimConditionName=Condicoes Financeiro, pcimreusablecontainername=Objects
Container,  ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=Horario Bancario, pcimreusablecontainername=Time Period Container,
ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionName: GF3
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimPolicyRule
objectClass: rbpimPermission
pcimActionList: pcimActionName=OperacoesFinanceiro3, rbpimPermissionName=GF3,
pcimreusablecontainername=Permissions Container, pcimGroupName=PoliticaRBPIM,
pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC

dn:pcimActionName=OperacoesFinanceiro3,rbpimPermissionName=GF3,pcimreusablecontainername=
Permissions Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimOperationList: Efetuar Pagamentos
pcimActionName: OperacoesFinanceiro3
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionOrder: 1

dn:pcimActionName=OperacoesFinanceiro1,rbpimPermissionName=GF1,pcimreusablecontainername=
Permissions Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimOperationList: AgendarDOC
rbpimOperationList: AgendarTED
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionName: OperacoesFinanceiro1
pcimActionOrder: 1

dn: pcimActionName=OperacoesContabil1,rbpimPermissionName=GC1,pcimreusablecontainername=
Permissions Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimOperationList: Abrir Conta
pcimActionName: OperacoesContabil1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionOrder: 1

dn: pcimConditionName=Condicoes Auditoria,pcimreusablecontainername=Objects Container,ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=condicaosimples1, pcimConditionName=Condicoes Auditoria , pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=condicaosimples2, pcimConditionName=Condicoes Auditoria , pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionGroupNumber: 1
pcimConditionListType: 2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: pcimcompoundconditionauxclass
pcimConditionNegated: FALSE
pcimConditionName: Condicoes Auditoria

dn: cn=condicaosimples1,pcimConditionName=Condicoes Auditoria,pcimreusablecontainername=Objects Container,ou=Modelo RBPIM, dc=Banco ABC
pcimVariableModelProperty: dlmName
pcimStringList: GerFinanceiro
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimsimpleconditionauxclass
objectClass: pcimVariable
objectClass: pcimExplicitVariableAuxClass
objectClass: pcimValueAuxClass
objectClass: pcimStringValueAuxClass
objectClass: pcimconditionauxclass
pcimVariableModelClass: ApplicationSystem
cn: cn=condicaosimples1, pcimConditionName=Condicoes Auditoria, pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC

dn: cn=condicaosimples2,pcimConditionName=Condicoes Auditoria,pcimreusablecontainername=Objects Container,ou=Modelo RBPIM, dc=Banco ABC
pcimVariableModelProperty: dlmName
pcimStringList: GerCliente
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimsimpleconditionauxclass
objectClass: pcimVariable
objectClass: pcimExplicitVariableAuxClass
objectClass: pcimValueAuxClass
objectClass: pcimStringValueAuxClass
objectClass: pcimconditionauxclass
pcimVariableModelClass: ApplicationSystem
cn: cn=condicaosimples2, pcimConditionName=Condicoes Auditoria, pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC

dn: pcimConditionName=Condicoes
Financeiro,pcimreusablecontainername=ObjectsContainer,ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=condicaosimples1, pcimConditionName=Condicoes Financeiro,
pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionGroupNumber: 1
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: pcimcompoundconditionauxclass
pcimConditionNegated: FALSE
pcimConditionName: Condicoes Financeiro

dn: cn=condicaosimples1,pcimConditionName=Condicoes
Financeiro,pcimreusablecontainername=Objects Container,ou=Modelo RBPIM, dc=Banco ABC
pcimVariableModelProperty: dlmName
pcimStringList: GerFinanceiro
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimsimpleconditionauxclass
objectClass: pcimVariable
objectClass: pcimExplicitVariableAuxClass
objectClass: pcimValueAuxClass
objectClass: pcimStringValueAuxClass
objectClass: pcimconditionauxclass
pcimVariableModelClass: ApplicationSystem
cn: cn=condicaosimples1, pcimConditionName=Condicoes Financeiro,
pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC

dn: pcimConditionName=Condicoes Contabil,pcimreusablecontainername=Objects
Container,ou=Modelo RBPIM, dc=Banco ABC
pcimConditionList: cn=condicaosimples1, pcimConditionName=Condicoes Contabil,
pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC
pcimConditionGroupNumber: 1
pcimConditionListType: 1
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: pcimcompoundconditionauxclass
pcimConditionNegated: FALSE
pcimConditionName: Condicoes Contabil

