RICARDO CASSIANO NABHEN

RBPIM: UM MODELO DE POLÍTICAS DE SEGURANÇA BASEADO EM PAPÉIS

CURITIBA

2003

RICARDO CASSIANO NABHEN

RBPIM: UM MODELO DE POLÍTICAS DE SEGURANÇA BASEADO EM PAPÉIS

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

Área de Concentração:
Metodologia e Técnicas de Computação

Orientador:
Prof. Edgard Jamhour, Dr.

CURITIBA

2003

Nabhen, Ricardo Cassiano
RBPIM: Um Modelo de Políticas de Segurança Baseado em Papéis. Curitiba, 2003. 241 p.

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

1. Segurança 2. Rbac 3. Ldap 4. Política.
I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Informática Aplicada.

À minha esposa, Márcia, e aos meus filhos,
Jacqueline e Eduardo, pelo apoio e pela compreensão
dos inúmeros momentos dedicados a este trabalho.

Agradecimentos

Ao professor Edgard Jamhour, pela excelente orientação, confiança e apoio.

Ao pesquisador Larry Bartz, pelas valiosas contribuições.

À empresa Justino, Filhos & Cia Ltda, pelo apoio concedido para a viabilização deste trabalho de pesquisa.

Sumário

Agradecimentos i
Sumário ii
Lista de Figuras viii
Lista de Tabelas xi
Lista de Abreviaturas e Siglas xiii
Resumo xiv
Abstract xvi

Capítulo 1
Introdução **1**
1.1. Desafio 1
1.2. Motivação 1
1.3. Proposta 4
1.4. Organização 5

Capítulo 2
O Gerenciamento de Redes Baseado em Políticas **7**
2.1. Introdução 7
2.2. Serviços de Diretórios 7
2.3. Especificação X.500 8
2.4. LDAP 8
2.5. Políticas 9
2.6. Modelo de Informação 10
2.7. Trabalhos de Padronização 10
2.7.1. Directory Enabled Networks (DEN) 11
2.7.2. Common Information Model (CIM) 12
2.7.3. Policy Core Information Model (PCIM) 15
2.7.3.1 Descrição do Modelo 16
2.7.3.2 Arquitetura para Implementação 21

2.8. Outro trabalho relacionado..... 23

2.8.1. Integrando a especificação de objetivos no gerenciamento baseado em políticas..... 23

2.9. Conclusões do Capítulo 26

Capítulo 3

O Controle de Acesso Baseado em Papéis - RBAC..... 27

3.1. Introdução..... 27

3.2. O Controle de Acesso..... 27

3.2.1. A Implementação do Controle de Acesso..... 29

3.2.2. As Políticas de Controle de Acesso..... 30

3.2.2.1 O Modelo Discricionário..... 30

3.2.2.2 O Modelo Obrigatório..... 30

3.2.2.3 O Modelo Baseado em Papéis..... 31

3.2.2.4 Algumas considerações..... 31

3.3. O RBAC..... 31

3.4. RBAC: O Modelo de Referência 36

3.4.1. O Modelo de Referência..... 37

3.4.1.1. O *Core* RBAC..... 37

3.4.1.2. O *Hierarchical* RBAC..... 38

3.4.1.3. *Static Separation of Duty Relations (SSD)*..... 39

3.4.1.4. *Dynamic Separation of Duty Relations (DSD)* 39

3.4.2. A Especificação Funcional do Modelo RBAC..... 40

3.5. Outros Trabalhos Relacionados..... 42

3.5.1. Especificando e Gerenciando o Controle de Acesso Baseado em Papéis dentro de uma Intranet Corporativa..... 42

3.5.2. O RBAC no Ambiente Operacional do Solaris 43

3.5.3. O Controle de Acesso e o Gerenciamento de Sessão no Ambiente http..... 44

3.5.4. O Controle de Acesso Baseado em Papéis para o Modelo CORBA de Segurança..... 46

3.6. Conclusões do Capítulo..... 48

Capítulo 4

RBPIM: Uma Proposta de Extensão do PCIM para o Mapeamento do RBAC.... 50

4.1. Introdução..... 50

4.2. Trabalhos Relacionados..... 51

4.2.1. *hyperDrive*: Um esquema do LDAP para o controle de acesso baseado em papéis..... 51

4.2.2. O Modelo de Informação CADS-2 52

4.2.2.1. A Descrição do Modelo..... 52

4.2.2.2. Exemplo de Mapeamento do RBAC no CADS-2..... 54

4.3. A Proposta RBPIM..... 56

4.3.1. Considerações e Diretrizes..... 56

4.3.2. O Contexto do Mapeamento do RBAC..... 57

4.3.3. A Descrição do Modelo Proposto..... 58

4.3.3.1. O Mapeamento do *Core* e do *Hierarchical* RBAC no RBPIM.... 59

4.3.3.2. O Mapeamento do *Constrained* RBAC no RBPIM..... 63

4.3.3.3. Agregações no RBPIM..... 63

4.3.4. Exemplo do mapeamento de políticas RBAC no RBPIM..... 64

4.4. Conclusões do Capítulo..... 67

Capítulo 5

O Mapeamento do RBPIM nos Serviços de Diretório LDAP..... 69

5.1. Introdução..... 69

5.2. O Mapeamento do PCIM no LDAP..... 70

5.2.1. Tipos de Classes..... 70

5.2.2. Descrição do mapeamento LDAP do PCIM..... 70

5.3. O Mapeamento do RBPIM no LDAP..... 76

5.3.1. Descrição do esquema de diretório..... 76

5.3.2. Classes para o mapeamento de papéis, permissões e de separação de tarefas..... 80
5.3.3. Classes associativas relacionadas a papéis e permissões..... 83
5.3.4. Classes acessórias para agrupamento..... 86
5.3.5. Classes para representação de variáveis..... 87
5.3.6. Classes para representação de valores..... 88
5.4. Conclusões do Capítulo..... 90

Capítulo 6

A Implementação do RBPIM: Arquitetura, API e Algoritmos do PDP..... 92
6.1. Introdução..... 92
6.2. Padrões para Implementação 93
6.2.1.Modelos de Arquitetura..... 93
6.2.2. Interações entre o PDP e o PEP..... 94
6.3. A Arquitetura Proposta..... 95
6.3.1. Apresentação..... 95
6.3.2. Mensagens COPS para a arquitetura RBPIM..... 98
6.4. A API do RBPEP..... 104
6.4.1. Abertura do Serviço de Política..... 106
6.4.2. Abertura de Sessão RBPIM (Abertura de Sessão Fase 1)..... 107
6.4.3. Seleção de Papéis para a Sessão RBPIM (Abertura de Sessão Fase 2)..... 110
6.4.4. Fechamento de Sessão RBPIM..... 111
6.4.5. Verificação de Acesso a Objetos RBPIM..... 112
6.5. Algoritmos para Implementação do PDP..... 115
6.5.1. Algoritmo principal do PDP..... 117
6.5.2. Algoritmo do PDP para a mensagem OPN..... 117
6.5.3. Algoritmo do PDP para a mensagem REQ..... 118
6.5.3.1. Algoritmo do PDP para RBPEP_CreateSession (Fase 1 da

Criação da Sessão) 118

6.5.3.2. Algoritmo do PDP para RBPEP_SelectedRoles (Fase 2 da Criação da Sessão) 123

6.5.3.3. Algoritmo do PDP para RBPEP_CheckAccess..... 124

6.5.4. Algoritmo do PDP para a mensagem RPT..... 132

6.5.5. Algoritmo do PDP para a mensagem DRQ..... 132

6.5.6. Algoritmo do PDP para a mensagem CC..... 132

6.6. Conclusões do Capítulo..... 132

Capítulo 7

Estudo de Caso e a Avaliação da Proposta..... 134

7.1. Introdução..... 134

7.2. Estudo de Caso..... 134

7.2.1. Cenário..... 134

7.2.2. O Mapeamento da Política de Controle de Acesso através do RBPIM..... 136

7.2.2.1. Hierarquia de Papéis..... 137

7.2.2.2. Separação de tarefas entre Papéis..... 138

7.2.2.3. A DIT para o caso do BANCO ABC..... 140

7.2.2.4. Objetos do Diretório para o caso do BANCO ABC..... 141

7.3. Implementação do Protótipo..... 143

7.3.1. Estrutura do Protótipo..... 144

7.3.2. Cenário de Avaliação..... 145

7.4. Apresentação dos Resultados..... 145

7.4.1. Avaliação Funcional..... 146

7.4.2. Avaliação quanto ao Desempenho..... 148

7.5. Conclusões do Capítulo..... 152

Capítulo 8

Conclusões e Trabalhos Futuros..... 153

Referências Bibliográficas..... 156

Anexo A – Esquema de Diretório do RBPIM..... 160

1.1. Classes..... 160

1.2. Atributos..... 162

Anexo B – Especificações COPS para o Tipo de Cliente RBPIM..... 165

Anexo C – Arquivos LDIF do Estudo de Caso do BANCO ABC..... 166

Anexo D – Arquivos de Entrada e Saída do Estudo de Caso do BANCO ABC.... 182

Anexo E – Código Fonte dos Principais Módulos do Protótipo desenvolvido..... 204

1.1. Arquivo Pdp.java..... 204

1.2. Arquivo Rbpep.java..... 216

Lista de Figuras

Figura 2.1	Visão conceitual do *framework* de Políticas.....	10
Figura 2.2	Visão geral da filosofia do DEN.....	11
Figura 2.3	O Core Model do CIM	13
Figura 2.4	CIM – A arquitetura em camadas.....	13
Figura 2.5	O Modelo de Informação do DEN.....	14
Figura 2.6	O Meta Modelo de Política.....	16
Figura 2.7	PCIM: Visão Conceitual.....	17
Figura 2.8	PCIM: Principais classes do modelo.....	17
Figura 2.9	PCIM: *PolicyConditions* nas formas DNF e CNF.....	19
Figura 2.10	PCIM: Classes Associativas.....	21
Figura 2.11	Arquitetura para implementação do *framework* de Políticas.....	22
Figura 2.12	Visão simplificada do Core Schema proposto em [BS 1].....	25
Figura 3.1	O Controle de Acesso.....	28
Figura 3.2	Uma matriz de Acesso.....	29
Figura 3.3	ACLs correspondentes à Fig. 3.2.....	29
Figura 3.4	Visão do RBAC.....	31
Figura 3.5	Um exemplo de hierarquia entre papéis.....	33
Figura 3.6	Exemplo de política RBAC.....	36
Figura 3.7	O *Core* RBAC.....	38
Figura 3.8	O *Hierarchical* RBAC.....	38
Figura 3.9	O relacionamento de *SSD*.....	39
Figura 3.10	O relacionamento de *DSD*.....	40
Figura 3.11	Elementos e Relacionamentos RBAC no ambiente do Solaris.....	43
Figura 3.12	Um exemplo do RBAC no ambiente do Solaris.....	44
Figura 3.13	Extensões do esquema do LDAP para suportar o RBAC.....	45
Figura 3.14	Exemplo de Política RBAC apresentada em [FOW 01].....	47
Figura 3.15	Exemplo de um cenário do *framework* apresentado em [FOW 01].....	47
Figura 4.1	Relacionamentos entre os objetos do modelo *hyperDrive*.....	51

Figura 4.2 A DIT do CADS-2 52
Figura 4.3 CADS-2: Extensões da classe *PolicyCondition* 54
Figura 4.4 CADS-2: Extensões da classe *PolicyAction* 54
Figura 4.5 CADS-2: Um exemplo de mapeamento do RBAC 55
Figura 4.6 Hierarquia de Classes do RBPIM 60
Figura 4.7 RBPIM: Detalhe das Principais Classes e Associações 61
Figura 4.8 Hierarquia parcial de classes do *CIM_User Schema* 62
Figura 4.9 Exemplo do mapeamento da associação UA do RBPIM 65
Figura 4.10 Hierarquia parcial de classes do *CIM_System Schema* 65
Figura 4.11 Exemplo de um objeto de permissão do RBPIM 66
Figura 4.12 Exemplo das associações UA e PA do RBAC no RBPIM 67

Figura 5.1 Exemplo de um *attachment* de uma classe auxiliar 71
Figura 5.2 Exemplo de associação por *DIT containment* 72
Figura 5.3 Exemplo de uma regra de política simples 75
Figura 5.4 Exemplo de uma regra de política complexa 76
Figura 5.5 RBPIM: Classes do esquema de diretório LDAP (Parte 1) 78
Figura 5.6 RBPIM: Classes do esquema de diretório LDAP (Parte 2) 79
Figura 5.7 Exemplo 1: Objetos de políticas RBPIM (formato LDIF) 82
Figura 5.8 Exemplo 2: Objetos de políticas RBPIM (formato LDIF) 85
Figura 5.9 Exemplo 3: Objetos de políticas RBPIM (formato LDIF) 90

Figura 6.1 Arquiteturas para implementação do controle baseado em políticas 93
Figura 6.2 COPS: Modelos para o controle baseado em políticas 95
Figura 6.3 Arquitetura proposta para a implementação do RBPIM 96
Figura 6.4 O Protocolo COPS 98
Figura 6.5 Cenário típico da arquitetura do RBPIM 105
Figura 6.6 Diagrama de seqüência da chamada RBPEP_Open 106
Figura 6.7 Diagrama de seqüência da chamada RBPEP_CreateSession 108
Figura 6.8 Diagrama de seqüência da chamada RBPEP_SelectedRoles 110
Figura 6.9 Diagrama de seqüência da chamada RBPEP_CloseSession 112
Figura 6.10 Diagrama de seqüência da chamada RBPEP_CheckAccess 114
Figura 6.11 Entradas do diretório contendo a informação de política RBPIM 116

Figura 6.12 Exemplo da associação entre papéis e usuários..... 122

Figura 7.1 Hierarquia de papéis do Banco ABC..... 137
Figura 7.2 A DIT do diretório para o caso do Banco ABC..... 140
Figura 7.3 Um papel e seus principais objetos..... 141
Figura 7.4 Arquitetura do Protótipo..... 144
Figura 7.5 Cenário de Avaliação com 20 aplicações..... 145
Figura 7.6 Tempos Médios obtidos para os cenários do Estudo de Caso..... 151

Lista de Tabelas

Tabela 2.1 Exemplos de objetivos de política..... 15
Tabela 2.2 Principais características da ferramenta de ger. de políticas, do PDP e do PEP..... 22

Tabela 3.1 Um exemplo simples de política RBAC..... 32
Tabela 3.2 Primitivas de acesso ao modelo RBAC..... 34
Tabela 3.3 Regras de definição do modelo RBAC..... 35
Tabela 3.4 Especificações funcionais do RBAC..... 41

Tabela 5.1 O Mapeamento PCIM-LDAP: Principais classes estruturais e abstratas.. 73
Tabela 5.2 O Mapeamento PCIM-LDAP: Principais classes associativas..... 73
Tabela 5.3 RBPIM: Regras de *attachment*..... 79
Tabela 5.4 Esquema de diretório do RBPIM – Parte I..... 80
Tabela 5.5 Esquema de diretório do RBPIM – Parte II..... 84
Tabela 5.6 Esquema de diretório do RBPIM – Parte III..... 87
Tabela 5.7 Esquema de diretório do RBPIM – Parte IV..... 88
Tabela 5.8 Esquema de diretório do RBPIM – Parte V..... 89

Tabela 6.1 Entidades da arquitetura proposta para a implementação do RBPIM..... 97
Tabela 6.2 Campos do cabeçalho de mensagens COPS..... 99
Tabela 6.3 Campos dos objetos específicos COPS..... 99
Tabela 6.4 Mensagem OPN..... 101
Tabela 6.5 Mensagem CAT..... 101
Tabela 6.6 Mensagem CC..... 101
Tabela 6.7 Mensagem DRQ..... 102
Tabela 6.8 Mensagem REQ..... 102
Tabela 6.9 Mensagem DEC..... 103
Tabela 6.10 Mensagem RPT..... 103

Tabela 7.1 Banco ABC: Aplicações e Operações..... 135
Tabela 7.2 Banco ABC: Usuários do sistema e suas posições..... 135
Tabela 7.3 Banco ABC: Categorias funcionais e ocupações..... 135
Tabela 7.4 Banco ABC: Papéis e direitos de acesso..... 138
Tabela 7.5 Banco ABC: Relacionamento de SSD entre papéis..... 139
Tabela 7.6 Banco ABC: Relacionamento de DSD entre papéis..... 139
Tabela 7.7 Banco ABC: Prioridade entre papéis..... 139
Tabela 7.8 Banco ABC: Principais Classes e Objetos..... 142
Tabela 7.9 Banco ABC: O Papel Auditor e seus principais objetos..... 143
Tabela 7.10 Arquivos App1.in e App1.out..... 147
Tabela 7.11 Baterias de testes para avaliação do desempenho..... 148
Tabela 7.12 Tempos obtidos na chamada *RBPEP_CreateSession* (milisegundos)..... 149
Tabela 7.13 Tempos obtidos na chamada *RBPEP_SelectedRoles* (milisegundos)..... 149
Tabela 7.14 Tempos obtidos na chamada *RBPEP_CheckAccess* (milisegundos)..... 150

Lista de Abreviaturas e Siglas

ACL	*Access Control List*
API	*Application Program Interface*
CIM	*Common Information Model*
COPS	*Common Open Policy Service*
CORBA	*Common Object Request Broker Architecture*
DAP	*Directory Access Protocol*
DEN	*Directory Enabled Networks*
DMTF	*Distributed Management Task Force*
DN	*Distinguished Name*
IANA	*Internet Assigned Numbers Authority*
IETF	*Internet Engineering Task Force*
IPSEC	*IP Security Protocol*
ISO	*International Standards Organization*
ITU-T	*International Telecommunication Union - Telecommunication Standardization Sector*
LDAP	*Lightweight Directory Access Protocol*
LDIF	*LDAP Data Interchange Format*
LPDP	*Local Policy Decision Point*
PCIM	*Policy Core Information Model*
PCLS	*Policy Core LDAP Schema*
PDP	*Policy Decision Point*
PEP	*Policy Enforcement Point*
QoS	*Quality of Service*
RBAC	*Role Based Access Control*
RDN	*Relative Distinguished Name*
RFC	*Request for Comments*
RSVP	*Resource Reservation Protocol*
SAP	*Service Access Point*
SNMP	*Simple Network Management Protocol*
TLS	*Transport Layer Security*
UML	*Unified Modeling Language*

Resumo

Consideráveis avanços na infra-estrutura de comunicação têm possibilitado um enorme crescimento na criação de redes integradas. Esta integração permite o compartilhamento de recursos e serviços, e, ao mesmo tempo, coloca a segurança entre os principais problemas enfrentados pelas organizações. Este aspecto torna imprescindível a criação de políticas que visam evitar o acesso não autorizado a recursos.

O gerenciamento de políticas de segurança em ambientes distribuídos é uma tarefa complexa, especialmente quando é considerado que algumas políticas são voláteis e dinâmicas pela sua própria natureza. O PCIM - *Policy Core Information Model*, proposto em conjunto pelo IETF e o pelo DMTF, fornece um *framework* para representar, gerenciar, aplicar e compartilhar a informação de política. Em um ambiente distribuído, freqüentemente se faz necessário aplicar o mesmo conjunto de políticas em um conjunto amplo de aplicações heterogêneas. Neste contexto, o *framework* proposto pelo PCIM se apresenta como um candidato natural para implementar esse conjunto de políticas. O PCIM, todavia, não foi concebido para suportar áreas de aplicação específicas. Por exemplo, políticas de segurança relacionadas às regras de negócio exigem a representação de informação e procedimentos não suportados diretamente pelo PCIM. De acordo com o IETF, tais áreas de aplicações devem ser suportadas pela extensão do modelo PCIM através da especialização de suas classes. O RBAC - *Role Based Access Control* – é um dos mais significativos esforços no sentido de definir um método de controle de acesso que atenda de forma completa às necessidades da descrição de políticas de segurança relacionadas às regras de negócio. Para que o PCIM suporte o método de controle de acesso definido pelo RBAC suas classes precisam ser estendidas.

O propósito deste trabalho é definir uma arquitetura de segurança baseada no RBAC para a definição e o gerenciamento de políticas em ambientes distribuídos. Para tanto, pretende-se definir um modelo para suportar o RBAC, que será implementado em serviços de diretório baseados no LDAP - *Lightweight Directory Access Protocol*, utilizando um esquema de diretório derivado do PCIM para a adaptação ao modelo de objetos utilizado pelos serviços LDAP. Além do modelo proposto, este trabalho pretende definir os algoritmos relacionados com a implementação das regras RBAC envolvendo os objetos de políticas contidos no repositório LDAP e as APIs que atuarão como interface às facilidades fornecidas

pela arquitetura. O trabalho se mantém fiel ao *framework* proposto pelo PCIM, seguindo os padrões existentes para a definição do PDP, PEP e COPS.

Palavras-Chave: 1. Segurança 2. Rbac 3. Ldap 4. Política

Abstract

Considerable advances in the communication infrastructure have made possible a huge growth of interconnected networks. This integration allows the sharing of resources and services, and at the same time, the security becomes one of the main concerns faced by the organizations. This aspect becomes essential the creation of policies that aim to protect resources from unauthorized access.

Managing security policies in distributed environments is a complex task, especially considering that some policies are volatile and dynamic for their nature. PCIM - Policy Core Information Model -, a joint work *of the IETF and the DMTF, provides a framework to represent, to manage, to apply and to share the policy information. Usually, in a distributed environment, it becomes necessary to apply the same set of policies in a set of heterogeneous applications. In this context, PCIM framework acts as a natural candidate to implement this set of policies. PCIM, however, was not conceived to support specific applications areas. For example, security policies related to business rules demand procedures and information not supported by PCIM. In accordance with the IETF, such applications areas must be supported by PCIM model extension through the specialization of its classes. RBAC - Role Based Access Control - is one of the most significant efforts in the direction to define an access control method that meets the needs of the description of security policies related to business rules. So, PCIM's classes have to be extended in order to add the support for RBAC access control method.*

The purpose of this work is to define a security framework based on RBAC for policy definition and management in distributed environments. So, it is intended to define a model to support RBAC, that will be implemented in directory services based on LDAP - Lightweight Directory Access Protocol, using a directory schema derived from the PCIM for adaptation to the object model used by LDAP services. Beyond the considered model, this work intends to define the algorithms related with the implementation of RBAC rules involving the policies objects contained in the LDAP repository and the APIs that will act as interface to the facilities supplied by this framework. The work follows the PCIM framework and the existing standards for the definition of the PDP, PEP and COPS.

Keywords: *1. Security 2. Rbac 3. Ldap 4. Policy*

Capítulo 1

Introdução

1.1. Desafio

Os avanços tecnológicos e a conseqüente facilidade na utilização dos serviços de comunicação têm possibilitado um enorme crescimento na criação de redes integradas. Esta integração de redes permite o compartilhamento de recursos e serviços através da utilização de um meio de comunicação compartilhado, mas, no entanto, torna o gerenciamento da segurança uma questão de extrema relevância para as organizações.

A integração de redes através de uma infra-estrutura de comunicação não confiável torna imprescindível a criação de uma política de segurança que visa proteger recursos do acesso não autorizado. Este procedimento passa pela configuração de dispositivos de segurança e do próprio sistema operacional. Quando um determinado serviço é solicitado faz-se necessário, inicialmente, provar a identidade de quem está solicitando o serviço, e, em seguida, verificar se há privilégios suficientes para acessar tal serviço. Neste cenário, é importante reunir e gerenciar as informações de segurança de forma consistente e em um local onde seja possível um nível de centralização tal que permita uma visão lógica global do ambiente. Além desta centralização lógica, é igualmente importante a utilização de uma metodologia que permita que o controle de acesso seja feito de forma flexível, ou seja, que ele seja facilmente adaptável às novas exigências das corporações e que possibilite a descrição de políticas de segurança de maneira natural.

1.2. Motivação

As políticas estão relacionadas com o problema de alocar recursos baseando-se nas decisões impostas pelas regras de negócio corporativas. Uma rede baseada em política (*policy-based networking*) é a aplicação destas políticas organizacionais no âmbito das redes. Neste contexto, as políticas descrevem o relacionamento entre os recursos da rede e o usuário. Como citado no *Policy Based Networking Call For Pape*rs da *IEEE Network Magazine* : “as

políticas podem ser usadas para se obter uma melhor escala no gerenciamento da rede através da descrição de atributos comuns de classes de objetos, tipicamente associados com o papel que representam, tais como dispositivos de rede, serviços de *software* e usuários, em vez de definir individualmente os atributos destes elementos". As políticas podem ser definidas a partir de duas perspectivas [RFC 3198]: 1) Um objetivo ou método de ação definidos para guiar e determinar as decisões presentes e futuras; 2) Um conjunto de regras para administrar, gerenciar e controlar o acesso a recursos. Vale notar que ambas definições não são contraditórias, e que uma complementa a outra.

O gerenciamento de políticas de segurança corporativas em ambientes distribuídos é uma tarefa complexa, especialmente em políticas que são voláteis e dinâmicas pela sua própria natureza. Políticas de segurança relacionadas às regras de negócio geralmente exigem procedimentos de controle de acesso (autorização) bem elaborados. Privilégios e direitos de acesso de usuários mudam freqüentemente, algumas vezes relacionados às posições hierárquicas ocupadas em uma organização, outras vezes relacionadas com as funções que os usuários assumem em um projeto específico.

Os administradores de rede geralmente tentam garantir o cumprimento das políticas de segurança corporativas através da configuração de dispositivos e de sistemas operacionais. Normalmente, estes dispositivos de segurança e computadores se encontram espalhados por vários locais. Neste contexto, é muito difícil assegurar que todas políticas de segurança pré-definidas sejam realmente atendidas. A principal razão para esta dificuldade é que os dispositivos de segurança, tais como os *firewalls* e os sistemas operacionais, utilizam métodos de controle de acesso que são bem diferentes das necessidades da definição de políticas no mundo real.

O RBAC [FCK 95] é um esforço significativo 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 obrigatório. 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.

O RBAC, entretanto, não é implementado como um método de controle de acesso nativo em sistemas operacionais. O Solaris 8, por exemplo, usa o RBAC apenas como um esquema alternativo ao modelo de privilégios ilimitados concedidos ao acesso *root* nos sistemas UNIX, permitindo o gerenciamento e a atribuição de privilégios extras a usuários para a execução de funções administrativas [SUN 01]. Neste cenário, muitos autores têm proposto o uso do RBAC embutido em aplicações ou em aplicações *middleware*, tais como CORBA, como pode ser observado em [FOW 01] .

O DMTF - *Distributed Management Task Force* –, fundado em 1992[1], tem se dedicado bastante ao desenvolvimento de padrões de gerenciamento para ambientes distribuídos. Entre esses padrões se destaca o CIM – *Commom Information Model* –, que é um modelo conceitual utilizado para a descrição da informação de gerenciamento, objetivando o estabelecimento de um padrão que possibilite o compartilhamento desta informação entre sistemas e aplicações. Por exemplo, o CIM fornece modelos que permitem a representação de serviços baseados em rede, dispositivos, sistemas, redes, entre outros. Uma outra iniciativa, proposta em conjunto pela Cisco e a Microsoft em 1997, foi o DEN – *Directory Enabled Networks*. O DEN, que foi incorporado ao CIM no final de 1998, teve como objetivo central produzir uma especificação visando integrar serviços de diretório e dispositivos de rede como um meio para construir redes inteligentes. Ter um diretório como elemento central capacita a integração da informação de gerenciamento de uma grande variedade de fontes, fornecendo as bases para o gerenciamento baseado em políticas. A partir das padronizações do CIM, o IETF – *Internet Engineering Task Force* – e o DMTF propuseram um modelo mais refinado sobre o paradigma de políticas: O PCIM - *Policy Core Information Model* - [RFC 3060]. O PCIM fornece um modelo para representar, gerenciar, aplicar e compartilhar a informação de política. As classes que compõem o PCIM têm como objetivo servirem de uma hierarquia extensível (através de especialização) para a definição de objetos de política de diversos tipos [RFC 3060].

Seguindo a filosofia do DEN, que se baseia na utilização de serviços de diretórios como repositório para a informação de gerenciamento, [MES 01] propõe um esquema de

[1] No início, a sigla DMTF representava Desktop Management Task Force. Anos mais tarde, passou a forma como é conhecida atualmente: *Distributed Management Task Force.*

diretório para o mapeamento do PCIM em serviços de diretórios baseados no LDAP - *Lightweight Directory Access Protocol* [RFC 1777 ; RFC 2251]. O LDAP é um serviço de diretório aberto que já é implementado em muitos sistemas operacionais. Ainda, diversas linguagens de programação bastante utilizadas, como Java e C, têm APIs para o LDAP [BARTZ 97].

Além de definir um modelo orientado a objetos para a representação de políticas, o grupo de trabalho de políticas do IETF/DMTF 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]. Um aspecto fundamental nesta arquitetura é a interação entre o PDP e PEP. [RFC 2748] apresenta o COPS – *Commom Open Policy Service Protocol* , um protocolo que pode ser utilizado para a troca de informação de política entre o servidor de políticas (PDP) e seus clientes (PEPs).

As padronizações contidas nestes trabalhos se apresentam como os elementos necessários para a construção de um *framework* RBAC capaz de dar suporte a sistemas operacionais e aplicações legadas. O PCIM, entretanto, não suporta explicitamente o RBAC, fazendo com que seja necessária a extensão de classes para que ele possa acomodar objetos que descrevem políticas segundo o modelo RBAC. O *CADS-2 Information Model* [BARTZ 01] é um exemplo de como o PCIM pode ser estendido para suportar o modelo RBAC. Também, trabalhos anteriores que têm abordado implementações de modelos RBAC em serviços de diretórios LDAP, tais como o *hyperdrive* [BARTZ 97], precisam ser revisados com o intuito de suportar o PCIM.

1.3. Proposta

Esse trabalho tem por objetivo propor uma arquitetura de segurança baseada no RBAC para a definição e o gerenciamento de políticas em ambientes distribuídos. Para tanto, pretende-se definir um modelo, o RBPIM – *Role Based Policy Information Model* -, para suportar o RBAC, que será construído a partir da extensão do PCIM e implementado em serviços de diretório baseados no LDAP utilizando um esquema de diretório derivado do PCIM para a adaptação ao modelo de objetos utilizado pelos serviços LDAP. No contexto

do PCIM, este trabalho pretende definir o PDP, o PEP e mapear um conjunto padrão de APIs RBAC em mensagens COPS.

Do ponto de vista da sua utilização, este trabalho almeja fornecer uma aplicação de segurança, implementada em um serviço de diretório que atuará como repositório da informação RBAC, para fornecer um serviço de controle de acesso. Além do modelo proposto, este trabalho pretende definir os algoritmos relacionados com a implementação das regras RBAC envolvendo os objetos de políticas contidos no repositório LDAP e as APIs que atuarão como interface às facilidades oferecidas pela arquitetura. Através destas APIs, aplicações poderão verificar a possibilidade de uma entidade acessar um recurso em um determinado instante de tempo. A partir da utilização dos padrões definidos pelo DMTF, este trabalho permitirá que aplicações heterogêneas de um sistema distribuído compartilhem um conjunto coerente de políticas de acesso.

Diversos modelos RBAC têm sido propostos [FK 92; ON 94; FCK 95], sem que houvesse o estabelecimento de um padrão que especificasse de forma definitiva suas características. Este trabalho utiliza as definições RBAC contidas em [SANDHU 00][2].

1.4. Organização

O desenvolvimento do trabalho apresentado nesta dissertação está estruturado em oito capítulos, organizados da seguinte forma:

- Capítulo 1 - Introdução: Faz uma apresentação geral do contexto deste trabalho.
- Capítulo 2 – O Gerenciamento de Redes Baseado em Políticas: Descreve os principais conceitos e padronizações relacionados ao gerenciamento de redes baseado em políticas.
- Capítulo 3 – O Controle de Acesso Baseado em Papéis - RBAC: Apresenta os principais conceitos relacionados ao controle de acesso baseado em papéis, os quais foram utilizados para a concepção deste trabalho. Além disso, descreve alguns trabalhos que propõem a utilização do RBAC como mecanismo de controle de acesso.

[2] [SANDHU 00] tem por objetivo servir como um modelo de referência para desenvolvedores de aplicações baseadas no RBAC, pois ele define um vocabulário, um conjunto de modelos e funcionalidades, considerando toda a evolução do estado da arte na área de modelos RBAC.

- Capítulo 4 – RBPIM: Uma Proposta de Extensão do PCIM para o Mapeamento do RBAC: Apresenta o modelo proposto por este trabalho.

- Capítulo 5 – O Mapeamento do RBPIM nos Serviços de Diretório LDAP: Apresenta a proposta de mapeamento do RBPIM no LDAP bem como as estratégias e conceitos utilizados para alcançar tal mapeamento.
- Capítulo 6 – A Implementação do RBPIM: Arquitetura, API e Algoritmos do PDP: Propõe uma arquitetura para a implementação do controle de acesso baseado em papéis do RBPIM, define as principais chamadas da API relacionadas ao acesso às facilidades oferecidas pela arquitetura e os principais algoritmos para a implementação do PDP.

- Capítulo 7 – Estudo de Caso e Avaliação da Proposta: Realiza um estudo de caso envolvendo o controle de acesso baseado no *framework* do RBPIM em uma aplicação bancária. Além disso, apresenta o protótipo desenvolvido para validar a arquitetura proposta e os resultados decorrentes da utilização do protótipo para a implementação do estudo de caso.
- Capítulo 8 – Conclusões e Trabalhos futuros: Apresenta as conclusões obtidas neste trabalho e possíveis prosseguimentos do mesmo.

Capítulo 2

O Gerenciamento de Redes Baseado em Políticas

2.1. Introdução

O presente capítulo descreve os principais conceitos e trabalhos relacionados ao gerenciamento de redes baseado em políticas. Com este propósito, este capítulo apresenta nas seções de 2.2 a 2.6 algumas definições importantes relacionadas ao escopo deste trabalho, prosseguindo com a descrição e a evolução dos principais esforços de padronização da área na seção 2.7. A seção 2.8 mostra um exemplo de como o paradigma do gerenciamento baseado em política pode ser aplicado. A seção 2.9 conclui o capítulo.

2.2. Serviços de Diretórios

O serviço de diretório é um repositório que disponibiliza informações utilizadas por outras aplicações e serviços de um sistema distribuído. Um serviço de diretório é basicamente constituído de duas partes: um banco de dados e um protocolo de acesso. O banco de dados de um diretório é uma versão especializada de *DBMS - DataBase Management System*. Ele difere de um *DBMS* de propósito geral por ser especialmente concebido para otimizar as operações de leitura, isto é, ele oferece um tempo de resposta bastante rápido para operações de busca em grandes quantidades de dados, mas o desempenho nas operações de escrita é pobre. Desta forma, os diretórios são indicados como repositórios de dados de informações estáticas, tais como nome de usuários, endereços de email e parâmetros de configuração de dispositivos [YU 98]. O protocolo de acesso define o conjunto de *APIs* para efetuar as operações de consulta e atualização no banco de dados do diretório. A padronização das *APIs* de acesso permite que diferentes aplicativos na rede compartilhem o mesmo serviço de diretório.

Uma característica muito importante do banco de dados de diretório é sua escalabilidade. Um serviço de diretório pode ser implementado na forma de um único

servidor, ou distribuído em vários servidores. No caso da implementação distribuída, os servidores cooperam entre si oferecendo aos clientes do serviço de diretório uma visão unificada do serviço. Essa característica é conseguida devido à estrutura hierárquica adotada pelo serviço de diretório. O serviço de diretório é diferente dos tradicionais sistemas de banco de dados, pois a informação mantida nele é descrita sob a forma de atributos. Estes atributos fornecem a informação específica sobre os vários objetos mantidos no repositório aos clientes que usam o serviço de diretório.

2.3. Especificação X.500

X.500 [X500 88] é o nome dado a um conjunto de padrões desenvolvidos pela ISO/ITU-T que especifica como a informação pode ser armazenada e acessada em um serviço de diretório global. A especificação X500 adota uma estrutura orientada a objetos para armazenar suas informações, onde cada registro de informação é derivado de uma classe. Ela não especifica como o serviço de diretório deve ser implementado, dando foco à definição de como a informação deve ser estruturada e como o servidor e o cliente de diretório devem se comunicar. Segundo a especificação X.500, o cliente se comunica com o serviço de diretório através de um protocolo denominado DAP (*Directory Access Protocol*).

2.4. LDAP

A adoção do X.500 foi dificultada pela complexidade e necessidades impostas para sua utilização, principalmente em relação ao DAP. Do ponto de vista prático, era necessário propor uma versão mais “leve” do serviço e ainda compatível com a pilha TCP/IP.

O LDAP - *Lightweight Directory Access Protocol* [RFC 1777 ; RFC 2251] – foi proposto com o objetivo de desenvolver um protocolo que permitisse implementar um serviço de diretório mais apropriado para a arquitetura TCP/IP. De fato, o próprio nome LDAP (*Lightweigth DAP*) sugere que esse padrão é uma versão simplificada do DAP do X.500. Inicialmente, o LDAP não foi concebido para ser um serviço de diretório completo, mas simplesmente um substituto para o DAP, mantendo o próprio X.500 como serviço de diretório. Além de funcionar como "*front-end*" para o X.500, o padrão LDAP foi estendido, criando também uma versão LDAP do servidor de diretório, cabendo ao próprio servidor LDAP armazenar os dados. Essa versão do LDAP é usualmente referenciada na literatura como "*stand-alone*" LDAP.

O LDAP tornou-se rapidamente um padrão aberto na Internet, sendo que sua evolução foi documentada numa série de padrões elaborados na forma de RFCs. Isto motivou muitos fornecedores a proporem a utilização de um serviço de diretório para auxiliar na administração de serviços e aplicações em ambientes distribuídos. Alguns exemplos importantes são os serviços de diretório implementados em sistemas operacionais comerciais, como o *Microsoft Active Directory* [SL 98] **,** o *Novell's NetWare Directory Services* (*NDS*) [NVL 97] e o servidor LDAP da *SUN* [SDS 01].

2.5. Políticas

As políticas podem ser usadas para o gerenciamento de redes e sistemas distribuídos. Neste contexto, as políticas de gerenciamento têm se concentrado em procedimentos de autorização relacionados à alocação de recursos dentro de uma rede ou sistema e em procedimentos relacionados à qualidade de serviço (QoS – *Quality of Service*). Embora haja grande similaridade nestes conceitos e técnicas usados por diferentes comunidades, não há uma notação comum aceita para definir políticas. No contexto deste trabalho, as políticas estão relacionadas com o problema de alocar recursos baseando-se nos objetivos impostos pelas regras de negócio corporativas.

As políticas podem ser definidas a partir de duas perspectivas [RFC 3198]: 1) Um objetivo ou método de ação definidos para guiar e determinar as decisões presentes e futuras; 2) Um conjunto de regras para administrar, gerenciar e controlar o acesso a recursos. Neste contexto, para melhor visualizar uma rede controlada por política, primeiro deve-se modelar a rede como uma máquina de estados e então usar a política para controlar qual o estado em que seus dispositivos deveriam ou poderiam estar em um determinado instante de tempo [RFC 3060].

As políticas também podem ser vistas como um conjunto de regras que especificam como a rede deveria atuar para resolver os conflitos gerados em função das interações entre usuários e aplicações disponíveis na rede. Elas especificam quais recursos e aplicações estarão acessíveis a quais usuários, permitindo a classificação de diferentes aplicações e usuários em categorias, possibilitando um nível de seletividade quanto ao atendimento das decisões referentes às regras de negócio.

De maneira geral, implementar decisões de políticas passa pela configuração das diversas entidades envolvidas no processo de segurança. A Figura 2.1 mostra uma visão do

framework de políticas. A informação de política normalmente é armazenada em repositórios de dados baseados em serviços de diretórios. Através de um protocolo padronizado, a entidade de gerenciamento acessa este repositório, onde obtém os objetos que descrevem políticas. Então, o gerenciador de políticas implementa as especificações das regras de negócio corporativas descritas pelos objetos de políticas, as quais resultam em decisões que devem ser implementadas, quando os estados/configurações das diversas entidades controladas pelas políticas são modificados.

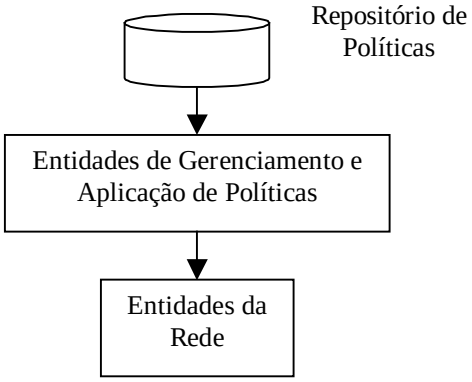

Figura 2.1: Visão conceitual do *framework* de políticas

2.6. Modelo de Informação

Um modelo de informação é uma abstração do conhecimento para que este possa ser entendido antes de ser implementado. No contexto deste trabalho, ele estrutura o conhecimento sobre os objetos da rede (usuários, aplicações, serviços, dispositivos, etc.) e suas interações através de um esquema orientado a objetos. Este esquema consiste em um conjunto de classes extensíveis que são utilizadas para representar os objetos reais da estrutura computacional.

2.7. Trabalhos de Padronização

Devido à sua simplicidade, o LDAP se tornou rapidamente um sucesso comercial, sendo implementado em diversos sistemas operacionais e servidores de email para o compartilhamento de informações sobre usuários entre diferentes aplicações. Muitos fabricantes de *software* e *hardware* perceberam que o LDAP também poderia ser usado para o armazenamento de outros tipos de informação, tais como configurações. Entretanto, para suportar estas novas aplicações LDAP, seria necessário criar padrões de esquemas[3] de

[3] Um esquema de diretório contém as definições formais das classes e atributos a partir das quais objetos podem ser inseridos no diretório.

diretório. Inicialmente, o IETF, responsável pela definição das especificações LDAP, não demonstrou intenção de criar esses novos padrões. Como conseqüência, outros organismos assumiram a tarefa de criá-los, objetivando dar suporte a interoperação entre produtos comerciais. Entre estas iniciativas está o DEN - *Directory Enabled Networks*, mostrado na próxima subseção.