dn: cn=condicaosimples1,pcimConditionName=Condicoes Contabil,pcimreusablecontainername=Objects Container,ou=Modelo RBPIM, dc=Banco ABC
pcimVariableModelProperty: dlmName
pcimStringList: GerCliente
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimsimpleconditionauxclass
objectClass: pcimVariable
objectClass: pcimExplicitVariableAuxClass
objectClass: pcimValueAuxClass
objectClass: pcimStringValueAuxClass
objectClass: pcimconditionauxclass
pcimVariableModelClass: ApplicationSystem
cn: cn=condicaosimples1, pcimConditionName=Condicoes Contabil,
pcimreusablecontainername=Objects Container, ou=Modelo RBPIM, dc=Banco ABC

dn: cn=Horario Comercial,pcimreusablecontainername=Time Period Container,ou=Modelo RBPIM, dc=Banco ABC
pcimTPCLocalOrUtcTime: 1
pcimTPCTimeOfDayMask: T080000/T180000
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimconditionauxclass
objectClass: pcimsimpleconditionauxclass
objectClass: pcimTPCAuxClass
cn: Horario Comercial

dn: cn=Horario Bancario,pcimreusablecontainername=Time Period Container,ou=Modelo RBPIM, dc=Banco ABC
pcimTPCLocalOrUtcTime: 1
pcimTPCTimeOfDayMask: T100000/T160000
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimconditionauxclass
objectClass: pcimsimpleconditionauxclass
objectClass: pcimTPCAuxClass
cn: Horario Bancario

dn: pcimreusablecontainername=Separation Container,ou=Modelo RBPIM, dc=Banco ABC
pcimReusableContainerName: User Container
objectClass: top
objectClass: dlm1managedElement
objectClass: dlm1managedSystemElement
objectClass: dlm1logicalElement
objectClass: dlm1system
objectClass: dlm1admindomain
objectClass: pcimreusablecontainer
objectClass: pcimreusablecontainerauxclass

dn: rbpimSSDname=SSD01,pcimreusablecontainername=Separation Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimRoleSet: rbpimRoleName=Auditor, ou=Modelo RBPIM ,dc=Banco ABC
rbpimRoleSet: rbpimRoleName=Atendente, ou=Modelo RBPIM ,dc=Banco ABC
rbpimSSDName: SSD01
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimSSD
rbpimCardinality: 2

dn: rbpimSSDname=SSD02,pcimreusablecontainername=Separation Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimRoleSet: rbpimRoleName=Auditor, ou=Modelo RBPIM ,dc=Banco ABC
rbpimRoleSet: rbpimRoleName=Supervisor, ou=Modelo RBPIM ,dc=Banco ABC
rbpimSSDName: SSD02
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimSSD
rbpimCardinality: 2

dn: rbpimSSDname=SSD03,pcimreusablecontainername=Separation Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimRoleSet: rbpimRoleName=Auditor, ou=Modelo RBPIM ,dc=Banco ABC
rbpimRoleSet: rbpimRoleName=Caixa, ou=Modelo RBPIM ,dc=Banco ABC
rbpimSSDName: SSD03
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimSSD
rbpimCardinality: 2

dn: rbpimDSDname=DSD01,pcimreusablecontainername=Separation Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimRoleSet: rbpimRoleName=Supervisor, ou=Modelo RBPIM ,dc=Banco ABC
rbpimRoleSet: rbpimRoleName=Atendente, ou=Modelo RBPIM ,dc=Banco ABC
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimDSD
rbpimDSDName: DSD01
rbpimCardinality: 2

dn:
pcimActionName=PermissaoAtendente2,rbpimRoleName=Atendente,pcimreusablecontainername=Roles Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionDN: rbpimPermissionName=GC1, pcimreusablecontainername=Permissions Container, pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo RBPIM, dc=Banco ABC
pcimActionName: PermissaoAtendente2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionOrder: 1

dn:pcimActionName=PermissaoSupervisor2,rbpimRoleName=Supervisor,pcimreusablecontainername=
Roles Container,pcimGroupName=PoliticaRBPIM,pcimreusablecontainername=Policy Group
Container,ou=Modelo RBPIM, dc=Banco ABC
rbpimPermissionDN: rbpimPermissionName=GC2, pcimreusablecontainername=Permissions Container,
pcimGroupName=PoliticaRBPIM, pcimreusablecontainername=Policy Group Container, ou=Modelo
RBPIM, dc=Banco ABC
pcimActionName: PermissaoSupervisor2
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionOrder: 1

---

# Anexo B – PIB Armazenada no PEP - Estudo de Caso Banco ABC

A PIB corresponde à estrutura definida para armazenamento das informações de política provisionada ao PEP. Esta estrutura é armazenada em memória no PEP estando disponível, tanto para possibilitar o provisionamento, como para prover informações às aplicações controladas.

A seguir, é listado um arquivo relativo ao conteúdo da memória para o estudo de caso, BancoABC, apresentado no trabalho.

---

```
<?xml version="1.0" encoding="UTF-8"?>
<Pep>
<Configuracao>
        <PepId>Pep1</PepId>
        <PdpAddr>localhost</PdpAddr>
        <PdpPorta>3288</PdpPorta>
        <ClientType>4002 </ClientType>
        <MaxHops>3</MaxHops>
</Configuracao>
```

```
<pep id="PepRBAC">
        <BasePib>

                <PrcSupport oid="1.3.6.1.2.2.2.1.1">
                        <Prid id="16">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.4.10</SupportedPrc>
                                <SupportedAttrs type="3">1</SupportedAttrs>
                        </Prid>
                        <Prid id="17">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.4.11</SupportedPrc>
                                <SupportedAttrs type="3">1</SupportedAttrs>
                        </Prid>
                        <Prid id="1">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.1.2</SupportedPrc>
                                <SupportedAttrs type="3">1111111</SupportedAttrs>
                        </Prid>
                        <Prid id="2">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.1.3</SupportedPrc>
                                <SupportedAttrs type="3">111</SupportedAttrs>
                        </Prid>
                        <Prid id="3">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.2.1</SupportedPrc>
                                <SupportedAttrs type="3">11</SupportedAttrs>
                        </Prid>
                        <Prid id="4">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.2.2</SupportedPrc>
                                <SupportedAttrs type="3">11</SupportedAttrs>
                        </Prid>
                        <Prid id="5">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.2.3</SupportedPrc>
                                <SupportedAttrs type="3">111</SupportedAttrs>
                        </Prid>
                        <Prid id="6">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.3.1</SupportedPrc>
                                <SupportedAttrs type="3">111111111111</SupportedAttrs>
                        </Prid>
                        <Prid id="7">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.4.1</SupportedPrc>
                                <SupportedAttrs type="3">1</SupportedAttrs>
                        </Prid>
                        <Prid id="8">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.4.2</SupportedPrc>
                                <SupportedAttrs type="3">1</SupportedAttrs>
                        </Prid>
                        <Prid id="9">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.4.3</SupportedPrc>
                                <SupportedAttrs type="3">1</SupportedAttrs>
                        </Prid>
                        <Prid id="10">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.4.4</SupportedPrc>
                                <SupportedAttrs type="3">1</SupportedAttrs>
                        </Prid>
                        <Prid id="11">
                                <SupportedPrc type="6">1.3.6.1.2.2.2.4.5</SupportedPrc>
                                <SupportedAttrs type="3">1</SupportedAttrs>
                        </Prid>
```

```
                <Prid id="12">
                        <SupportedPrc type="6">1.3.6.1.2.2.2.4.6</SupportedPrc>
                        <SupportedAttrs type="3">1</SupportedAttrs>
                </Prid>
                <Prid id="13">
                        <SupportedPrc type="6">1.3.6.1.2.2.2.4.7</SupportedPrc>
                        <SupportedAttrs type="3">1</SupportedAttrs>
                </Prid>
                <Prid id="14">
                        <SupportedPrc type="6">1.3.6.1.2.2.2.4.8</SupportedPrc>
                        <SupportedAttrs type="3">1</SupportedAttrs>
                </Prid>
                <Prid id="15">
                        <SupportedPrc type="6">1.3.6.1.2.2.2.4.9</SupportedPrc>