2.7.1. Directory Enabled Networks (DEN)

Os sistemas distribuídos são caracterizados pela heterogeneidade dos elementos a serem administrados. Além disso, o uso de repositórios de dados específicos para cada aplicação limita a interoperabilidade, tornando o processo de integração mais complicado e custoso. No contexto do gerenciamento baseado em políticas, ocorre a necessidade de gerenciar dispositivos de modo semelhante ao que é feito com usuários e, ainda, é preciso estabelecer uma ligação entre estes dispositivos, aplicações e serviços.

Uma rede "*directory enabled*" é uma rede onde perfis de usuários, aplicações e serviços de rede são integrados através de um modelo de informação comum que armazena o estado da rede e expõe a informação associada a ela [YU 98]. O DEN fornece um modelo de informação que define as abstrações para o gerenciamento de políticas, dispositivos, protocolos e serviços. Com isto, ele disponibiliza um modelo unificado para a integração de usuários, aplicações e serviços baseados em rede. A Figura 2.2 mostra a filosofia da operação do DEN [STR 99].

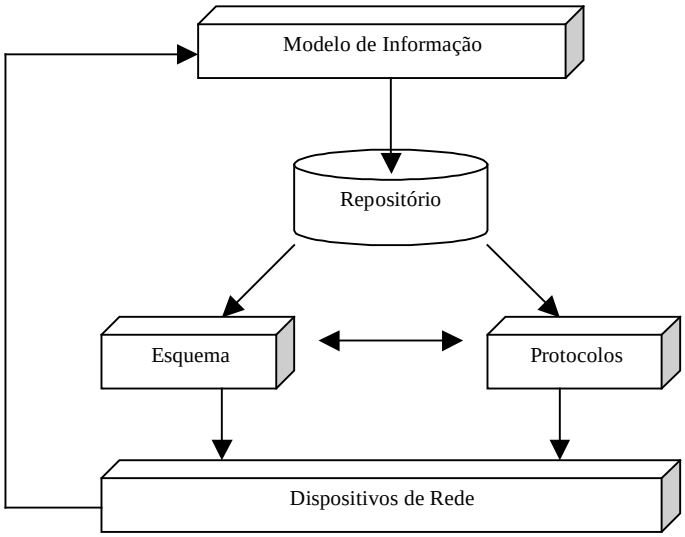

Figura 2.2: Visão geral da filosofia do DEN

Além deste modelo de informação, o DEN especifica também um esquema, permitindo o uso, de maneira padronizada, de diretórios como repositórios de informações de gerenciamento. Uma arquitetura baseada no uso de diretórios como repositórios centralizados de dados possibilita a integração de informações de gerenciamento provenientes de uma grande diversidade de fontes, construindo-se, assim, as bases para o gerenciamento baseado em políticas. Os benefícios da adoção da proposta do DEN são evidentes, entres os quais estão a eliminação de tarefas redundantes, alcançada pela centralização do gerenciamento, e os procedimentos de autenticação e autorização centralizados. A estrutura do modelo de informação do DEN segue o paradigma da modelagem orientada a objetos. O DEN usa o protocolo LDAP para acessar e gerenciar a informação do diretório.

2.7.2. Commom Information Model (CIM)

A definição de esquemas de diretórios suficientemente genéricos para acomodar as particularidades de dispositivos de redes e políticas de segurança é uma tarefa bastante difícil. Por causa disto, vários grupos de trabalhos dentro do DMTF continuam envolvidos. Inicialmente, o DEN foi lançado como uma especificação individual e independente. Hoje, o DEN é considerado um trabalho concluído e seus conceitos foram incorporados a um modelo de informação mais genérico chamado CIM -*Common Information Model*. O CIM [DMTF 99] é um modelo de informação orientado a objetos que descreve como um sistema e seus componentes podem ser gerenciados, permitindo o estabelecimento de uma visão consistente do ambiente, independente dos protocolos e formatos de dados utilizados. A implementação do CIM é neutra, ou seja, suas especificações não são particulares a uma arquitetura. Ele define como a informação de gerenciamento pode ser representada de maneira lógica, mas não como ela deve ser armazenada, sendo composto por dois componentes: 1) Uma Especificação, a qual define detalhes de integração com outros modelos de gerenciamento; 2) Um Esquema, o qual fornece a própria descrição do modelo. O CIM é um modelo de informação dividido em camadas: *Core Model*, *Commom Model* e os *Extension Schemas*. O *Core Model*, mostrado de forma simplificada na Figura 2.3, descreve os conceitos comuns aplicáveis a todas as áreas do gerenciamento.

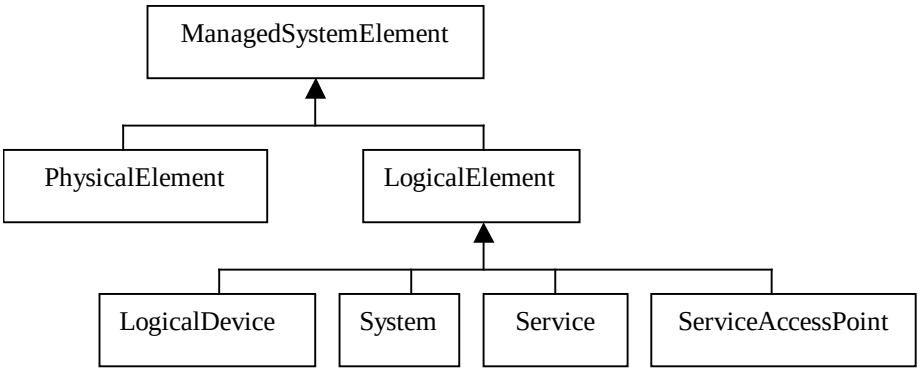

Figura 2.3. – O Core Model do CIM

ManagedSystemElement é a classe base para toda a hierarquia. *LogicalElement* é a classe base para todos os elementos que representam componentes abstratos do sistema, tais como perfis, processos ou funcionalidades. *System* é um agrupamento de *LogicalElements*. *Service* e *ServiceAccessPoint* definem mecanismos para representar as estruturas que fornecem o acesso às funcionalidades de um sistema. *LogicalDevice* é uma abstração (emulação) de uma entidade de *hardware*. Uma subcamada, chamada de *Common Model*, refina os conceitos genéricos apresentados pelo *Core Model*, definindo modelos para dar suporte às abstrações que ocorrem em domínios específicos de gerenciamento, como mostrado parcialmente na Figura 2.4.

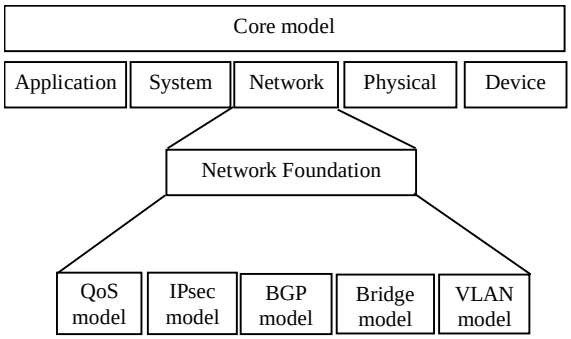

Figura 2.4: CIM – A arquitetura em camadas

Entre eles estão os modelos *System, Device, Application, Network* e *Physical*. Por exemplo, o *Core Model* define genericamente um serviço. O *Network Model* (*Common Model*) refina este conceito para descrever diferentes tipos de serviços que são específicos para redes, tais como *forwarding* e *routing traffic*. O *Commom Model* fornece um conjunto de classes que serve como base de extensão para esquemas específicos a uma tecnologia. O *Core Model* e o *Common Model* juntos compõem o *CIM Schema*.

A terceira camada do CIM, *Extension Schemas*, representa extensões do *Common Model* particulares a uma determinada tecnologia. Estes esquemas são específicos a um ambiente, tais como os sistemas operacionais. Ainda na Figura 2.4 podem ser observadas algumas extensões ao *Network Model*, como o *QoS* e o *IPsec*. O DEN estende o CIM em diversas maneiras, entre as quais se destacam: a) a adição da modelagem de dispositivos e serviços de redes; b) o estabelecimento do mapeamento dos modelos do CIM em serviços de diretórios baseados no LDAP; e c) a integração dos conceitos dos diretórios X.500 com os serviços de diretórios que usam o LDAP como protocolo de acesso. A Figura 2.5 mostra uma representação simplificada do modelo de informação definido pelo DEN. Como pode ser visto, o DEN incorpora conceitos do CIM e do X.500. Os conceitos definidos pelo CIM são usados pelo DEN para dar suporte à modelagem dos serviços e elementos da rede, enquanto o modelo do X.500 é usado para a integração destas entidades a usuários e aplicações [YU 98].

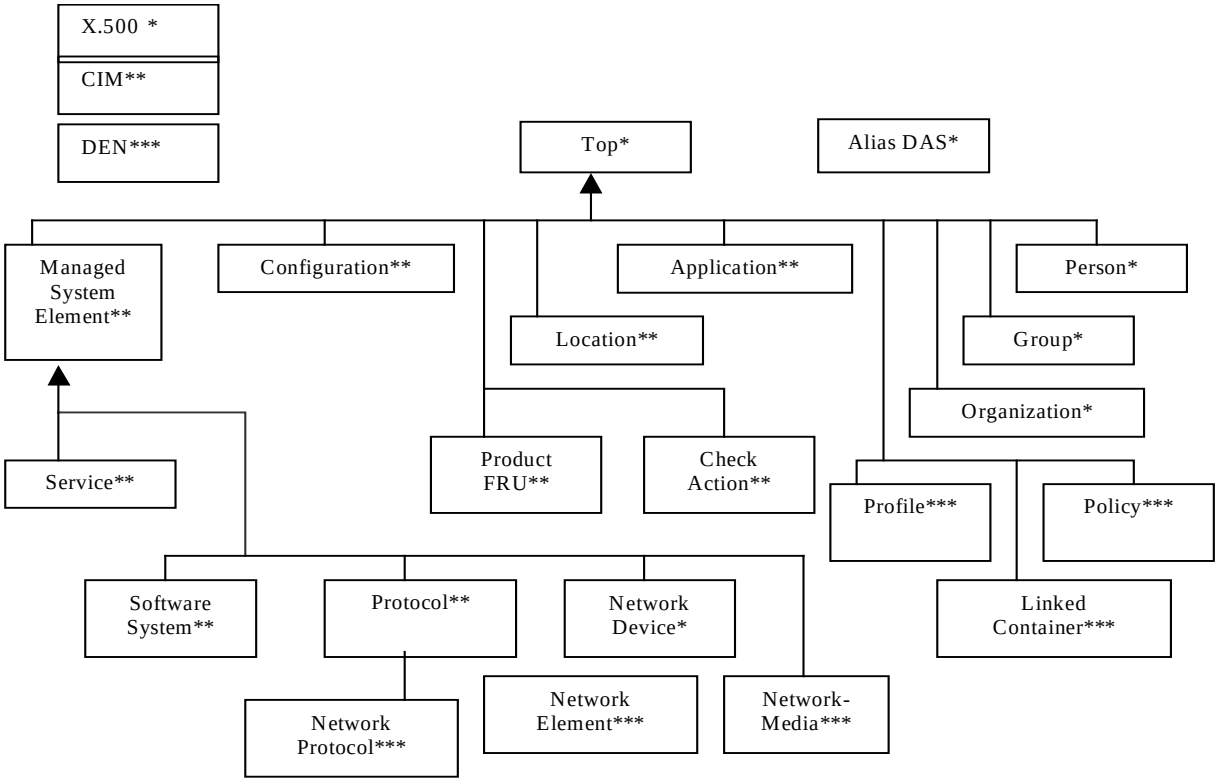

Figura 2.5: O Modelo de Informação do DEN

Muitas classes do CIM ainda requerem especificações adicionais, especialmente aquelas que visam dar suporte à descrição de políticas[4]. Recentemente, o DMTF e o IETF juntaram forças para elaborar um conjunto mais refinado de classes de política. Como

[4] Referidas pelo DMTF como *Policy Classes*

resultado deste trabalho, foi definido um novo modelo de informação o qual foi chamado de PCIM- *Policy Core Information Model* - [RFC 3060], mostrado na próxima subseção.

2.7.3. Policy Core Information Model (PCIM)

O PCIM é um modelo orientado a objetos que permite a representação da informação de política. Este modelo foi especificado de uma maneira genérica tal que possibilite a representação de política de qualquer natureza.

As políticas representam os objetivos (metas) relacionados às regras de negócio. Estes objetivos são descritos de forma intuitiva, através de uma linguagem de alto nível. Faz-se necessário, assim, a realização da tradução destes objetivos para uma forma na qual eles possam ser implementados na rede [RFC 3060]. Como ilustração, a Tabela 2.1 apresenta alguns objetivos de política.

• "Todos os diretores podem acessar o módulo de relatórios da aplicação de informações gerenciais da empresa";
• "Todos usuários pertencentes ao departamento de engenharia precisam que o tempo de acesso à base de dados de projetos seja de no máximo 2 segundos";
• "Todo acesso realizado via protocolo HTTP ao servidor central proveniente da Filial 1 deve usar criptografia";
• "Não permitir o acesso aos serviços de *home banking* no servidor central entre 24:00hs e 6:00hs".

Tabela 2.1: Exemplos de objetivos de política

Um exemplo no qual este processo de tradução se aplica é o *Service Level Agreement* (SLA) e seus objetivos *Service Level Objetives (*SLOs). A [RFC 3198] define SLA como o resultado documentado da negociação entre um cliente e um fornecedor de serviços que especifica os níveis de disponibilidade, operação, *performance* e outros atributos do serviço; e SLO como o particionamento de um SLA em métricas e informações operacionais para o cumprimento e o monitoramento das especificações do SLA. Este esquema é ilustrado na Figura 2.6. Normalmente, as especificações SLAs são escritas em uma linguagem de alto nível baseada em uma terminologia relacionada às regras de negócio. Os objetivos SLOs são descritos através de informações mais específicas para dar suporte às SLAs [RFC 3060].

Figura 2.6 – O Meta Modelo de Política

A modelagem de classes proposta pelo PCIM tem por objetivo servir como base para a tradução da descrição dos serviços de rede e métricas para um nível tal que possa indicar quais mecanismos ativar e quais parâmetros suprir para que os objetivos sejam alcançados.

2.7.3.1 Descrição do Modelo

O PCIM define duas hierarquias de classes de objetos: classes estruturais, as quais definem a forma de representar e controlar a informação de política; e classes associativas, que indicam como as instâncias das classes estruturais se relacionam [RFC 3060].

As classes que compõem o PCIM foram especificadas com o intuito de servir como uma hierarquia de classes extensível para a definição de objetos de políticas que visam capacitar os desenvolvedores de aplicações, os administradores de rede e os administradores de políticas a representarem a informação de política de diferentes tipos. Nesta abordagem, uma política é aplicada usando um conjunto de *policy rules* (regras de políticas). Cada *policy rule* consiste de um conjunto de *conditions* (condições) e um conjunto de *actions* (ações). Se o conjunto de condições associado a uma regra de política for validado como verdadeiro, então o conjunto de ações associado à mesma regra será executado, possibilitando, desta forma, que procedimentos relacionados ao(s) objeto(s) controlado(s) sejam executados, como mostra

esquematicamente a Figura 2.7. O *framework* do PCIM permite também a definição de um conjunto de períodos, os quais podem informar os intervalos de tempo que uma determinada regra de política pode ser ativada.

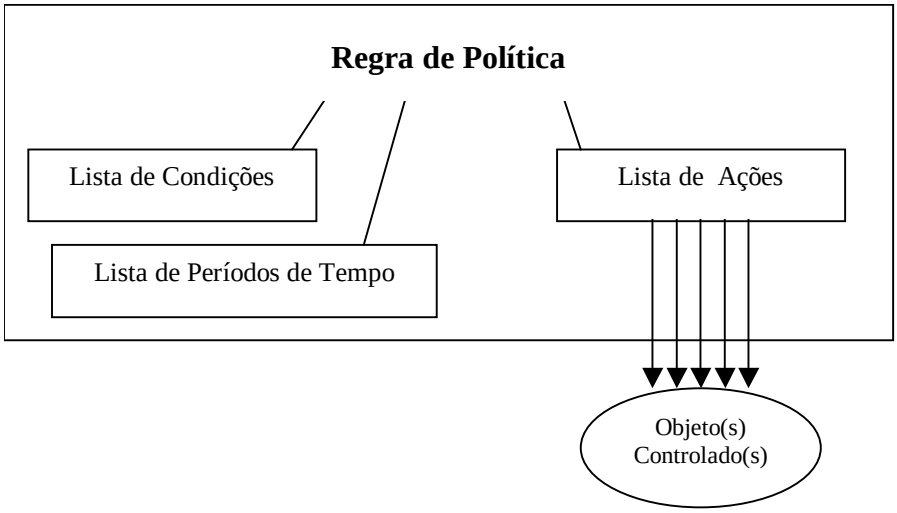

Figura 2.7 – PCIM: Visão Conceitual

A Figura 2.8 mostra a hierarquia das classes estruturais do PCIM.

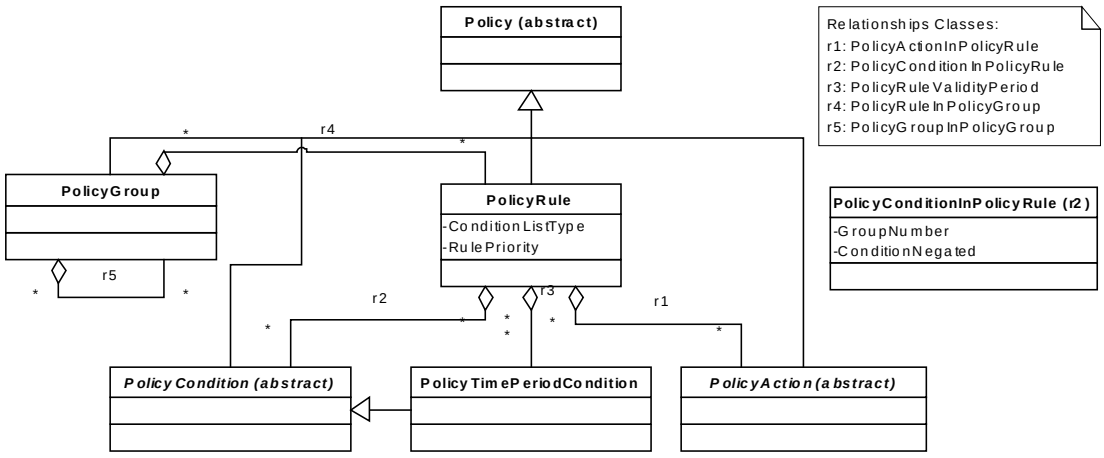

Figura 2.8 – PCIM: Principais classes do modelo

A classe abstrata *Policy* atua como base da hierarquia do PCIM. A classe *PolicyGroup* permite que *PolicyRules* ou *PolicyGroups* sejam agrupados, conforme indicado na figura pelos relacionamentos *r4* e *r5*. Por exemplo, uma determinada empresa poderia criar um conjunto de regras de políticas específicas para cada uma de suas filiais. Assim, poderia existir um *PolicyGroup* para cada uma dessas filiais contendo suas regras específicas. Para saber quais as regras de políticas estão associadas a cada filial bastaria, portanto, obter o

PolicyGroup relacionado a ela. Um *PolicyGroup* pode ser visto como um *container* que agrega regras de políticas e outros *PolicyGroups*. Por exemplo, em um serviço de diretório LDAP, um *PolicyGroup* poderia ser implementado como uma entrada de nível superior que possui outras entradas hierarquicamente inferiores referentes às regras de políticas e outros *PolicyGroups*.

Na Figura 2.8, a classe *PolicyRule* representa a semântica "Se *Condição* Então faça a *Ação*" associada com a política. As condições e ações associadas com uma regra de política, respectivamente, pelos relacionamentos *r2* e *r1*, são modeladas com subclasses de *PolicyCondition* e *PolicyAction*. Uma regra de política pode também ser associada com um ou mais períodos de tempo, indicando quando ela poderá ser aplicada ou não. Períodos de tempo são modelados pela classe *PolicyTimePeriodCondition* e são associados a uma regra pelo relacionamento *r3*.

Os objetos que representam regras (*PolicyRule*) podem ser priorizados em relação a outros através do atributo *RulePriority*, fornecendo os instrumentos para a solução de conflitos entre eles. As ações associadas a cada uma destas regras podem ser ordenadas, possibilitando a especificação da seqüência das ações a serem executadas.

As condições associadas às regras são construídas de duas maneiras: 1) DNF (*Disjunctive Normal Form*), um conjunto de condições construídas com o operador lógico AND e unidas pelo operador lógico OR; 2) CNF (*Conjunctive Normal Form*), um conjunto de condições construídas com o operador lógico OR e unidas pelo operador lógico AND. O valor da propriedade *ConditionListType* da classe *PolicyRule* define qual destas duas formas será utilizada para agrupar as condições associadas à regra de política: 1 para DNF e 2 para CNF. Este esquema é mostrado na Figura 2.9. Além disso, as condições ainda podem ser negadas individualmente (atributo *ConditionNegated* de *r2*) e agrupadas através da atribuição de um valor numérico do tipo inteiro (atributo *GroupNumber* de *r2*), onde todas aquelas que possuírem valores idênticos são consideradas pertencentes ao mesmo grupo de condições. Por exemplo, admitindo uma regra que agregue cinco *PolicyConditions*, chamadas de C1, C2, C3, C4 e C5, e com os seguintes valores de agrupamento e negação [RFC 3060]:

- C1: Grupo = 1, Negação = FALSE
- C2: Grupo = 1, Negação = TRUE
- C3: Grupo = 1, Negação = FALSE
- C4: Grupo = 2, Negação = FALSE
- C5: Grupo = 2, Negação = FALSE

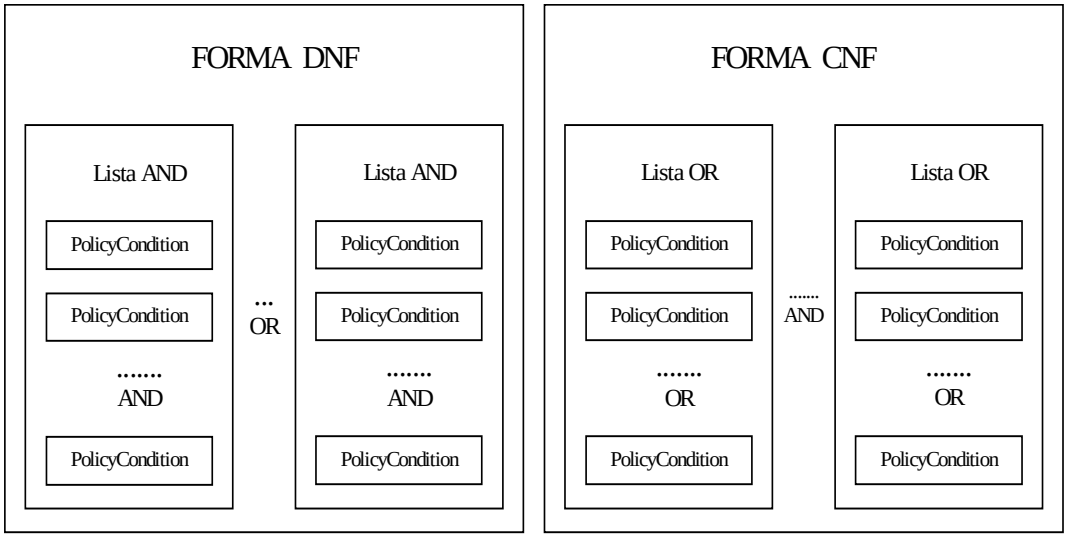

Figura 2.9 – PCIM: *PolicyConditions* nas formas DNF e CNF

Para a forma DNF, a condição geral para validação desta regra será:

a) (C1 AND (NOT C2) AND C3) OR (C4 AND C5)

No caso da forma CNF:

b) (C1 OR (NOT C2) OR C3) AND (C4 OR C5)

Em ambos os casos, *a* e *b*, estas expressões lógicas são testadas para verificar a possibilidade de execução das ações associadas a esta regra [RFC 3060]. Vale lembrar que, caso esta regra ainda agregue objetos que indiquem períodos de tempo (classe *PolicyTimePeriodCondition*), estes intervalos serão considerados para a ativação da regra.

O *framework* do PCIM também permite a associação de papéis a cada regra de política. Para tanto, ele 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. Assim, através da associação de papéis aos recursos da rede, regras de políticas podem ser associadas a estes recursos, permitindo que o *framework* de política fique responsável pela configuração dos recursos da rede, de tal forma que ele passe a ter seu comportamento controlado de acordo com as políticas especificadas para aquele papel [RFC 3060]. Neste trabalho, pensou-se inicialmente em aproveitar esta propriedade para fazer o mapeamento de políticas RBAC no PCIM através

da associação de papéis RBAC a ela, no entanto, como será mostrado no Capítulo 3, a riqueza da descrição de políticas de segurança RBAC exige um esquema mais poderoso, que vai além da mera utilização do atributo *PolicyRoles.*

O PCIM divide condições e ações em duas categorias: 1) "*rule-specific*", quando as condições e ações estão associadas a apenas uma regra; 2) "*reusable*", quando as condições e ações podem estar associadas a mais de uma regra. Uma classe denominada *PolicyRepository* (não mostrada na Figura 2.8) representa um *container* para o armazenamento de condições e ações que estão na categoria 2, ou seja, são condições e ações "reusáveis". Como poderá ser visto no Capítulo 4, o mapeamento do *framework* RBAC proposto por este trabalho utiliza apenas *PolicyConditions* e *PolicyActions* enquadradas na categoria 1. Optou-se por esta categoria por questões de simplificação do modelo, ficando a incorporação das classes que dão suporte ao reuso de condições e ações para um trabalho futuro.

Além das classes estruturais, o PCIM fornece diversas classes que dão o suporte para as associações e agregações que surgem a partir do relacionamento entre elas. Conforme os relacionamentos presentes na Figura 2.8, uma associação é modelada como uma classe, normalmente contendo referências para dois objetos. Uma classe associativa não afeta de nenhuma forma as classes associadas. Uma agregação é uma forma "forte" de associação, o que geralmente representa um relacionamento de coleção. Por exemplo, o CIM usa uma agregação para representar o relacionamento entre um sistema e seus componentes, onde um sistema agrega ("é composto por") seus componentes [RFC 3060]. A Figura 2.10 mostra a hierarquia das classes associativas. Uma *PolicyRule*, por exemplo, agrega zero ou mais instâncias de objetos da classe *PolicyCondition* através da associação *PolicyConditionInPolicyRule*. As classes *Dependency, Component,* e *SystemComponent* são definidas pelo esquema do CIM. Uma associação sempre conecta duas classes, as quais, eventualmente, podem ser a mesma classe, como *PolicyGroupInPolicyGroup.*

No Capítulo 5 será mostrada a estratégia de mapeamento destas classes em serviços de diretórios LDAP.

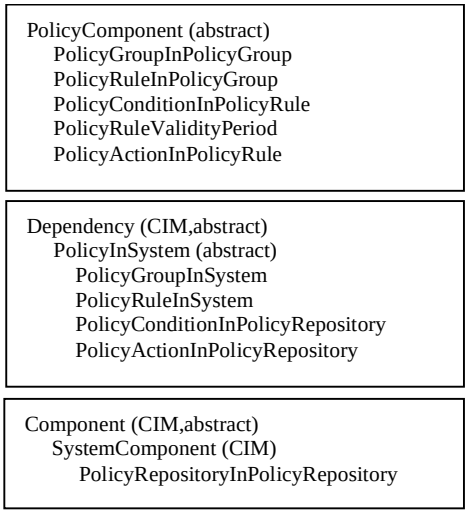

Figura 2.10 – PCIM: Classes Associativas

2.7.3.2 Arquitetura para Implementação

A arquitetura para implementação deste *framework* é centrada em duas entidades principais:

- **PDP** – *Policy Decision Point*: Entidade responsável pelas decisões de política.
- **PEP** – *Policy Enforcement Point*: Entidade responsável pela implementação das decisões de políticas apuradas pelo PDP.

A Figura 2.11 mostra uma ilustração desta arquitetura, a qual será utilizada como base para a implementação do modelo proposto por este trabalho. Em um cenário típico da interação entre as entidades, o PEP precisa verificar a possibilidade de acesso de uma determinada aplicação a um recurso da rede. Neste caso, ele formula um pedido e o envia ao PDP. O PDP retorna a decisão, ficando o PEP responsável pela realização dos procedimentos necessários para que ela seja cumprida. Através de uma ferramenta de gerenciamento, as informações de políticas são inseridas em um repositório, o qual é tipicamente implementado por um serviço de diretório como o LDAP. O PDP utiliza as informações contidas no repositório para efetuar as decisões de política. Estas informações de política podem ser provenientes de várias fontes, ficando o PDP responsável pela sua tradução e validação.

Com a responsabilidade de implementar as decisões de políticas, procedimentos como a filtragem de pacotes, o controle de acesso e a criptografia são exemplos de tarefas que podem ser executadas pelo PEP. O PEP pode ser visto como uma entidade presente em um

nó da rede enquanto o PDP é uma entidade remota que fornece um serviço baseado em decisões de políticas.

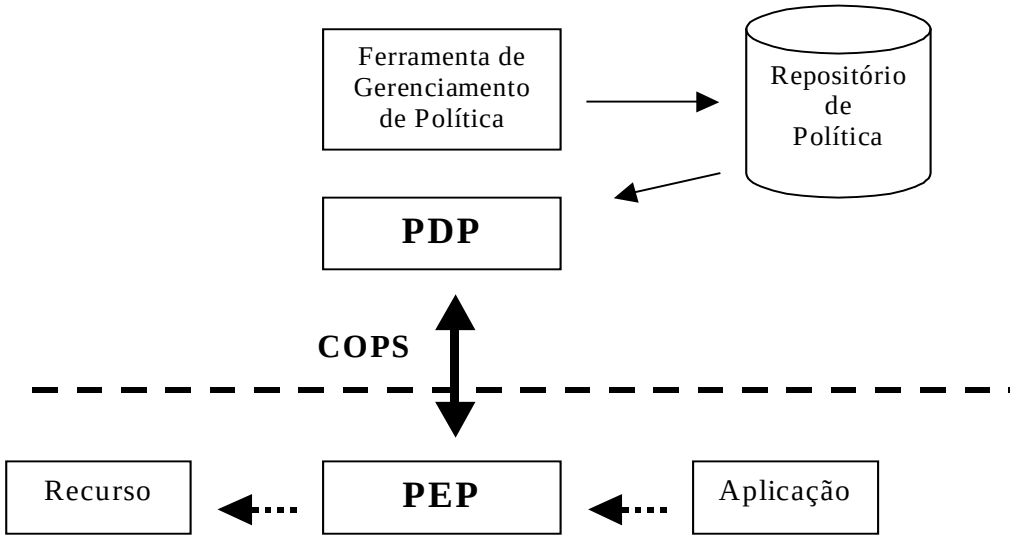

Figura 2.11 – Arquitetura para implementação do *framework* de Políticas

Nesta arquitetura, a interação entre o PDP e o PEP se dá através do protocolo COPS [RFC 2748]. O COPS propõe dois modelos para o controle baseado em políticas, o *Outsourcing* e o *Provisioning*. O modelo *Outsourcing* é usado tipicamente em cenários onde o PEP precisa de uma decisão de política instantânea. Neste caso, ele envia uma requisição ao PDP e aguarda a decisão de política. No modelo *Provisioning* não há correlação direta entre os eventos do PEP e decisões do PDP. Nesta situação, o PDP passa suprir o PEP, de forma pró-ativa ou a pedido do próprio PEP, com a informação de política. Como será detalhado no Capítulo 6, este trabalho optou pela utilização do modelo *Outsourcing* para a implementação da arquitetura definida para o RBPIM. Para resumir, a Tabela 2.2 apresenta as principais características das entidades desta arquitetura.

Ferramenta de Ger. de Política	**PDP**	**PEP**
• Interação com o administrador; • Gerenciamento da informação de políticas no repositório.	• Consome informações de políticas armazenadas no repositório; • Realiza decisões de políticas.	• Solicita decisões de políticas ao PDP; • Implementa as decisões provenientes do PDP.

Tabela 2.2: Principais características da ferramenta de ger. de políticas, do PDP e do PEP

A título de ilustração, suponha a seguinte política:

"O acesso a uma aplicação financeira (porta 8080) que está sendo executada em um servidor WEB(W) deve ser feito através do protocolo HTTP e apenas pelos usuários provenientes da rede da filial 1."

Neste contexto, o administrador do sistema, através da ferramenta de gerenciamento, insere uma regra de política expressa segundo o modelo do PCIM como:

***If* protocolo = HTTP and destino = W and porta = 8080 and origem = filial 1**
autorizar acesso
else
negar acesso
endif

Quando necessário, o PEP, que interage diretamente com as entidades da rede, solicita ao PDP uma decisão baseada nas regras de políticas pré-definidas para verificar se ele deve ou não autorizar o acesso, suprindo-o com os parâmetros necessários para que ele realize tal procedimento. Então, o PDP consulta os objetos de política inseridos no repositório, seleciona a(s) regra(s) de política relacionada(s) ao acesso requerido e toma a decisão baseada nesta(s) regras(s). Neste exemplo, a regra de política selecionada é representada pela expressão lógica mostrada acima, cujo resultado, autorizar ou negar o acesso, é enviado ao PEP. Obviamente, a decisão de autorizar ou negar o acesso precisaria, nesse ponto, ser implementada. Um exemplo desta situação seria a aplicação de novas configurações aos dispositivos da rede que implementam o controle de acesso. Este é um processo complexo que deve levar em consideração o conhecimento da forma de configuração dos dispositivos envolvidos e da topologia da rede. Um trabalho nesta direção pode ser visto em [CISCO 99], onde é apresentado um *framework* que utiliza um processo chamado *policy compilation*. Este processo aplica técnicas de compilação no ambiente do gerenciamento baseado em políticas para a geração da informação de configuração de dispositivos.

2.8. Outro trabalho relacionado

2.8.1 Integrando a especificação de objetivos no gerenciamento baseado em políticas

O trabalho apresentado resumidamente nesta subseção mostra como [BS 01] utilizou e estendeu o PCIM para realizar a representação e o gerenciamento da informação de política.

Conforme mostrado anteriormente no exemplo contido na Tabela 2.1, os objetivos de políticas são representados por especificações declarativas que relatam o comportamento esperado do sistema a partir da execução de eventuais procedimentos, ou seja, eles informam (1)*o que o usuário deseja.* Por outro lado, eles não especificam (2)*como* atingir estes objetivos, ou seja, qual a seqüência de ações necessárias para alcançá-los. A partir do ponto de vista dos usuários, os trabalhos de padronização na área do gerenciamento baseado em políticas fornecem apenas o suporte ao segundo aspecto (2), não disponibilizando uma forma para a especificação de (1). O trabalho apresentado em [BS 01] propõe um modelo objetivando dar o suporte necessário à representação do primeiro aspecto, o qual foi chamado de ***goal,*** em conjunto com (2), chamado de ***rule***, derivado do PCIM.

Este modelo, construído a partir da extensão do PCIM, é composto por um *core schema*, o qual organiza e define objetos *goals* aplicáveis a várias áreas do gerenciamento e por extensões, resultantes do refinamento do *core schema*, para atender a áreas específicas, tais como QoS e segurança [BS 01]. São definidas duas classes principais:

- **policygoal**: Especifica qual o comportamento esperado do sistema;
- **policyrule**: Derivada do PCIM, indicando quais ações devem ser tomadas para que o sistema passe a ter o comportamento esperado.

Conjuntos de *Goals* e *Rules* podem ser associados para permitir que, em função da execução das ações associadas a regras, os objetivos a elas relacionados sejam satisfeitos [BS 01]. A Figura 2.12 mostra o *core schema* proposto. Uma instância da classe *PolicyGoal* representa uma expressão lógica relacionada ao comportamento do sistema. Um objetivo de política é alcançado através da associação de um conjunto de instâncias da classe *SimpleGoal* com uma instância de *PolicyGoal*. Os atributos definidos pela classe associativa *SimpleGoalInGoal* e pela classe estrutural *PolicyGoal* são utilizados para definir como as instâncias de *SimpleGoal* serão combinadas para a construção das expressões lógicas que representam os objetivos de política. É importante notar que as classes e associações da hierarquia *PolicyGoal* seguem o mesmo esquema proposto pelo PCIM, mostrado na subseção 2.7.3. A classe *PolicyGroup* é definida como uma agregação de outros objetos *PolicyGroup* e combinações de objetos *PolicyRule* e *PolicyGoal*. Ela foi estendida com a inserção do atributo *GroupSemantics*, o qual fornece a semântica de relacionamento entre *goals* e *rules*.

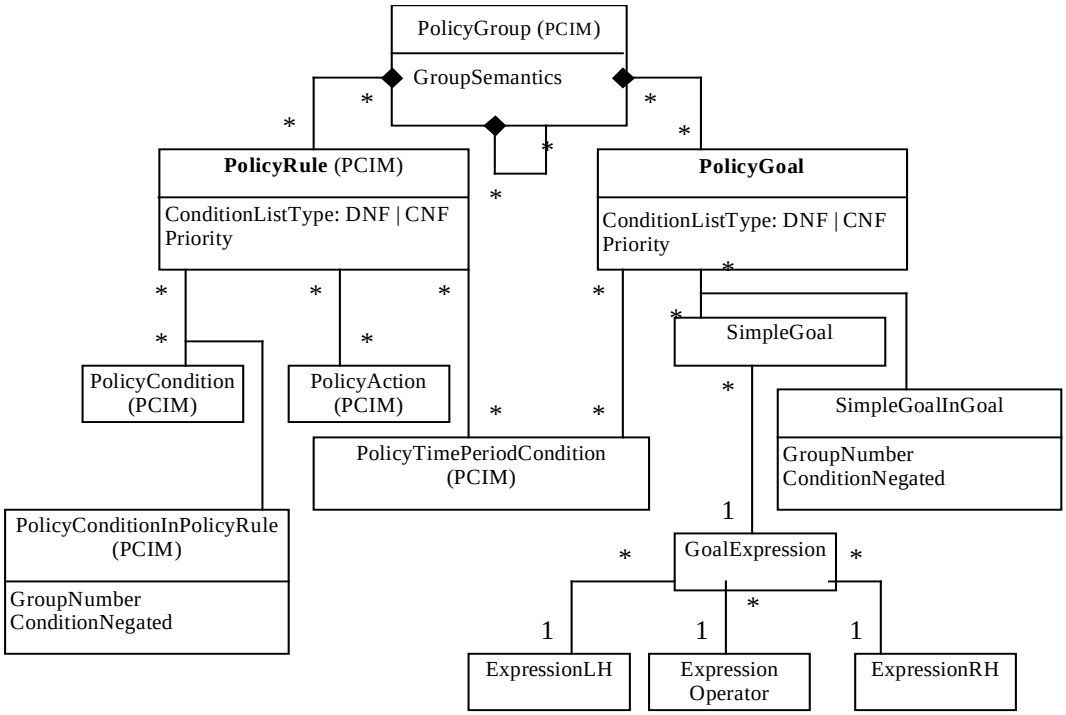

Figura 2.12 – Visão simplificada do Core Schema proposto em [BS 01]

Como exemplo, suponha o seguinte objetivo de política:

"O Tempo de acesso ao servidor WEB A deve ser inferior a 4 segundos"

Através do esquema proposto por este modelo, tem-se a seguinte ***GoalExpression*** compondo um ***SimpleGoal***:

AccessTime < **4**

ExpressionLH *ExpressionOperator* *ExpressionRH*

A partir deste objetivo de política definido e de sua associação a um *PolicyGroup*, objetos *PolicyRule* poderão ser criados e associados ao mesmo *PolicyGroup* para definir as ações necessárias para alcançar este objetivo. Admitindo que o endereço IP do servidor A seja 192.168.1.1, poderia existir a seguinte ***PolicyRule***:

if (dest = 192.168.1.1:80) then set Diffserv=1

PolicyCondition *PolicyAction*

A regra indica que será aplicado um nível diferenciado de serviço, indicado por Diffserv=1, sempre que o destino for a máquina 192.168.1.1 na porta 80. Para maiores informações sobre este modelo, consulte [BS 01].

2.9. Conclusões do Capítulo

O paradigma do gerenciamento baseado em política tem emergido, concentrando-se em procedimentos de autorização relacionados à alocação de recursos dentro de uma rede e na área da qualidade de serviço. As políticas podem ser vistas como um conjunto de regras para administrar, gerenciar e controlar o acesso a recursos.

Em função da popularização dos serviços de diretórios, vários trabalhos de padronização têm sido desenvolvidos dentro do DMTF, objetivando o uso destes serviços como repositórios logicamente centralizados para a informação de gerenciamento, entre os quais se destacam o DEN e o CIM.

Devido à necessidade de criar um padrão para o gerenciamento baseado em política, o DMTF e o IETF juntaram forças para elaborar um modelo mais refinado, o PCIM. O PCIM define um modelo orientado a objetos onde a política é aplicada através de um conjunto de regras. Pela sua própria proposta, o PCIM tem como objetivo servir de base de extensão. De fato, suas classes precisam ser estendidas para suportar a definição da informação de política que atenda as necessidades individuais de aplicações. Um exemplo desta situação é mapeamento de políticas de segurança escritas segundo o modelo RBAC, mostrado no Capítulo 4 deste trabalho. A seção 2.8 apresentou um trabalho que mostra como o PCIM pode ser estendido.

Capítulo 3

O Controle de Acesso Baseado em Papéis - RBAC

3.1. Introdução

O RBAC é uma tecnologia ao mesmo tempo nova e velha. As raízes do RBAC incluem o uso dos grupos em sistemas operacionais e o controle de privilégios nos sistemas gerenciadores de banco de dados nos primeiros sistemas computacionais multiusuários interativos no início da década de 70 [SANDHU 00]. No entanto, a utilização do conceito de papéis em aplicações tem tido mais destaque nos últimos anos. Ela tem sido motivada pela grande quantidade de trabalhos que tem contribuído para a evolução do modelo. Apesar desta quantidade de trabalhos, não existe ainda um padrão aceito globalmente para a definição do RBAC. Algumas características e nomenclaturas utilizadas por um trabalho muitas vezes não são contempladas por outros, o que torna mais difícil a adoção de um determinado modelo.

O presente capítulo descreve os principais conceitos relacionados ao controle de acesso baseado em papéis, os quais foram utilizados para a concepção deste trabalho. Com este propósito, este capítulo apresenta na seção 3.2 algumas definições importantes relacionadas ao controle de acesso e ao RBAC, prosseguindo com a descrição do modelo RBAC segundo dois dos principais trabalhos de padronização nas seções 3.3 e 3.4, um dos quais adotado como referência por este trabalho. A seção 3.5 descreve alguns trabalhos que propõem a utilização do RBAC como mecanismo de controle de acesso. A seção 3.6 conclui o capítulo.

3.2. O Controle de Acesso

O propósito do controle de acesso é limitar as atividades que um usuário legítimo de um sistema computacional pode executar. Ele controla o que o usuário pode fazer diretamente

ou o que aplicações por ele executadas são permitidas fazer [SS 94]. A Figura 3.1 mostra a representação deste esquema.

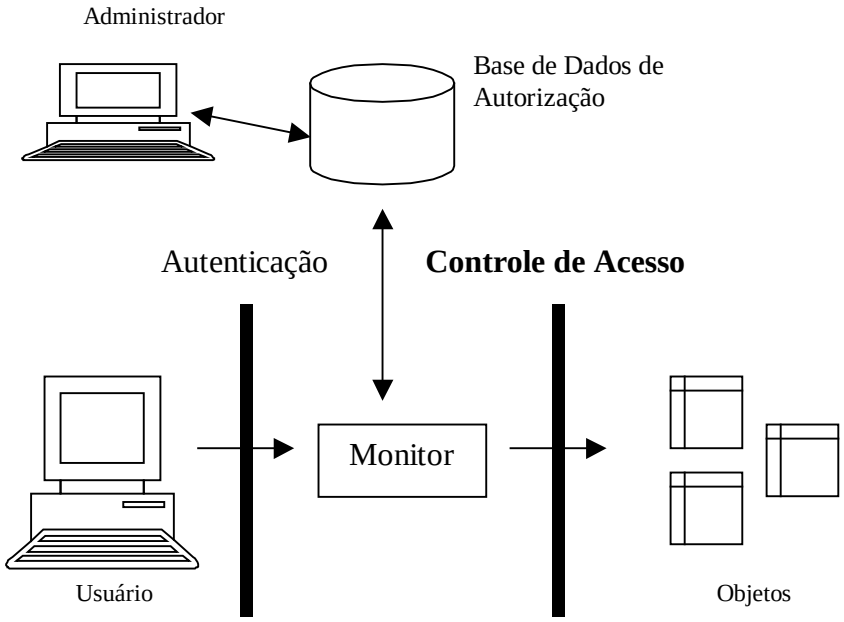

Figura 3.1 – O Controle de Acesso

O processo de autenticação consiste em provar a identidade do usuário. Após esta fase, um processo monitor fica responsável pelo controle de acesso aos objetos do sistema, realizando consultas à base de dados de autorização para verificar se um determinado usuário tem direito de acesso a um dado recurso.

Dentro de um sistema computacional as atividades são iniciadas por entidades conhecidas como *subjects*. Usuários são *subjects* e aplicações que são executadas pela requisição de usuários também. Um usuário pode ser um s*ubject* em uma ocasião e ser um *subject* diferente em outra, dependendo dos privilégios que ele possui em cada situação. Um conceito básico na área do controle de acesso é a associação entre *subjects* e *objects*, ou seja, é preciso controlar o acesso de *subjects* aos objetos do sistema, permitindo ou negando o acesso a cada um deles. Neste caso, a autorização é expressa através de direitos ou modos de acesso. Em um sistema de arquivos, por exemplo, alguns arquivos podem ser lidos(r) e alterados(w), e outros, apenas lidos(r).

Uma maneira natural de implementar o controle de acesso dentro de um sistema é através do uso da matriz de acesso. Uma matriz de acesso é um modelo conceitual que especifica quais os direitos de acesso cada s*ubject* tem sobre cada o*bject* [SS 94] . A Figura

3.2 mostra um exemplo de uma matriz de acesso. Neste exemplo, o Usuário 1 é proprietário do Arquivo 1, possuindo livre acesso a ele, mas apenas tem direito de leitura sobre o arquivo 2. Já o usuário 2 é o proprietário do Arquivo 2, mas não possui direito de acesso ao Arquivo 1.

Object / ***Subject***	Arquivo 1	Arquivo 2
Usuário 1	*Own* *R* *W*	*R*
Usuário 2		*Own* *R* *W*

Figura 3.2 – Uma matriz de acesso

3.2.1. A Implementação do Controle de Acesso

As matrizes de acesso tendem a ser ineficientes para a realização do controle de acesso em sistemas com grandes volumes de dados. Na prática, elas não são implementadas na forma de matrizes, mas através de outras estratégias mais eficientes, como mostrado a seguir.

a) ACL (*Access Control List*):

Nesta implementação, cada objeto possui uma ACL associada, a qual contém informações dos direitos de acesso que cada *subject* possui em relação ao objeto. Com as ACLs fica fácil a obtenção dos direitos de acesso que cada objeto possui, no entanto, fica difícil saber quais os direitos de acesso cada *subject* possui dentro do sistema.

Arquivo 1	Usuário 1: *Own, R,W*
Arquivo 2	Usuário 2: *Own, R,W* Usuário 1: R

Figura 3.3 – ACLs correspondentes à Fig. 3.2

Por exemplo, para saber quais direitos de acesso e *subjects* estão associados ao Arquivo 1 basta ler sua ACL, onde, neste caso, tem-se o Usuário 1 com controle total. Por

outro lado, para saber quais direitos de acesso o Usuário 1 possui, é preciso varrer todos os objetos e verificar se ele tem associação com cada uma das ACLs relacionadas.

b) *Capabilities* (Lista de Permissões):

É uma variação do conceito das ACLs. Enquanto nas ACLs é mantida uma lista de controle de acesso para cada objeto do sistema, através da implementação das c*apabilities* é tomado como referência cada *subject* do sistema, ou seja, para cada um deles é mantida uma lista de permissões contendo todos os objetos e permissões associados.

3.2.2. As Políticas de Controle de Acesso

O controle de acesso nos sistemas computacionais normalmente segue as seguintes políticas [SS 94]:

- Modelo Discricionário;
- Modelo Obrigatório (*Mandatory*);
- Modelo Baseado em Papéis.

3.2.2.1 O Modelo Discricionário

As políticas de controle deste modelo são baseadas na identidade dos usuários. As autorizações especificam para cada usuário (ou grupo de usuários) e cada objeto do sistema os modos de acesso permitidos[SS 94].

3.2.2.2 O Modelo Obrigatório

As políticas de controle deste modelo são baseadas na classificação dos *subjects* e *objects* do sistema. Para cada objeto e usuário do sistema é associado um nível de segurança. O nível de segurança associado ao objeto reflete o nível de importância da informação contida no objeto, classificando o objeto quanto ao dano potencial que um acesso não autorizado traria. O nível de segurança associado ao usuário, também chamado de *clearance*, reflete o nível de confiabilidade do usuário em não expor a informação a um usuário que não esteja autorizado para tal [SS 94]. De maneira geral, o acesso de um *subject* a um objeto é verificado em função do relacionamento entre os níveis de segurança deles e levando-se em conta o tipo de acesso solicitado.

3.2.2.3 O Modelo Baseado em Papéis

As políticas de controle de acesso deste modelo são baseadas nas atividades que os usuários desempenham no sistema. Esta estratégia torna necessária a identificação dos papéis dentro do sistema. Um papel pode ser visto como um conjunto de ações e responsabilidades associadas com uma atividade de trabalho [SS 94]. Uma grande vantagem decorre desta abordagem: Em vez da atribuição de direitos de acesso de forma individualizada a cada usuário, esta atribuição é feita a papéis. Em uma segunda etapa, os usuários são associados a papéis em função das atividades desempenhadas no sistema.

3.2.2.4 Algumas considerações

O modelo RBAC oferece um controle mais flexível do que os modelos discricionários e obrigatórios para o estabelecimento de políticas de autorização. Como pode ser visto, no modelo de acesso discricionário os direitos de acesso são mapeados individualmente a usuários ou a grupos de usuários, um esquema pouco flexível e que limita bastante o gerenciamento das permissões de acesso. Já no modelo obrigatório, ocorre o uso de um esquema de classificação em níveis de segurança, também muito rígido para as necessidades da maioria das organizações.

3.3. O RBAC

Nesta seção serão apresentadas as definições do RBAC contidas em [FCK 95]. No modelo RBAC, os usuários são feitos membros de "papéis", os quais têm direitos de acesso. Ele usa o conceito de "operações", representando as ações que um papel pode executar sobre os objetos compartilhados da rede, e o conceito de "usuários", que representam os membros associados aos papéis. A Figura 3.4 mostra esta situação.

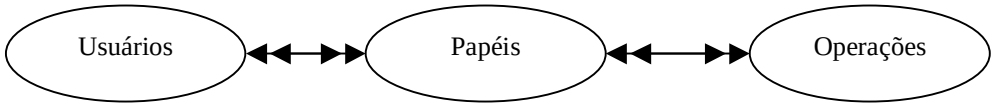

Figura 3.4 – Visão do RBAC

Um usuário pode assumir um ou mais papéis e um papel pode ser assumido por um ou mais usuários, como indicado pelas setas duplas na Figura 3.4. As operações podem ser

associadas ou excluídas dos papéis objetivando atender às eventuais alterações das atividades desempenhadas e representadas por um papel dentro da organização. Observe que os papéis podem ser atualizados sem a necessidade da modificação individual de cada usuário ou grupos de usuários. As alterações passam automaticamente a ser refletidas em todos usuários que possuem associação com o papel.

Um dos principais aspectos motivadores para o uso do RBAC é conceito da separação de tarefas entre papéis (*Separation of duties*). Por exemplo, suponha uma política de autorização construída no cenário de uma agência bancária. Neste caso, podem existir papéis como *Atendente*, *Supervisor* e *Fiscal* e operações como *Agendar Operação de Crédito*, *Autorizar Operação de Crédito* e *Revisar Operações*. Imagine as associações entre operações e papéis mostradas na Tabela 3.1. Duas questões podem observadas: 1) Um funcionário da agência bancária que assume os papéis *Atendente* e *Supervisor* jamais poderá assumir o papel *Fiscal*, uma vez que este último tem por atribuição revisar operações efetuadas; 2) Durante uma sessão não pode ser permitido que um usuário assuma simultaneamente os papéis *Atendente* e *Supervisor*, pois este último tem por objetivo autorizar operações de crédito agendadas. Neste caso, o RBAC permite que se estabeleça um relacionamento de separação de tarefas entre estes papéis para evitar que os casos 1 e 2 possam ocorrer.

Papéis	Funcionário	Operações
Atendente	A	Agendar Operação de Crédito
Supervisor		Autorizar Operação de Crédito
Fiscal	B	Revisar Operações

Tabela 3.1 – Um exemplo simples de política RBAC

Além desta facilidade de gerenciamento, o *framework* proposto pelo RBAC é construído sob o princípio do Privilégio Mínimo (*Least Privilege)*, ou seja, um usuário não precisa ser autorizado a executar operações além das tarefas necessárias para desempenhar sua função dentro da organização. Por exemplo, referindo-se novamente à Tabela 3.1, em uma sessão onde um usuário assume o papel *Atendente* ele somente poderá realizar a operação *Agendar Operação de Crédito*.

Um importante conceito também empregado pelo RBAC é a hierarquia entre papéis. Em um cenário típico de uma organização, indivíduos que desempenham diferentes funções podem executar tarefas em comum. Neste contexto, a hierarquia entre papéis simplifica bastante a descrição de direitos de usuários, pois evitaria a necessidade de especificar de

forma repetida operações em comum. Ainda, este relacionamento hierárquico pode ser balizado pela estrutura hierárquica das próprias organizações, permitindo que a política de segurança seja descrita de maneira extremamente natural. Em uma hierarquia entre papéis, um papel pode "conter" outros papéis. No RBAC, um papel hierarquicamente superior pode conter um ou mais papéis hierarquicamente inferiores. Isto significa que um papel superior tem todos os privilégios e pode executar todas as operações que os papéis de níveis inferiores contêm. A Figura 3.5 ilustra este conceito.

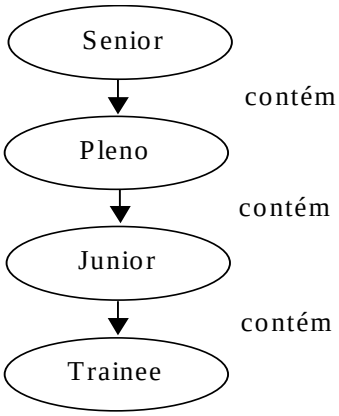

Figura 3.5 – Um exemplo de hierarquia entre papéis

O papel "analista pleno", por exemplo, pode executar todas as operações que o papel "analista junior" pode, ocorrendo o mesmo caso entre os papéis "analista sênior" e "analista pleno". Neste exemplo, o "analista pleno" herda os privilégios do "analista junior" e o "analista sênior" os do "analista pleno". Obviamente, cada papel, além dos privilégios herdados, possui seus próprios privilégios associados.

No RBAC, as políticas são descritas em termos de usuários (*users*), *subjects*, papéis (*roles*) e operações (*operations*). Para que um dado usuário ganhe acesso a um objeto controlado pelo RBAC, inicialmente ele deve ser incluído como membro de um papel e, em seguida, o papel deve ser ativado para aquele usuário. *Subjects* representam os processos ativados pelo usuário. Conseqüentemente, diversos *subjects* podem ser associados a um usuário.

O modelo RBAC pode ser descrito em termos de primitivas, que agem como interface ao modelo, e por regras, que definem precisamente as propriedades do modelo. A seguir, a tabela 3.2 mostra um resumo das primitivas que compõem a definição do modelo.

PRIMITIVA	DESCRIÇÃO
subject-user(s:subject)	Retorna o usuário associado ao subject s.
authorized-roles(s:subject)	Retorna os papéis associados ao subject s.
role-members(r:roles)	Retorna os usuários autorizados para o papel r.
user-authorized-roles(u:user)	Retorna os papéis associados ao usuário u.
role-operations(r:roles)	Retorna as operações associadas ao papel r.
operation-objects(op:operation)	Retorna os objetos pelos quais a operação op pode ser aplicada.
mutually-exclusive-authorization(r:roles)	Retorna a lista de papéis mutuamente exclusivos com r.
membership-limit(r:roles)	Retorna o número limite de usuários que podem receber o papel r.
number-of-members(r:roles)	Retorna o número de membros existentes no papel r.
exec(s:subject, op:operation)	Retorna true: O processo ativo s pode executar a operação op; false: não pode.
active-roles(s:subject)	Retorna a lista de papéis ativos para o subject s.
mutually-exclusive-activation(r:roles)	Retorna a lista de papéis mutuamente exclusivos dinamicamente com r.
function-operations(f:function)	Retorna o conjunto de todas as operações necessárias para executar a função de negocio f.
access(s:subject, o:object)	Retorna true: O processo ativo s pode acessar o objeto o; false: não pode.

Tabela 3.2 – Primitivas de acesso ao modelo RBAC

Por exemplo, baseado na Tabela 3.1 e nas primitivas da Tabela 3.2, as seguintes definições de política estão presentes:

role-members(Atendente) : Funcionário A

role-members(Fiscal) : Funcionário B

role-members(Supervisor) : NULL

role-operations(Supervisor): Autorizar Operação de Crédito

role-operations(Atendente): Agendar Operação de Crédito

number-of-members(Atendente): 1

mutually-exclusive-authorization(Supervisor): Fiscal

Para completar a definição concisa do modelo RBAC, a Tabela 3.3 relaciona as principais regras associadas ao modelo.

REGRA	DESCRIÇÃO
1) *Role Hierarchy* $\forall$ s:subject, $r_{i,j}$: roles $r_j \in$ authorized-roles(s) Λ $r_j > r_i \Rightarrow r_i \in$ authorized-roles(s)	Caso um papel r_j contenha um papel r_i e r_j esteja no conjunto de papéis autorizados ao subject s então r_i também está neste conjunto.
2) *Static Separation of Duty* $\forall$ u:user, $r_{i,j}$: roles : $i \neq j$: $u \in$ role-member(r_i) Λ $u \in$ role-member(r_j) $\Rightarrow r_i \notin$ mutually-exclusive-authorization(r_j)	Alguns papéis não podem ser assumidos simultaneamente por um mesmo usuário, assim, são mutuamente exclusivos.
3) *Cardinality* $\forall$ r:roles membership-limit(r) $\geq$ number-of-members(r)	Esta regra indica que o número de membros de um determinado papel não pode ultrapassar o número máximo de membros permitidos para ele.
4) *Role Authorization* $\forall$ s:subject active-roles(s) $\subseteq$ authorized-roles(s)	Esta regra indica que um determinado subject s não pode ter um papel ativo que não seja autorizado para ele.
5) *Role Execution* $\forall$ s:subject, op: operation: exec(s,op) $\Rightarrow$ active-roles(s) $\neq \varnothing$	Esta regra indica que um determinado subject s pode executar uma dada operação op se ele está atuando dentro de um papel ativo.
6) *Dynamic Separation of Duty* $\forall$ s:subject, $r_{i,j}$: roles : $i \neq j$: $r_i \in$ active-roles(s) Λ $r_j \in$ active-roles(s) $\Rightarrow r_i \notin$ mutually-exclusive-activation(r_j)	Esta regra indica que um determinado subject s pode se tornar ativo em outro papel desde que este novo papel não seja mutuamente exclusivo com qualquer um dos papéis no qual o subject s está correntemente ativo.
7) *Operation Authorization* $\forall$ s:subject, op: operation, r:roles: exec(s,op) $\Rightarrow$ r $\in$ active-roles(s) Λ op $\in$ role-operations(r)	Esta regra indica que um determinado subject s pode executar uma determinada operação somente se ela está autorizada para o papel no qual o subject s está correntemente ativo.

Tabela 3.3 – Regras de definição do modelo RBAC

As regras foram construídas baseadas nas primitivas relacionadas na Tabela 3.2. A regra 1 trata da descrição da hierarquia de papéis, indicando a característica de herança entre eles. As regras 2 e 3 estão relacionadas à associação de usuários a papéis e estão baseadas no fato de que não pode ser dado mais privilégio do que o necessário a um usuário, de que não pode haver conflito entre os papéis assumidos por um usuário e de que, caso exista uma limitação numérica com relação à atribuição de papéis, ela não pode ser excedida. As regras 4,5,6 e 7 tratam do aspecto de ativação de papéis.

Com intuito de mostrar como políticas são mapeadas através do RBAC, a seguir, baseado na hierarquia de papéis mostrada na Figura 3.5 e nas definições indicadas pelas Tabelas 3.2 e 3.3 , será mostrado na Figura 3.6 um exemplo simplificado da descrição de uma política RBAC.

User: Alberto, Luiz**,** Carlos
Subject: Equipe de Projeto, Equipe de Análise, Equipe de programação
Roles: Senior, Pleno, Junior e Trainee.
Operations: Liderar projeto, Orientar analista, Aprovar projeto, Programar módulo do sistema , Analisar sistema
Objects: Módulo do sistema, Projeto
subject user (Equipe de Análise): Alberto, Luiz
subject user (Equipe de Projeto) : Alberto, Luiz**,** Carlos
subject user (Equipe de Programação): Carlos
authorized-roles (Equipe de Análise): Pleno
role-members (Senior): Luiz, Alberto
role-Operations (Pleno): Orientar analista junior,Analisar sistema
mutually-exclusive-authorization (Trainee): Senior
membership-limit (Senior) = 2
membership-limit (Pleno) = 2

Figura 3.6 – Exemplo de política RBAC

3.4. RBAC: O Modelo de Referência

O RBAC possui uma série de funcionalidades e propriedades não cobertas pelos modelos tradicionais de controle de acesso. Se isto por um lado aumenta a riqueza e a capacidade da descrição de políticas de segurança, por outro lado aumenta a complexidade da formalização do próprio modelo.

Esta característica cria a necessidade de refinamentos e de divisão do modelo em submodelos, para que seja possível explorar as várias dimensões do RBAC. [SC 96] e [SANDHU 00] discutem as várias formas de RBAC, objetivando estabelecer um modelo de referência com a padronização de terminologias e funcionalidades para que desenvolvedores de aplicações com mecanismos RBAC embutidos possam especificar de forma mais específica seus produtos e os consumidores de TI possam avaliar e comparar estes produtos. [SC 96] define uma família de quatro modelos conceituais, os quais foram chamados de $RBAC_0$, $RBAC_1$, $RBAC_2$ e $RBAC_3$. O $RBAC_0$ é o modelo base do RBAC, o qual especifica as entidades essenciais e o relacionamento entre elas. O $RBAC_1$ estende o $RBAC_0$, adicionando a característica de relacionamento hierárquico entre papéis. Da mesma forma, o $RBAC_2$ também estende o $RBAC_0$, introduzindo o conceito de restrições (*constraints)* na ativação de papéis, mas sem levar em conta a hierarquia introduzida pelo $RBAC_1$. Neste caso, são especificados os esquemas de Conflitos entre papéis (Estático e Dinâmico) e de Cardinalidade. O $RBAC_3$ é modelo RBAC consolidado, ou seja, incorpora todas as

características dos modelos anteriores, combinando as entidades com a hierarquia de papéis e restrições.

[SANDHU 00] propõe uma padronização sem introduzir novas características ao modelo, mas apenas levando em consideração todo avanço conseguido com a publicação dos diversos trabalhos da área, dando enfoque à definição dos componentes fundamentais do RBAC. Este trabalho está dividido em duas partes: Um modelo de referência, que fornece uma definição formal das entidades do modelo e seus relacionamentos, e uma especificação de necessidades, que define operações administrativas, funções de revisão e funções de sistema que podem ser utilizadas, respectivamente, para a criação e manutenção dos elementos e relações do modelo, para a execução de consultas administrativas e para criar e gerenciar sessões de usuários.

3.4.1. O Modelo de Referência

O objetivo central deste modelo de referência é 1) definir uma nomenclatura contendo termos para serem utilizados em aplicações RBAC e 2) especificar as características e funcionalidades do modelo. Este modelo de referência divide o RBAC em 4 componentes: a) *Core* RBAC, o qual define as entidades básicas do modelo e seus relacionamentos; b) *Hierarchical* RBAC, o qual define as especificações e relacionamentos para dar suporte à hierarquia entre papéis; c) *Static Separation of Duty Relations*, e d) *Dynamic Separation of Duty Relations* os quais definem os relacionamentos de separação de tarefas para assegurar que indivíduos não assumam papéis conflitantes de forma estática e dinâmica, garantindo as políticas de interestes das organizações. Os componentes c) e d) compõem o *Constrained RBAC.*

3.4.1.1. O *Core* RBAC

A Figura 3.7 apresenta o *Core* RBAC. Esta figura mostra os cinco elementos básicos do RBAC: usuários(*USERS*), papéis(*ROLES*), objetos(*OBS*), operações(*OPS*) e permissões(*PRMS)*. Um papel é uma função desempenhada dentro do contexto de uma organização. Uma permissão é uma autorização para realizar uma operação em um objeto controlado pelo RBAC. Estas operações dependem do tipo de sistema no qual o controle de acesso está sendo implementado. Procedimentos de leitura, escrita e inclusão de registros em um tabela de um banco de dados são exemplos de operações.

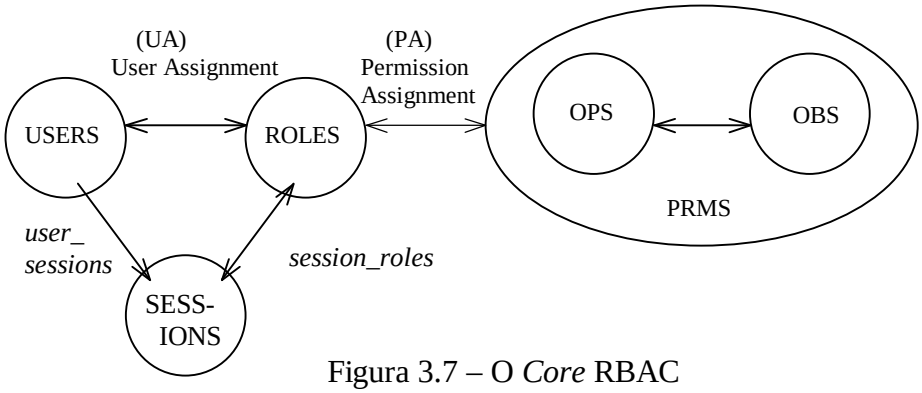

Figura 3.7 – O *Core* RBAC

Um objeto é uma entidade que contém ou recebe informações e o acesso a essas informações precisa ser controlado. Arquivos, diretórios, aplicações, tabelas de um banco de dados e serviços são exemplos objetos passíveis de serem controlados pelo RBAC.

A Figura 3.7 mostra dois relacionamentos importantes: A associação entre usuários e papéis (*UA*) e associação entre permissões e papéis (*PA*). Conforme mostrado anteriormente, este relacionamento é de muitos-para-muitos, indicando que um usuário pode ser associado a vários papéis e um papel pode ser associado a vários usuários. De forma análoga, o mesmo comentário vale para a associação *PA*. Sessões (*SESSIONS*) são criadas quando da ativação de um usuário no sistema, onde papéis são mapeados a um usuário, os quais devem pertencer à lista de papéis autorizados para ele.

3.4.1.2. O *Hierarchical* RBAC

O *Hierarchical* RBAC define o relacionamento de herança entre papéis, uma característica idêntica à apresentada na seção 3.3, relacionada à herança de privilégios. A Figura 3.8 mostra a visão do RBAC hierárquico.

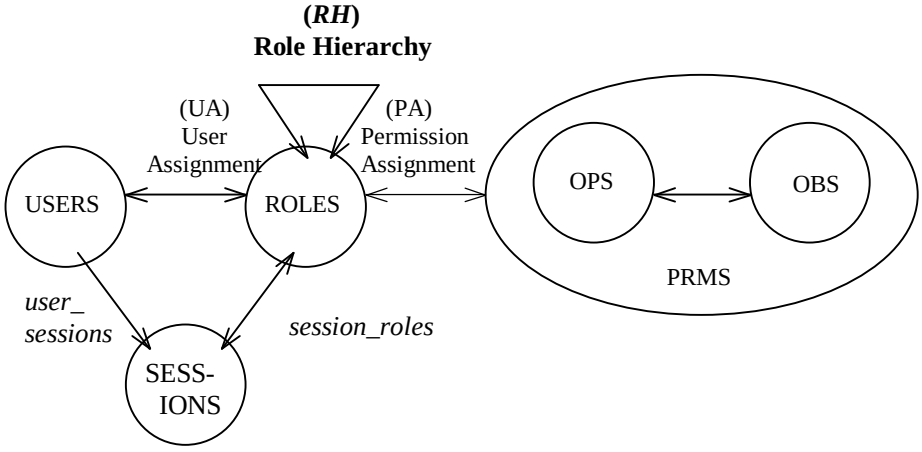

Figura 3.8– O *Hierarchical* RBAC

3.4.1.3. *Static Separation of Duty Relations(SSD)*

O relacionamento de separação estática de tarefas entre papéis é uma forma "forte" de estabelecer o conflito entre as atividades desempenhadas dentro de uma organização. Neste caso, se um usuário está incluído na lista de membros de um papel A e há um relacionamento de *SSD* entre o papel A e um papel B, este usuário não poderá ser incluído na lista de membros do papel B. Para dar maior flexibilidade à construção deste relacionamento, [SANDHU 00] define a relação *SSD* com dois argumentos, o primeiro indica um conjunto de papéis que pode ser composto por dois ou mais papéis, e o segundo informa a cardinalidade do conjunto, ou seja, a quantidade máxima maior do que um de papéis pertencentes ao conjunto correspondente que um usuário pode assumir, qualquer que seja a combinação entre eles. A presença do relacionamento de *SSD* é ilustrada na Figura 3.9.

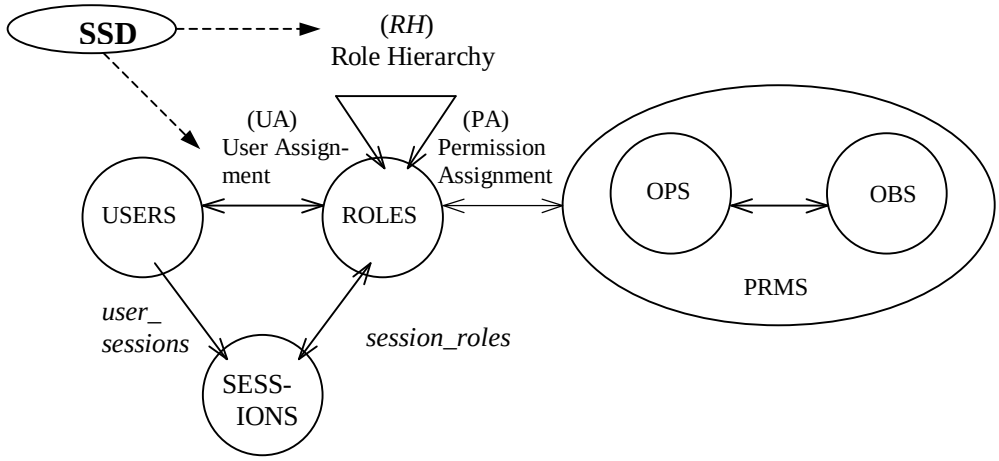

Figura 3.9– O relacionamento de *SSD*

Este relacionamento de *SSD* também deve ser respeitado pelo *Hierarchical* RBAC. Por exemplo, admita um cenário contendo três papéis: A, B e C. Supondo que exista um conflito por *SSD* entre os papéis B e C e que o papel A herda o papel B, então A e C também possuem um conflito por *SSD* entre si.

3.4.1.4. *Dynamic Separation of Duty Relations(DSD)*

De forma semelhante ao relacionamento de *SSD*, o relacionamento de *DSD* também é utilizado para a limitação de permissões, no entanto, de forma menos restritiva, o *DSD* é criado no contexto dinâmico da ativação da sessão do usuário. Neste caso, se um usuário está incluído na lista de membros de um papel A e há um relacionamento de *DSD* entre este papel

e um papel B, o usuário também pode estar incluído na lista de membros do papel B, entretanto, durante uma sessão de usuário, ele não poderá assumir simultaneamente os dois papéis. O relacionamento de *DSD* é utilizado quando os papéis não criam conflito quando agem de forma independente, mas apenas simultaneamente. A Figura 3.10 ilustra esta situação. Também, de forma similar ao relacionamento de *SSD*, os relacionamentos de *DSD* são especificados com dois argumentos, o primeiro indicando o conjunto de papéis e o segundo indicando a cardinalidade do conjunto.

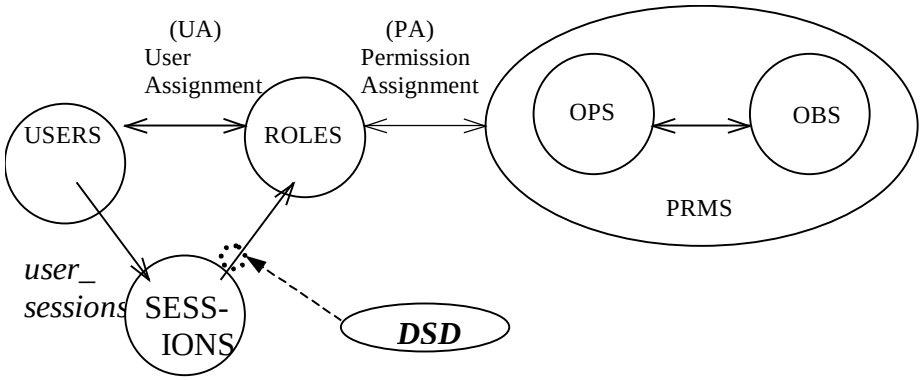

Figura 3.10– O relacionamento de *DSD*

3.4.2. A Especificação Funcional do Modelo RBAC

O modelo de referência apresentado na subseção anterior definiu uma série de entidades, componentes e relacionamentos. Para a criação e manutenção destes elementos do modelo RBAC, [SANDHU 00] propõe uma especificação de necessidades funcionais que fornecem a semântica de operação do modelo. Ele agrupa esta especificação em três categorias:

- Funções Administrativas – Funções de criação e manutenção dos elementos e relacionamentos para a construção dos vários modelos RBAC;
- Funções do Sistema – Funções de suporte necessárias durante a interação de um usuário com sistema (por exemplo, uma sessão de usuário);
- Funções de Revisão – Funções de verificação dos resultados das ações realizadas pelas funções administrativas;

A Tabela 3.4 resume estas especificações. As funções administrativas como *AddUser* e *AddRole* possibilitam, respectivamente, a inclusão de usuários e papéis no sistema.

ESPECIFICAÇÕES FUNCIONAIS DO RBAC				
	CORE RBAC	***HIERARCHICAL RBAC***	***RELACIONAMENTO DE SSD***	***RELACIONAMENTO DE DSD***
FUNÇÕES ADMINISTRATIVAS	AddUser DeleteUser AddRole DeleteRole AssignUser DeassignUser GrantPermission RevokePermission	AddInheritance DeleteInheritance AddAscendant AddDescendant + *CORE RBAC*	CreateSSDSet DeleteSSDSet AddSSDRoleMember DeleteSSDRoleMember SetSSDCardinality + *CORE RBAC*	CreateDSDSet DeleteDSDSet AddDSDRoleMember DeleteDSDRoleMember SetDSDCardinality + *CORE RBAC*
FUNÇÕES DO SISTEMA	CreateSession AddActiveRole DropActiveRole CheckAccess	CreateSession AddActiveRole DropActiveRole CheckAccess	CreateSession AddActiveRole DropActiveRole CheckAccess	CreateSession AddActiveRole DropActiveRole CheckAccess
FUNÇÕES DE REVISÃO	AssignedUsers AssignedRoles RolePermissions UserPermissions SessionRoles SessionPermissions	AssignedUsers AssignedRoles RolePermissions UserPermissions SessionRoles SessionPermissions AuthorizedUsers AuthorizedRoles	SSDRoleSets SSDRoleSetRoles SSDRoleSetCardinality + *CORE RBAC*	DSDRoleSets DSDRoleSetRoles DSDRoleSetCardinality + *CORE RBAC*

Tabela 3.4 – Especificações funcionais do RBAC

AssignUser é a função responsável pela associação *UA* mostrada do modelo de referência, ou seja, é através desta função que usuários são incluídos na lista de membros de papéis. De maneira similar, *GrantPermission* realiza o relacionamento *PA* do mesmo modelo. *AddInheritance* é responsável pela criação de um relacionamento hierárquico entre dois papéis existentes. *AddDescendant* cria um novo papel e o insere como descendente de um papel já existente. *CreateSession* cria uma sessão de usuário, ativando o usuário com um conjunto *default* de papéis ou permitindo a opção de escolha entre os papéis possíveis de serem ativados. *AddActiveRole* possibilita que um papel se torne ativo em uma dada sessão já estabelecida. *CheckAccess* verifica se um determinado *subject* de uma sessão tem permissão para executar a operação solicitada em um objeto. *CreateSSDSet* e *CreateDSDSet* criam, respectivamente, novos conjuntos de relacionamentos SSD e DSD. *AddSSDRoleMember* insere um papel como membro de um conjunto SSD existente, enquanto *SetSSDCardinality* especifica a cardinalidade da composição dos papéis do conjunto correspondente. *AssignedUsers* é uma função de revisão que retorna os usuários contidos na lista membros de um dado papel. *RolePermissions* retorna o conjunto de permissões autorizadas para um determinado papel. Como pode ser observado na Tabela 3.4, há uma repetição das funções ao

longo dos diversos modelos do RBAC. Na realidade, as funções operam de forma idêntica, entretanto, faz-se necessário o suporte à nova funcionalidade introduzida pelo modelo correspondente. Para maiores informações sobre a especificação e o modelo de referência apresentados, consulte [SANDHU 00].

3.5. Outros Trabalhos Relacionados

O RBAC tem se mostrado como uma alternativa atrativa em relação aos modelos de controle de acesso tradicionais. Esta seção mostra alguns trabalhos que ilustram como o RBAC tem sido utilizado como método de controle de acesso e como suas funcionalidades estão sendo incorporadas a aplicações e sistemas.

3.5.1. Especificando e Gerenciando o Controle de Acesso Baseado em Papéis dentro de uma Intranet Corporativa

O uso das *ACLs* na administração de políticas de segurança corporativas exige uma intensa coordenação nos procedimentos administrativos, tornando este processo custoso e sujeito a erro. Por esta falta de confiabilidade, algumas organizações têm resistido em publicar certas informações em seus servidores WEB [FB 97].

[FB 97] propõe uma implementação do RBAC para o uso em servidores WEB, a qual foi chamada de RBAC/Web. Nesta proposta, a informação de segurança é organizada em uma base de dados de autorização. A partir desta base de dados, o RBAC/Web mantém a informação de autorização em um formato relacional, atuando também como ferramenta de visualização e manutenção da informação RBAC. A base de dados do RBAC/Web inclui informações sobre papéis, hierarquia de papéis, associações entre usuários e papéis, associações entre papéis e permissões, entre outras. Um privilégio representa um método de acesso a um ou mais objetos controlados pelo RBAC. Nesta implementação, os privilégios são métodos de acesso *HTTP* que um usuário pode executar sobre *URL*s controladas pelo RBAC.

Em um cenário típico de uso, o usuário deve, antes de acessar a *URL* controlada, estabelecer uma sessão RBAC, momento no qual é permitido que ele selecione os papéis que estarão ativos na sessão. Obviamente, estes papéis pertencem à lista de papéis autorizados para o usuário. Após esta fase, para cada acesso a uma determinada *URL*, o gerenciador de

sessão determina o método *HTTP* que o usuário poderá executar, autorizando ou não o acesso a esta *URL* controlada. Para maiores informações, consulte [FB 97].

3.5.2. O RBAC no Ambiente Operacional do Solaris

A implementação RBAC introduzida no ambiente do sistema operacional Solaris 8 é baseada no processo de *login* de usuários, no qual eles assumem identidades especiais que os permitem executar ferramentas e utilitários de administração restritos [SUN 01]. São introduzidos três elementos ao ambiente operacional do sistema:

- Papel *(Role*): Identidade especial assumida por usuários autorizados;
- Autorização (*Authorization*): Permissão que pode ser associada a um papel para executar uma classe de ações controladas por uma política de segurança;
- Perfil de Direitos(*Rights Profile*): Um pacote que pode ser associado a um papel ou usuário, podendo ser composto por autorizações, comandos com atributos de segurança e outros perfis de direitos encadeados.

A Figura 3.11 mostra os elementos do RBAC no Solaris e suas associações. A seta aponta a partir de um elemento para o qual ele poderá ser associado. Papéis são associados a usuários. Perfis de direitos são associados a papéis. Autorizações e comandos com atributos de segurança são componentes dos perfis de direitos. Apesar de serem procedimentos considerados inseguros, autorizações e perfis de direitos podem ser associados diretamente a usuários (indicado pelas setas tracejadas).

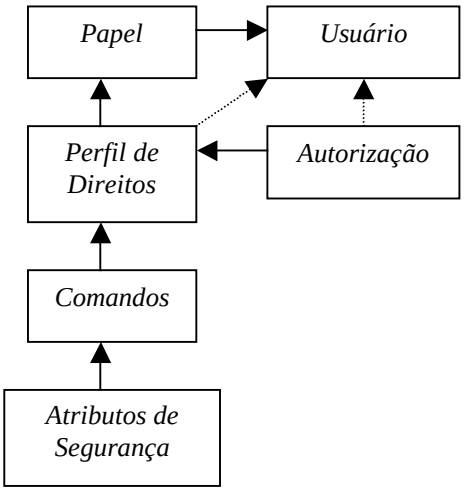

Figura 3.11 – Elementos e Relacionamentos RBAC no ambiente do Solaris

Um papel é criado de forma semelhante a um usuário do sistema. Todos usuários que

podem assumir um papel em comum operam no mesmo ambiente e têm acesso aos mesmos arquivos. Usuários não podem se logar diretamente com um papel; eles devem se logar inicialmente através da sua conta normal, e, através do comando *su* e do *password* do papel solicitado, o usuário poderá assumir o papel. Não há herança entre papéis. Quando um usuário assume um papel os atributos do papel substituem todos os atributos do usuário. A Figura 3.12 mostra um exemplo como ilustração da utilização do RBAC no Solaris. Neste exemplo, há o papel Operador, que é responsável pelo gerenciamento de impressoras e pela execução de *backups*. Este papel é atribuído ao usuário A. O perfil de direitos de Operador é associado ao papel Operador. O perfil de direitos é utilizado para o gerenciamento de impressoras, *daemons* de impressão e *spoolers*. Ele tem três autorizações: *solaris.admin.printer.read*, *solaris.admin.printer.modify* e *solaris.admin.printer.delete*, utilizados para a manipulação da fila de impressão. Além disso, este perfil tem dois comandos com atributos de segurança associados, o */usr/sbin/accept* com *uid=0* e */usr/ucb/lpq* com *euid=lp*. Para maiores informações, consulte [SUN 01].