                        <SupportedAttrs type="3">1</SupportedAttrs>
                </Prid>
        </PrcSupport>

        <PibIncarnation oid="1.3.6.1.2.2.2.1.2">
                <Prid id="1">
                        <Name type="3">PDP</Name>
                        <Id type="3">PDPrbac</Id>
                        <Longevity type="2">1</Longevity>
                        <Ttl type="2">0</Ttl>
                        <InCtxtSet type="1">True</InCtxtSet>
                        <Active type="1">True</Active>
                        <FullState type="1">TRue</FullState>
                </Prid>
        </PibIncarnation>

        <DeviceId oid="1.3.6.1.2.2.2.1.3">
                <Prid id="1">
                        <Descr type="3">Pep RBAC Ver 1</Descr>
                        <MaxMsg type="2">4294967295</MaxMsg>
                        <MaxContexts type="2">4294967295</MaxContexts>
                </Prid>
        </DeviceId>

</BasePib>

<DeviceCapabilities>

        <CapabilitiesSet oid="1.3.6.1.2.2.2.2.1">
                <Prid id="1">
                        <Name type="3">Politicas RBAC</Name>
                        <Capability type="6">1.3.6.1.2.2.2.2.2.1</Capability>
                </Prid>
        </CapabilitiesSet>

        <InterfaceRoleCombo oid="1.3.6.1.2.2.2.2.2">
                <Prid id="1">
                        <Role type="3">PoliticaRBPIM</Role>
                        <CapSetName type="3">Politicas RBAC</CapSetName>
                </Prid>
        </InterfaceRoleCombo>
```

```
        <InterfaceIndex oid="1.3.6.1.2.2.2.2.3">
                <Prid id="1">
                        <ifRoles type="3">PoliticaRBPIM</ifRoles>
                        <CapSetName type="3">Politicas RBAC</CapSetName>
                        <IfIndex type="3">1</IfIndex>
                </Prid>
        </InterfaceIndex>

</DeviceCapabilities>

<ClassifierGroup>

        <IPFilter oid="1.3.6.1.2.2.2.3.1">
                <Prid id="1">
                        <Negation type="1">FALSE</Negation>
                        <AddrType type="3"/>
                        <DstAddr type="3">192.168.1.10</DstAddr>
                        <DstPrefixLength type="3">255.255.255.0</DstPrefixLength>
                        <SrcAddr type="3">192.168.1.0</SrcAddr>
                        <SrcPrefixLength type="3">255.255.255.0</SrcPrefixLength>
                        <Dscp type="2"/>
                        <FlowId type="2"/>
                        <Protocol type="3"/>
                        <DstL4PortMin type="2">80</DstL4PortMin>
                        <DstL4PortMax type="2">80</DstL4PortMax>
                        <SrcL4PortMin type="2"/>
                        <SrcL4PortMax type="2"/>
                </Prid>
        </IPFilter>

</ClassifierGroup>

<Rbac>

  <User oid="1.3.6.1.2.2.2.4.1">
          <Prid id="1">
                  <uid type="6">Pedro</uid>
          </Prid>
  </User>

  <UserAssignement oid="1.3.6.1.2.2.2.4.2">
    <Prid id="1">
      <user type="6">1.3.6.1.2.2.2.4.1.1</user>
      <role type="6">1.3.6.1.2.2.2.4.3.3</role>
    </Prid>
    <Prid id="2">
      <user type="6">1.3.6.1.2.2.2.4.1.1</user>
      <role type="6">1.3.6.1.2.2.2.4.3.5</role>
    </Prid>
```

```
  <Prid id="3">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.6</role>
  </Prid>
  <Prid id="4">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.7</role>
  </Prid>
  <Prid id="5">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.8</role>
  </Prid>
  <Prid id="6">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.9</role>
  </Prid>
  <Prid id="7">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.10</role>
  </Prid>
  <Prid id="8">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.11</role>
  </Prid>
  <Prid id="9">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.12</role>
  </Prid>
  <Prid id="10">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.13</role>
  </Prid><
  Prid id="11">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.14</role>
  </Prid>