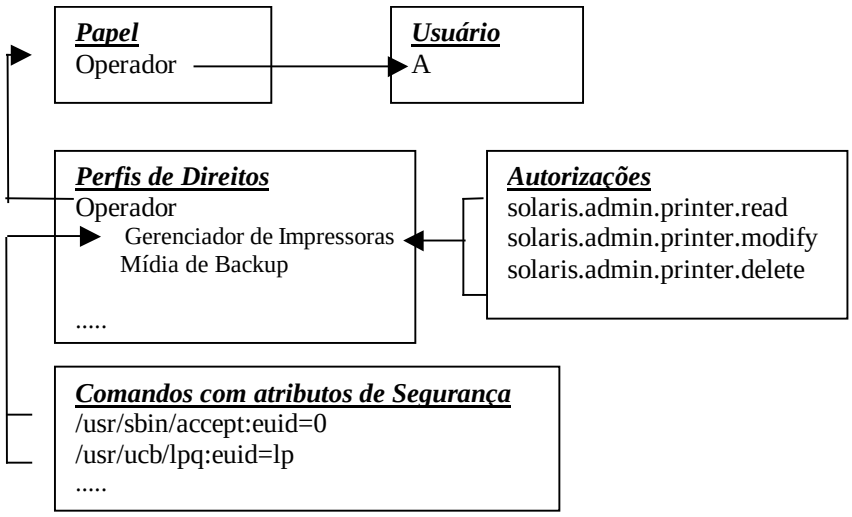

Figura 3.12 – Um exemplo do RBAC no ambiente do Solaris

3.5.3. O Controle de Acesso e o Gerenciamento de Sessão no Ambiente HTTP

[GUTZMANN 01] propõe uma estratégia que usa o RBAC e o gerenciamento de sessão WEB para a proteção das redes contra as brechas de segurança presentes no ambiente *HTTP*. O gerenciamento de sessão é implementado através de mecanismos de concessão de *tickets* baseados em *cookies*. A partir das definições RBAC presentes em [SC 96], [GUTZMANN 01] utiliza o $RBAC_0$, ou seja, estão presentes na sua proposta as entidades e os relacionamentos essenciais do RBAC: usuários, papéis, permissões, associação entre

permissões e papéis, associação entre usuários e papéis, e sessões. Assim, não há o suporte à hierarquia entre papéis.

O $RBAC_0$ é implementado em serviços de diretórios baseados no LDAP. Nesta estratégia, são efetuadas duas extensões ao esquema do diretório. A Figura 3.13 apresenta estas extensões. Na primeira, uma nova subclasse, *xPerson*, é adicionada à hierarquia de classes que representa um usuário no modelo. Esta classe (*objectclass)* especifica um novo atributo obrigatório, o *securityrole*. A segunda extensão é baseada na definição de uma nova classe, a *securityRoleObject*. Cada papel no sistema é representado por uma instância desta classe. Ela define a associação entre papéis e permissões do modelo RBAC. Quando um usuário é inserido no sistema, ele é adicionado como uma instância da classe *xPerson*. O atributo *securityrole* aponta para um papel (como indicado pela seta tracejada), ou seja, aponta para uma instância de *securityRoleObject*, realizando, assim, a associação entre usuários e papéis do $RBAC_0$. Cada instância de *securityRoleObject* possui atributos e valores associados que indicam se o papel pode ou não (atributos TRUE ou FALSE) executar as funções de acesso aos objetos controlados.

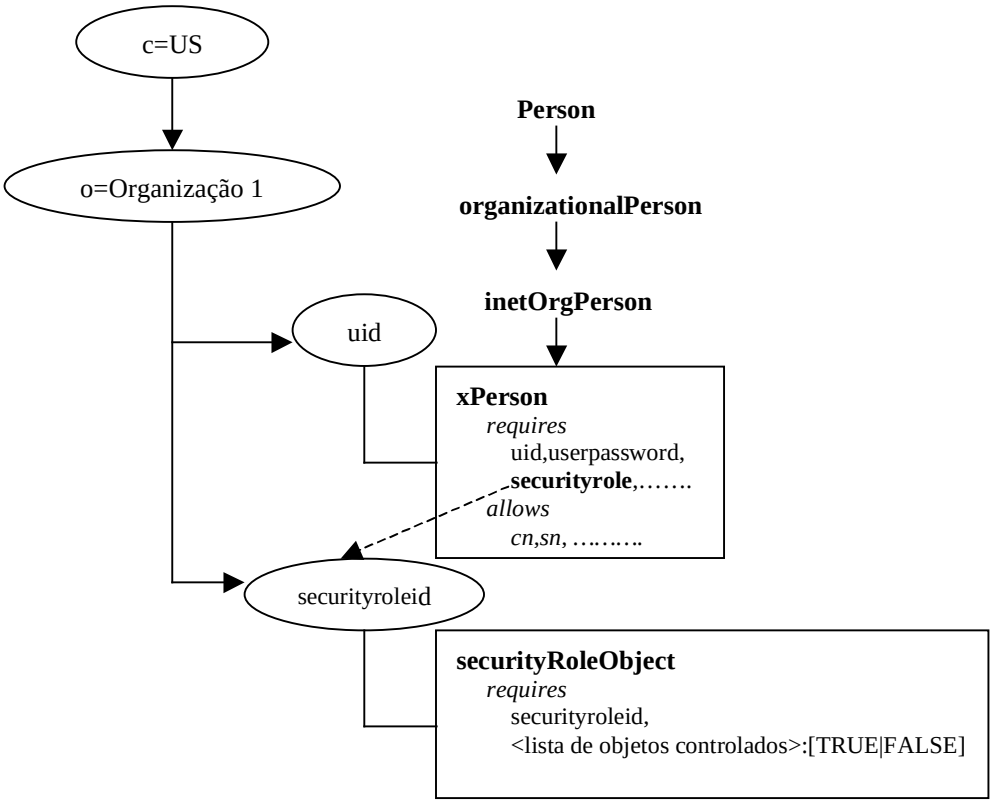

Figura 3.13 – Extensões do esquema do LDAP para suportar o RBAC

Esta subseção deu ênfase apenas à implementação do RBAC utilizada na estratégia descrita. Para outras informações, consulte [GUTZMANN 01].

3.5.4. O Controle de Acesso Baseado em Papéis para o Modelo CORBA de Segurança

O trabalho apresentado em [FOW 01] propõe a integração do controle de acesso baseado em papéis em sistemas distribuídos que seguem os padrões de segurança CORBA [OMG 99], através da adoção de uma estratégia que busca a ativação automática de papéis pelos componentes de segurança do *middleware*, sem que haja a necessidade da interação de usuários e aplicações com o subsistema de segurança para a seleção dos papéis que estarão ativos em uma dada sessão.

A especificação CORBAsec define uma série de objetos de serviço utilizados na implementação dos controles de segurança, tais como autenticação e autorização. As políticas de segurança são expressas em termos de atributos de controle e de privilégios [FOW 01]. No contexto do RBAC, políticas de autorização são representadas pelo objeto *AccessPolicy* da especificação CORBAsec, o qual contém permissões concedidas aos principais para a execução de operações no sistema. Ainda, o objeto *RequiredRights* armazena os atributos de controle utilizados para a execução de operações em objetos servidores. A partir das informações mantidas nestes dois objetos, o objeto *AccessDecision* fica responsável por verificar se uma determinada operação é ou não permitida.

Nesta proposta, chamada de RBAC-JaCoWeb, o controle de acesso é realizado de forma transparente pelo *middleware*, onde os objetos de serviço localizados do lado do cliente verificam se o acesso solicitado está em conformidade com a política de segurança, e se ele deve ou não ser autorizado [FOW 01]. Utilizando as especificações RBAC contidas em [SFK 00], [FOW 01] mostra a incorporação do controle de acesso baseado em papéis dentro do esquema de autorização proposto pelo projeto JaCoWeb, projeto que visa o estudo de esquemas de autorização em sistemas distribuídos utilizando o suporte do modelo CORBA de segurança. O RBAC é aplicado através da extensão do serviço de políticas para objetos distribuídos, o PoliCap, definido anteriormente em [WFW 00], para que ele gerencie o RBAC no contexto do CORBAsec [FOW 01].

Para ilustrar o mapeamento do RBAC na arquitetura proposta, [FOW 01] mostra um exemplo do controle de acesso baseado em papéis realizado no cenário de uma agência bancária. O conjunto de papéis definidos é: R = {cli, cxpf, cxpf, ger}, representando, respectivamente, clientes, caixas para pessoa física, caixas para pessoa jurídica e gerente. O serviço PoliCap mantém a informação RBAC, conforme mostra a Figura 3.14. Os direitos de acessos predefinidos pela especificação CORBAsec são os direitos g(*get*), s(*set*), m(*manage*)

e u(*use*). Os direitos requeridos são especificados por *interface* e não por instâncias de objetos individuais. O combinador indica se um principal precisa possuir todos os direitos requeridos(*All*) para executar uma operação ou apenas um deles (*Any*). Não há relacionamentos de SSD, mas apenas de DSD. Por exemplo, em uma dada sessão, um usuário não pode assumir simultaneamente o papel de gerente e caixa de pessoa jurídica.

Tomando a informação RBAC da Figura 3.14, [FOW 01] mostra como o principal *bia* assume novos papéis ativos em cenários estabelecidos antes e depois da decisão de acesso da operação correspondente, detalhando como o PoliCap e os objetos CORBA interagem para a realização do controle de acesso. A Figura 3.15 mostra o resultado das decisões de acesso efetuadas.

AccessPolicy

Atributo de Privilégio	Direitos Concedidos
role: cli	corba: g
role: cxpf	corba: g,s
role: cxpj	corba: g,u
role: ger	corba: g,m

Associação UA

Principal	Papéis
ana	cli,cxpj,ger
bia	cli,cxpj,cxpf
cris	cli,cxpf

Relacionamento DSD

cli	cxpf	cxpj	ger
cxpf	cli	cli	cli
cxpj	ger	ger	cxpf
ger			cxpj

RequiredRights

Direitos Requeridos	Combinador	Operação	Interface
corba: g	all	ver_saldo	ContaPFis
corba: g	all	ver_saldo	ContaPJur
corba: s	all	depositar	ContaPFis
corba: u	all	depositar	ContaPJur
corba: s,m	any	abrir	ContaPFis
corba: g,m	all	abrir	ContaPJur

Figura 3.14 – Exemplo de Política RBAC apresentada em [FOW 01]

Operação	Papéis Ativos Antes	Papéis Ativos Depois	Acesso
ContaPFis::abrir	nenhum	{cxpf}	**Permitido**
ContaPFis::depositar	{cxpf}	{cxpf}	**Permitido**
ContaPJur::depositar	{cxpf}	{cxpj,cxpj}	**Permitido**
ContaPJur::abrir	{cxpf,cxpj}	{cxpj,cxpj}	**Negado**

Figura 3.15 – Exemplo de um cenário do *framework* apresentado em [FOW 01]

O acesso à operação ContaPFis::abrir foi permitido, pois esta operação requer que apenas um dos direitos *s* e *m*, e o papel *cxpf*, entre os seus direitos associados, confere o *s*. Neste instante, o papel *cxpf* é ativado para *bia*. Como o principal já está com o papel cxpf ativo, o acesso à operação ContaPFis::depositar também foi permitido, pois ele já concedeu na

invocação da operação anterior o direito *s*, requerido para esta operação. De maneira análoga, o papel *cxpj* é ativado para o principal *bia*, uma vez que ele é requerido para a operação ContaPJur::depositar, a qual passa a ser permitida. Já o acesso à operação ContaPJur::abrir foi negado, pois ela requer os direitos g e m, não conferidos a nenhum papel associado ao principal *bia*.

Para ver detalhes das fases de autenticação e do *binding*, da interação entre os objetos do CORBAsec e outras informações sobre o funcionamento deste *framework*, consulte [FOW 01].

3.6. Conclusões do Capítulo

A arquitetura distribuída nos sistemas de informação corporativos está cada vez mais sendo utilizada. Neste contexto, o controle da segurança nestas aplicações passa a ser uma tarefa difícil, exigindo a busca por novos métodos que reduzam a complexidade no gerenciamento da informação de segurança.

Neste sentido, este capítulo apresentou o RBAC, um modelo de controle de acesso que se enquadra como alternativa aos modelos discricionário e obrigatório, cujas diferenças foram discutidas na seção 3.2.

Várias formas e definições de RBAC têm sido propostas. As seções 3.3 e 3.4 apresentaram dois importantes trabalhos que definem as características e funcionalidades do modelo. O primeiro apresentou uma definição formal do modelo contendo suas entidades básicas e associações, baseando estas definições em primitivas de acesso e regras que especificam as propriedades do RBAC. O segundo, mais recente, tomou como base todo o conhecimento e evolução na área de modelos RBAC e apresentou um modelo de referência e uma especificação que visam nortear desenvolvedores e consumidores de tecnologia quanto ao desenvolvimento e à avaliação de aplicações RBAC.

A seção 3.5 apresentou quatro trabalhos que propõem a utilização do RBAC. A subseção 3.5.1 mostrou a utilização do RBAC no controle de acesso a *URLs* via os métodos de acesso do protocolo *HTTP*. A subseção 3.5.2 mostrou como o sistema Solaris 8 incorporou os conceitos de papéis no controle da autorização das tarefas administrativas do sistema. A subseção 3.5.3 apresentou uma implementação do RBAC em serviços de diretórios LDAP, o qual realizou algumas extensões ao modelo de objetos do LDAP para suportar a semântica do controle de acesso baseado em papéis. Por fim, a subseção 3.5.4 apresentou uma proposta de

implementação do RBAC utilizando as especificações e os objetos de segurança da arquitetura CORBAsec.

Capítulo 4

RBPIM: Uma Proposta de Extensão do PCIM para o Mapeamento do RBAC

4.1. Introdução

O *framework* introduzido pela RFC 3060 foi proposto com o intuito de estabelecer um padrão para representar e gerenciar informação de políticas. As classes e associações produzidas pelo PCIM são genéricas, fornecendo a base necessária à representação de políticas de qualquer natureza. Assim, novas subclasses derivadas das classes do PCIM devem ser propostas para atender às necessidades específicas de aplicações, como as políticas RBAC. Por exemplo, novas subclasses derivadas de *PolicyCondition* e *PolicyAction* podem suprir as necessidades específicas de definição de condições e ações de políticas. O Capítulo 3 mostrou que o esquema de autorização através do conceito de papéis, introduzido pelo RBAC, representa um grande avanço no gerenciamento de políticas de segurança, uma vez que oferece o suporte a funcionalidades não presentes nos modelos tradicionais e ainda possibilita a descrição de políticas de forma natural à estrutura hierárquica das organizações.

Neste contexto, o presente capítulo apresenta o RBPIM – *Role Based Policy Information Model* – uma proposta derivada do PCIM capaz de suportar a descrição e o gerenciamento de políticas RBAC. Este modelo será implementado em serviços de diretórios baseados no LDAP, conforme será mostrado no Capítulo 5. Para tanto, será utilizado um esquema de diretório derivado do PCIM para a adaptação ao modelo de objetos utilizado pelos serviços LDAP. As definições RBAC utilizadas neste trabalho foram baseadas no modelo de referência e nas especificações funcionais propostos por [SANDHU 00], os quais foram apresentados no Capítulo 3. Este capítulo está organizado da seguinte forma. A seção 4.2 apresenta dois trabalhos que serviram de motivação para a concepção deste modelo, prosseguindo com a apresentação do modelo proposto na seção 4.3. A seção 4.4 conclui o capítulo.

4.2. Trabalhos Relacionados

4.2.1 hyperDrive: Um esquema do LDAP para o controle de acesso baseado em papéis

[BARTZ 97] propõe um conjunto de novas classes e atributos LDAP para suportar o *framework* do RBAC, objetivando capacitar a interoperação entre diferentes implementações RBAC que usam serviços de diretórios. Esta estratégia de mapeamento do RBAC no LDAP foi chamada de "hyperDRIVE".

O RBAC no modelo de objetos proposto é centrado em duas responsabilidades principais no contexto da segurança: a autorização e o acesso. Na terminologia do RBAC, um acesso corresponde a uma operação ou privilégio com autorização restrita. Na Figura 4.1, o acesso é representado pela classe (*objectclass) operationAccessor*. A autorização significa o mapeamento ou agregação de uma ou mais operações ou privilégios a um determinado papel. A autorização neste modelo é representada pela classe *Role*.

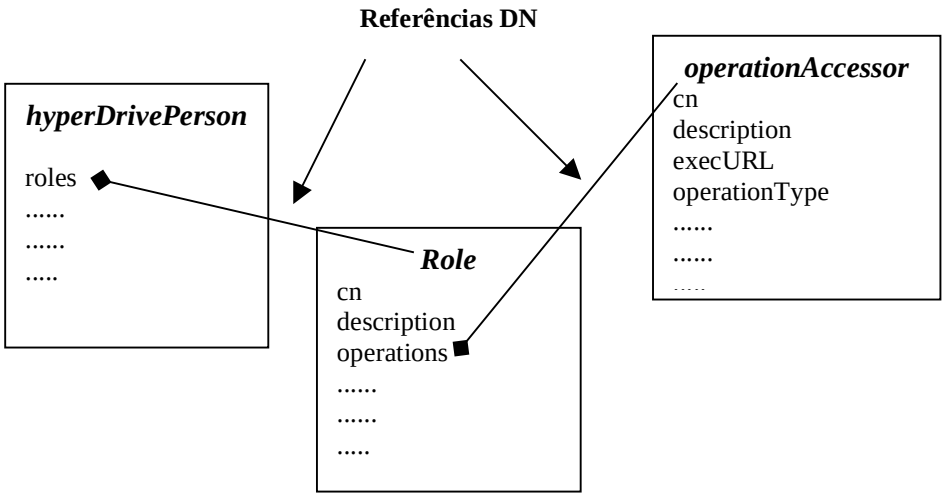

Figura 4.1 – Relacionamentos entre os objetos do modelo hyperDrive

A Figura 4.1 mostra o relacionamento entre as classes de objetos do modelo. A *objectclass Role* possui um atributo chamado *operations*, o qual representa uma agregação de referências a instâncias de *operationAccessor*. Da mesma forma, a *objectclass hyperDrivePerson*, uma extensão da *objectclass person* do LDAP, possui um atributo chamado *roles*, o qual representa uma agregação de referências para instâncias de *Role*.

Um usuário é inserido no diretório como uma instância de *hyperDrivePerson*. Papéis são associados a usuários através da referência *roles*. O atributo *execURL* da *objectclass*

operationAccessor informa a *URL* através da qual o a operação será executada. As referências mantidas pelos atributos *roles* e *operations* são especificadas pelos *DNs* (*distinguished names*) dos objetos referenciados. Para outras informações sobre o funcionamento deste *framework*, consulte [BARTZ 97].

4.2.2. O Modelo de Informação CADS-2

O CADS-2 [BARTZ 01] é um modelo de informação orientado a objetos implementado em um serviço de diretório. Este modelo de informação usa o RBAC como modelo para o gerenciamento da informação de segurança e serviços de diretórios LDAP como repositórios para o armazenamento da informação RBAC.

4.2.2.1 A Descrição do Modelo

A implementação do RBAC é baseada nas especificações de política do PCIM. De maneira geral, a função principal do CADS-2 é permitir/negar o acesso de um determinado usuário a um dado serviço. Neste contexto, a semântica do RBAC é estabelecida através da associação de papéis a serviços e da associação de usuários a papéis. A Figura 4.2 mostra como a informação é mantida no diretório, através da apresentação das entradas da DIT (*Directory Information Tree*).

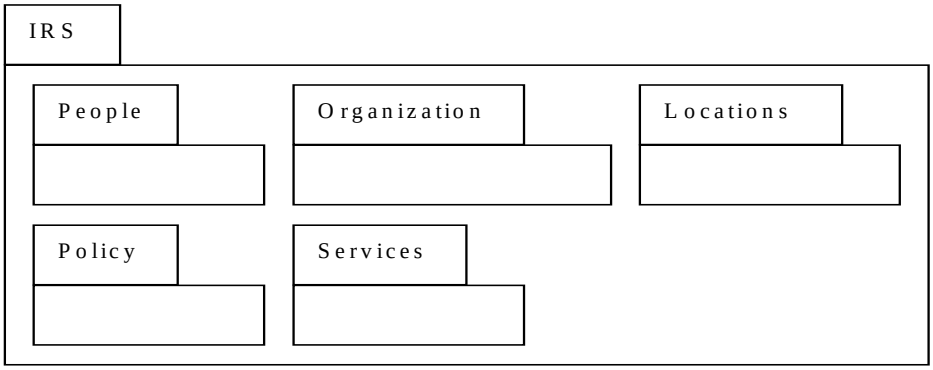

Figura 4.2 – A DIT do CADS-2

Na Figura 4.2 o *folder* (em UML, conhecido como *container*) principal é o IRS[5]. Dentro de IRS existem 5 sub-folders. *People* mantém as informações de pessoas no diretório. Neste *folder* há vários objetos do tipo pessoa, instâncias da *objectclass* Person.

[5] A sigla IRS significa Internal Revenue Service. É um órgão ligado ao Departamento do Tesouro dos Estados Unidos.

Um objeto pessoa no diretório possui diversos atributos, como nome e telefone. Também há atributos que atuam como referências para outros objetos no diretório, tais como referências para unidades de trabalho e localidades associados ao objeto pessoa. O *folder Locations* mantém informações em três níveis. No primeiro nível, são mantidas informações sobre estados. No segundo nível, dentro de estados, são mantidas informações sobre cidades e, por último, as informações de endereços são mantidas dentro de cidades. Cada um destes *folders* ou *containers* é um objeto do diretório. Uma das características de um diretório é que os objetos podem conter outros objetos, não importando o nível desta associação. Qualquer tipo de objeto pode ser contido dentro de um objeto. Este processo é conhecido como DIT *containment*. As informações sobre as unidades de negócio são mantidas no *folder Organization*. O *folder Services* mantém informações sobre serviços. Serviços podem ser entendidos como serviços de computação que serão executados na rede, como por exemplo o acesso de recursos via Web, repositórios Ftp, uma aplicação CGI, entre outros. Assim, os objetos armazenados neste *folder* representam os serviços os quais eles descrevem. Um determinado serviço pode ter mais de uma maneira de ser acessado. Isto implica na necessidade de representar estas formas acesso. Desta forma, um objeto do tipo *Services* tem objetos subordinados chamados de *ServiceAccessPoint (SAP).* Vale lembrar que *Services e ServiceAccessPoint* utilizados neste *framework* seguem as definições propostas pelo CIM [DMTF 00].

Uma vez definida a informação sobre pessoas e serviços, faz-se necessário estabelecer uma sistemática que possibilite as conexões entre eles. Para suprir tal necessidade este *framework* adicionou um folder que mantém objetos que descrevem políticas, chamado *Policy*. Os objetos deste folder são baseados no *framework* proposto pelo PCIM, mostrado no Capítulo 2. A RFC 3060 define um objeto de política abstrato. Cabe ao desenvolvedor criar subclasses que fornecem novas definições para a implementação. Este *framework* faz extensões à classe *PolicyCondition,* criando tipos diferentes para cada operação de comparação, como, por exemplo, *equal,greaterThan* e *lessThan,* conforme mostra a Figura 4.3. Estes operadores podem ser utilizados para construir comparações complexas como, por exemplo, operações com valores dos atributos dos objetos mantidos no repositório LDAP. De maneira análoga, novas subclasses foram adicionadas também à classe *PolicyAction* do PCIM.

A construção da *PolicyAction* utilizada no CADS-2 é baseada na "atribuição de um valor a uma variável". Assim, através das subclasses de *AssignorPolicyAction* é possível atribuir a um papel o SAP para possibilitar o acesso a um serviço. A Fig. 4.4. mostra esta extensão.

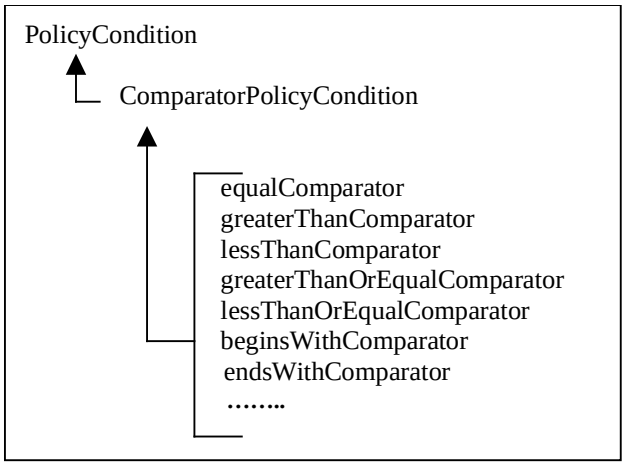

Figura 4.3 – CADS-2: Extensões da classe *PolicyCondition*

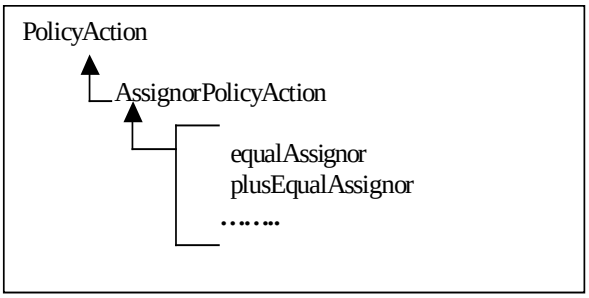

Figura 4.4 – CADS-2: Extensões da classe *PolicyAction*

4.2.2.2. Exemplo do Mapeamento RBAC no CADS-2

Com o intuito de demonstrar como o CADS-2 fornece o suporte ao controle de acesso baseado em papéis, a seguir será ilustrado um simples processo de autorização de acesso a uma operação dentro do contexto de uma agência bancária. Nesta situação, suponha que o *framework* precise autorizar o acesso de um determinado funcionário *A* ao serviço que permite realizar o processo de abertura de contas, que tem como ponto de acesso a URL indicada por *https://BankOnline.com.br/Admin/openAccount*. As informações sobre funcionários, serviços, pontos de acesso e políticas estão na forma de objetos armazenados nos *folders* do diretório, conforme mostrado anteriormente na Figura 4.2.

Segundo as especificações de uma política de segurança hipotética, apenas os funcionários que têm os dois caracteres iniciais do atributo *Setor* com valor 52 e o atributo *Cargo* com valor igual a 5878 podem ter acesso a este serviço. Ou seja, funcionários com esses requisitos satisfeitos poderão assumir o papel *operador_I* e, portanto, terão direito de acesso ao serviço de abertura de contas, que é uma operação autorizada a este papel. A Figura 4.5 mostra como esta regra de negócio baseada no RBAC é mapeada no modelo proposto pelo CADS-2.

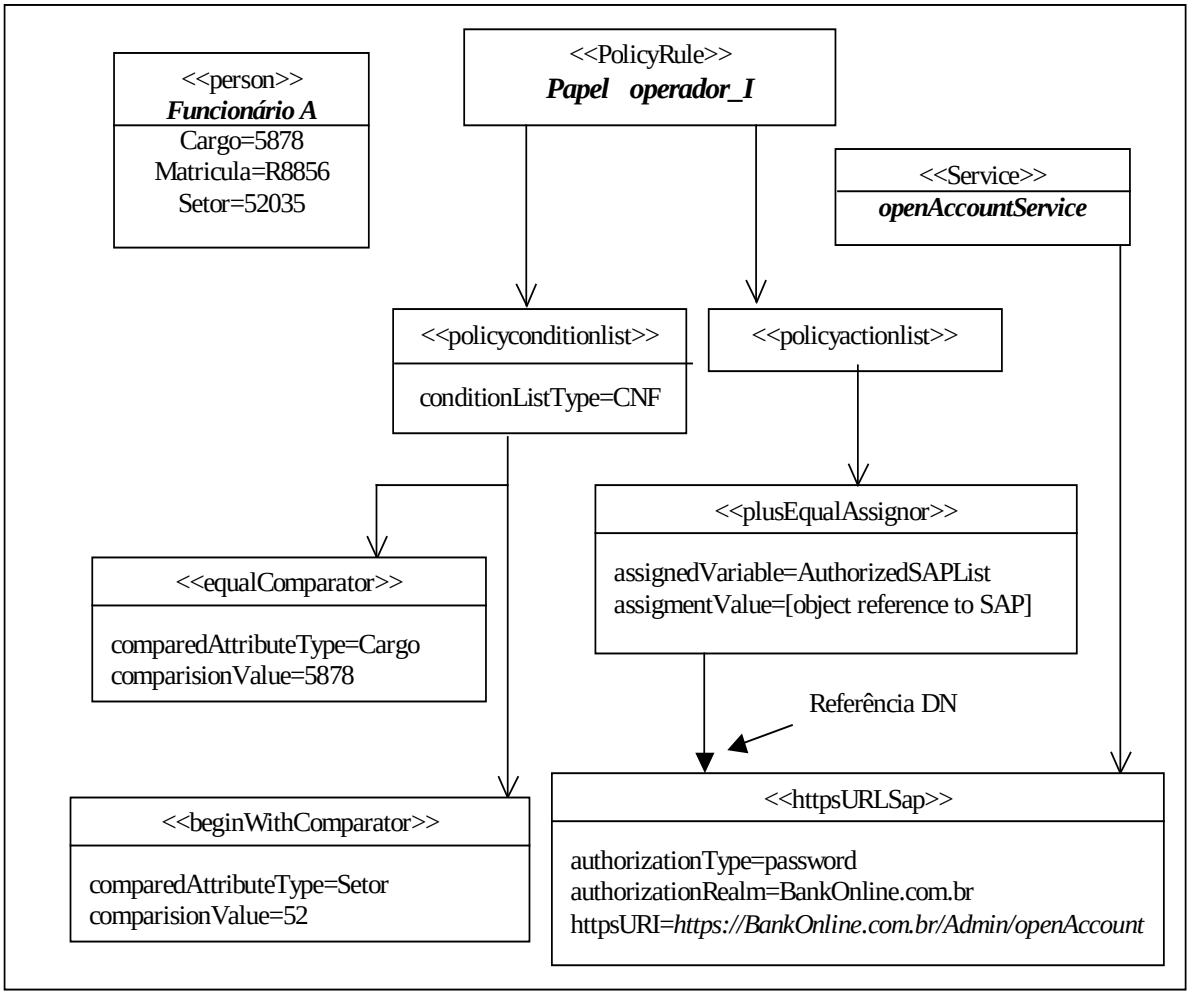

Figura 4.5 – CADS-2: Um exemplo de mapeamento do RBAC

Com exceção do relacionamento entre os objetos das classes *plusEqualAssignor* e *httpsURIsap*, que indica uma referência entre objetos, os demais relacionamentos indicam *DIT containment*, apontando a partir do objeto *container* em direção ao objeto contido. Neste modelo, um papel RBAC é representado por uma instância da classe *PolicyRule* do *PCIM*. Associadas à *PolicyRule*, observa-se a lista de condições que permitem a seleção dos

principals que poderão assumir o papel em questão. Como a lista de condições é do tipo CNF e considerando que cada condição está em um grupo diferente, é construída uma expressão lógica com o operador AND, indicando que todas as condições associadas a ela devem ser verdadeiras. A partir da análise dos atributos do funcionário *A*, esta expressão resulta em verdadeiro. Com isso, o papel *operador_I* é autorizado a este principal. Assim, a associação entre usuários e papéis (UA) do RBAC é mapeada pelo CADS-2. Para completar, é necessário indicar quais são as permissões de acesso concedidas a um dado papel (associação PA). No CADS-2, este procedimento é alcançado pela autorização de serviços a papéis. Como dito na seção anterior, um serviço pode ter várias maneiras de ser acessado, ou seja, vários SAPs. Por questões de simplificação, neste exemplo foi considerado apenas um, o *httpsURIsap*. Esta subclasse indica que o serviço será acessado via protocolo HTTP sobre SSL. A forma de autenticação escolhida foi via *password*, mas poderia ser, por exemplo, via certificados X.509. O domínio do serviço é BankOnline.com.br, sendo acessado pela URL *https://BankOnline.com.br/Admin/openAccount.*

4.3. A Proposta RBPIM

A presente seção detalha o modelo de informação proposto. Este modelo propõe a utilização do *framework* do PCIM para o mapeamento da informação RBAC. Para tal, utiliza os conceitos e especificações apresentados nos capítulos anteriores, propondo uma extensão ao PCIM. Ainda, este modelo utiliza parte das estratégias de mapeamento adotadas pelo CADS-2, inovando com uma alternativa que possibilite a descrição de políticas mais genéricas, não apenas relacionadas ao processo de autorização a serviços baseados em rede, além de possibilitar a descrição de políticas RBAC em completa conformidade com as funcionalidades apresentadas em [SANDHU 00].

4.3.1. Considerações e Diretrizes

Como mostrado no Capítulo 2, o DMTF tem especificado uma série de padrões (modelos) para a representação da informação necessária para o gerenciamento de sistemas computacionais, tais como a informação sobre configurações, sistemas, redes, serviços, aplicações e componentes que podem gerenciados. A especificação X.500 define uma série de padrões que especificam como a informação pode ser armazenada e acessada em um diretório. Além disso, ela disponibiliza uma série de classes básicas para a representação da informação sobre pessoas, grupos, organizações, entre outros. Ainda, as regras de nomes e o

modelo de informação do X.500 foram mantidos pelas implementações dos serviços de diretórios LDAP. A estratégia adotada para a concepção deste modelo visa o aproveitamento das padronizações disponibilizadas pelo CIM. Essa estratégia é importante, pois é muito provável que os serviços de diretório comerciais adotem a proposta do DMTF, passando a fornecer as classes do CIM de forma pré-definida. A compatibilidade com o CIM permitirá reduzir bastante o número de informações a serem adicionadas ao diretório, a fim de suportar esta proposta.

Neste contexto, o modelo proposto por este trabalho tem como objetivo a aplicação do controle de acesso baseado em papéis no gerenciamento de políticas de autorização em sistemas distribuídos, utilizando serviços de diretórios baseados no LDAP como repositório da informação de segurança. As políticas especificadas a partir deste modelo são construídas segundo o RBAC, utilizando as informações das entidades gerenciáveis descritas em conformidade com os padrões definidos pelo DMTF e pelos serviços de diretórios. Desta forma, organizações que usam serviços de diretórios baseados no LDAP como repositórios para a informação de gerenciamento e descrevem os objetos armazenados nestes repositórios através dos esquemas de diretórios do CIM serão beneficiadas. Elas poderão incorporar o controle de acesso baseado em papéis ao gerenciamento do processo de autorização, através da adição de objetos de políticas RBAC descritos segundo o modelo proposto, sem que haja a necessidade de efetuar alterações nos objetos que representam a estrutura computacional existente.

4.3.2 O Contexto do Mapeamento do RBAC

Para o mapeamento do RBAC a partir do *framework* de políticas do PCIM pensou-se, inicialmente, em utilizar o suporte da propriedade *PolicyRoles*, definida pela classe *PolicyRule*. A idéia da utilização desta propriedade é a otimização do processo de configuração de recursos em uma rede, uma vez que regras de políticas (*PolicyRules*) podem ser associadas aos recursos da rede através dela. Desta forma, isto possibilita que cada recurso passe a ter seu comportamento controlado pelo *framework* de políticas, como indicado pelas regras de política associadas a ele. Se por um lado este esquema simplifica o processo de mapeamento de políticas a recursos, por outro lado ele é insuficiente para o estabelecimento até mesmo do *Core* RBAC por completo. Obviamente, novas classes devem ser adicionadas ao modelo para que ele possa suportar as funcionalidades RBAC, na mesma linha das propostas apresentadas em [GUTZMANN 01], [BS 01] e [BARTZ 97].

O CADS-2, mostrado na subseção 4.2.2, se apresenta como uma estratégia viável de mapeamento do RBAC. Um papel, que é o conceito central do RBAC, é mapeado pela principal classe do *framework* de políticas do PCIM, a classe *PolicyRule*. Neste contexto, as associações *UA (User-Role assignment)* e *PA (Permission-Role assignment)* são mapeadas diretamente pelo *framework* de políticas, onde, respectivamente, as condições construídas a partir das subclasses de *PolicyCondition* fornecem os usuários autorizados a assumir um papel e as subclasses de *PolicyAction* especificam as permissões de acesso associadas ao papel.

A análise do *framework* proposto pelo CADS-2 mostra que, no seu atual estágio de desenvolvimento, ele apenas contempla as associações *UA* e *PA* do *Core* RBAC, não apresentando funcionalidades para o estabelecimento de hierarquias e separação de tarefas entre papéis. Além disso, as expressões das *PolicyConditions* são construídas baseadas em operadores representados por classes pré-definidas e a extensão realizada a partir da classe *PolicyAction* do PCIM restringe a associação PA do RBAC ao mapeamento de papéis a serviços, limitando a descrição de políticas de segurança. Neste contexto, o modelo proposto por este trabalho adotou as seguintes premissas:

- Extensão do PCIM, com a adição de novas classes para dar suporte ao *Hierarchical* RBAC e ao *Constrained* RBAC.
- Substituição da forma de construção das associações UA e PA do RBAC por um esquema mais aberto, baseado nas extensões do PCIM (PCIMe) propostas em [MOORE 02], onde novas classes estabelecem refinamentos às classes *PolicyCondition* e *PolicyAction*, dando o suporte necessário à criação de, respectivamente, expressões booleanas e operações de atribuição, tipicamente utilizadas na especificação de regras de políticas.
- Adição de novas classes que permitirão a especificação do período de tempo que um dado papel poderá estar ativo, utilizando, para isso, o suporte das próprias classes definidas pelo PCIM para tal finalidade.

4.3.3 A Descrição do Modelo Proposto

O diagrama apresentado na figura a seguir foi construído tendo-se como base o modelo de informação do PCIM [RFC 3060], o *CIM_Policy Schema ve*rsão 2.4 [DMTF 00]

e algumas atualizações do PCIM propostas no *Internet-Draft* PCIMe [MOORE 02]. Eles mostram apenas as classes e associações destas especificações relevantes ao escopo do modelo proposto, bem como as novas classes e associações introduzidas por este trabalho, indicadas na legenda por (3)-RBPIM. A maior parte das propriedades das classes deste modelo não está sendo mostrada[6]. A Figura 4.6 apresenta o diagrama do modelo proposto. A classe *ManagedSystemElement* é a classe base da hierarquia do *Core Model* do CIM. Diversos refinamentos para áreas específicas do gerenciamento foram propostos, dando origem ao *Commom Model* do CIM. Apesar desta possibilidade de adicionar novas classes ao *Core Model*, algumas extensões não podem ser feitas, uma vez que representam a informação sobre objetos que estão fora do domínio do sistema gerenciado [DMTF 00], como, por exemplo, informações sobre organizações e usuários. Assim, o CIM especifica uma abstração da classe *ManagedSystemElement*, a classe *ManagedElement*, a qual, em conjunto com classe *Policy*, atua como classe base da hierarquia do PCIM.

4.3.3.1 O Mapeamento do *Core* e do *Hierarchical* RBAC no RBPIM

Duas classes derivadas da classe *PolicyRule* do PCIM representam as principais entidades do RBPIM: *RBACRole* e *RBACPermission*, respectivamente, um papel e uma permissão RBAC. Através da hierarquia mostrada na Figura 4.6, estas duas classes herdam as propriedades e a semântica *IF condition THEN action* da classe *PolicyRule.* Além disso, para dar suporte ao *Hierarchical* RBAC, este trabalho propõe a introdução da propriedade *InheritedRoles*. Através dela, poderá ser especificado um conjunto de papéis "juniores" que são herdados por um dado papel, uma vez que esta propriedade admite a especificação de múltiplos valores (*Multi-valued Property*).

[6] A descrição completa das classes e suas respectivas propriedades será apresenta no Capítulo 5.