  <Prid id="12">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.15</role>
  </Prid>
  <Prid id="13">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.16</role>
  </Prid>
  <Prid id="14">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.17</role>
  </Prid>
  <Prid id="15">
    <user type="6">1.3.6.1.2.2.2.4.1.1</user>
    <role type="6">1.3.6.1.2.2.2.4.3.18</role>
  </Prid>
</UserAssignement>
```

```
 <Roles oid="1.3.6.1.2.2.2.4.3">
    <Prid id="1">
        <rolename type="3">Auditor</rolename>
        <priority type="1">1</priority>
    </Prid>
    <Prid id="2">
        <rolename type="3">Funcionario</rolename>
        <priority type="1">5</priority>
    </Prid>
    <Prid id="3">
        <rolename type="3">Atendente</rolename>
        <priority type="1">4</priority>
    </Prid>
    <Prid id="4">
        <rolename type="3">Caixa</rolename>
        <priority type="1">3</priority>
    </Prid>
    <Prid id="5">
        <rolename type="3">Supervisor</rolename>
        <priority type="1">2</priority>
    </Prid>

</Roles>

<TimeFilterRole oid="1.3.6.1.2.2.2.4.4">
        <Prid id="3">
                <timeFilter type="6">1.3.6.1.2.2.2.4.12.2</timeFilter>
                <role type="6">1.3.6.1.2.2.2.4.3.4</role>
        </Prid>
        <Prid id="4">
                <timeFilter type="6">1.3.6.1.2.2.2.4.12.2</timeFilter>
                <role type="6">1.3.6.1.2.2.2.4.3.5</role>
        </Prid>
        <Prid id="1">
                <timeFilter type="6">1.3.6.1.2.2.2.4.12.1</timeFilter>
                <role type="6">1.3.6.1.2.2.2.4.3.1</role>
        </Prid>
        <Prid id="2">
                <timeFilter type="6">1.3.6.1.2.2.2.4.12.2</timeFilter>
                <role type="6">1.3.6.1.2.2.2.4.3.3</role>
        </Prid>
</TimeFilterRole>

<PermissionAssignement oid="1.3.6.1.2.2.2.4.5">

        <Prid id="1">
                <role type="6">1.3.6.1.2.2.2.4.3.1</role>
                <permission type="6">1.3.6.1.2.2.2.4.6.1</permission>
        </Prid>
        <Prid id="2">
                <role type="6">1.3.6.1.2.2.2.4.3.3</role>
                <permission type="6">1.3.6.1.2.2.2.4.6.2</permission>
        </Prid>
        <Prid id="3">
                <role type="6">1.3.6.1.2.2.2.4.3.3</role>
                <permission type="6">1.3.6.1.2.2.2.4.6.3</permission>
        </Prid>
        <Prid id="4">
                <role type="6">1.3.6.1.2.2.2.4.3.3</role>
                <permission type="6">1.3.6.1.2.2.2.4.6.4</permission>
```

```
		</Prid>
		<Prid id="5">
			<role type="6">1.3.6.1.2.2.2.4.3.4</role>
			<permission type="6">1.3.6.1.2.2.2.4.6.5</permission>
		</Prid>
		<Prid id="6">
			<role type="6">1.3.6.1.2.2.2.4.3.5</role>
			<permission type="6">1.3.6.1.2.2.2.4.6.6</permission>
		</Prid>
		<Prid id="7">
			<role type="6">1.3.6.1.2.2.2.4.3.5</role>
			<permission type="6">1.3.6.1.2.2.2.4.6.7</permission>
		</Prid>
		<Prid id="8">
			<role type="6">1.3.6.1.2.2.2.4.3.5</role>
			<permission type="6">1.3.6.1.2.2.2.4.6.8</permission>
		</Prid>
</PermissionAssignement>

<Permissions oid="1.3.6.1.2.2.2.4.6">
		<Prid id="1">
			<object type="6">1.3.6.1.2.2.2.4.9.1</object>
			<operation type="3">Auditar_Transacoes</operation>
		</Prid>
		<Prid id="2">
			<object type="6">1.3.6.1.2.2.2.4.9.2</object>
			<operation type="3">AgendarDOC</operation>
		</Prid>
		<Prid id="3">
			<object type="6">1.3.6.1.2.2.2.4.9.2</object>
			<operation type="3">AgendarTED</operation>
		</Prid>
		<Prid id="4">
			<object type="6">1.3.6.1.2.2.2.4.9.3</object>
			<operation type="3">Abrir Conta</operation>
		</Prid>
		<Prid id="5">
			<object type="6">1.3.6.1.2.2.2.4.9.2</object>
			<operation type="3">Efetuar Pagamentos</operation>
		</Prid>
		<Prid id="6">
			<object type="6">1.3.6.1.2.2.2.4.9.2</object>
			<operation type="3">AutorizarDOC</operation>
		</Prid>
		<Prid id="7">
			<object type="6">1.3.6.1.2.2.2.4.9.2</object>
			<operation type="3">AutorizarTED</operation>
		</Prid>
		<Prid id="8">
			<object type="6">1.3.6.1.2.2.2.4.9.3</object>
			<operation type="3">Conceder Limite</operation>