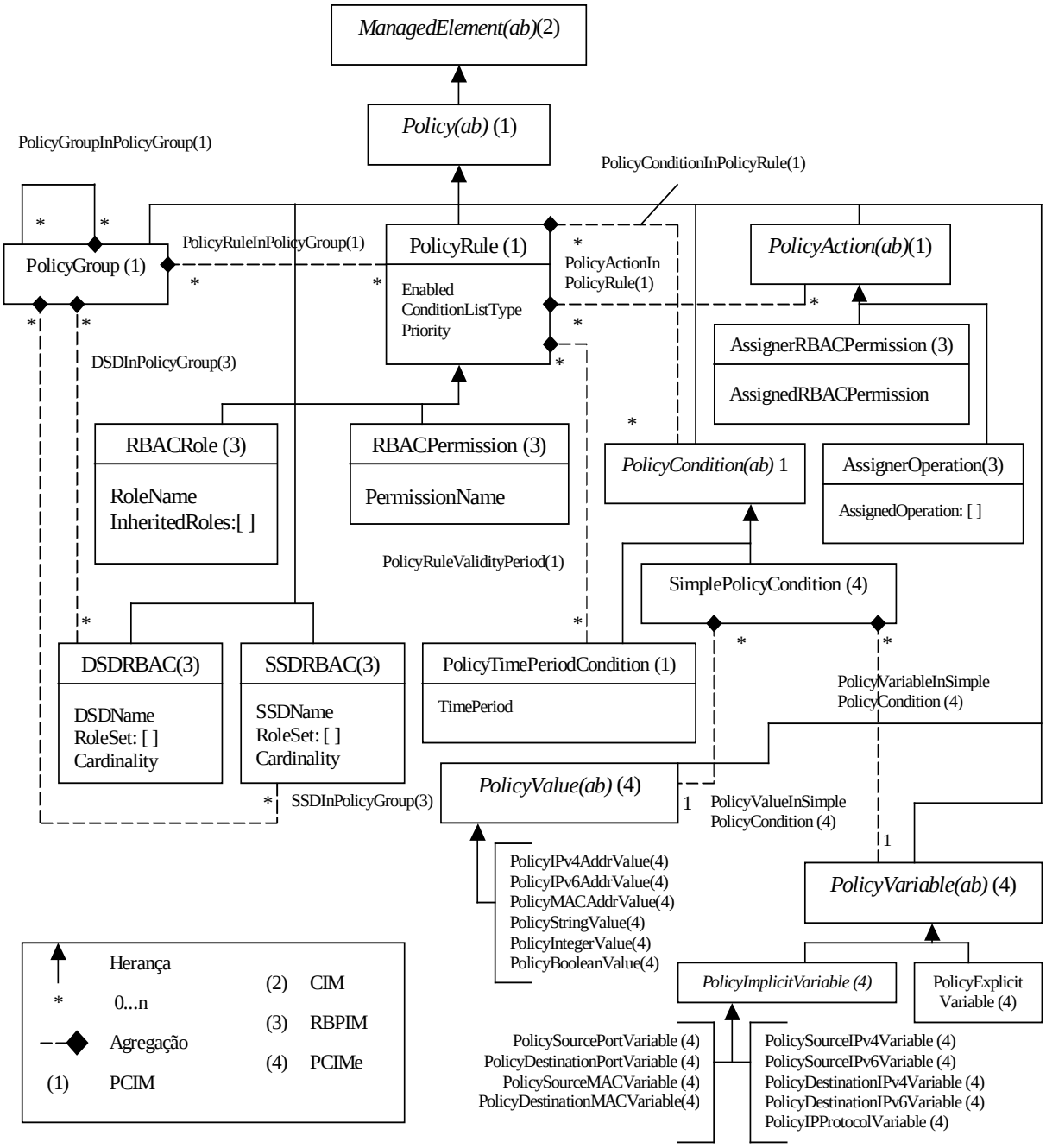

Figura 4.6 – Hierarquia de Classes do RBPIM

A Figura 4.7 mostra em detalhe as principais classes e associações indicadas pelas classes do RBPIM. Objetos da classe *RBACRole* podem ter associados objetos das classes *SimplePolicyCondition* (2), *AssignerRBACPermission* (5) e *PolicyTimePeriodCondition* (6). Objetos *RBACRole* podem ter associados vários objetos de (2). O atributo *ConditionListType* herdado de *PolicyRule* define como as expressões indicadas pelas classes (2), (3) e (4) serão construídas, ou seja, se a composição será DNF ou CNF. O RBPIM mapeia a associação *UA* do RBAC através das expressões definidas por (2), (3) e (4). Como será exemplificado logo

a seguir, estas expressões são construídas baseadas nas propriedades dos objetos que representam a informação sobre pessoas, definidas pelo *CIM_User Schema* [DMTF 00]. A figura 4.8 mostra a hierarquia da classe *Person* deste diagrama. Usuários que atenderem os requisitos indicados pelas expressões definidas por (2), (3) e (4) de um dado papel *RBACRole* pertencerão à lista de usuários autorizados ao papel.

Na Figura 4.7, a associação *PA* do RBAC é mapeada através da classe indicada por (5). Um papel RBPIM pode ter vários objetos de (5) associados. Cada objeto de (5) indica um objeto de permissão associado ao papel. A propriedade *AssignedRBACPermission* da classe *AssignerRBACPermission* mantém uma referência a um objeto de permissão associado ao papel. A semântica herdada de *PolicyRule* por *RBACRole* pode ser resumida na forma *IF (2)(3)(4) THEN (5)*, ou seja, (2)(3)(4) fornecem a lista de usuários enquanto (5) as permissões autorizadas ao papel.

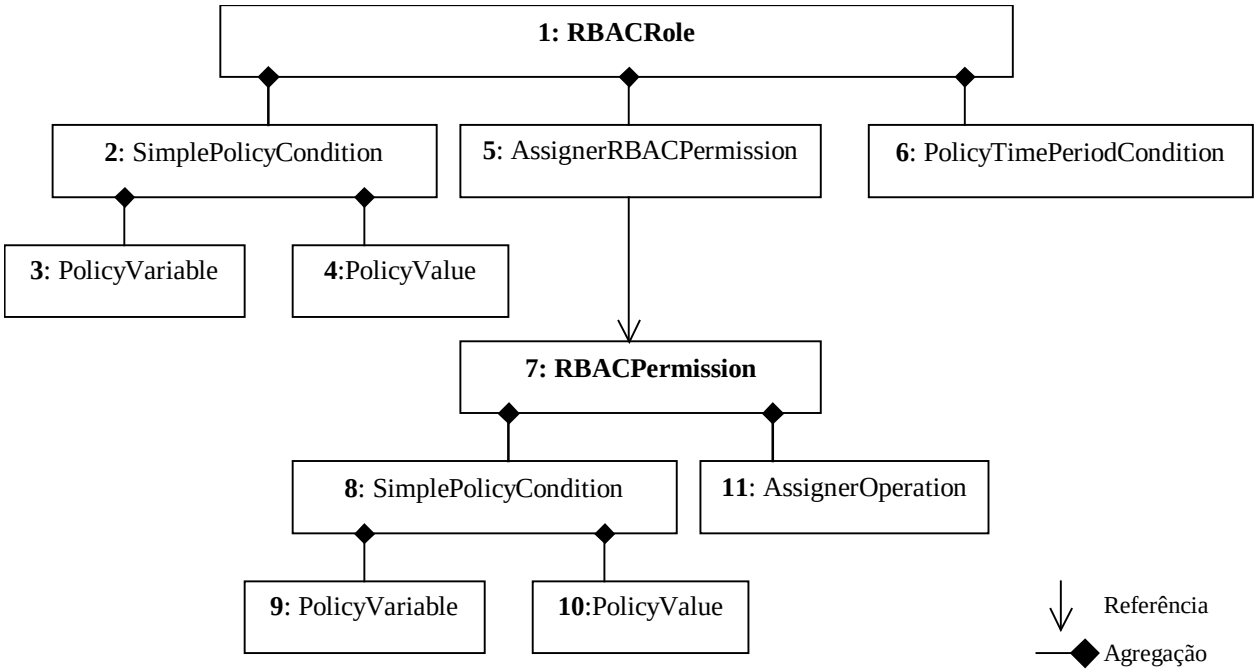

Figura 4.7: RBPIM: Detalhe das Principais Classes e Associações

A classe *PolicyTimePeriodCondition* (6) permite que seja representado um período no qual um dado papel poderá se encontrar ativo. Vários períodos poderão ser associados a um dado papel. Assim, para que um dado papel esteja ativo, é preciso que pelo menos um desses objetos associados ao papel indique um período válido para ativação. Por exemplo, a

propriedade *TimePeriod* com valor 20000101T080000/20000131T120000 [RFC 3060] representa uma faixa para ativação que vai de 01/01/2000 às 08h00m até 31/01/2000 12h00m, onde a letra T indica a porção que representa a hora e o caracter / indica a data e hora final do período.

Uma permissão RBAC pode ser vista como o mapeamento de um conjunto de operações a um conjunto de recursos. Referindo-se ainda à Figura 4.7, a classe *RBACPermission* (7) definida neste trabalho também possui semântica *IF condition THEN action.* No caso desta classe, tem-se *IF (8)(9)(10) THEN (11),* onde as expressões definidas por (8), (9) e (10) definem os recursos associados com a permissão enquanto (11) as operações autorizadas ao conjunto de recursos. A lista de operações autorizadas é mantida pela propriedade *AssignedOperation* da classe *AssignerOperation*.

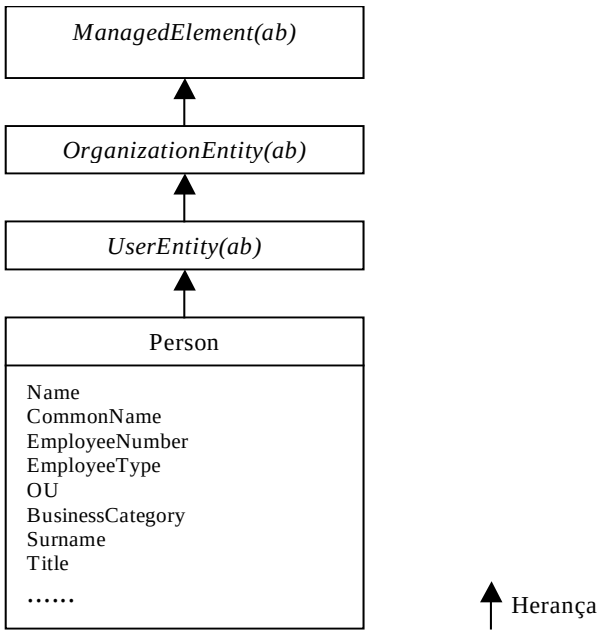

Figura 4.8 – Hierarquia parcial de classes do CIM_User Schema

Nas Figuras 4.6 e 4.7, a classe *SimplePolicyCondition* refina a estrutura básica da classe *PolicyCondition*, introduzindo o uso do par <variável>,<valor> para formar uma condição [MOORE 02], criando a semântica *"<variable> MATCH <value>"* para as expressões booleanas, onde o operador *MATCH* fica implícito às expressões, não tendo uma representação formal nas classes. O par variável/valor associado a uma *SimplePolicyCondion* é criado, respectivamente, por instâncias das subclasses de *PolicyVariable* e *PolicyValue.* [MOORE 02] define duas especializações de *PolicyVariable*: *PolicyExplicitVariable e*

PolicyImplicitVariable. A diferença entre estes dois tipos de variáveis está relacionada à representação da informação que elas mapeiam. Uma variável explícita está ligada a uma informação cuja representação está bem definida no contexto da modelagem do CIM, como, por exemplo, diretórios, arquivos, serviços, aplicações, entre outros. Uma variável implícita, por sua vez, não possui tal definição. Por exemplo, uma *PolicyCondition* não pode fazer uma referência a uma construção do modelo que representa um endereço IP de origem de um pacote que trafega na rede [MOORE 02]. O PCIMe define uma série de especializações de *PolicyImplicitVariable*, como, por exemplo, *PolicySourceIPv4Variable* e *PolicyDestinationIPv4Variable*.

Uma instância da classe *PolicyExplicitVariable*, que possui as propriedades *ModelClass* e *ModelProperty*, indica os nomes da classe e da propriedade do modelo que deverá ser considerado em uma determinada condição. Eles são definidos no contexto do *CIM Schema*. Instâncias das subclasses de *PolicyValue* representam os valores a serem comparados com estas propriedades referidas. Por exemplo, uma instância de *PolicyStringValue* pode representar um string ou conjunto de strings utilizados em uma dada condição.

4.3.3.2 O Mapeamento do *Constrained* RBAC no RBPIM

Os relacionamentos de separação estática e dinâmica de tarefas são modelados, respectivamente, pelas classes *SSDRBAC* e *DSDRBAC*. Instâncias destas classes representam conjuntos de papéis e a cardinalidade associada a cada um destes conjuntos. Seguindo as definições de [SANDHU 00], suponha a existência dos papéis Gerente, Caixa e Auditor no ambiente de uma agência bancária. Devido às regras de negócios, um usuário nunca poderá ter mais do que um destes papéis em sua lista de papéis autorizados. Assim, existe um relacionamento de separação estática de tarefas entre estes papéis, que pode ser estabelecido por uma instância da classe *SSDRBAC* contendo o seguinte: *RoleSet* = {Gerente, Caixa, Auditor} e *Cardinality* = 2, ou seja, para qualquer combinação de 2 destes papéis ocorre a violação por SSD.

4.3.3.3 Agregações no RBPIM

Como definido no Capítulo 2, uma agregação normalmente representa um relacionamento de coleção. A Figura 4.6 apresenta uma série de agregações. Observe o caso da classe *PolicyGroup*. Esta classe permite que papéis, permissões e relacionamentos do *constrained* RBAC sejam agrupados em função das necessidades dos contextos de suas utilizações. Neste caso, uma empresa poderia agrupar as informações de controle de acesso

de seus recursos de forma segmentada por filiais e departamentos. As agregações são modeladas por classes que contêm propriedades que atuam como referências para as classes envolvidas. Por exemplo, a classe *PolicyRuleInPolicyGroup* modela a agregação de agrupamento de papéis e permissões e a classe *PolicyConditionInPolicyRule* modela a agregação de objetos de condições em objetos *PolicyRule*, ou seja, modela a associação entre papéis e permissões aos seus objetos de condição subordinados.

4.3.4 Exemplo do mapeamento de políticas RBAC no RBPIM

Considere o seguinte cenário: Uma determinada empresa mantém todas as funções de negócio de seus funcionários representadas por papéis. Cada funcionário é classificado por categoria. Suponha que a informação referente a esta classificação seja mantida pela propriedade *BusinessCategory* da classe *Person*. Suponha, ainda, que todos os funcionários, cujos dois primeiros caracteres de suas categorias sejam CT (Contabilidade) possam assumir o papel Contador_I. Além disso, eles só poderão assumir este papel quando estiverem realizando um acesso a partir da rede interna 192.168.10.0/24. A figura 4.9 mostra como este cenário pode ser mapeado pelo modelo proposto por este trabalho. As classes *PolicyRule* e *PolicyCondition* não estão sendo mostradas. Nesta figura, a classe *RBACRole* representa o papel Contador_I. Esta classe agrega duas *SimplePolicyConditions*, que, com suas próprias agregações, indicam que a seguinte condição (*default* DNF)

IF BusinessCategory MATCH "CT*" and PolicySourceIPv4Variable MATCH "192.168.10.0/24"

será associada a esta regra. No contexto deste trabalho, indica um requisito para que um determinado usuário assuma tal papel.

Suponha, agora, a necessidade de controlar o acesso a um determinado diretório, *"/etc/application"*, representado, no escopo do CIM, pela classe *Directory*. A figura 4.10 apresenta de forma parcial o *CIM_System 2.4 Schema* [DMTF 00].

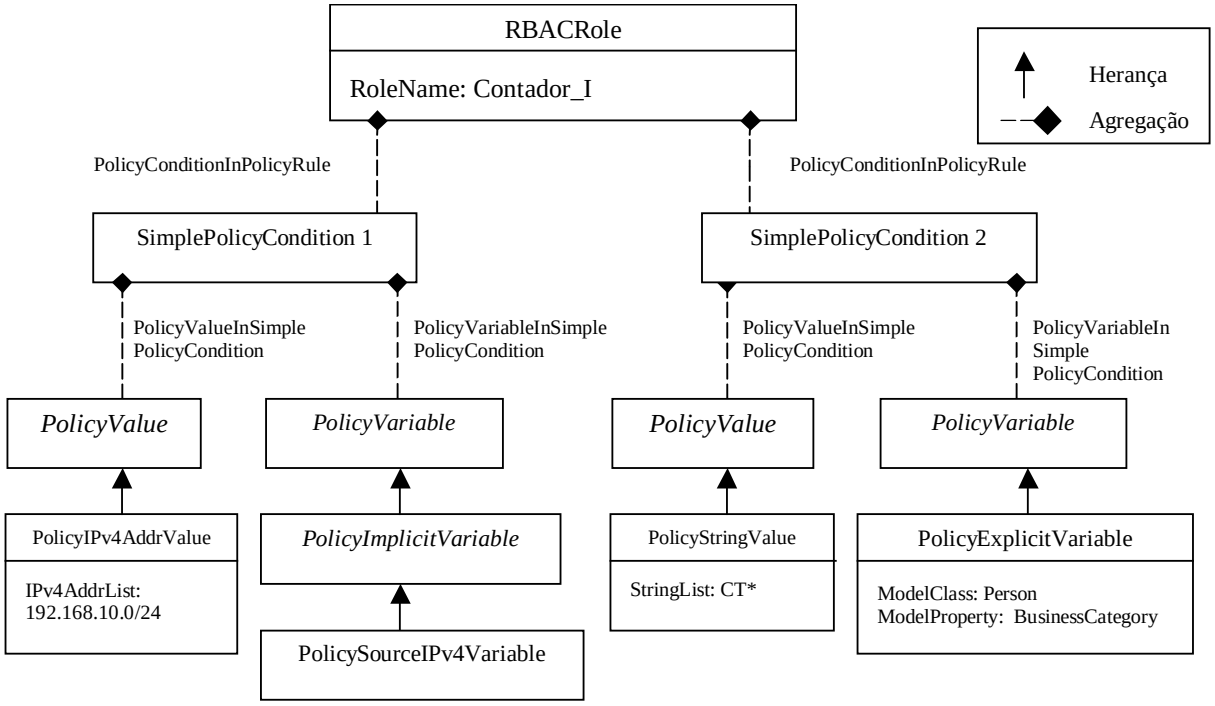

Figura 4.9 – Exemplo do mapeamento da associação UA do RBAC

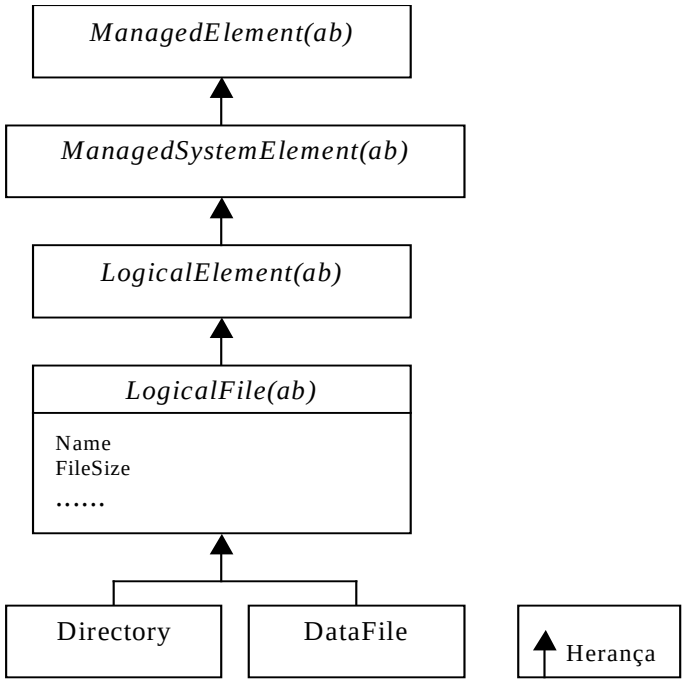

Figura 4.10 – Hierarquia parcial de classes do CIM_System Schema

Um objeto *RBACPermission* será criado para representar as permissões de acesso autorizadas para tal diretório. Através da agregação *PolicyConditionInPolicyRule*, um objeto *SimplePolicyCondition* pode ser associado a este objeto de permissão. Ainda, através da agregação *PolicyActionInPolicyRule*, um objeto *AssignerOperation* também pode ser

associado a este objeto de permissão, indicando quais são as operações permitidas. Este cenário é mostrado na figura 4.11. Novamente, as classes *PolicyRule*, *PolicyCondition* e *PolicyAction* não estão sendo mostradas nesta figura.

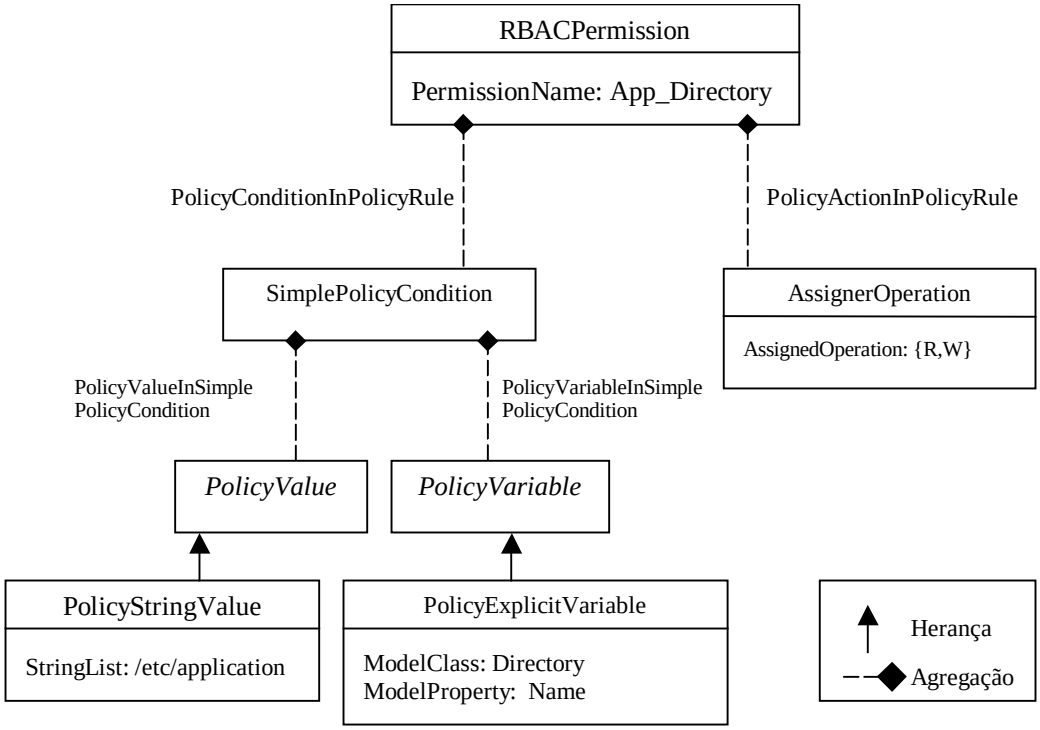

Figura 4.11 – Exemplo de um objeto de permissão do RBPIM

O objeto de permissão mostrado na figura 4.11 representa a seguinte semântica:

IF Name **MATCH** "/etc/application" **THEN** SET AssignedOperation = {R,W}

Obviamente, esta condição tem sentido se executada dentro do contexto de objetos da classe *Directory*, conforme valor da propriedade *ModelClass*. Assim, a permissão *APP_Directory* indica que operações de leitura e escrita poderão ser efetuadas no diretório */etc/application*. Vale ressaltar que este mesmo objeto de permissão pode ter associado a ele outras instâncias de *SimplePolicyCondition*, permitindo a composição de condições mais completas, como, por exemplo:

IF Name **MATCH** "/etc/application" **AND** PolicySourceIPv4Variable **MATCH** "192.168.1.1" **AND** PolicySourcePortVariable **MATCH** {"1024 TO 65535"}
THEN
SET AssignedOperation = {R,W}

Neste caso, as operações de leitura e escrita serão autorizadas para o diretório */etc/application* caso o endereço IP do *host* de origem seja 192.168.1.1 e o número da porta correspondente ao processo de origem seja maior que 1023.

Para finalizar, a figura 4.12 mostra como o papel *Contador_I*, ilustrado previamente na figura 4.9, pode ter associado a permissão *APP_Directory*, definida na figura 4.11. Ainda, a figura 4.12 mostra uma instância de *PolicyTimePeriodCondition*, indicando um período válido para ativação do papel. As classes relativas aos objetos *PolicyVariables* e *PolicyValues* de cada uma das *SimplePolicyConditions* foram suprimidas.

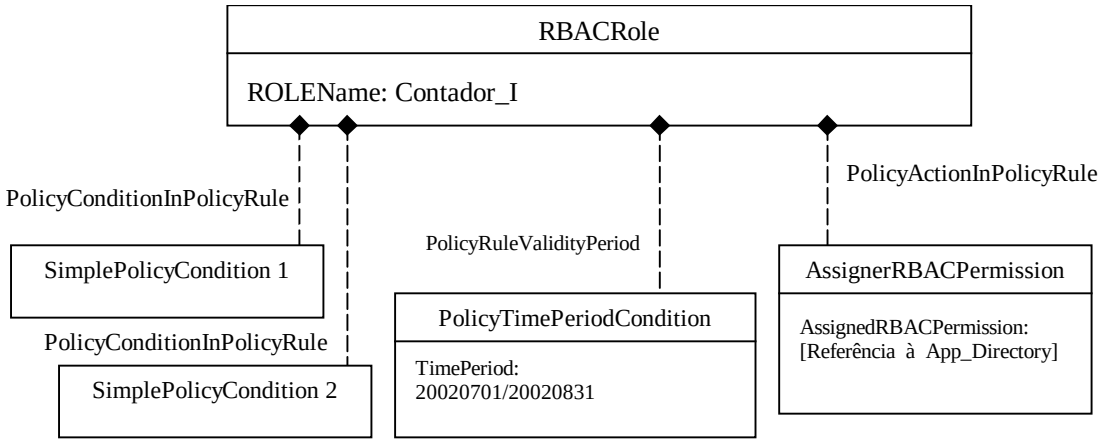

Figura 4.12 – Exemplo das associações UA e PA do RBAC no RBPIM

4.4. Conclusões do Capítulo

O RBPIM, apresentado neste capítulo, tem como objetivo central servir de base para o mapeamento e aplicação do controle de acesso baseado papéis no ambiente dos sistemas distribuídos. Neste sentido, este modelo utilizou as definições RBAC contidas no modelo de referência proposto em [SANDHU 00].

Os trabalhos propostos pelo DMTF têm buscado estabelecer uma série de padrões para a representação da informação necessária para o gerenciamento de sistemas computacionais. O RBPIM adotou a estratégia de aproveitar estas padronizações, possibilitando que o controle de acesso baseado em papéis seja aplicado às entidades do ambiente computacional, cujas representações seguem os esquemas do CIM. Essa estratégia é importante, pois é muito provável que os serviços de diretório comerciais adotem a proposta do DMTF, passando a fornecer as classes do CIM de forma pré-definida. A compatibilidade

com o CIM permitirá reduzir bastante o número de informações a serem adicionadas ao diretório, a fim de suportar esta proposta.

O RBPIM foi construído a partir do modelo de informação do PCIM, realizando extensões para dar suporte à descrição de políticas RBAC. Além disso, foram utilizadas algumas estratégias de mapeamento RBAC do CADS-2 e algumas extensões propostas pelo PCIMe. O capítulo 5 apresentará o mapeamento do RBPIM no LDAP. O esquema resultante deste mapeamento será utilizado como base para o armazenamento da informação de política utilizada pela arquitetura proposta no capítulo 6.

Capítulo 5

O Mapeamento do RBPIM nos Serviços de Diretório LDAP

5.1. Introdução

Segundo a proposta do IETF, o modelo de informação do PCIM é neutro, não especificando nada quanto à forma de sua implementação, ficando a cargo dos desenvolvedores a escolha da tecnologia mais adequada. O PCIM informa que trabalhos subseqüentes com os mapeamentos específicos deste modelo de informação a uma dada implementação serão propostos. Um trabalho nesta linha é o PCLS [MOORE 01]- *Policy Core Ldap Schema* -, um *Internet Draft* que define o mapeamento das classes do PCIM ao modelo de objetos utilizado pelos serviços de diretórios baseados no protocolo LDAP.

Este capítulo mostra como o PCLS pode ser utilizado para realizar o mapeamento do RBPIM no LDAP. No entanto, as especificações do PCLS apenas definem os mapeamentos das classes estruturais e associativas contidas no PCIM. As novas classes introduzidas pelo RBPIM e PCIMe não possuem mapeamento no LDAP definido. Assim, este capítulo também propõe o mapeamento das classes do PCIMe (apenas aquelas que foram incorporadas ao RBPIM) e das novas classes propostas neste trabalho.

Este capítulo está organizado da seguinte forma. A seção 5.2 apresenta o mapeamento do PCIM do LDAP bem como as estratégias e conceitos utilizados para alcançar tal mapeamento, prosseguindo com a apresentação dos mapeamentos referidos na seção 5.3. A seção 5.4 conclui o capítulo.

5.2. O Mapeamento do PCIM no LDAP

5.2.1 Tipos de classes

Como definido na [RFC 2252], uma classe (*objectclass*) em um serviço de diretório baseado no protocolo LDAP pode ser abstrata (*abstract*), estrutural (*structural*) ou auxiliar (*auxiliary*). Um objeto em um serviço de diretório não pode ser instanciado a partir de uma classe abstrata. Uma classe estrutural é o único tipo de classe que pode ter instâncias em um diretório. Enquanto uma classe estrutural pode ser uma subclasse de uma classe abstrata ou de uma outra classe estrutural, uma classe abstrata pode ser uma subclasse apenas de outra classe abstrata. Uma classe auxiliar define um conjunto de propriedades (atributos) que podem ser adicionados a objetos instanciados no diretório. Desta forma, é possível adicionar atributos a um objeto do diretório sem que a classe utilizada para instanciá-los contenha estes atributos em sua definição. Como no caso das classes abstratas, um objeto em um serviço de diretório não pode ser instanciado a partir de uma classe auxiliar. Classes auxiliares podem ser subclasses de classes abstratas e de outras classes auxiliares.

5.2.2 Descrição do mapeamento LDAP do PCIM

O mapeamento LDAP das classes do PCIM, PCIMe e do RBPIM não é feito na base de um para um, ou seja, as classes provenientes destes modelos não possuem necessariamente uma classe correspondente no LDAP *Schema*. [DMTF 02] apresenta as seguintes estratégias de mapeamento LDAP:

- 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 vinculadas[7] a objetos pré-existentes no diretório;

Como no PCIM, as associações do RBPIM também são modeladas através de classes. [MOORE 01] define que estas classes de relacionamentos podem ser modeladas de duas formas:

1) Atributos atuando como referências DN;

2) Relacionamentos superior-subordinado inerentes à DIT do diretório.

A Figura 5.1 mostra um exemplo típico de um *attachment* de uma classe auxiliar a um objeto do diretório. Neste exemplo, existem dois objetos instanciados no diretório, um a partir da classe estrutural *A* e outro da classe estrutural *B*. O objetivo do *attachment* da classe auxiliar *X* é permitir que haja a associação entre a instância 1 de *A* e a instância 1 de *B*. Neste caso, o atributo que pertence à definição da classe auxiliar *X* passa a integrar a lista de atributos da instância 1 de A. Assim, é possível que o atributo *REF* armazene uma referência DN a instância 1 de *B*, concretizando a associação entre os objetos. Note que, o *attachment* de uma classe auxiliar possibilita uma flexibilização do esquema de diretório, pois atributos podem ser inseridos a objetos sem a necessidade da alteração das definições das classes base do objeto (Este procedimento não muda a definição da classe estrutural A. Isto significa que novos objetos instanciados a partir desta classe estrutural não terão este atributo, a não ser que, novamente, um *attachment* seja realizado).

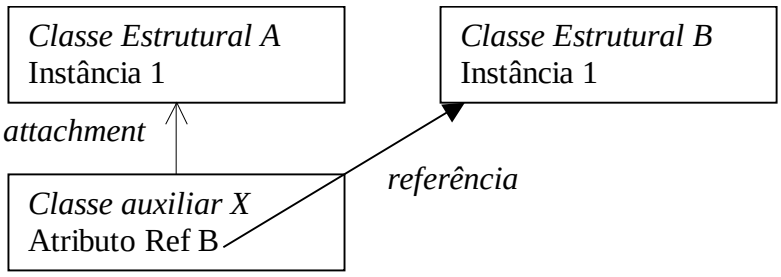

Figura 5.1 – Exemplo de um *attachment* de um classe auxiliar

[7] No inglês o termo utilizado é *attach* (atachar). Os termos vincular e *attachment* serão usados neste trabalho para indicar que os atributos definidos por uma classe auxiliar serão adicionados a uma instância (objeto) do diretório. Quando se diz que uma classe auxiliar será vinvulada a uma classe estrutural significa dizer que objetos desta classe estrutural poderão utilizar os atributos definidos por esta classe auxiliar.

A estrutura hierárquica da *DIT* (*Directory Information Tree)* de um diretório possibilita um relacionamento superior-subordinado de forma inerente. Veja o exemplo da Figura 5.2. Neste caso não existem atributos atuando como referências a objetos. A associação é estabelecida naturalmente, pois a instância 1 de B é uma entrada no diretório inserido a partir da entrada da instância 1 de A. Neste caso, este associação é caracterizada por um *DIT containment*, ou seja, a instância 1 de A contém a instância 1 de B em função da organização da *DIT* do diretório.

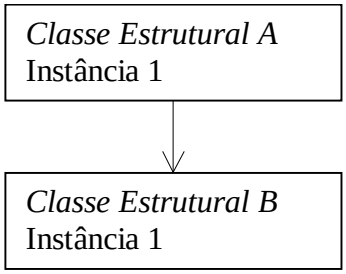

Figura 5.2 – Exemplo de associação por *DIT containment*

[MOORE 01] propõe o mapeamento das classes abstratas e estruturais, e das classes decorrentes dos relacionamentos conforme resumem, respectivamente, as tabelas 5.1 e 5.2. A classe *Policy* é mapeada pela classe abstrata *pcimPolicy* no modelo de objetos do LDAP. Ela atua como classe base da hierarquia de classes que definem objetos relacionados a políticas. A classe *PolicyGroup* é mapeada por três classes. Esta estratégia de mapeamento dá flexibilidade à utilização do modelo, uma vez que possibilita duas alternativas para a especificação de *containers PolicyGroup*: 1) Através do *attachment* da classe *pcimGroupAuxClass* a objetos já existentes no diretório; 2) Novos objetos inseridos a partir da classe estrutural *PolicyGroupInstance*. Estas duas classes são especializações da classe abstrata *pcimGroup*. De maneira similar, a classe *PolicyRule* do PCIM também é mapeada por três classes: Uma classe abstrata chamada *pcimRule*, a qual define os atributos dos objetos *PolicyRule*, e duas especializações criadas com o mesmo propósito daquelas da classe *pcimGroup*: *pcimRuleInstance* e *pcimRuleAuxClass*.

PCIM	Classe LDAP
Policy	pcimPolicy (ab)
PolicyGroup	pcimGroup (ab) pcimGroupAuxClass (aux) pcimGroupInstance (struc)
PolicyRule	pcimRule (ab) pcimRuleAuxClass (aux) pcimRuleInstance (struc)
PolicyCondition	pcimConditionAuxClass (aux)
PolicyAction	pcimActionAuxClass (aux)
PolicyTimePeriodCondition	pcimTPCAuxClass (aux)

Tabela 5.1 – O Mapeamento PCIM-LDAP: Principais classes estruturais e abstratas

PCIM	Atributo / Classe LDAP
PolicyGroupInPolicyGroup	pcimGroupsAuxContainedSet em pcimGroupContainmentAuxClass
PolicyRuleInPolicyGroup	pcimRulesAuxContainedSet em pcimRuleContainmentAuxClass
PolicyConditionInPolicyRule	DIT *containment* ou pcimRuleConditionList em pcimRule
PolicyActionInPolicyRule	DIT *containment* ou pcimRuleActionList em pcimRule
PolicyRuleValidityPeriod	pcimRuleValidityPeriodList em pcimRule

Tabela 5.2 – O Mapeamento PCIM-LDAP: Principais classes associativas

As classes *PolicyCondition, PolicyAction* e *PolicyTimePeriodCondition* são mapeadas por três classes auxiliares do LDAP, respectivamente, *pcimConditionAuxClass, pcimActionAuxClass* e *pcimTPCAuxClass*. Na realidade, como especificado claramente no PCIM, as condições e ações associadas a uma regra devem ser criadas a partir de novas especializações de *PolicyCondition* e *PolicyAction*. Assim, no contexto do mapeamento LDAP proposto pelo PCLS, novas especializações de *pcimConditionAuxClass* e *pcimActionAuxClass* devem ser especificadas para atender às necessidades de aplicações. Com intuito de manter a flexibilidade do modelo, [MOORE 01] recomenda que estas novas subclasses também sejam do tipo auxiliar, estratégia que foi adotada por este trabalho, conforme será mostrado na próxima seção.

As associações *PolicyGroupInPolicyGroup* e *PolicyRuleInPolicyGroup* são mapeadas através de classes auxiliares. Por exemplo, suponha a necessidade de uma empresa agrupar as regras de políticas de acordo com suas filiais. Neste caso, o administrador do sistema poderia

criar no diretório duas instâncias de *pcimGroupInstance*, uma para cada filial. Em seguida, vinculando a estes dois objetos a classe auxiliar *pcimRuleContainmentAuxClass*, ele usaria o atributo *multi-valued pcimRulesAuxContainedSet* desta classe para indicar os *DNs* dos objetos *pcimRuleInstance* ou outros objetos que representam políticas (que tenham vinculados a si a classe *pcimRuleAuxClass*) envolvidos no agrupamento. De maneira análoga, grupos podem agregar outros grupos, através do *attachment* da classe *pcimGroupContainmentAuxClass.*

Conforme mostrado no Capitulo 2, o PCIM define condições e ações 1) específicas a uma regra e 2) reusáveis, as quais podem ser aplicadas a mais de uma regra. Neste sentido, o PCLS propõe classes que possibilitam o mapeamento LDAP destas associações para os dois casos. No entanto, como a construção de condições e ações proposta pelo RBPIM apenas contempla o caso 1, o mapeamento do caso 2 não será discutido por este trabalho. [MOORE 01] sugere duas alternativas para o mapeamento das associações entre regras e condições e entre regras e ações para o caso 1: Regras de Políticas Simples e Regras de Políticas Complexas, conforme mostram, respectivamente, as figuras 5.3 e 5.4. Na figura 5.3, as condições e ações associadas a uma regra são construídas a partir do *attachment* das classes auxiliares que as representam diretamente ao objeto de regra de política. Neste exemplo, o objeto que representa a regra de política *Rule1* tem associado a si duas condições *(condition 1 e condition 2*) e uma ação (*action 1*). A figura 5.4 apresenta um exemplo do mapeamento LDAP de uma regra de política complexa. Neste caso, as classes auxiliares que representam condições e ações não são vinculadas diretamente ao objeto de regra, mas sim, respectivamente, às classes estruturais *pcimRuleConditionAssociation* e *pcimRuleActionAssociation*, as quais representam as próprias associações e mantém os atributos decorrentes delas. Estes atributos permitem o agrupamento e a negação de expressões lógicas que farão parte das condições associadas às regras e, ainda, permitem o estabelecimento de uma ordem na execução das ações relacionadas com uma regra.

As regras de políticas simples permitem um esquema mais eficiente, uma vez que as informações sobre as condições e ações associadas à regra estão vinculadas diretamente a ela, não havendo a necessidade de novas consultas no diretório para sua obtenção. Por outro lado, regras de políticas complexas dão mais flexibilidade à descrição de políticas, pois permitem a construção de expressões *booleanas* complexas nos esquemas DNF/CNF[8] e a ordenação na

[8] Regras de políticas simples apenas realizam um AND com as várias condições vinculadas à regra.

seqüência de execução das ações associadas a uma regra. Além disso, elas fornecem as bases para o reuso de condições e ações.

No exemplo da figura 5.4, as classes auxiliares *condition1* e *condition2* estão vinculadas às classes estruturais *CA1* e *CA2*. Por sua vez, *CA1* e *CA2* estão contidas (via *DIT containment*) dentro da entrada do diretório da sua própria regra associada, *Rule1*. Além disso, o objeto *Rule1* possui referências aos objetos CA1 e CA2 (DN_CA1 e DN_CA2). [MOORE 01] ressalta que a utilização das referências DN pode, aparentemente, ser redundante, uma vez que os objetos CA1 e CA2 já possuem um relacionamento de *DIT containment* com *Rule1*. No entanto, ele sugere que esta dupla associação dá mais consistência e que cabe ao desenvolvedor fazer a opção pela utilização de ambas ou de usar apenas *DIT containment*. Para a implementação do RBPIM, este trabalho optou pelo cenário descrito no exemplo da figura 5.4. A expressão lógica construída a partir de CA1 e CA2 está no formato DNF, conforme discutido no Capítulo 2 . De maneira análoga, a classe estrutural AC1 é associada a *Rule1*, indicando que, caso a expressão resultante de *condition1* e *condition2* seja verdadeira, a ação *action1* será executada. Os atributos da classe *pcimTPCAuxClass* indicam os intervalos nos quais as regras poderão ser ativadas. Neste caso, a associação entre regras e períodos de tempo pode ser especificada de duas formas: 1) *Attachment* da classe *pcimTPCAuxClass* diretamente ao objeto que representa regra; 2) *Attachment* da classe *pcimTPCAuxClass* a um objeto da classe estrutural *pcimRuleValidityAssociation*, a qual modela a associação. Em seguida, este objeto é referenciado pelo atributo *pcimRuleValidityPeriodList* da classe *pcimRule*. Este trabalho adotou a estratégia 2 para o mapeamento de períodos de ativação de papéis RBAC.