		</Prid>

</Permissions>



<TimeFilterPermission oid="1.3.6.1.2.2.2.4.7">
		<Prid id="1">
```

```
                <TimeFilterIndex type="6">1.3.6.1.2.2.2.4.12.1</TimeFilterIndex>
                <PermissionIndex type="6">1.3.6.1.2.2.2.4.6.1</PermissionIndex>
        </Prid>
        <Prid id="2">
                <TimeFilterIndex type="6">1.3.6.1.2.2.2.4.12.2</TimeFilterIndex>
                <PermissionIndex type="6">1.3.6.1.2.2.2.4.6.2</PermissionIndex>
        </Prid>
        <Prid id="3">
                <TimeFilterIndex type="6">1.3.6.1.2.2.2.4.12.2</TimeFilterIndex>
                <PermissionIndex type="6">1.3.6.1.2.2.2.4.6.3</PermissionIndex>
        </Prid>
        <Prid id="4">
                <TimeFilterIndex type="6">1.3.6.1.2.2.2.4.12.2</TimeFilterIndex>
                <PermissionIndex type="6">1.3.6.1.2.2.2.4.6.4</PermissionIndex>
        </Prid>
        <Prid id="5">
                <TimeFilterIndex type="6">1.3.6.1.2.2.2.4.12.2</TimeFilterIndex>
                <PermissionIndex type="6">1.3.6.1.2.2.2.4.6.5</PermissionIndex>
        </Prid>
        <Prid id="6">
                <TimeFilterIndex type="6">1.3.6.1.2.2.2.4.12.2</TimeFilterIndex>
                <PermissionIndex type="6">1.3.6.1.2.2.2.4.6.6</PermissionIndex>
        </Prid>
        <Prid id="7">
                <TimeFilterIndex type="6">1.3.6.1.2.2.2.4.12.2</TimeFilterIndex>
                <PermissionIndex type="6">1.3.6.1.2.2.2.4.6.7</PermissionIndex>
        </Prid>
        <Prid id="8">
                <TimeFilterIndex type="6">1.3.6.1.2.2.2.4.12.2</TimeFilterIndex>
                <PermissionIndex type="6">1.3.6.1.2.2.2.4.6.8</PermissionIndex>
        </Prid>

</TimeFilterPermission>

<IPHeaderFilterPermission oid="1.3.6.1.2.2.2.4.8">
        <Prid id="1">
                <ipFilter type="6">1.3.6.1.2.2.2.3.1.1</ipFilter>
                <permission type="6">1.3.6.1.2.2.2.4.6.1</permission>
        </Prid>
</IPHeaderFilterPermission>

<Objects oid="1.3.6.1.2.2.2.4.9">
        <Prid id="2">
                <expression type="3">(ApplicationSystem.dlmName=GerFinanceiro)
                </expression>
        </Prid>
        <Prid id="3">
                <expression type="3">(ApplicationSystem.dlmName=GerCliente)
                </expression>
        </Prid>
        <Prid id="1">
                <expression type="3">(ApplicationSystem.dlmName=GerFinanceiro ||
                ApplicationSystem.dlmName=GerCliente)
                </expression>
        </Prid>
</Objects>

<DSD oid="1.3.6.1.2.2.2.4.10">
        <Prid id="1">
                <Cardinality type="3">2</Cardinality>
```

```
            </Prid>
        </DSD>

        <DSDEntries oid="1.3.6.1.2.2.2.4.11">
            <Prid id="1">
                <Role type="6">1.3.6.1.2.2.2.4.3.5</Role>
                <DSDIndex type="6">1.3.6.1.2.2.2.4.10.1</DSDIndex>
            </Prid>
            <Prid id="2">
                <Role type="6">1.3.6.1.2.2.2.4.3.3</Role>
                <DSDIndex type="6">1.3.6.1.2.2.2.4.10.1</DSDIndex>
            </Prid>
        </DSDEntries>

        <TimeFilter oid="1.3.6.1.2.2.2.4.12">
            <Prid id="1">
                <LocalUtcTime type="3">1</LocalUtcTime>
                <TpcTime type="3"/>
                <MesesdoAno type="3"/>
                <DiasdoMes type="3"/>
                <DiasdaSemana type="3"/>
                <HoradoDia type="3">T080000/T180000</HoradoDia>
            </Prid>
            <Prid id="2">
                <LocalUtcTime type="3">1</LocalUtcTime>
                <TpcTime type="3"/>
                <MesesdoAno type="3"/>
                <DiasdoMes type="3"/>
                <DiasdaSemana type="3"/>
                <HoradoDia type="3">T100000/T160000</HoradoDia>
            </Prid>
        </TimeFilter>

</Rbac>

</pep>
<ControleSessao>
        <UsuarioAtivo Nome="">
            <Sessao id="">
                <PapelAtivo item="">
                   <CodigoPapel type="3"> </CodigoPapel>
                </PapelAtivo>
            </Sessao>
        </UsuarioAtivo>
</ControleSessao>
</Pep>
```