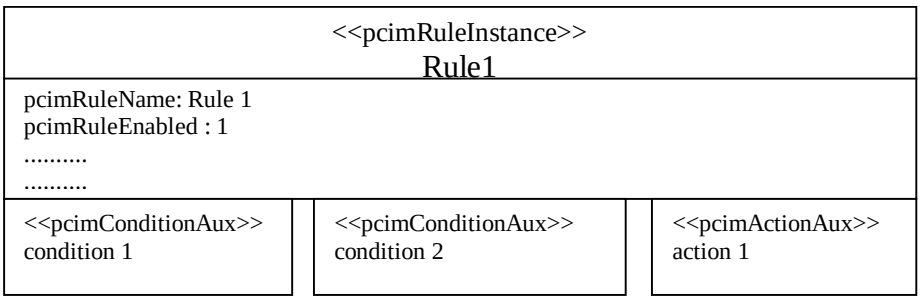

Figura 5.3 – Exemplo de uma regra de política simples

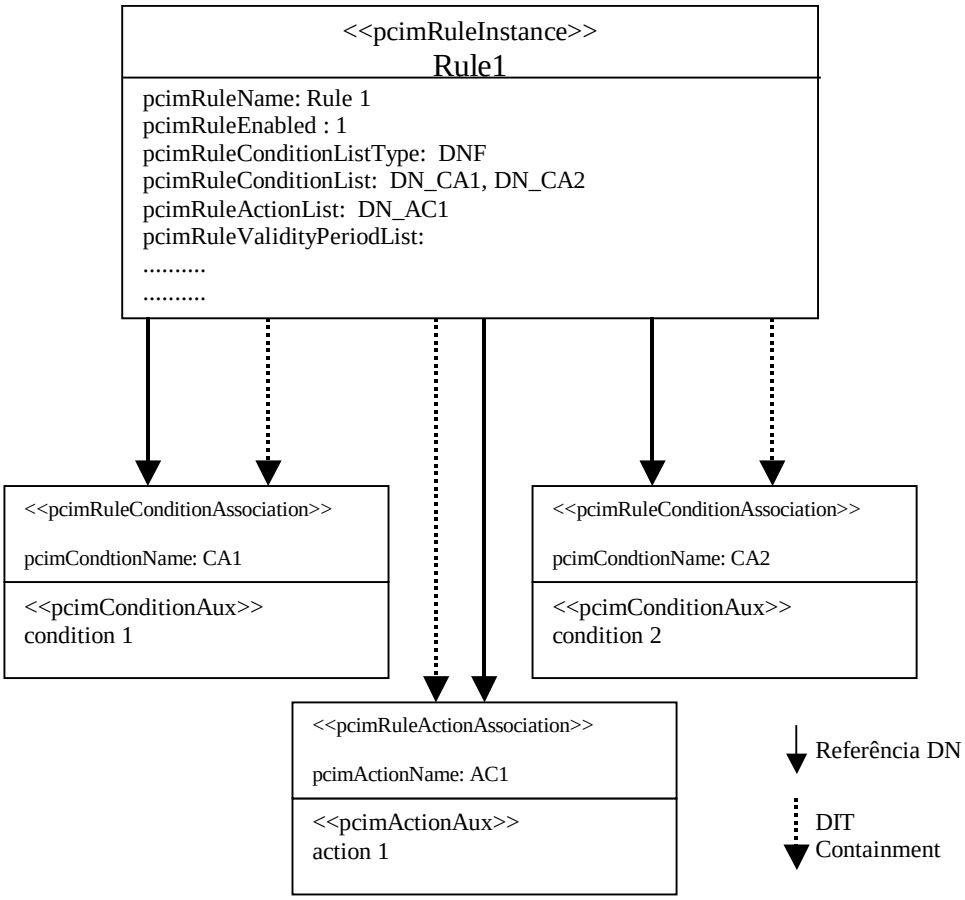

Figura 5.4 – Exemplo de uma regra de política complexa

5.3. O Mapeamento do RBPIM no LDAP

5.3.1 Descrição do esquema de diretório

A presente seção mostra o mapeamento proposto do RBPIM no LDAP. Para a construção deste esquema de diretório foram utilizadas as estratégias apresentadas na seção 5.2. As novas classes introduzidas pelo RBPIM, como o caso de *RBACRole* e *SimplePolicyCondition*, obviamente, não possuem mapeamento definido pelo PCLS, o qual será proposto nesta seção.

As Figuras 5.5 e 5.6 apresentam o esquema de diretório para o mapeamento do RBPIM no modelo de objetos do LDAP. Os tipos de cada classe estão indicados juntamente com os nomes de cada uma delas, onde (s), (ab) e (ax) referem-se, respectivamente, a classes estruturais, abstratas e auxiliares do LDAP. O prefixo *pcim* em cada classe indica que ela foi proposta pelo *PCLS*, enquanto *rbpim* indica que é uma nova classe proposta por este trabalho.

A classe *top* é classe base para toda a hierarquia e *dlm1ManagedElement* é proveniente do CIM. Em destaque encontram-se as duas classes centrais do esquema proposto: *rbpimRole* e *rbpimPermission*. Os papéis definidos em uma política de segurança descrita segundo o modelo RBPIM são inseridos como objetos da classe *rbpimRole*. Face à herança inerente à hierarquia de classes, cada objeto *rbpimRole* herda os atributos de suas classes superiores. Desta forma, segundo a notação do LDAP, cada objeto *rbpimRole* possuirá seu atributo multi-valued *objectclass* com os valores *top, dlm1ManagedElement, pcimPolicy, pcimRule* e *rbpimRole*. Composto de forma semelhante, um objeto de permissão é representado por instâncias da classe *rbpimPermission*. Objetos das classes *pcimRuleConditionAssociation* e *pcimRuleActionAssociation* são associados a objetos de papéis *rbpimRole* e de permissões *rbpimPermission.* Para objetos de papéis elas permitem, respectivamente, a especificação das condições de políticas que realizam a associação *UA* do RBAC e as atribuições de permissões relacionadas à associação *PA* do RBAC. Para objetos de permissões elas permitem, respectivamente, a especificação das condições de políticas que descrevem os recursos associados com a permissão e quais as operações permitidas para os recursos referidos. Objetos da classe *pcimRuleValidityAssociation* são associados a objetos *rbpimRole,* permitindo, desta forma, que sejam determinados os períodos que um dado papel poderá ser ativado.

Cada objeto de condição *pcimRuleConditionAssociation* contém (via *DIT containment*) um objeto *rbpimConditionAssociation,* o qual define o par variável/valor associado à condição. Objetos das classes *rbpimSSD* e *rbpimDSD* permitem que sejam especificados as cardinalidades e os conjuntos de papéis *rbpimRole* que possuem conflitos por separação, respectivamente, estática e dinâmica de tarefas.

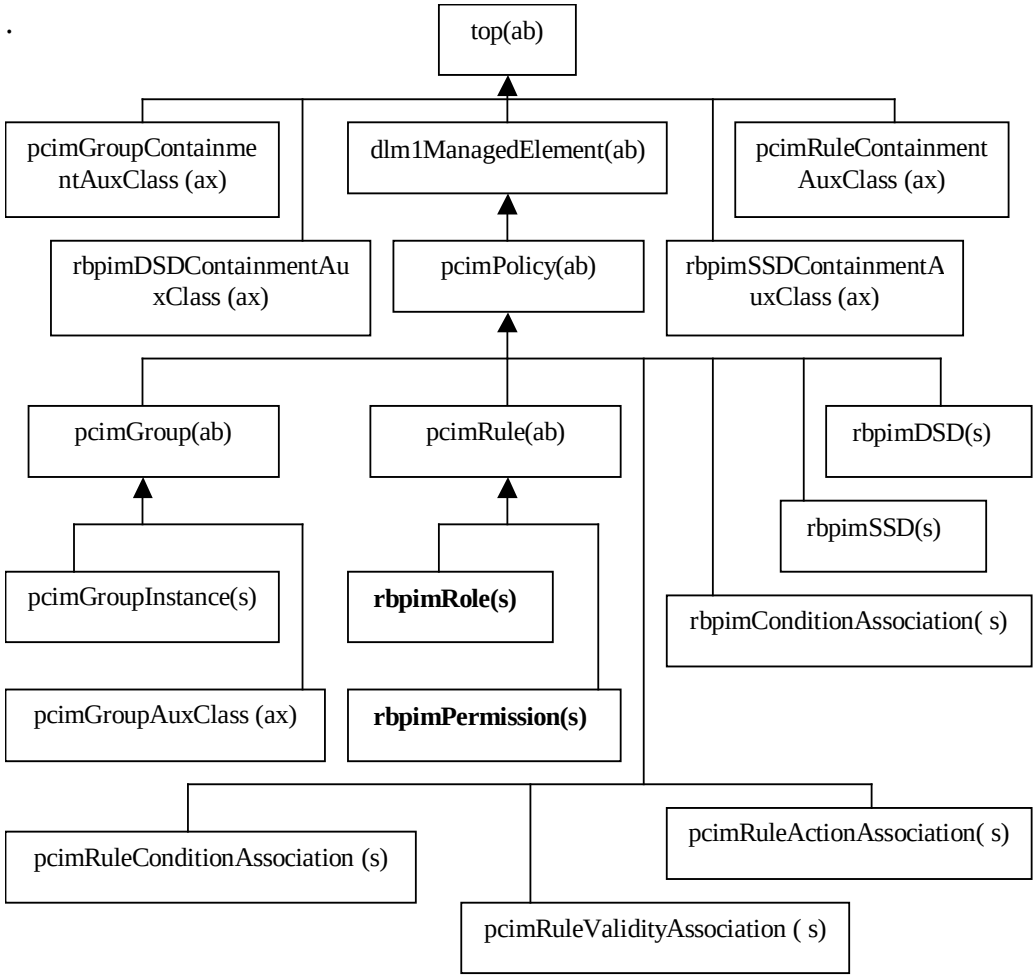

Figura 5.5 – RBPIM: Classes do Esquema de diretório LDAP (Parte 1)

Instâncias das classes auxiliares são vinculadas às instâncias das classes estruturais. [MOORE 01] indica que, quando o esquema for aplicado, devem existir regras para a construção da DIT do diretório que permitam especificar quais classes auxiliares e classes estruturais poderão ter instâncias associadas nestes *attachments*. A Tabela 5.3 apresenta as regras que devem existir no esquema LDAP proposto para o RBPIM.

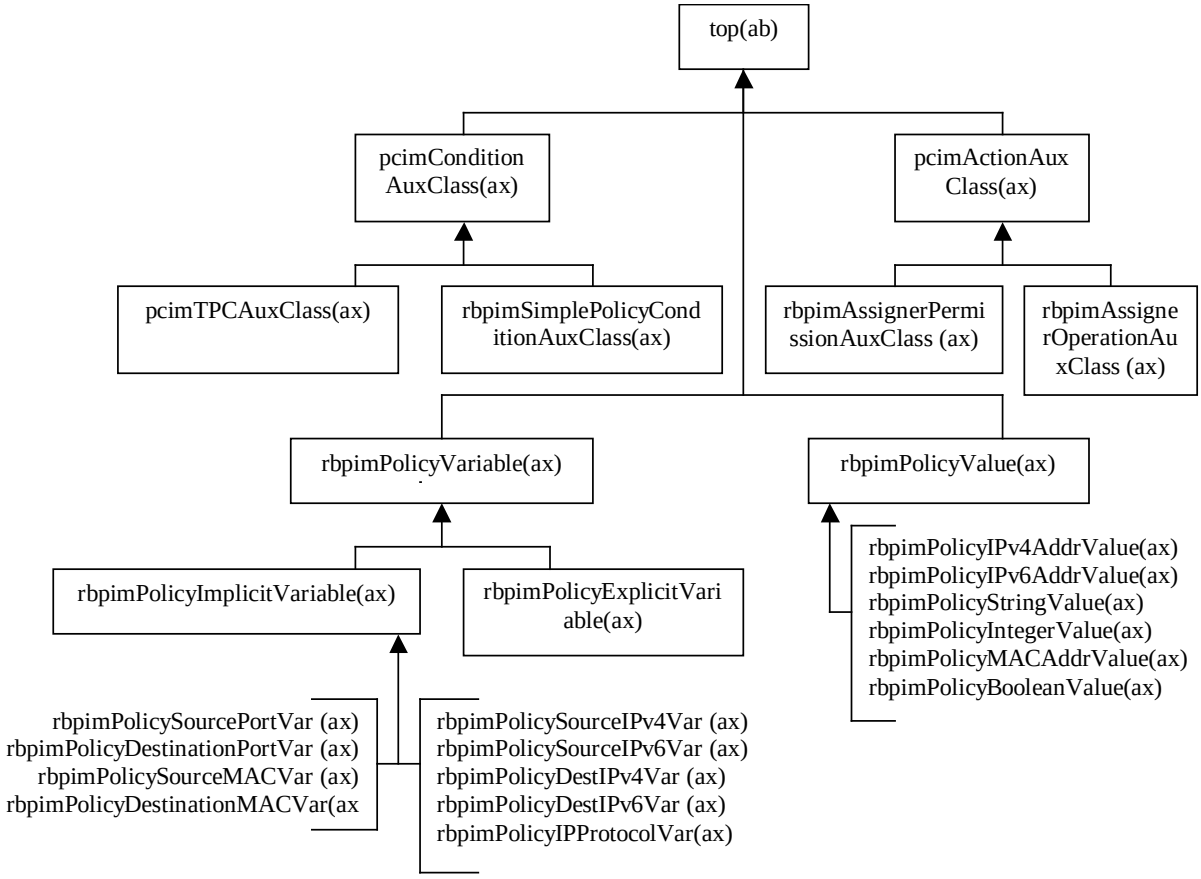

Figura 5.6 – RBPIM: Classes do Esquema de diretório LDAP (Parte 2)

Classe Estrutural	Classes Auxiliares
pcimGroupInstance	*pcimGroupContainmentAuxClass* p*cimRuleContainmentAuxClass* *pcimSSDContainmentAuxClass* *pcimDSDContainmentAuxClass*
pcimRuleConditionAssociation	*pcimConditionAuxClass* *rbpimSimplePolicyConditionAuxClass*
pcimRuleActionAssociation	*pcimActionAuxClass* e suas subclasses
pcimRuleValidityAssociation	*pcimTPCAuxClass*
rbpimConditionAssociation	*rbpimPolicyVariable* e *rbpimPolicyValue* e suas subclasses

Tabela 5.3 – RBPIM: Regras de *attachment*

As Tabelas de 5.4 a 5.7 descrevem as classes do esquema de diretório das figuras 5.5 e 5.6 seus respectivos atributos. As classes abstratas, como por exemplo *pcimPolicy* e *pcimRule*, não serão apresentadas, no entanto, os atributos por elas definidos serão mostrados em suas subclasses, uma vez que podem ter um significado diferente em cada especialização.

Na descrição apresentada nestas tabelas, os atributos de cada *objectclass* são qualificados como:

*****: RDN (*relative distinguished name*) do objeto	**s** : String
m: *multi-valued*	**b**: Boolean
d: Referência DN*(distinguished name)* a um outro objeto	**i** : Inteiro

5.3.2 Classes para o mapeamento papéis, permissões e de separação de tarefas

As classes mostradas na Tabela 5.4 permitem que sejam instanciados no diretório objetos para representar papéis (*rbpimRole*), permissões (*rbpimPermission*) e relacionamentos de separação de tarefas entre papéis (*rbpimDSD* e *rbpimSSD*).

Classe	Descrição	Atributos	Descrição
rbpimRole	Representa um papel RBAC.	rbpimRoleName (*,s) pcimRuleEnabled (i) pcimRuleConditionListType(i) pcimRuleConditionList(d,m) pcimRuleActionList(d,m) pcimRuleValidityPeriodList(d,m) pcimRulePriority (i) rbpimInheritedRoles(d,m)	Nome do papel. Indica se o papel está habilitado(1) ou desabilitado (2). Indica se as condições associadas com o papel são DNF(1) ou CNF(2). Indica os DNs de objetos pcimRuleConditionAssociation. Indica os DNs de objetos pcimRuleActionAssociation. Indica os DNs de objetos pcimRuleValidityAssociation. Indica um valor inteiro para a representação de prioridades entre papéis. Indica os DNs de outros papéis que serão herdados por um dado papel.
rbpimPermission	Representa uma permissão RBAC.	rbpimPermissionName (*,s) pcimRuleConditionListType(i) pcimRuleConditionList(d,m) pcimRuleActionList(d,m)	Nome da permissão. Idem anterior. Idem anterior. Idem anterior.
rbpimDSD	Representa um relacionamento de DSD entre papéis.	rbpimDSDname(*,s) rbpimRoleSet(d,m) rbpimCardinality(i)	Nome do conjunto. Indica os DNs dos objetos rbpimRole envolvidos. Valor inteiro que especifica a cardinalidade do conjunto.
rbpimSSD	Representa um relacionamento de SSD entre papéis.	rbpimSSDname(*,s) rbpimRoleSet(d,m) rbpimCardinality(i)	Nome do conjunto. Indica os DNs dos objetos rbpimRole envolvidos. Valor inteiro que especifica a cardinalidade do conjunto.

Tabela 5.4 - Esquema de diretório do RBPIM – Parte I

As referências *DNs* indicadas pelo atributo *pcimRuleConditionList* de *rbpimRole* associam objetos de condições *pcimRuleConditionAssociation* a um papel. Estes objetos combinados na forma DNF ou CNF (definido pelo atributo *pcimRuleConditionListType*) formam a expressão lógica que especifica os usuários autorizados a assumir um papel (Associação *UA* do RBAC). De maneira semelhante, as referências *DNs* indicadas pelo atributo *pcimRuleConditionList* de *rbpimPermission* associam objetos de condições *pcimRuleConditionAssociation* a uma permissão. Estes objetos combinados na forma DNF ou CNF (definido pelo atributo *pcimRuleConditionListType*) formam a expressão lógica que especifica os recursos associados com a permissão. O atributo *pcimRuleActionList* de *rbpimRole* define referências *DNs* a objetos *pcimRuleActionAssociation.* Estes objetos definem quais as permissões associadas ao papel (Associação *PA* do RBAC). O atributo *pcimRuleActionList* de *rbpimPermission* define referências *DNs* a objetos *pcimRuleActionAssociation,* os quais indicam quais as operações autorizadas ao conjunto de recursos associados à permissão. O atributo *rbpimInheritedRoles* de *rbpimRole* permite a indicação de referências DNs aos papéis juniores herdados por um papel. O atributo *pcimRuleValidityPeriodList* de *rbpimRole* permite a associação de objetos *pcimRuleValidityAssociation* a um papel. Estes objetos indicam períodos de tempo no qual um dado papel se encontrará ativo. O atributo *pcimRulePriority* permite o estabelecimento de prioridades entre papéis. Como será visto no Capitulo 6, este trabalho utiliza este atributo para selecionar os papéis de maior prioridade dentre um conjunto de papéis conflitantes por SSD.

Considere o exemplo da aplicação deste esquema de diretório na Figura 5.7. Ele apresenta um trecho de um arquivo no formato LDIF (*LDAP Data Interchange Format*)[9] onde são mostrados objetos no diretório que descrevem a política RBAC apresentada nos exemplos das figuras 4.9, 4.11 e 4.12 do Capítulo 4. A entrada do diretório *o=BANK123.net* representa um *folder* de uma instituição financeira hipotética (BANK123.net) onde são inseridos todos os objetos LDAP associados a ela. A entrada *ou=RBACPolicyDept* refere-se a um *folder* da unidade organizacional dentro da instituição responsável pela especificação de políticas RBAC. Por fim, todos os objetos de papéis e permissões deste exemplo estão agrupados no

[9] LDIF é um formato para definição de entradas de um diretório no formato texto. A maioria dos serviços de diretórios permite a importação de entradas a partir de um arquivo no formato LDIF.

folder pcimGroupName=FINANCE,ou=RBACPolicyDept,o=BANK123.net, uma vez que se referem a papéis e permissões relacionados com o departamento financeiro.

```
dn: rbpimRoleName=Contador_I, pcimGroupName= Finance, ou= RBACPolicyDept,
    o=Bank123.net
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimRole
rbpimRoleName: Contador_I
pcimRuleEnabled: 1
pcimRuleConditionListType: 1
pcimRuleConditionList: pcimConditionName=UsersCond1, rbpimRoleName=
   Contador_I, pcimGroupName= Finance, ou= RBACPolicyDept, o=Bank123.net
pcimRuleConditionList: pcimConditionName=UsersCond2, rbpimRoleName=
   Contador_I, pcimGroupName= Finance, ou= RBACPolicy Dept, o=Bank123.net
pcimRuleActionList: pcimActionName=Permission1, rbpimRoleName=
   Contador_I, pcimGroupName= Finance, ou= RBACPolicyDept, o=Bank123.net
pcimRuleValidityPeriodList: pcimValidityConditionName=Period1, rbpimRoleName=
   Contador_I, pcimGroupName= Finance, ou= RBACPolicyDept, o=Bank123.net

dn: rbpimPermissionName=App_Directory, pcimGroupName= Finance, ou=
    RBACPolicyDept, o=Bank123.net
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimPermission
pcimRuleConditionListType: 1
pcimRuleConditionList: pcimConditionName=Directory1, rbpimPermissionName=
   App_Directory, pcimGroupName= Finance, ou= RBACPolicyDept, o=Bank123.net
pcimRuleActionList: pcimActionName=Operations1, rbpimPermissionName=
   App_Directory, pcimGroupName= Finance, ou= RBACPolicyDept, o=Bank123.net

dn: rbpimSSDname=SSD01, pcimGroupName= Finance, ou= RBACPolicyDept,
    o=Bank123.net
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimSSD
rbpimSSDname: SSD01
rbpimRoleSet: rbpimRoleName=Contador_I, pcimGroupName= Finance, ou=
RBACPolicyDept, o=Bank123.net
rbpimRoleSet: rbpimRoleName=Contador_II, pcimGroupName= Finance, ou=
RBACPolicyDept, o=Bank123.net
rbpimCardinality: 2
```

Figura 5.7 – Exemplo 1: Objetos de políticas RBPIM (formato LDIF)

A Figura 5.7 apresenta três instâncias do diretório. O objeto identificado por *dn: rbpimRoleName=Contador_I, pcimGroupName= Finance, ou= RBACPolicyDept, o=Bank123.net* representa um papel (*Contador_I*). O DN (*Distinguished Name*) de um objeto é o identificador único de uma entrada no diretório. Além de identificar a entrada, ele também indica a localização desta entrada no diretório. O objeto identificado por *dn: rbpimPermissionName=App_Directory, pcimGroupName= Finance, ou=RBACPolicyDept, o=Bank123.net* representa uma permissão RBPIM. Por fim, *dn: rbpimSSDname=SSD01, pcimGroupName= Finance, ou= RBACPolicyDept, o=Bank123.net* representa um relacionamento de SSD, onde há um conflito entre os papéis *Contador*_I e *Contador*_II. A cardinalidade 2 informa que para qualquer seleção de 2 papéis definidos no conjunto indicado por *rbpimRoleSet* ocorre violação por SSD.

5.3.3 Classes associativas relacionadas a papéis e permissões

As classes apresentadas na Tabela 5.5 estão relacionadas com os objetos associados a um papel e a uma permissão. Cada objeto da classe estrutural *pcimRuleConditionAssociation* tem vinculado a si a classe auxiliar *rbpimSimplePolicyConditionAuxClass*, a qual define a semântica de uma expressão formada pelo par <variável>/<valor>. Os valores de cada par <variável>/<valor> são especificados por um objeto da classe estrutural *rbpimConditionAssociation*, através do *attachment* da classe *rbpimPolicyExplicitVariable*, das classes da hierarquia *rbpimPolicyImplicitVariable* e das classes da hierarquia *rbpimPolicyValue*, conforme o tipo de informação representada na expressão. Objetos *rbpimConditionAssociation* estão associados a objetos *pcimRuleConditionAssociation* através de *DIT containment.*

Objetos *pcimRuleActionAssociation* associados a objetos de permissão (*rbpimPermission*) têm vinculados a si a classe auxiliar *rbpimAssignerOperationAuxClass*, através da qual são especificadas quais operações poderão ser efetuadas no(s) recurso(s) selecionado(s) pela condição associada ao mesmo objeto de permissão. Por outro lado, objetos *pcimRuleActionAssociation* associados a objetos que representam papéis (*rbpimRole*) têm vinculados a si a classe auxiliar *rbpimAssignerPermissionAuxClass*, com a qual são definidas referências a objetos de permissão *rbpimPermission.* Cada objeto da classe *pcimRuleValidityAssociation* tem vinculado a classe auxiliar *pcimTPCAuxClass*, a qual define os atributos necessários à especificação do intervalos de tempo para ativação de papéis.

Classe	Descrição	Atributos	Descrição
pcimRuleConditionAssociation	Representa a associação entre papéis e condições e entre permissões e condições.	pcimConditionName(*,s) pcimConditionGroupNumber(i) pcimConditionNegated (b)	Nome da associação. Valor inteiro utilizado para agrupar condições nas expressões DNF e CNF. Valor boolean que indica se a condição será negada(true) ou não (false) nas expressões DNF e CNF.
rbpimSimplePolicyConditionAuxClass	Representa uma condição no formato <variable> MATCH <value>.	Não tem.	Não tem.
rbpimConditionAssociation	Representa a associação entre uma condição e o par variável/valor relacionados.	rbpimConditionName(*,s)	Nome da associação.
pcimRuleActionAssociation	Representa a associação entre papéis e permissões e entre permissões e operações.	pcimActionName(*,s)	Nome da associação.
rbpimAssignerPermissionAuxClass	Representa uma referência a um objeto permissão do RPIM.	rbpimPermissionDN(d)	Indica uma referência DN para um objeto rbpimPermission.
rbpimAssignerOperationAuxClass	Representa as operações autorizadas para um determinado objeto rbpimPermission.	rbpimOperationList(s,m)	Indica um string contendo as operações permitidas.
PcimRuleValidityAssociation	Representa a associação entre papéis e seus períodos de ativação.	pcimValidityConditionName (*,s)	Nome da associação.
pcimTPCAuxClass	Representa um período em que um determinado papel poderá ser ativado.	pcimTPCTime (s) pcimTPCMonthOfYearMask(bs) pcimTPCDayOfMonthMask(bs) pcimTPCDayOfWeekMask(bs) pcimTPCTimeOfDayMask(bs) pcimTPCLocalOrUtcTime (i)	Indica um intervalo no formato yyyymmddThhmmss/yyyymmddThhmmss. Indica um string de 12 bits. Indica um string de 31 bits. Indica um string de 7 bits. Indica um intervalo Thhmmss/Thhmmss. Inteiro indicando hora local(1) ou UTC(2).

Tabela 5.5 - Esquema de diretório do RBPIM – Parte II

Como exemplo, a Figura 5.8 apresenta o trecho do arquivo LDIF referente aos objetos de condição e de período de ativação associados ao papel *Contador_I* contido na Figura 5.7. Por questões de simplificação, os atributos *objectclass* referentes às classes base da hierarquia (*top*, *dlm1ManagedElement* e *pcimPolicy*) de cada instância foram suprimidos. Dois objetos de condição, *pcimConditionName=UsersCond1* e *pcimConditionName=UsersCond2,* estão sendo mostrados. Eles possuem, via *DIT Containment,* objetos que definem o par

"variável/expressão" associado à condição, os quais são, respectivamente, *rbpimConditionName=Expression1* e *rbpimConditionName=Expression2*.

```
dn:  pcimConditionName=UsersCond1,rbpimRoleName= Contador_I,  pcimGroupName=
      Finance,  ou= RBACPolicy Dept,   o=Bank123.net
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName:UsersCond1
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn:   rbpimConditionName=Expression1,  pcimConditionName=UsersCond1,
      rbpimRoleName= Contador_I,pcimGroupName= Finance, ou= RBACPolicyDept,
      o=Bank123.net
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyImplicitVariable
objectClass: rbpimPolicySourceIPv4Var
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyIPv4AddrValue
rbpimConditionName: Expression1
rbpimIPv4AddrList: 192.168.10.0/24

dn:  pcimConditionName=UsersCond2, rbpimRoleName= Contador_I,   pcimGroupName=
     Finance,  ou= RBACPolicyDept,   o=Bank123.net
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName:UsersCond2
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Expression2, pcimConditionName=UsersCond2,
     rbpimRoleName=  Contador_I,  pcimGroupName=  Finance, ou= RBACPolicyDept,
     o=Bank123.net
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Expression2
rbpimModelClass: Person
rbpimModelProperty: BusinessCategory
rbpimStringList: CT*

dn:  pcimValidityConditionName=Period1, rbpimRoleName= Contador_I, pcimGroupName=
     Finance,  ou= RBACPolicyDept,   o=Bank123.net
objectClass: pcimRuleValidityAssociation
objectClass: pcimTPCAuxClass
pcimValidityConditionName: Period1
pcimTPCTime: 20020701T000000/20020831T240000
```

Figura 5.8 – Exemplo 2: Objetos de políticas RBPIM (formato LDIF)

O papel *Contador_I* (ver Figura 5.7) define o atributo *pcimRuleConditionListType* como 1, ou seja, *DNF*. Assim, a expressão que define os usuários pertencentes à lista de usuários autorizados ao papel é dada por:

(UsersCond1 = TRUE AND UserCond2 = TRUE)

ou,

(rbpimPolicySourceIPv4Var MATCH 192.168.10.0/24 AND Person.BusinessCategory MATCH CT*)

Esta regra define que apenas os usuários cujos acessos sejam provenientes da rede *192.168.10.0/24* e cujos dois primeiros caracteres do atributo *BusinessCategory* dos objetos da classe *Person* do diretório que os representam forem *CT*. A Figura 5.8 ainda apresenta o objeto *pcimValidityConditionName=Period1*,o qual define o intervalo de 01/07/2002 à 31/08/2002 permitido para a ativação do papel.

5.3.4 Classes acessórias para agrupamento

As classes apresentadas na Tabela 5.6 são classes acessórias utilizadas para o estabelecimento do agrupamento da informação de política. Agrupamentos podem ser utilizados para facilitar a localização da informação de política. Por exemplo, suponha que uma determinada empresa deseje agrupar as informações sobre papéis e permissões em função dos diversos departamentos existentes. Uma prática usual consiste em criar uma entrada no diretório para cada um destes departamentos. Estas entradas podem ser instanciadas a partir da classe estrutural *pcimGroupInstance*, caracterizando um *container* de agrupamento para cada um dos departamentos. Caso o diretório já contenha objetos que representam os departamentos em questão não faz sentido criar novas entradas para os departamentos. Neste caso, deve-se vincular a classe *pcimGroupAuxClass* a cada uma destas entradas o que, segundo o *framework* do PCIM/PCLS, também as transforma em *containers* para agrupamento da informação de política. Estas entradas *containers* para objetos de política precisam referenciar os objetos que estão sendo agrupados. As demais classes auxiliares apresentadas na Tabela 5.6 são utilizadas para este fim, ou seja, são vincular às entradas de agrupamento (*pcimGroupInstance e pcimGroupAuxClass)* para fornecer os atributos com as referências DNs necessárias para os objetos contidos. Assim, para referenciar objetos de papéis e permissão faz-se necessário a *attachment* da classe *pcimRuleContainmentAuxClass.*

Classe	Descrição	Atributos	Descrição
pcimGroupInstance pcimGroupAuxClass	Representam um container para agrupamento de papéis, permissões, relacionamentos de separação de tarefas.	pcimGroupName (*)(s)	Nome do agrupamento.
pcimGroupContainmentAuxClass	Representa a agregação entre grupos.	pcimGroupsAuxContainedSet (d,m)	Indica referências DNs de objetos pcimGroup que serão agregados em um container de grupo.
pcimRuleContainmentAuxClass	Representa a agregação entre um objeto container de grupo e seus papéis e permissões subordinados.	pcimRulesAuxContainedSet (d,m)	Indica referências DNs de objetos que representam papéis e permissões, os quais serão agregados em um container de grupo.
rbpimDSDContainmentAuxClass	Representa a agregação entre um objeto container de grupo e objetos DSD subordinados.	rbpimDSDAuxContainedSet (d,m)	Indica referências DNs de objetos que representam o relacionamento de separação de tarefas dinâmica que serão agregados em um container de grupo.
rbpimSSDContainmentAuxClass	Representa a agregação entre um objeto container de grupo e objetos SSD subordinados.	rbpimSSDAuxContainedSet (d,m)	Indica referências DNs de objetos que representam o relacionamento de separação de tarefas estático e que serão agregados em um container de grupo.

Tabela 5.6 - Esquema de diretório do RBPIM – Parte III

5.3.5 Classes para representação de variáveis

As variáveis são utilizadas para a construção de condições individuais. As classes apresentadas na Tabela 5.7 definem as variáveis que podem ser utilizadas para a construção de expressões segundo o *framework* do RBPIM. Conforme definido anteriormente na Tabela 5.3, estas variáveis são vinculadas a objetos da classe estrutural *rbpimConditionAssociation* para representar uma condição em específico. A classe *rbpimPolicyExplicitVariable* permite que sejam referenciados objetos cujas informações estão definidas no contexto da modelagem do CIM, como diretórios e aplicações. As demais classes desta tabela referem-se à representação de informações não definidas segundo os modelos do CIM. Por exemplo, as classes *rbpimPolicySourceIPv4Var* e *rbpimPolicyDestIPv4Var* permitem à representação de um endereço *IPv4* de origem e destino respectivamente. Todas as variáveis implícitas não possuem atributos, uma vez que a própria classe indica qual a informação a ser considerada em uma condição.

Classe	Descrição	Atributos	Descrição
rbpimPolicyExplicitVariable(ax)	Representa uma variável cuja semântica é definida explicitamente no contexto dos modelos do CIM.	rbpimModelClass(s) rbpimModelProperty(s)	Indica o nome da classe que possui o atributo a ser utilizado em uma condição. Indica o nome do atributo a ser utilizado nas condições.
rbpimPolicyImplicitVariable(ax)	Representa uma variável cuja semântica não é definida no contexto dos modelos do CIM.	Não tem.	Não tem.
rbpimPolicySourceIPv4Var (ax) rbpimPolicyDestIPv4Var (ax)	Classes que representam variáveis que indicam, respectivamente, endereços IPv4 de origem e destino.	Não tem.	Não tem.
rbpimPolicySourceIPv6Var (ax) rbpimPolicyDestIPv6Var (ax)	Classes que representam variáveis que indicam, respectivamente, endereços IPv6 de origem e destino.	Não tem.	Não tem.
rbpimPolicySourcePortVar (ax) rbpimPolicyDestinationPortVar (ax)	Classes que representam variáveis que indicam, respectivamente, portas TCP/UDP de origem e destino.	Não tem.	Não tem.
rbpimPolicySourceMACVar (ax) rbpimPolicyDestinationMACVar(ax)	Classes que representam variáveis que indicam, respectivamente, endereços MAC de origem e destino.	Não tem.	Não tem.
rbpimPolicyIPProtocolVar(ax)	Classe que representa o protocolo de nível superior encapsulado no pacote IP.	Não tem.	Não tem.

Tabela 5.7 - Esquema de diretório do RBPIM – Parte IV

5.3.6 Classes para representação de valores

As classes apresentadas na Tabela 5.8 são empregadas na representação de valores utilizados para a construção de condições. Estas classes, que são especializações de *rbpimPolicyValue*, fornecem uma lista de valores que podem ser utilizados nos formato de *Valor Simples*, *Faixa* e *Conjunto*. Por exemplo, para a classe *rbpimPolicyIntegerValue, o* atributo *rbpimIntegerList* poderia ter, respectivamente para cada um dos formatos, *80, (80-88) e (80,81,82).* Da mesma forma que as classes apresentadas para a representação de variáveis apresentadas na subseção anterior, estas classes são vinculadas a objetos da classe

estrutural *rbpimConditionAssociation* para compor a expressão *variável MATCH valor* utilizado nas condições.

Classe	Descrição	Atributos	Descrição
rbpimPolicyStringValue	Representa uma string ou um conjunto de strings utilizado(s) na construção de condições.	rbpimStringList(s,m)	Conjunto de strings utilizado na composição de condições.
rbpimPolicyIntegerValue	Representa um inteiro ou um conjunto de inteiros utilizado(s) na construção de condições.	rbpimIntegerList(s,m)	Conjunto de inteiros utilizado na composição de condições.
rbpimPolicyBooleanValue	Representa um *boolean* ou um conjunto de *booleans* utilizado(s) na construção de condições.	rbpimBooleanList(s,m)	Conjunto de *booleans* utilizado na composição de condições.
rbpimPolicyIPv4AddrValue	Representa um endereço IPv4 ou um conjunto de endereços Ipv4 utilizado(s) na construção de condições.	rbpimIPv4AddrList(s,m)	Conjunto de endereços IPv4 utilizado na composição de condições.
rbpimPolicyIPv6AddrValue	Representa um endereço IPv6 ou um conjunto de endereços IPv6 utilizado(s) na construção de condições.	rbpimIPv6AddrList(s,m)	Conjunto de endereços IPv6 utilizado na composição de condições.
rbpimPolicyMACAddrValue	Representa um endereço MAC ou um conjunto de endereços MAC utilizado(s) na construção de condições.	rbpimMACAddrList(s,m)	Conjunto de endereços MAC utilizado na composição de condições.

Tabela 5.8 - Esquema de diretório do RBPIM – Parte V

Para ilustrar, a Figura 5.9 apresenta em destaque o trecho do arquivo LDIF referente aos objetos de condição associados ao papel *Contador_I* contido na Figura 5.7. Na condição 1, a *PolicyVariable* implícita, que representa um endereço *IPv4* de origem (classe *rbpimPolicySourceIPv4Var*), e o *PolicyValue,* que representa um valor *IPv4* (classe *rbpimPolicyIPv4AddrValue*), são vinculados ao objeto da classe *rbpimConditionAssociation (Expression1)* para formar o par *rbpimPolicySourceIPv4Var MATCH rbpimIPv4AddrList, ou seja, rbpimPolicySourceIPv4Var MATCH 192.168.10.0/24.* De maneira análoga, a condição 2 mostra o *attachment* das classes *rbpimPolicyExplicitVariable* e *rbpimPolicyStringValue* ao objeto *Expression2* para indicar a expressão *Person.BusinessCategory MATCH CT**.

```
dn:  rbpimConditionName=Expression1,  pcimConditionName=UsersCond1,
     rbpimRoleName= Contador_I,pcimGroupName= Finance, ou= RBACPolicyDept,
     o=Bank123.net
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyImplicitVariable
objectClass: rbpimPolicySourceIPv4Var
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyIPv4AddrValue
rbpimConditionName: Expression1
rbpimIPv4AddrList: 192.168.10.0/24

dn: rbpimConditionName=Expression2, pcimConditionName=UsersCond2,
    rbpimRoleName=  Contador_I,  pcimGroupName=  Finance, ou= RBACPolicyDept,
    o=Bank123.net
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Expression2
rbpimModelClass: Person
rbpimModelProperty: BusinessCategory
rbpimStringList: CT*
```

1

2

Figura 5.9 – Exemplo 3: Objetos de política RBPIM (formato LDIF)

5.4. Conclusões do Capítulo

O presente Capítulo apresentou um esquema de diretório para dar suporte à representação de políticas RBAC no LDAP construídas segundo o *framework* do RBPIM. A maioria das classes que compõem este esquema é derivada do PCLS. No entanto, as classes do PCIMe e as novas classes introduzidas pelo RBPIM tiveram um mapeamento LDAP proposto neste Capítulo.

O uso de um repositório único, com a centralização lógica da informação do controle de acesso, facilita a criação de implementações baseadas em políticas, como a que será proposta no Capítulo 6. O esquema proposto se baseia nos objetos existentes na infraestrutura computacional, os quais são representados no repositório segundo os modelos do CIM. O uso de padrões como o PCIM, o PCLS, o PCIMe e o CIM, que forneceram a base do modelo e do esquema do RBPIM, é importante, pois, além das facilidades de integração dos serviços de diretório, possibilita a interoperabilidade entre outras aplicações e o *framework* proposto neste trabalho.

Uma análise do esquema proposto permite observar que a estratégia adotada para a especificação de políticas RBAC é bastante flexível, pois permite, por exemplo, que o administrador crie uma regra que associe desde um simples usuário a um papel, através de uma condição que apenas selecione o usuário, até a associação de um conjunto de usuários a um papel, especificada através da criação de uma regra de política igualmente simples, mas, no entanto, que permite a seleção de um grupo de usuários. Da mesma forma, diversos recursos podem ser associados com um único objeto de permissão RBPIM através da criação de uma única regra de política. Em seguida, a partir da associação deste objeto de permissão a um papel, a mesma flexibilidade é alcançada no diz respeito à autorização dos usuários do referido papel aos recursos em questão. Os algoritmos que serão apresentados no Capítulo 6 evidenciarão esta estratégia.

Capítulo 6

A Implementação do RBPIM: Arquitetura, API e Algoritmos do PDP

6.1. Introdução

O Capítulo 5 apresentou um esquema de diretório para o mapeamento do RBPIM no LDAP, o qual foi proposto no Capítulo 4. Através da utilização deste esquema, as organizações poderão especificar políticas RBAC para o estabelecimento do controle de acesso a recursos, dispondo, ainda, de todos os benefícios inerentes ao uso de serviços de diretórios como repositórios, onde se destacam a interoperabilidade, a centralização lógica e a distribuição física. No entanto, um serviço de diretório é apenas um repositório com facilidades de integração com outros sistemas. A implementação das políticas RBAC exige, obviamente, uma arquitetura composta de outras entidades que possam agir dinamicamente, através da interação com as entidades da rede e com o próprio repositório de informação RBAC para a tomada de decisão do controle de acesso.

O presente Capítulo, neste contexto, apresenta uma arquitetura para a implementação do controle de acesso baseado em papéis proposto pelo *framework* do RBPIM. Com este propósito, a seção 6.2 mostra alguns trabalhos de padronização importantes para definição da arquitetura em questão. A seção 6.3 apresenta a arquitetura proposta, a qual foi derivada da arquitetura definida em [RFC 2753]. A seção 6.4 apresenta as principais chamadas relacionadas ao acesso às facilidades oferecidas pela arquitetura. A seção 6.5 define os principais algoritmos para a implementação do PDP. Por fim, a seção 6.6 conclui o Capítulo.

6.2. Padrões para Implementação

6.2.1 Modelos de Arquitetura

[RFC 2753] define três arquiteturas possíveis para a implementação do controle baseado em política, todas centradas em duas entidades principais, o PDP e o PEP, conforme mostra a figura 6.1. No cenário da figura 6.1.a, O PEP (*Policy Enforcement Point*) é um componente presente em um nó da rede enquanto o PDP (*Policy Decision Point*) é uma entidade remota [RFC 2753] que fornece um serviço baseado em decisões de política. O PEP interage com as entidades da rede e tem como responsabilidade garantir que uma decisão de política seja cumprida.

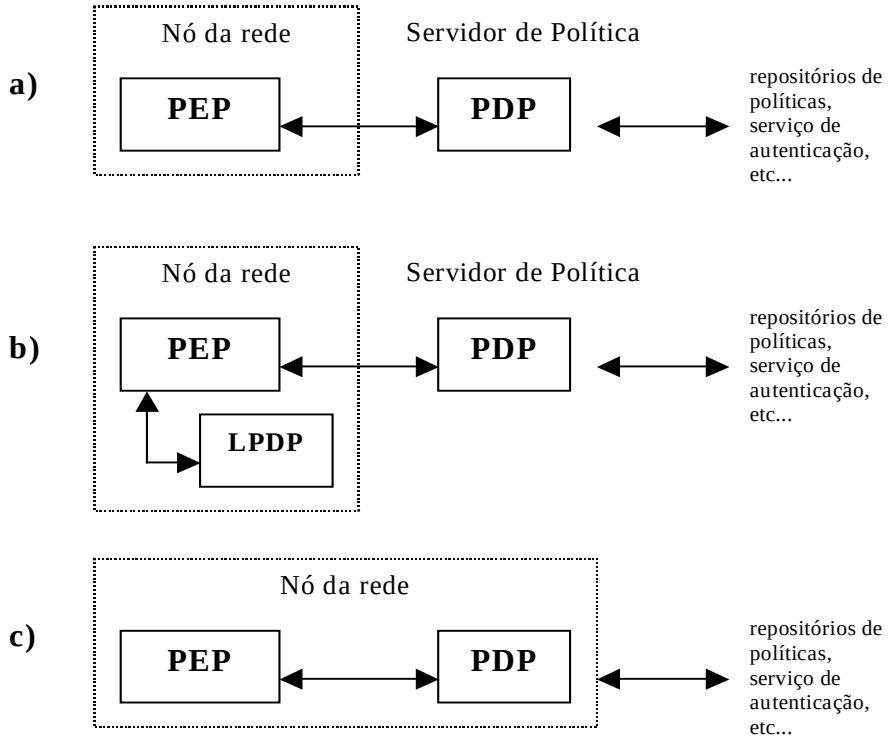

Figura 6.1 – Arquiteturas para implementação do controle baseado em políticas

O PDP, a partir de solicitações enviadas pelo PEP ou da aplicação de novas regras, realiza decisões de política, retornando o resultado desta decisão ao PEP para que elas sejam implementadas. Obviamente, o PDP precisa utilizar mecanismos adicionais para executar suas

funções. Entre eles estão protocolos, como o SNMP[10] e o LDAP, e serviços, como de autenticação e de repositório de informações.

As figuras 6.1.b e 6.1.c são variações da arquitetura indicada pela figura 6.1.a, onde o PDP também se apresenta em um nó da rede. Esta variação é indicada no caso de políticas que dependem fortemente da informação e do estado de uma entidade em particular na rede, situação caracterizada pelo aspecto dinâmico das informações [RFC 2753].

No cenário da figura 6.1.b, a partir de um evento onde o PEP precisa tomar uma decisão baseada em política, ele contata seu LPDP – *Local Policy Decision Point* – e obtém um resultado parcial. Este resultado parcial de política é encaminhado ao PDP, juntamente com as outras informações necessárias para a tomada de decisão. Por fim, o PDP toma a decisão final e a retorna ao PEP. Nesta situação, o LPDP e o PDP trabalham cooperativamente, onde o primeiro atua de forma subordinada ao segundo. Na figura 6.1.c, o PDP se apresenta inteiramente no próprio nó da rede, não havendo a necessidade do PEP contatar um servidor de políticas remoto.

6.2.2. Interações entre o PDP e PEP

Um aspecto vital na arquitetura mostrada na seção anterior é a interação entre o PDP e o PEP. No sentido de estabelecer um padrão para esta comunicação, o *IETF Networking Group* definiu um protocolo específico, o COPS – *Common Open Policy Service*, proposto pela [RFC 2748]. Este protocolo emprega um modelo cliente/servidor onde o PEP envia requisições de políticas ao PDP, o qual retorna as decisões relacionadas de volta ao PEP. Com intuito de estabelecer a confiabilidade na troca de mensagens, o COPS utiliza o TCP como protocolo de transporte. Conforme padronização do IANA, entidades PDP devem aguardar pedidos de conexões na porta 3288, ficando o PEP responsável por iniciar a conexão TCP. O PEP deve manter uma conexão TCP permanente com o PDP.

O COPS é um protocolo *query/response* que suporta dois modelos para o controle baseado em políticas, o *Outsourcing* e o *Provisioning*, conforme ilustrado na Figura 6.2. O modelo *Outsourcing* está relacionado a um cenário onde ocorrem eventos em que o PEP precisa de uma decisão de política instantânea (autorização), no qual ele envia uma requisição ao PDP e aguarda a decisão de política. Neste modelo, o PEP delega todas decisões de

[10] O SNMP(*Simple Network Management Protocol*) é um protocolo utilizado no gerenciamento de redes.

políticas a um PDP externo. No modelo *Provisioning*[11] não há correlação direta entre os eventos do PEP e decisões do PDP. Neste caso, o PDP também pode agir de maneira pró-ativa, provisionando o PEP com informações de política [RFC 3084], esquema tipicamente utilizado na atualização da configuração de dispositivos resultantes das especificações *SLAs* (*Service Level Agreements)*.

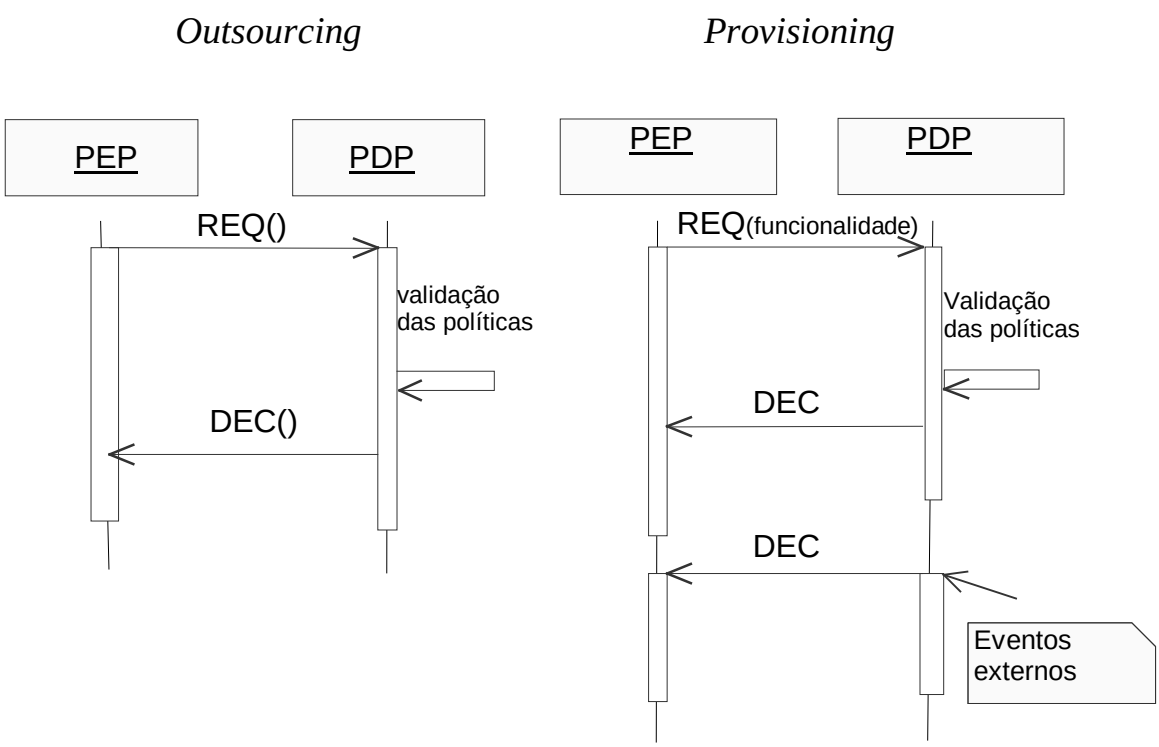

Figura 6.2 – COPS: Modelos para o controle baseado em políticas

O COPS fornece um esquema de segurança para autenticação, proteção contra *replays* e integridade, podendo, também, usar outros protocolos para segurança como o *IPSec* ou *TLS* para autenticar e estabelecer um canal de comunicação seguro entre o PEP e o PDP [RFC 2748]. [WALKER 02] descreve como usar o *TLS* para estabelecer conexões COPS seguras sobre a *Internet*.

6.3. A Arquitetura Proposta

6.3.1. Apresentação

A partir do referencial das padronizações expostas na seção anterior, este trabalho propõe uma arquitetura para a implementação do RBPIM baseada na arquitetura ilustrada pela

[11] A RFC 3084 utiliza os termos *Provisioning* e *Configuration* para identificar o mesmo modelo.

Figura 6.1.a., utilizando o COPS na interação PDP/RBPEP[12] no modelo *Outsourcing*. Esta arquitetura está apresentada na Figura 6.3.

O COPS e a arquitetura para o controle baseado em política apresentada na seção 6.2 foram propostos com intuito de mover a informação de política para dispositivos, focando os aspectos relacionados ao domínio do *QoS*, no entanto, eles são suficientemente genéricos para que possam ser utilizados em outros domínios [CISCO 99], como o domínio das políticas de segurança, escopo deste trabalho.

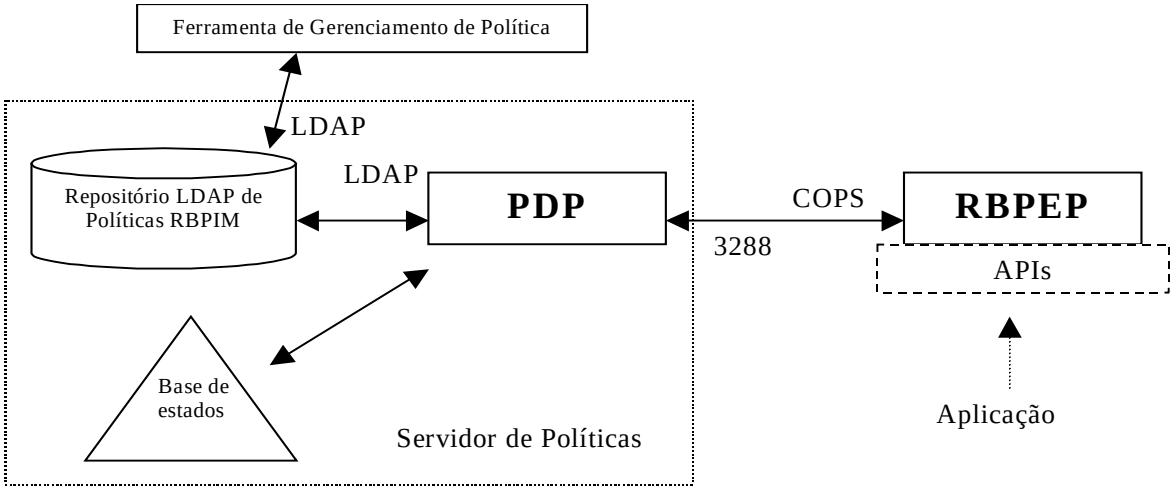

Figura 6.3 – Arquitetura proposta para a implementação do RBPIM

A adoção desta arquitetura se justifica pelo fato de que este trabalho busca o desenvolvimento de uma aplicação para o fornecimento de um serviço de controle de acesso baseado em papéis. Em um cenário típico da utilização deste serviço, uma aplicação precisa autorizar o acesso de um usuário a um recurso submetido ao controle do RBPIM. Assim, esta aplicação poderia solicitar esta decisão de política através de uma chamada da API disponibilizada pelo RBPEP. Este, por sua vez, consultaria o servidor de políticas via COPS para obter o resultado da decisão para, em seguida, retorná-lo à aplicação. A justificativa para a utilização do modelo *Outsourcing* está relacionada com a complexidade e a forma da aplicação do *framework* de políticas. O modelo *Provisioning* exige um RBPEP mais complexo, enquanto o modelo *Outsourcing* torna o RBPEP mais simples, pois ele (RBPEP) delega toda a responsabilidade das decisões de política para um PDP externo. As políticas baseadas em papéis, como a proposta do RBPIM, exigem procedimentos de controle e

[12] A entidade está definida em [RFC 2753] como PEP, no entanto, neste trabalho, ela foi chamada de RBPEP, ou seja, Role-Based PEP.

avaliação bastante complexos, como a separação de tarefas entre papéis e o gerenciamento de sessões de usuários, o que torna impraticável a escolha do modelo *Provisioning* para este esquema de controle de acesso.

A Tabela 6.1 resume as principais características das entidades desta arquitetura. [SANDHU 00] define uma entidade denominada *Sessão* para representar a ativação de papéis a usuários. Através dela, o PDP mantém informações sobre os usuários ativos para a realização das decisões de políticas RBPIM. Esta situação pode ser exemplificada em dois cenários:

1) Quando o PDP precisa verificar se um usuário pode acessar um recurso ele necessita saber quais os papéis estão ativos para aquele usuário dentro do contexto da sessão corrente;
2) O PDP precisa saber quais os papéis estão ativos para um determinado usuário para verificar os conflitos de separação de tarefas, caso o usuário solicite um papel adicional no contexto da sessão corrente.

Neste sentido, este trabalho define uma entidade denominada *Base de Estados* para dar suporte ao PDP. Ela mantém a informação sobre sessões ativas de usuários, como um ID que identifica a sessão do usuário, um ID que identifica o usuário e uma lista contendo os papéis ativos no contexto de uma sessão, como será mostrado nas próximas seções.

Entidade	Função
PDP	Processo servidor que fornece um serviço de controle de acesso baseado em papéis segundo as informações de políticas RBPIM contidas no repositório LDAP. Aguarda requisições de políticas provenientes do RBPEP.
RBPEP	Processo cliente que consulta o PDP para obter decisões de políticas em função de chamadas de aplicações. Disponibiliza APIs para o acesso às facilidades da arquitetura.
Repositório LDAP de Políticas RBPIM	Repositório de política baseado em serviços de diretório LDAP. A informação de política é descrita segundo o esquema de diretório apresentado no Capítulo 5.
Base de Estados	Repositório de dados para o armazenando da informação decorrente das sessões de usuários.
Ferramenta de Gerenciamento de Política	Ferramenta de apoio ao administrador de política. Permite o gerenciamento da informação de política RBPIM no repositório LDAP.

Tabela 6.1 – Entidades da arquitetura proposta para a implementação do RBPIM

6.3.2 Mensagens COPS para a arquitetura RBPIM

A RFC 2748 define que cada mensagem COPS consiste em um cabeçalho seguido por um ou mais objetos específicos auto-identificáveis que contêm a informação de política relacionada com a mensagem, conforme mostra a figura 6.4. Observe que os campos do protocolo são organizados em palavras de 32 bits. Por exemplo, *client-type* ocupa os bytes número 2 e 3 do cabeçalho de cada mensagem COPS enquanto os campos *version* e *flags* são alocados dentro do primeiro byte da mensagem (4 bits para cada um).

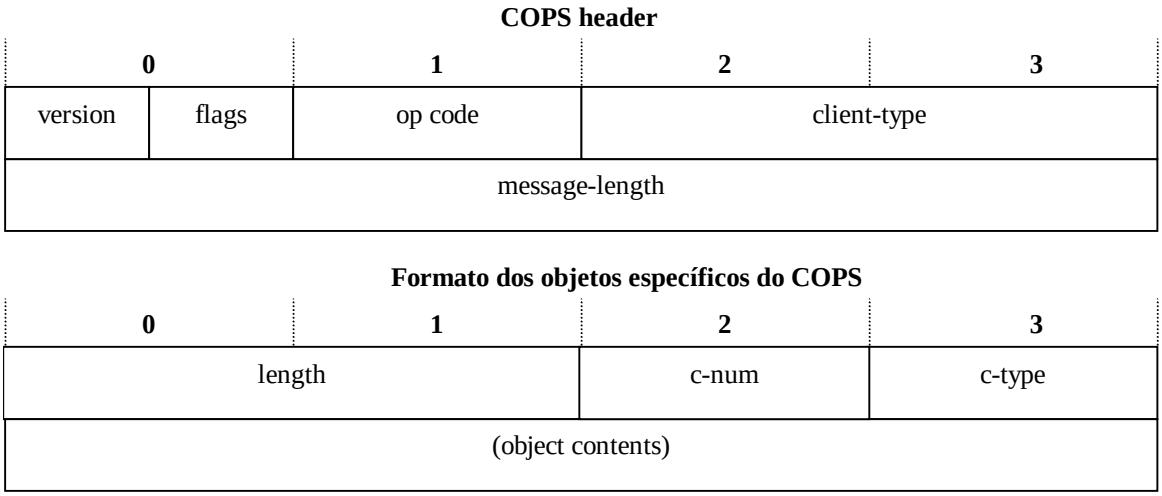

Figura 6.4 – O Protocolo COPS

A Tabela 6.2 apresenta os tipos de mensagens COPS. Em um cenário típico, o PEP solicita ao PDP a abertura do serviço de política através de uma mensagem OPN a qual é respondida através do envio de uma mensagem CAT (pedido aceito) ou CC (pedido não aceito). O PEP pode (caso o serviço esteja aberto) solicitar uma decisão de política ao PDP através de uma mensagem REQ. O PDP, assim, realiza a decisão de política, retornando seu resultado ao PEP através de uma mensagem DEC. Após o recebimento da decisão de política, o PEP deve enviar ao PDP uma mensagem RPT para indicar o *status* do recebimento/efetivação da decisão de política.

Uma característica importante do COPS é a sua natureza *stateful*, ou seja, as solicitações dos clientes do PEP são instaladas no PDP remoto até que elas sejam explicitamente excluídas pelo PEP [RFC 2748]. Uma outra questão importante relacionada com este protocolo é o esquema de tolerância a faltas. Este esquema é alcançado através do envio constante de mensagens *keep-alive* entre o PDP e PEP. Quando a conexão TCP é perdida o PDP pode excluir todos os estados instalados referentes ao PEP cujo contato foi

perdido. Quando o PEP detecta a perda da conexão ele deve continuar tentando o restabelecimento da conexão e, assim que conseguir, deve explicitamente mandar uma mensagem CC contendo um objeto de erro. Também, é esperado que o PEP notifique o PDP sobre eventuais solicitações efetuadas durante o período que a conexão estava perdida. Ainda, o PDP pode solicitar, através de uma mensagem SSQ, que todos os estados internos do PEP sejam sincronizados com os estados instalados no PDP [RFC 2748].

Campo	Valor	Significado
version	**1**	Campo de 4 bits que indica a versão do protocolo.
flags	**0x0** **0x1**	Campo de 4 bits com flags de controle. Deve ser 0x1 quando a mensagem está relacionada com outra mensagem COPS(DEC,RPT).
op code	**1**-REQ **2**-DEC **3**-RPT **4**-DRQ **5**-SSQ **6**-OPN **7**-CAT **8**-CC **9**-KA **10**-SSC	Campo de 8 bits que indica a operação solicitada. 1-Request PEP -> PDP 2-Decision PDP-> PEP 3-Report state PEP -> PDP 4-Delete request state PEP -> PDP 5-Synchronize state PDP -> PEP 6-Client Open PEP -> PDP 7-Client Accept PDP -> PEP 8-Client Close PEP -> PDP, PDP -> PEP 9-Keep alive PEP -> PDP, PDP -> PEP 10-Synchronize state complete PEP -> PDP
client-type	**0x8000**-RBPEP	Campo de 16 bits que indica o tipo de cliente de política. A RFC 2748 indica a faixa de 0x8000 a 0xFFFF para clientes específicos(não padronizados).
message-length		Campo de 32 bits que informa o tamanho da mensagem em octetos, incluindo o cabeçalho e os objetos específicos de política.

Tabela 6.2 – Campos do cabeçalho de mensagens COPS

Campo	Valor	Significado
length		Campo de 8 bits que descreve o número de octetos que compõem o objeto, incluindo seu próprio cabeçalho.
c-num	**1**-Handle **2**-Context **3**-In Interface **4**-Out Interface **5**-Reason **6**-Decision **7**-LPDP Decision **8**-Error **9**-Client Spec. Info **10**-Keep alive tim. **11**-PEP Identif. **12**-Report Type **13**-PDP Redirect Add. **14**-Last PDP Add. **15**-Accounting Timer **16**-Message Integrity	Campo de 8 bits que identifica o tipo de informação contida no objeto específico de política.
c-type		Campo de 8 bits definido no contexto de cada c-num que indica o subtipo da informação contida no objeto.
object-content		Campo de tamanho variável que contém a informação de política com a mensagem.

Tabela 6.3 – Campos dos objetos específicos COPS

O PEP e o PDP podem suportar vários tipos de clientes. Um exemplo de cliente de política é um roteador RSVP que deve exercer controle baseado em política a partir do uso do protocolo RSVP [RFC 2748]. Tipos diferentes de clientes normalmente exigem dados de política específicos. O PDP definido pela arquitetura proposta neste trabalho visa atender apenas a clientes de políticas RBPIM, aqui denominados RBPEP *Clients*. O termo RBPEP *Client*, utilizado daqui para frente neste trabalho, refere-se a uma aplicação que foi modificada com a incorporação de funcionalidades que a permite solicitar decisões de políticas ao PDP através do seu RBPEP. Este trabalho define um *client-type* com número *0x8000*, o qual pertence a faixa de clientes não padronizados, para identificar um RBPEP *Client*.

Na Tabela 6.3, cada *c-num* define um tipo de objeto de política COPS auto-identificável com o seguinte significado: 1 - Identifica de forma única um estado de um cliente dentro do contexto de uma conexão com o mesmo PEP para cada tipo de cliente; 2 - Identifica o tipo de evento que deu origem à mensagem; 3 e 4 - Usados para identificar as interfaces às quais uma dada mensagem se aplica; 5 - Usado para especificar o motivo pelo qual um determinado estado foi excluído; 6 - Indica uma decisão realizada pelo PDP; 7 - Indica uma decisão realizada pelo LPDP; 8 - Indica que um erro específico do COPS ocorreu; 9 - Contém informações específicas a um tipo de cliente; 10 - Usada para validar uma conexão entre o cliente e o servidor; 11 - Objeto para identificação do PEP junto ao PDP em uma mensagem OPN; 12 - Usada pelo PEP para comunicar ao PDP o sucesso ou falha no cumprimento da decisão de política; 13 - Usada pelo PDP para redirecionamento do PEP a um outro PDP; 14 - Usada pelo PEP para informar o último PDP ao qual ele se conectou anteriormente; 15 - Indicação de um *timer* para o estabelecimento de um intervalo mínimo de tempo para relatórios de auditoria; 16 - Objeto de integridade utilizado para a inclusão de um número de seqüência e de um *digest*, úteis para a autenticação e validação da mensagem.

Um *client-handle*, definido por um objeto específico COPS contendo *c-num=1* e *c-type=1*, permite a definição de um valor único que identifica um estado em particular mantido pelo PDP no contexto da conexão com um mesmo RBPEP para um tipo de cliente específico. Cada mensagem COPS deve ser associada a um determinado *client-handle*. No contexto deste trabalho, cada sessão de usuário RBPIM recebe um *client-handle* único, de tal forma a permitir que o PDP mantenha e identifique na *Base de Estados* um estado específico para cada sessão (veja seção 6.4).

A interação básica de solicitação e resposta de decisões de política é realizada pelas mensagens *REQ* e *DEC*. No entanto, como pode ser visto nas Tabelas 6.2 e 6.3, o COPS define uma série de outras mensagens que permitem o controle e o gerenciamento da conexão PDP-RBPEP. Entre estas mensagens estão o envio de uma notificação do RBPEP ao PDP indicando o sucesso ou falha na implementação da decisão de política, muito útil para monitoramento e auditoria; solicitação/confirmação de sincronização; notificação de erros; verificação da conexão através de mensagens *keep-alive* entre o RBPEP e o PDP; mensagens de redirecionamento; suporte à integridade. Para a implementação da arquitetura proposta, este trabalho utilizará as mensagens *REQ,DEC,RPT,OPN,CAT,CC,DRQ*, conforme mostram as tabelas de 6.4 a 6.10. Todas mensagens são especificadas para <*Common Header*> com *client-type*=0x8000 e *flags*=0x0, a não ser quando indicado o contrário.

Tabela 6.4 - Mensagem OPN

<Client-Open>:	**<Common Header>** **<PEPID>**
<Common Header>:	op code=6
<PEPID>:	C-Num = 11, C-Type = 1 object content = IP ou DNS name do RBPEP

OPN: Usada após o estabelecimento da conexão TCP entre RBPEP e PDP para início de uma conexão de serviço de política para um tipo específico de cliente. <PEPID>: Objeto que transporta o endereço IP ou DNS *name* do RBPEP. Pode ser utilizado na localização de políticas particulares a um determinado domínio.

Tabela 6.5 - Mensagem CAT

<Client-Accept>:	**<Common Header>** **<KA Timer>**
<Common Header>:	op code=7
<KA Timer>:	C-Num = 10, C-Type = 1 object content = de 1 a 65535

CAT: Usada pelo PDP para responder positivamente a um <Client-Open>. O objeto <KA Timer> indica o *time-out* relacionado ao recebimento e envio de uma mensagem que será utilizado durante a conexão.

Tabela 6.6 - Mensagem CC

<Client-Close>:	**<Common Header>** **<Error>**
<Common Header>:	op code=8
<Error>:	C-Num = 8, C-Type = 1 object content = error code + sub-code (ver Anexo B)

CC: Usada pelo PDP ou RBPEP para notificar seu par que um determinado *client-type* não é mais suportado. Error Code+Sub-Code indica o erro associado à mensagem.

Tabela 6.7 - Mensagem DRQ	
<Delete Request>:	**<Common Header>** **<Client Handle>** **<Reason>**
<Common Header>:	op code=4
<Client Handle>:	C-Num = 1, C-Type = 1 object content = SessionID
<Reason>:	C-Num = 5, C-Type = 1 object content=Reason Code+Sub-Code (veja [RFC 2748])

DRQ: Usada pelo RBPEP para informar o PDP que um estado mantido para um *client-handle* não é mais necessário. O objeto <Reason> especifica o motivo pelo qual o RBPEP está solicitando a exclusão do estado. Neste trabalho, esta mensagem será utilizada para indicar um pedido de encerramento de uma sessão RBPIM ativa (veja subseção 6.4.4).

Tabela 6.8 – Mensagem REQ	
<Request Message>:	**<Common Header>** **<Client Handle>** **<Context >** **[<ClientSI(s)>]**
<Common Header>:	op code = 1
<Client Handle>:	C-Num = 1, C-Type = 1 object content = SessionID
<Context>:	C-num = 2, C-Type = 1 object content = R-Type+ M-Type (ver seção 6.4)
<ClientSI(s)>:	<ClientSI> | <ClientSI(s)>
< ClientSI >:	C-num = 9, C-Type = 2 object content = (ver seção 6.4)

REQ: Usada pelo RBPEP para solicitar ao PDP uma decisão de política RBPIM. <Context> indica qual o evento gerador da solicitação do cliente RBPEP. <ClientSI> é o objeto de política específico do cliente que carrega a informação de política relacionada à mensagem.

Tabela 6.9 - Mensagem DEC	
<Deciosion Message>:	**<Common Header>** **<Client Handle>** **<Decision> | <Error>**
<Common Header>:	op code=2, flags=0x1
<Client Handle>:	C-Num = 1, C-Type = 1 object content = SessionID
<Decision>:	<Context> <Decision: Flags> [<Decision: ClientSI Data>]
<Context>:	C-num = 2, C-Type = 1 object content = R-Type+ M-Type (ver seção 6.4)
<Decision: Flags>:	C-num = 6, C-Type = 1 object content = Commands+Flags (ver seção 6.4)
<ClientSI Data>:	C-num = 6, C-Type = 4 object content = (ver seção 6.4)
<Error>:	C-Num = 8, C-Type = 1 object content = error code + sub-code (ver Anexo B)

DEC: Usada pelo PDP para o envio de uma decisão de política RBPIM ao RBPEP. Em função do contexto da utilização mensagem, o objeto <ClientSI Data> transporta informações específicas associadas à decisão. Por exemplo, para uma resposta relativa a uma abertura de sessão, este objeto pode retornar a lista papéis ativados para o usuário, como será mostrado na próxima seção.

Tabela 6.10 - Mensagem RPT	
<Report State>:	**<Common Header>** **<Client Handle>** **<Report-Type>**
<Common Header>:	op code = 3 , flags=0x1
<Client Handle>:	C-Num = 1, C-Type = 1 object content = SessionID
<Report-Type>:	C-Num = 12, C-Type = 1 object content = Report-Type (ver seção 6.4)

RPT: Usada pelo RBPEP para que ele possa notificar o PDP sobre o resultado da efetivação da decisão de política recebida pela mensagem DEC anterior.

Na próxima seção será apresentada a API para o acesso às facilidades da arquitetura RBPIM. Também será mostrado como as mensagens definidas nas tabelas de 6.4 a 6.10 serão utilizadas para suprir a informação de política requerida pelas chamadas desta API.

6.4. A API do RBPEP

Conforme mostrado na tabela 3.4, [SANDHU 00] propõe uma especificação de necessidades funcionais que fornecem a semântica de operação do modelo RBAC. A maioria das especificações funcionais contidas nesta tabela está relacionada com as funcionalidades da *Ferramenta de Gerenciamento de Política*, como, por exemplo, *AddUser*, *CreateSSDSet* e *DSDRoleSets*. Do ponto de vista do fornecimento do serviço, as funções do sistema são mais relevantes, como *CreateSession, AddActiveRole* e *CheckAccess*. Neste contexto, a partir destas especificações, a presente seção deriva um conjunto de chamadas que permitirão que aplicações utilizem o serviço de controle de acesso baseado em papéis proposto pelo *framework* do RBPIM. Assim, será dado enfoque às chamadas do serviço RBPIM (funções de sistema), ficando as chamadas das funções administrativas e de revisão para um trabalho posterior.

Para maior compreensão do contexto da utilização das chamadas da API do RBPEP, considere o cenário da figura 6.5. Neste caso, o cliente RBPEP é a aplicação que provê serviços WEB, a qual está sendo executada no Servidor WEB. Esta aplicação controla o acesso a seus recursos segundo o *framework* de políticas do RBPIM. Neste exemplo, as funcionalidades do RBPEP são disponibilizadas a esta aplicação através de um componente. Um componente é uma unidade binária de código destinada à construção de sistemas de *software*. Um componente pode ter várias interfaces, as quais definem os seus pontos de acesso. Para maiores informações sobre componentes, consulte [BOOCH 00].

Para utilizar o controle de acesso baseado nas políticas do RBPIM, a aplicação, ao ser inicializada, requisita o serviço de políticas RBPIM através do seu RBPEP. O RBPEP, por sua vez, deve estabelecer uma conexão com o PDP, indicando que ele precisa prover o serviço de política RBPIM. A partir deste ponto, para cada usuário, a aplicação deve solicitar ao seu RBPEP a criação de uma sessão RBPIM. Com isto, o PDP manterá um estado com as informações da sessão ativa de cada cliente do servidor WEB. Após a abertura de sessão, a aplicação poderá solicitar decisões de políticas RBPIM no contexto de cada sessão de usuário. Cada uma destas sessões é identificada por um *SessionID* único, criado pelo próprio RBPEP, que é transportado como um objeto específico *client-handle* em todas as mensagens COPS. A RFC 2748 especifica que este *client-handle* tem formato livre, ficando a cargo das implementações do COPS realizar tal especificação.

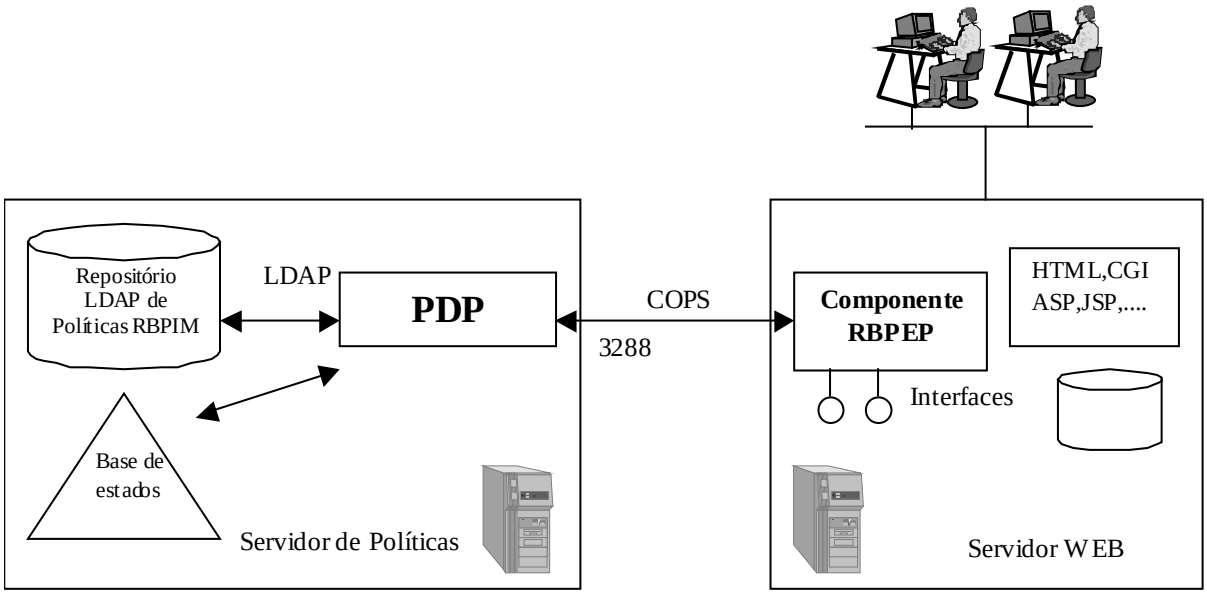

Figura 6.5 – Cenário típico da arquitetura do RBPIM

Este trabalho define um string com formato <*PEPID*>_<*Seqüência*> para o *SessionID* utilizado nas chamadas COPS, onde <*PEPID*> identifica o RBPEP junto ao PDP enquanto <*Seqüência*> é um valor numérico, iniciado por 1, que representa a seqüência de sessões abertas pelo RBPEP. O *framework* do COPS indica que o PEP deve manter uma conexão TCP para cada tipo de cliente. Como este trabalho aborda apenas um tipo de cliente (RBPEP *client*), múltiplos clientes (múltiplas sessões) compartilharão a mesma conexão TCP para interagir com o servidor de políticas.

Cada chamada retorna um valor RBPEP_Error para indicar o resultado de sua execução. Este trabalho estende o conjunto de mensagens definidas na RFC 2748 para atender as especificidades do serviço de política de RBPIM, definindo objetos *COPS* do tipo *error* com valor do campo <*ERROR CODE*> = 16 para indicar mensagens de erro padronizadas a clientes RBPIM, onde os valores do campo <*ERROR SUB_CODE*> do mesmo objeto *error* definem as mensagens propriamente ditas.

Neste trabalho, as mensagens contidas nos objetos *error* enviados pelo PDP ao RBPEP são repassadas à aplicação no retorno da chamada, com a correspondência dos valores <*ERROR SUB_CODE*> dos objetos COPS *error* aos valores *RBPEP_Error* retornados. Os valores encontram-se definidos no Anexo B deste trabalho. Os parâmetros passados ao RBPEP e retornados à aplicação estão mostrados em cada chamada, respectivamente, como

uma lista de valores ***in***: e ***out:***. A seguir, as subseções de 6.4.1 a 6.4.5 apresentam as chamadas da API do RBPEP.

6.4.1 Abertura do Serviço de Política

Chamada:	RBPEP_Error **RBPEP_Open**()
Descrição:	Inicia a conexão com o serviço de política RBPIM. Faz com que o RBPEP solicite a abertura de um serviço de política junto ao PDP para clientes RBPEP.
Retorno:	Consultar Anexo B.
Parâmetro:	Nenhum.

Na Figura 6.6, inicialmente, a aplicação solicita (1) um pedido de conexão com o serviço de políticas RBPIM, fazendo com que o RBPEP (2) estabeleça uma conexão TCP com o PDP. Uma vez que a conexão tenha sido estabelecida, o RBPEP (3) emite uma mensagem indicando que ele deseja abrir um serviço de política para clientes RBPEP (consumidores de políticas RBPIM). Neste instante, o PDP pode utilizar o PEPID recebido na mensagem OPN para verificar se o RBPEP está entre aqueles que ele pode prover o serviço. Caso afirmativo, ele (4) responde positivamente e passa a aguardar novas requisições de políticas para este tipo de cliente. Caso contrário, (5) ele notifica o RBPEP que não pode prover tal serviço. Por fim, a aplicação é notificada pelo RBPEP sobre o status da solicitação de abertura do serviço.

RBPEP_Open

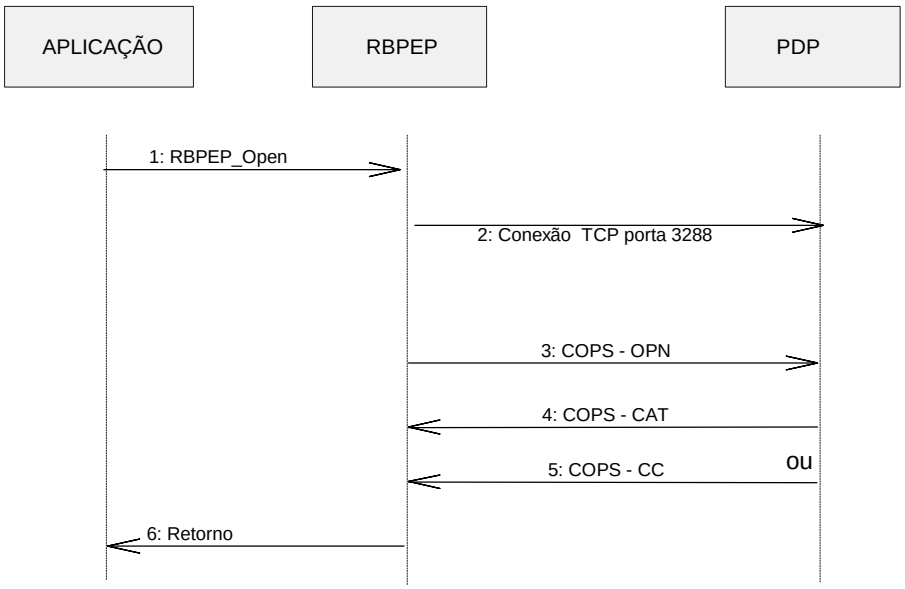

Figura 6.6 - Diagrama de seqüência da chamada RBPEP_Open

Cada uma das mensagens COPS possui a seguinte estrutura:

3:	**OPN:**	CH(1,0,6,0x8000,length) + OE(length,11,1,PEPID)
4:	**CAT:**	CH(1,0,7,0x8000,length) + OE(length,10,1,0)
5:	**CC:**	CH(1,0,8,0x8000,length) + OE(length,8,1,ErrorCode+Sub-Code)

CH: Common Header OE: Objeto Específico COPS (Ver tabelas 6.4,6.5 e 6.6)

6.4.2 Abertura de Sessão RBPIM (Abertura de Sessão Fase 1)

Chamada:	RBPEP_Error **RBPEP_CreateSession**(***in:*** *String* UserID; ***out:*** *Int* SessionQuant, *String* SessionID, *String* RoleSet[ ])
Descrição:	Cria uma sessão de usuário.
Retorno:	Consultar Anexo B.
Parâmetro:	**UserID** Identifica o usuário de forma unívoca no contexto do repositório de políticas. Presume-se que este valor tenha sido obtido a partir de um procedimento anterior de autenticação do usuário em questão. **SessionQuant** Indica o número de sessões ativas do usuário. **SessionID** Valor de retorno gerado pelo RBPEP que identifica a sessão do usuário de forma unívoca no contexto da conexão com o PDP. SessionID=<*String* PEPID>+<*Int* Sequência>. A cada nova sessão criada pelo RBPEP o valor inteiro Sequência é incrementado de um. **RoleSet[ ]** Valor de retorno que indica a lista de papéis elegíveis para o usuário.

A abertura de uma sessão RBPIM é composta por duas fases. Na primeira, realizada pela chamada RBPEP_CreateSession, a aplicação solicita a criação da sessão informando o usuário envolvido. Ao final desta fase, a aplicação recebe uma lista contendo os papéis elegíveis ao usuário. Na segunda fase, realizada pela chamada RBPEP_SelectedRoles, a aplicação envia ao PDP os papéis selecionados pelo usuário, momento no qual o PDP finaliza a abertura da sessão.

Cada usuário pode abrir múltiplas sessões RBPIM. O valor contido em UserID deve identificar um determinado usuário no contexto da informação de política contida no repositório. Este trabalho assume que o UserID é obtido a partir de um processo de autenticação realizado antes da chamada da API. A separação deste processo do *framework* proposto neste trabalho é importante, uma vez que permite que o administrador opte pela maneira mais adequada ao seu ambiente operacional. Por exemplo, a autenticação poderia ser realizada através de uma aplicação existente ou através de serviços externos como a autenticação LDAP [RFC 1777] e o Kerberos [RFC 1510]. Estes mecanismos devem

mapear a autenticação a um atributo do objeto do diretório que representa o usuário. Uma estratégia interessante, e que foi adotada neste trabalho, consiste em mapear este processo ao atributo *CN* (*CommonName*) de objetos *Person* (ver figura 4.8 do Capítulo 4) contidos no repositório de políticas. Neste caso, o UserID usado na chamada conteria o *CN* do usuário, o que permitiria, facilmente, que informações adicionais sobre ele sejam obtidas.

Na Figura 6.7, ao receber uma chamada para abertura (1) de uma sessão de usuário, o RBPEP gera o SessionID que identificará a sessão e envia uma mensagem (2) REQ ao PDP solicitando que seja alocado e mantido um estado (sessão) para o usuário indicado por UserID.

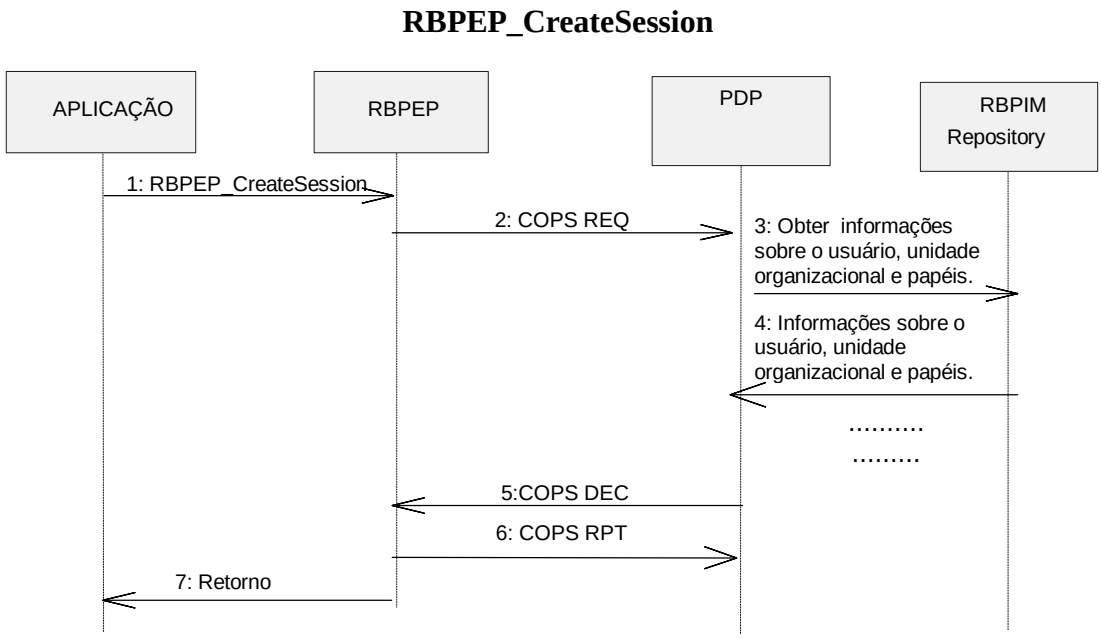

Figura 6.7 - Diagrama de seqüência da chamada RBPEP_CreateSession

Em seguida, o PDP (3) acessa o repositório de política onde obtém (4) informações sobre os papéis e usuários associados. Após uma filtragem, apenas os papéis autorizados para o usuário em questão são retornados ao RBPEP através uma mensagem (5) DEC, juntamente com a quantidade de sessões abertas previamente pelo usuário. Ao receber esta mensagem, o RBPEP (6) notifica o PDP sobre o recebimento da decisão. Assim, o PDP pode efetivar esta fase da criação da sessão com a inserção de uma entrada na Base de Estados contendo o UserID e o SessionID. Por fim, a aplicação recebe o (7) retorno da chamada da API juntamente com o ID da sessão, a quantidade de sessões abertas e a lista de papéis autorizados. A estrutura de cada mensagem COPS é mostrada a seguir:

2:	**REQ:**	CH(1,0,1,0x8000,length) + OE(length,1,1,SessionID) + OE(length,2,1,R-TYPE+M-TYPE) + OE(length,9,2,UserID)
5:	**DEC:**	CH(1,1,2,0x8000,length) + OE(length,1,1,SessionID) + OE(length,2,1, R-TYPE+M-TYPE) + OE(length,6,1,Command-Code) | OE(length,8,1,ErrorCode+Sub-Code)+ [OE(length,6,4,SessionQuant)+OE(length,6,4,RoleSet)]
6:	**RPT:**	CH(1,1,3,0x8000,length) + OE(length,1,1,SessionID) + OE(length,12,1, Report-Type)

CH: Common Header OE: Objeto Específico COPS (Ver tabelas 6.8, 6.9 e 6.10)

A mensagem REQ transporta, além do seu cabeçalho e do objeto *client-handle*(SessionID), um objeto *context* e um objeto contento a informação específica do cliente(*ClientSI*). O campo R-TYPE do objeto *context* indica o evento que originou a chamada e seus valores são padronizados pela [RFC 2748]. Já o campo M-TYPE, do mesmo objeto, permite a definição de informações específicas a um tipo de cliente, ficando a cargo de trabalhos subseqüentes sua padronização. Assim, este trabalho define o valor *2* para R-TYPE, o qual indica um pedido de alocação de recursos e o valor 1 para M-TYPE para indicar um pedido de abertura de sessão RBPIM (veja o Anexo B). Ambos valores serão utilizados em chamadas REQ de abertura de sessão fase 1. O *ClientSI* tem tamanho livre e transporta, nesta mensagem, o UserID provido pela aplicação ao RBPEP, o qual foi obtido no processo de autenticação do usuário realizado anteriormente a esta chamada.

Toda mensagem DEC possui um objeto *context* contendo a mesma informação do objeto *context* da mensagem REQ associada. Ainda, a mensagem DEC pode retornar o resultado da decisão ou, eventualmente, um objeto transportando a informação sobre um erro ocorrido. O campo *Command-Code*, contido no objeto que transporta a decisão de política, é utilizado neste trabalho com os seguintes valores: 0-Nenhuma decisão disponível; 1-Requisição aceita; e 2-Requisição negada. Por fim, caso a decisão tenha sido positiva (valor 1), a quantidade de sessões previamente assumidas pelo usuário e a lista de papéis elegíveis são retornadas em dois objetos do tipo *Decision*. Uma questão importante durante a criação de uma sessão RBAC é a separação estática(*SSD*) e dinâmica(*DSD*) de tarefas entre papéis. Nesta chamada da API, o PDP retorna a lista de papéis elegíveis para a sessão de usuário. Ao apurar esta lista, o PDP verifica os possíveis conflitos de separação estática de tarefas entre papéis, estendendo esta checagem até os papéis de níveis inferiores, eventualmente herdados por papéis presentes na lista. A lista resultante, que é enviada ao RBPEP, compreende um conjunto de papéis livres de conflitos por *SSD*. A aplicação, ao receber esta lista de papéis do RBPEP, deve propiciar funcionalidades ao usuário de tal forma que ele possa selecionar quais

papéis ele deseja assumir naquela sessão. Após a seleção dos papéis por parte do usuário, a aplicação realiza uma segunda chamada (veja a subseção a seguir) onde o PDP é informado sobre os papéis selecionados.

Na mensagem RPT, o campo *Report-Type* permite que o PDP seja informado sobre o recebimento da mensagem DEC pelo RBPEP. O PDP efetiva as decisões enviadas ao RBPEP apenas após o recebimento de uma mensagem RPT positiva. Neste trabalho, os seguintes valores foram convencionados: 1-Sucesso e 2-Falha.

6.4.3 Seleção de Papéis para a Sessão RBPIM (Abertura de Sessão Fase 2)

Chamada:	RBPEP_Error **RBPEP_SelectedRoles**(***in:*** *String* SessionID, *String* SelectedRoleSet[ ]; ***out:*** *Boolean* Result)
Descrição:	Chamada que completa a abertura de sessão e permite o RBPEP informar ao PDP os papéis selecionados pelo usuário para uma dada sessão.
Retorno:	Consultar Anexo B.
Parâmetro:	**SessionID** Valor que identifica a sessão do usuário. **SelectedRoleSet[ ]** Indica a lista de papéis selecionados pelo usuário. **Result** [TRUE | FALSE] Resultado da alocação dos papéis.

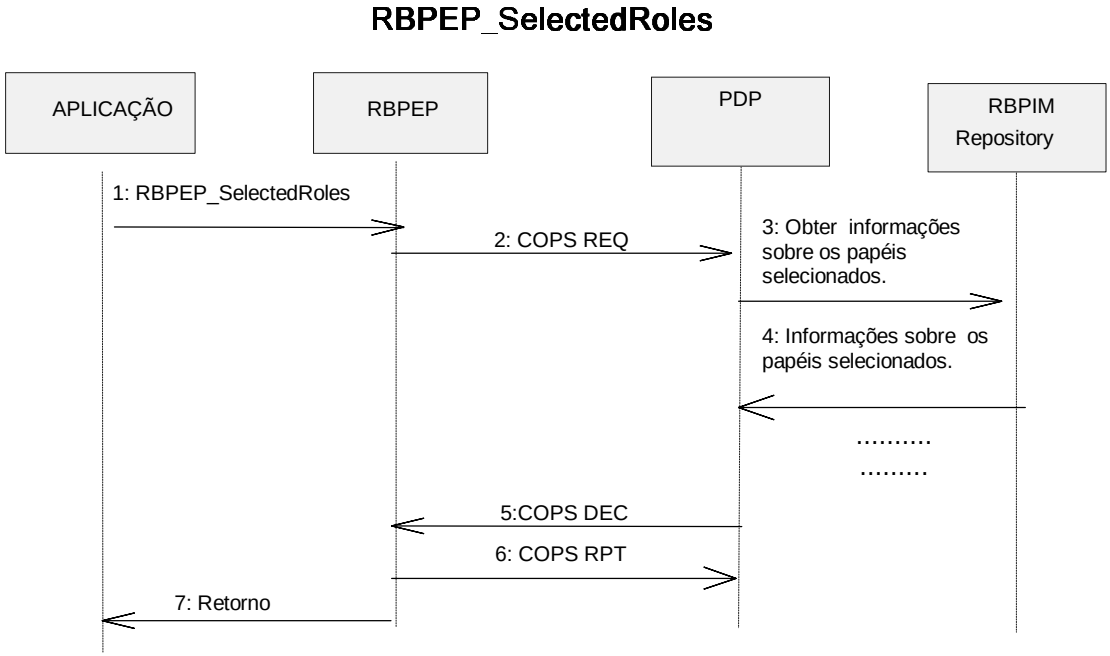

Figura 6.8 - Diagrama de seqüência da chamada RBPEP_SelectedRoles

Após o usuário selecionar os papéis para a sessão, a aplicação realiza uma chamada(1) informando a lista de papéis selecionados. Ao receber esta chamada, o RBPEP envia uma mensagem (2) REQ ao PDP solicitando que este conjunto de papéis seja ativado para a sessão. O PDP consulta o (3)(4) repositório de políticas RBPIM onde obtém informações sobre os papéis selecionados. Entre outras avaliações, o PDP verifica se há conflito por DSD entre eles. Após a verificação, a decisão do PDP é enviada ao RBPEP através uma mensagem (5) DEC. Ao receber esta mensagem, o RBPEP (6) notifica o PDP sobre o recebimento da decisão. Na seqüência, caso nenhum conflito tenha ocorrido entre os papéis, o PDP pode finalizar a ativação da sessão iniciada na chamada *RBPEP_CreateSession*, atualizando a entrada já existente na Base de Estados para a sessão indicada pelo SessionID com a lista de papéis selecionados. Por fim, a aplicação recebe o (7) o *status* da chamada juntamente com o resultado da alocação dos papéis selecionados [*TRUE*|*FALSE*]. A estrutura de cada mensagem é mostrada a seguir:

2:	**REQ:**	CH(1,0,1,0x8000,length) + OE(length,1,1,SessionID) + OE(length,2,1,R-TYPE+M-TYPE) + OE(length,9,2,SelectedRoleSet)
5:	**DEC:**	CH(2,1,1,0x8000,length) + OE(length,1,1,SessionID) + OE(length,2,1, R-TYPE+M-TYPE) + OE(length,6,1,Command-Code) | OE(length,8,1,ErrorCode+Sub-Code)
6:	**RPT:**	CH(3,1,1,0x8000,length) + OE(length,1,1,SessionID) + OE(length,12,1, Report-Type)

CH: Common Header OE: Objeto Específico COPS (Ver tabelas 6.8, 6.9 e 6.10)

A composição das mensagens associadas a esta chamada é semelhante às utilizadas na chamada RBPEP_CreateSession. Nesta chamada, o objeto *context* da mensagem REQ transporta o valor *0x02* para M-TYPE para indicar seleção de papéis para ativação da sessão RBPIM. O objeto *ClientSI* da mensagem REQ transporta a lista de papéis selecionados pelo usuário. Caso o PDP tenha descoberto um conflito por DSD na ativação dos papéis selecionados, o campo *Command-Code*, contido no objeto que transporta a decisão de política da mensagem DEC, conterá valor 2, indicando que a requisição foi negada.

6.4.4 Fechamento de Sessão RBPIM

Chamada:	RBPEP_Error **RBPEP_CloseSession**(***in:*** *String* SessionID)
Descrição:	Fecha uma sessão de usuário.
Retorno:	Consultar Anexo B.
Parâmetro:	**SessionID** Valor que identifica a sessão do usuário.

No fechamento de uma sessão RBPIM, todas as informações relativas à sessão indicada por SessionID devem ser removidas. Dada a natureza *stateful* do COPS, é importante que, quando um estado (sessão) é removido do RBPEP, uma mensagem DRQ seja enviada ao PDP para que o correspondente estado possa ser da mesma forma removido[RFC 2748]. Assim, após a solicitação da aplicação, o RBPEP remove todas as informações locais referente à sessão e envia uma mensagem DRQ ao PDP para que ele remova as informações correspondentes que estão sendo mantidas na Base de Estados. Esta mensagem tem a seguinte estrutura:

2:	**DRQ:**	CH(4,0,1,0x8000,length) + OE(length,1,1,SessionID) +[OE(length,5,1, Reason Code + Sub-Code)]

CH: Common Header OE: Objeto Específico COPS (Ver tabela 6.7)

RBPEP_CloseSession

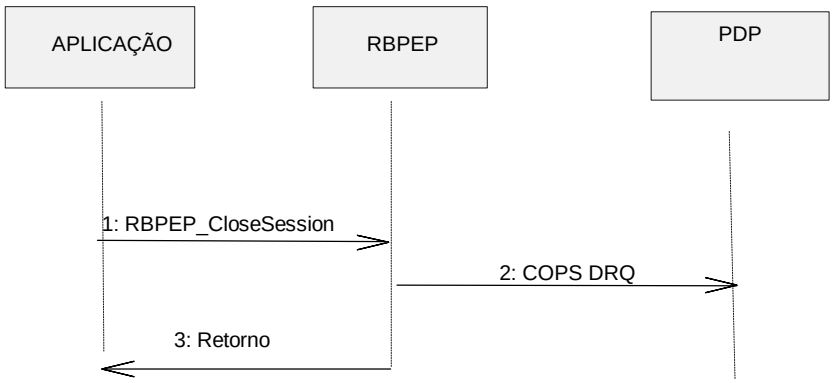

Figura 6.9 - Diagrama de seqüência da chamada RBPEP_CloseSession

Como nas outras mensagens, o SessionID é transportado em um objeto *client-handle*. Um objeto *reason* pode especificar o motivo pelo qual o estado (a sessão) está sendo removido. Para uma relação completa dos valores *Reason-Code+Sub-Code* , consulte [RFC 2748].

6.4.5 Verificação de Acesso a Objetos RBPIM

Chamada:	RBPEP_Error **RBPEP_CheckAccess**(***in:*** *String* SessionID, *String* Operation, *String* ObjectInfo[ ]; ***out:*** *Boolean* Result)
Descrição:	Verifica a possibilidade do usuário associado à sessão realizar uma operação em objetos submetidos ao controle de acesso efetuado pelo RBPIM.
Retorno:	Consultar Anexo B.

Parâmetro:	**SessionID** Valor que identifica a sessão do usuário. **Operation** Valor que identifica a operação requerida no contexto da informação de política. **ObjectInfo[ ]** Conjunto de valores que descreve quais objetos cujo acesso está sendo solicitado. **Result** [TRUE | FALSE] Resultado da verificação do acesso.

A aplicação efetua a chamada da API provendo ao RBPEP o *SessionID* da sessão, a operação requerida (*Operation*) e um array (*ObjectInfo*) contendo as informações que descrevem o(s) objeto(s) cujo acesso é controlado segundo o *framework* do RBPIM. Uma operação (*Operation)* é um procedimento executável de um programa, cuja invocação realiza alguma função para o usuário. Os tipos de operações e objetos que o RBAC controla são dependentes do tipo de sistema no qual ele será implementado. Por exemplo, em um sistema de arquivos, operações podem incluir *ler, escrever* e *executar*; em um sistema gerenciador de banco de dados, operações podem incluir *inserir, excluir* e *atualizar* [SANDHU 00]. Nesta linha, o modelo proposto por este trabalho dá ao administrador total flexibilidade à especificação de operações, as quais, obviamente, devem possuir significado inerente ao objeto especificado por *ObjectInfo*. Como detalhado no Capítulo 4, os objetos controlados pelo RBPIM possuem sua representação segundo as classes definidas pelos diversos *Cim Schemas*. Assim, é possível definir em um *ObjectInfo* informações que especificam um arquivo, um diretório, um usuário ou uma aplicação, por exemplo. Cabe ao administrador, portanto, definir operações relacionadas a cada tipo de objeto em específico.

O esquema de diretório resultante do mapeamento do RBPIM no LDAP, mostrado no Capítulo 5, definiu dois tipos de variáveis para serem utilizadas na composição de regras de políticas: *rbimPolicyExplicitVariable* e *rbimPolicyImplicitVariable*. A primeira especifica dois atributos que possibilitam a definição do par classe/atributo que identifica objetos descritos segundo as especificações dos modelos do CIM. A segunda é utilizada para que condições de políticas possam referenciar informações não mapeadas diretamente pelo CIM como endereços IP, endereços MAC e número de portas. Assim, este trabalho define a seguinte sintaxe para a informação contida no array *ObjectInfo*:

1) *<Nome da ObjectClass do CIM>.< Nome da propriedade do CIM >=<Valor>*
2) *[rbpimPolicySourceIPv4Var | rbpimPolicyDestIPv4Var | rbpimPolicySourceIpv6Var | rbpimPolicyDestIPv6Var | rbpimPolicySourcePortVar | rbpimPolicyDestinationPortVar | rbpimPolicySourceMACVar | rbpimPolicyDestMACVar | rbpimPolicyIPProtocolVar = <Valor>]*

Por exemplo, um diretório poderia ser especificado como *"Directory.Name=/usr/application"* e um arquivo de dados como *"DataFile.Name=financeiro.mdb"*. As classes *Directory* e *DataFile* estão definidas no *CIM_System Schema,* conforme mostrou a figura 4.10 do Capítulo 4. O segundo formato permite que a aplicação forneça ao RBPEP informações dos pacotes relacionados ao acesso requerido. Estas informações são necessárias quando o PDP precisa verificar o acesso a um recurso cujas regras de política fazem referência a elas. Neste caso, *Source* (origem) refere-se ao sistema que está requerendo o acesso enquanto *Dst/Destination* (destino) ao sistema onde se encontra o recurso a ser acessado. Este trabalho assume que a aplicação que utiliza as facilidades da arquitetura possui mecanismos suficientes para prover estas informações ao RBPEP. Por exemplo, juntamente com o diretório citado no exemplo acima, poderiam existir outras entradas em *ObjectInfo* como *rbpimPolicySourceIPv4Var=192.168.2.3* e *rbpimPolicySourcePortVar =1034*, indicando que a máquina que está solicitando o acesso possui o endereço IP 192.168.2.3 e que o processo é identificado pela porta 1034. Assim, caso as *PolicyConditions* que definem os recursos associados a uma permissão RBPIM também restringirem o acesso a determinadas máquinas e processos, o PDP pode verificar se os valores supridos na chamada da API estão contidos na faixa de endereços IPs e portas indicados nas próprias *PolicyConditions* das permissões.

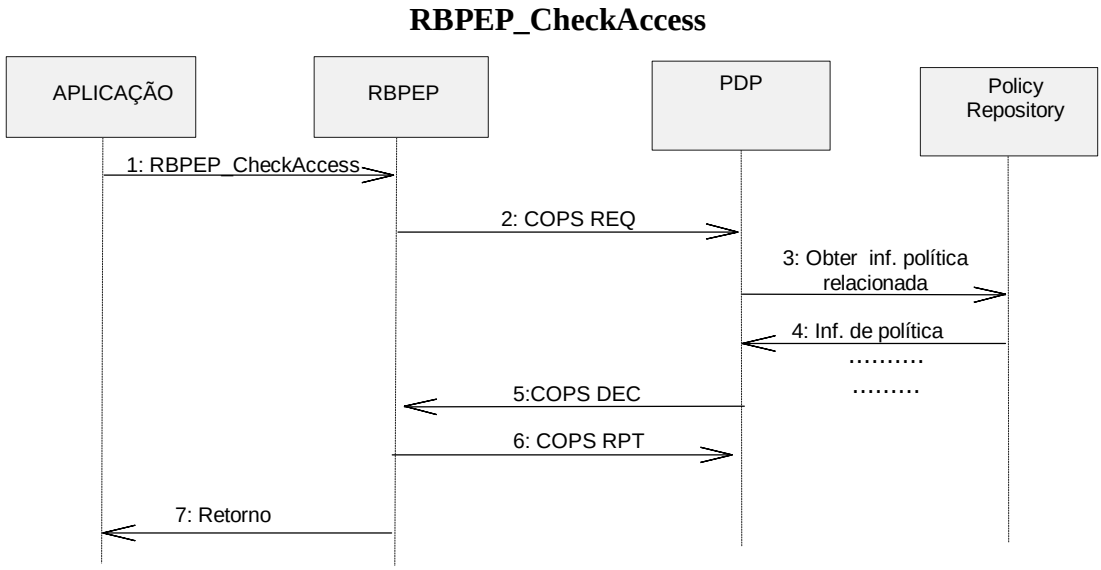

Figura 6.10 - Diagrama de seqüência da chamada RBPEP_CheckAccess

Na Figura 6.10, ao receber uma (1) chamada para verificar a execução da operação em um objeto no contexto da sessão identificada por SessionID, o RBPEP envia uma mensagem (2) REQ ao PDP solicitando que tal verificação seja efetuada. A partir da informação mantida

na Base Estados para a sessão identificada por SessionID, o PDP (3)(4) passa a acessar o repositório onde obtém a informação de política relevante à requisição, como será detalhado na próxima seção. Após a verificação, o PDP retorna sua decisão ao RBPEP através de uma mensagem (5) DEC. Ao receber esta mensagem, o RBPEP (6) notifica o PDP sobre seu recebimento. Por fim, a aplicação recebe o (7) *status* da chamada juntamente com o resultado da verificação do acesso[*TRUE*|*FALSE*]. A estrutura de cada mensagem é mostrada a seguir:

2:	**REQ:**	CH(1,0,1,0x8000,length) + OE(length,1,1,SessionID) + OE(length,2,1,R-TYPE+M-TYPE) + OE(length,9,2,Operation) + OE(length,9,2,ObjectInfo_Explicit) + [OE(length,9,2,ObjectInfo_Implicit)]
5:	**DEC:**	CH(2,1,1,0x8000,length) + OE(length,1,1,SessionID) + OE(length,2,1, R-TYPE+M-TYPE) + OE(length,6,1,Command-Code) | OE(length,8,1,ErrorCode+Sub-Code)
6:	**RPT:**	CH(3,1,1,0x8000,length) + OE(length,1,1,SessionID) + OE(length,12,1, Report-Type)

CH: Common Header OE: Objeto Específico COPS (Ver tabelas 6.8, 6.9 e 6.10)

A mensagem REQ transporta, além do seu cabeçalho e do objeto *client-handle* com o SessionID, um objeto *context* e dois objetos contento a informação específica do cliente(*ClientSI*). Nesta chamada, assim como na chamada de criação de sessão, este trabalho define o valor *0x02* para o campo R-TYPE do objeto *context*, o qual indica um pedido de alocação de recursos, e *0x03* para M-TYPE (Ver Anexo B), o qual indica um pedido de verificação de execução de uma operação em objetos controlados pelo RBPIM. Os dois *ClientSI* transportam, respectivamente, a operação e a informação sobre os objetos referenciados. A mensagem DEC associada a esta chamada é semelhante a aquela utilizada na chamada de abertura de sessão, com o campo *Command-Code* do objeto *Decision* podendo assumir os valores: 0-Nenhuma decisão disponível; 1-Requisição autorizada; e 2-Requisição negada. A mensagem RPT informa o PDP sobre o recebimento da mensagem DEC recebida pelo RBPEP.

6.5. Algoritmos para Implementação do PDP

O propósito desta seção é definir os algoritmos necessários à implementação do PDP referentes aos procedimentos inerentes às chamadas da API definidas na seção anterior.

Um procedimento importante quando se utiliza um serviço de diretório como repositório de informação é a construção de sua DIT, ou seja, a definição da forma na qual os objetos criados a partir do esquema de diretório serão organizados. Este procedimento tem um impacto importante no acesso à informação contida no diretório, uma vez que, se elaborado de maneira errônea, pode tornar este processo complexo e custoso. Este trabalho propõe a

hierarquia de objetos mostrada na figura 6.11 com intuito de organizar as entradas necessárias à descrição da informação de política RBPIM. Com exceção das classes estruturais que representam a informação referente ao relacionamento de separação de tarefas entre papéis[13], todas classes estruturais do esquema do RBPIM estão sendo mostradas juntamente com seu atributo RDN. Os algoritmos do PDP, descritos no decorrer desta seção, tomam como base esta hierarquia de diretório.

Tipicamente, as informações estão contidas em um diretório a partir de uma entrada que representa uma dada organização (atributo RDN **o**). Dentro desta entrada existem unidades organizacionais (atributo RDN **ou**), as quais permitem agrupamentos lógicos de entradas em função de suas representatividades dentro da organização.

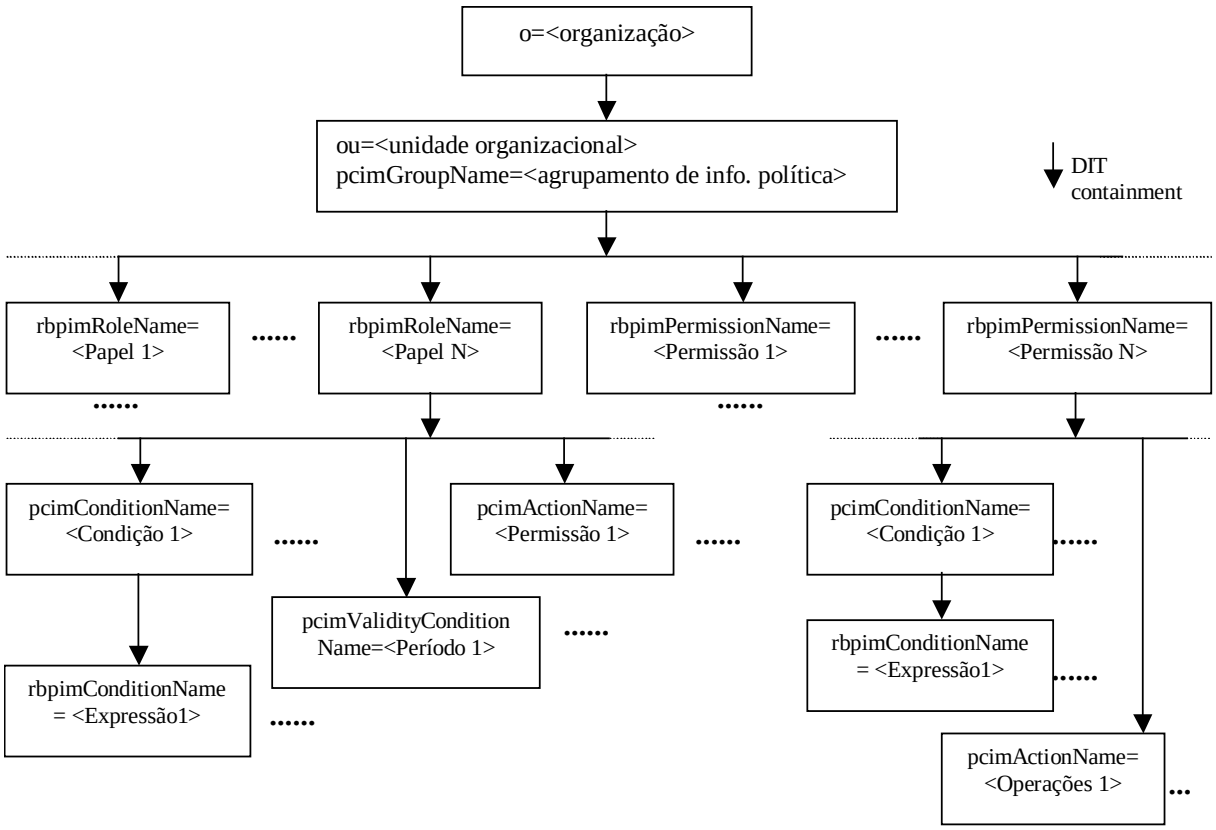

Figura 6.11 – Entradas do diretório contendo a informação de política RBPIM

Por exemplo, pode existir uma unidade organizacional para cada departamento da organização. Nesta linha, este trabalho define que entradas referentes às unidades organizacionais serão *containers* para entradas referente a papéis, permissões e relacionamentos de separação de tarefas. Seguindo o esquema do RBPIM, isto é conseguido

[13] Apesar de não mostradas nesta figura, elas são inseridas no mesmo nível das entradas referentes aos papéis e permissões.

com o *attachment* da classe auxiliar *pcimGroupAuxClass* à entrada da unidade organizacional, o que a transforma em um *PolicyGroup*. Ainda, ela pode receber o *attachment* das classes *pcimRuleContainmentAuxClass*, *rbpimDSDContainmentAuxClass* e *rbpimSSDContainmentAuxClass*, as quais fornecem os atributos necessários para a especificação das referências DNs aos objetos contidos. Segundo a estratégia adotada na organização da DIT proposta neste trabalho, estas três últimas classes não se fazem necessárias, uma vez que todas entradas de papéis, permissões e relacionamentos de separação de tarefas estão hierarquicamente localizados logo abaixo da unidade organizacional, o que permite que elas sejam facilmente acessadas. Vale ainda ressaltar que o PCIMe introduz um novo formato para agrupamento de regras, com a especificação do conceito de conjuntos de políticas (*Policy Sets*) e de encadeamento de regras (*Nested Policy Rules*), as quais, em função da época da proposição deste trabalho, foi deixado para implementação em um trabalho futuro.

6.5.1 Algoritmo principal do PDP

Enquanto (*verdade*)
    Receber mensagem COPS
    Verificar tipo da mensagem
        caso OPN: Chamar Algoritmo do PDP para a mensagem OPN
        caso REQ: Chamar Algoritmo do PDP para a mensagem REQ
        caso RPT: Chamar Algoritmo do PDP para a mensagem RPT
        caso DRQ: Chamar Algoritmo do PDP para a mensagem DRQ
        caso CC: Chamar Algoritmo do PDP para a mensagem CC
    Fim.
Fim_Enquanto..

6.5.2 Algoritmo do PDP para a mensagem OPN

Verificar se o tipo de cliente é suportado (RBPEP *Client* 0x8000).
Se o tipo de cliente é suportado *então*:
    Verificar se o PEPID contido na mensagem pertence à lista de RBPEPs autorizados para utilizar o serviço de política RBPIM.
    Se o RBPEP está autorizado a utilizar o serviço *então*:
        Verificar se não existe um serviço aberto para o RBPEP indicado pelo PEPID.
        Se NÃO existe o serviço aberto *então*:
            Retornar mensagem CAT ao RBPEP indicando o sucesso da abertura do serviço.
        Se o serviço já está aberto *então*:
            Retornar mensagem CC ao RBPEP contendo um objeto ERROR indicando que o serviço já está aberto.
        Fim.
    Se o RBPEP NÃO está autorizado a utilizar o serviço *então*:
        Retornar mensagem CC ao RBPEP contendo um objeto ERROR indicando que o RBPEP não está autorizado a utilizar o serviço de política.
    Fim.

Se o RBPEP NÃO é suportado *então*:
Retornar mensagem CC ao RBPEP contendo um objeto ERROR indicando que o tipo de cliente não é suportado pelo serviço de política.
Fim.

6.5.3 Algoritmo do PDP para a mensagem REQ

Verificar se o serviço de políticas está aberto para o RBPEP indicado pelo PEPID contido no SessionID.
Se o serviço de políticas está aberto *então:*
Verificar o tipo da requisição de política indicada por R-TYPE e M-TYPE da mensagem
caso RBPEP_CreateSession: Chamar Algoritmo do PDP para RBPEP_CreateSession
caso RBPEP_SelectedRoles: Chamar Algoritmo do PDP para RBPEP_SelectedRoles
caso RBPEP_CheckAccess: Chamar Algoritmo do PDP para RBPEP_CheckAccess
caso contrário: Retornar mensagem DEC ao RBPEP contendo um objeto ERROR indicando que o PDP não possui informação de política relacionada com a requisição.
Fim.
Se o serviço NÃO está aberto *então*:
Retornar mensagem DEC ao RBPEP contendo um objeto ERROR indicando que o RBPEP não possui serviço de política aberto.
Fim.

6.5.3.1 Algoritmo do PDP para RBPEP_CreateSession (Fase 1 da Criação da Sessão)

Este algoritmo determina os papéis RBAC que irão constituir a lista de papéis elegíveis do usuário. Esta lista será composta de papéis que possuem períodos de ativação válidos para a data e hora correntes e que estão livres de conflitos por SSD. Após a apresentação do algoritmo, serão apresentadas as consultas LDAP necessárias para a obtenção dos objetos que contêm a informação de política relacionada com os procedimentos associados com cada passo.

<u>*Passo 1:*</u> Se já existe uma sessão ativa na Base de Estados indicada pelo SessionID retornar um objeto COPS <Error> através da mensagem DEC. Caso contrário, ir para o *Passo* 2.

<u>*Passo 2:*</u> *Determinar* $\mathbf{R}=\{R_i\}_{i=1,..,n1}$ como sendo a lista de papéis (objetos *rbpimRole*) associados à unidade organizacional (*ou*) do usuário.

<u>*Passo 3:*</u> *Determinar* $\mathbf{C}_i=\{C_{ij}\}_{j=1,..,n2}$ como sendo a lista de condições (objetos *pcimRuleConditionAssociation*) associadas ao papel R_i.

<u>*Passo 4:*</u> *Determinar* $\mathbf{U}_i=\{U_{ik}\}_{k=1,..,n3}$ como sendo a lista CNs (*Common Name*) dos objetos que representam usuários que satisfazem às condições $\mathbf{C_i}$ associadas ao papel R_i.

<u>*Passo 5:*</u> *Determinar* $\mathbf{R^a}=\{R_j\}_{j\leq n1}$ como um subconjunto de **R** onde o UserID suprido pelo RBPEP na mensagem REQ $\in \mathbf{U}_i$.

<u>*Passo 6:*</u> *Determinar* $\mathbf{R^b}=\{R_m\}_{m\leq n1}$ como um subconjunto de $\mathbf{R^a}$ que inclui os papéis que

satisfazem às restrições determinadas pelos períodos de ativação indicados pelos objetos *pcimRuleValidityAssociation* associados.

Passo 7: *Determinar* $\mathbf{R^c}=\{R_n\}_{n=1,..,n4}$ como sendo a lista de papéis herdados indicados pelo atributo *rbpimInheritedRoles* de todos os objetos *rbpimRole* $\in \mathbf{R^b}$.

Passo 8: *Determinar* $\mathbf{R^d} =\{R_p\}_{p\leq n4}$ como sendo um subconjunto de $\mathbf{R^c}$ que inclui os papéis que satisfazem às restrições determinadas pelos períodos de ativação indicados pelos objetos *pcimRuleValidityAssociation* associados.

Passo 9: *Determinar* $\mathbf{R^e} =\{R_o\}_{o=1,..,n5}$ como $\mathbf{R^b} \cup^* \mathbf{R^d}$ (União disjunta).

Passo 10: *Determinar* $\mathbf{SSD}=\{SSD_p\}_{p=1,...,n6}$ como sendo a lista de objetos *rbpimSSD* associados aos papéis contidos em $\mathbf{R^e}$.

Passo 11: *Determinar* $\mathbf{R^f} =\{R_q\}_{q\leq n5}$ removendo de $\mathbf{R^e}$ os papéis conflitantes em **SSD**. Os papéis com menor prioridade (atributo *pcimRulePriority* herdado por *rbpimRole* da classe *pcimPolicyRule*) são removidos primeiro, até que as cardinalidades dos conjuntos sejam satisfeitas.

Passo 12: Criar na Base de Estados com *Status* = Fase 1 um registro com o SessionID, o UserID e a lista de papéis definida por $\mathbf{R^f}$. Em seguida, enviar uma mensagem DEC ao RBPEP com esta lista de papéis e o número de sessões abertas pelo usuário encapsulados em objetos COPS <Decision>.

a) Passo 2:

Consulta 1: Obter a unidade organizacional (*ou*) do usuário

O UserID indica o valor *CN* (*Common Name*) do objeto contido no diretório que representa o usuário. Este objeto é definido pela hierarquia de classes do *CIM_User Schema*, mostrado na Figura 4.8 do Capítulo 4. A partir da figura 6.11, assumindo que exista uma unidade organizacional (ou=PEOPLE) dentro do *folder* o=<organização> para o armazenamento da informação referente às pessoas da organização, esta consulta LDAP será:

Base_dn:	ou=People, o=<organização>
Atributos:	Ou
Escopo:	One
Filtro:	(&(objectClass=inetorgperson)(cn=<UserID>))

Consulta 2: Obter lista de todos os papéis contidos na unidade organizacional

Representando por <ou_usuário> o valor retornado pela consulta 1, tem-se a consulta LDAP mostrada abaixo para obtenção dos papéis habilitados. Note que, como não foram especificados os atributos, todos os atributos dos objetos em questão serão retornados.

Base_dn:	ou=<ou_usuário>, o=<organização>
Atributos:	
Escopo:	one
Filtro:	(&(objectClass=rbpimRole)(pcimRuleEnabled =1))

b) Passo 3:

Consulta 3: Obter a lista de condições associadas ao papel

A Consulta 2 permitiu que todos as atributos associados a um determinado papel fossem retornados. Entre eles, o atributo *pcimRuleConditionList* fornece uma lista de referências *DNs* a objetos de condições que indicarão quais usuários podem assumir o papel (*objectclass=pcimRuleConditionAssociation+rbpimSimplePolicyConditionAuxClass*). Cada objeto de condição possui apenas uma entrada (*objectclass=rbpimConditionAssociation*), a qual indica o par <*PolicyVariable*>,<*PolicyValue*> associado à condição.

Representando cada uma destas referências *DN* por <*CondiçãoDN*>, tem-se a consulta LDAP mostrada abaixo. Neste caso, em função do escopo *sub*, os atributos do objeto indicado pela Base_dn e do objeto de expressão (*PolicyVariable+PolicyValue*) serão retornados.

Base_dn:	<CondiçãoDN>
Atributos:	pcimConditionGroupNumber,pcimConditionNegated,rbpimModelClass, rbpimModelProperty,rbpimStringList
Escopo:	sub
Filtro:	

c) Passo 4:

Consulta 4: Obter o cn (*CommonName*) dos usuários selecionados pela lista de condições

O atributo *pcimRuleConditionListType*, retornado na Consulta 2, informa como as várias condições associadas ao papel, que foram obtidas pela Consulta 3, serão agrupadas (DNF ou CNF), levando-se em conta os atributos *pcimConditionGroupNumber* e

pcimConditionNegated de cada uma delas. Todas as condições associadas a um papel são especificadas através dos atributos dos objetos que representam os usuários no repositório de política. Assim, para a realização desta consulta, faz-se necessário representar a expressão lógica destas condições no formato de uma expressão usada no filtro da consulta LDAP. Admitindo que este filtro seja representado por <*Expressão_Filtro*>, tem-se a seguinte consulta:

Base_dn:	ou=People, o=<organização>
Atributos:	cn
Escopo:	one
Filtro:	(<Expressão_Filtro>)

Como ilustração, considere as entradas de diretório da *Companhia X* apresentadas na Figura 6.12. Existem 4 entradas da classe *Person* dentro do folder *People*. Elas representam os usuários de um sistema de *software* utilizado pela companhia. Admita que este sistema de *software* controle o acesso aos seus recursos segundo o *framework* do RBPIM. Neste caso, suponha a necessidade de selecionar os usuários que poderão assumir o papel *Gerente*. Segundo a <*Expressão_Filtro*> definida acima, esta expressão deve ser construída com variáveis explícitas envolvendo os atributos de objetos da classe *Person*. Assim, considerando o esquema CNF *(pcimRuleConditionListType*=2) e os objetos de condição *C1* e *C2* associados ao papel *Gerente*, obtém-se a seguinte consulta LDAP:

Base_dn:	ou=People, o=Companhia X
Atributos:	cn
Escopo:	one
Filtro:	(|(&(objectclass=Person)(BusinessCategory=10))(&(objectclass=Person)(cn=Antunes)))

Em função do esquema CNF, o filtro é construído como *C1 or C2*, ou seja, ele filtra os objetos *Person* cujo atributo *BusinessCategory* seja 10 ou cujo atributo *cn* seja *Antunes*, indicando, por fim, que os usuários *Carlos, Paulo* e *Antunes* terão o papel *Gerente* incluído em suas listas de papéis elegíveis.

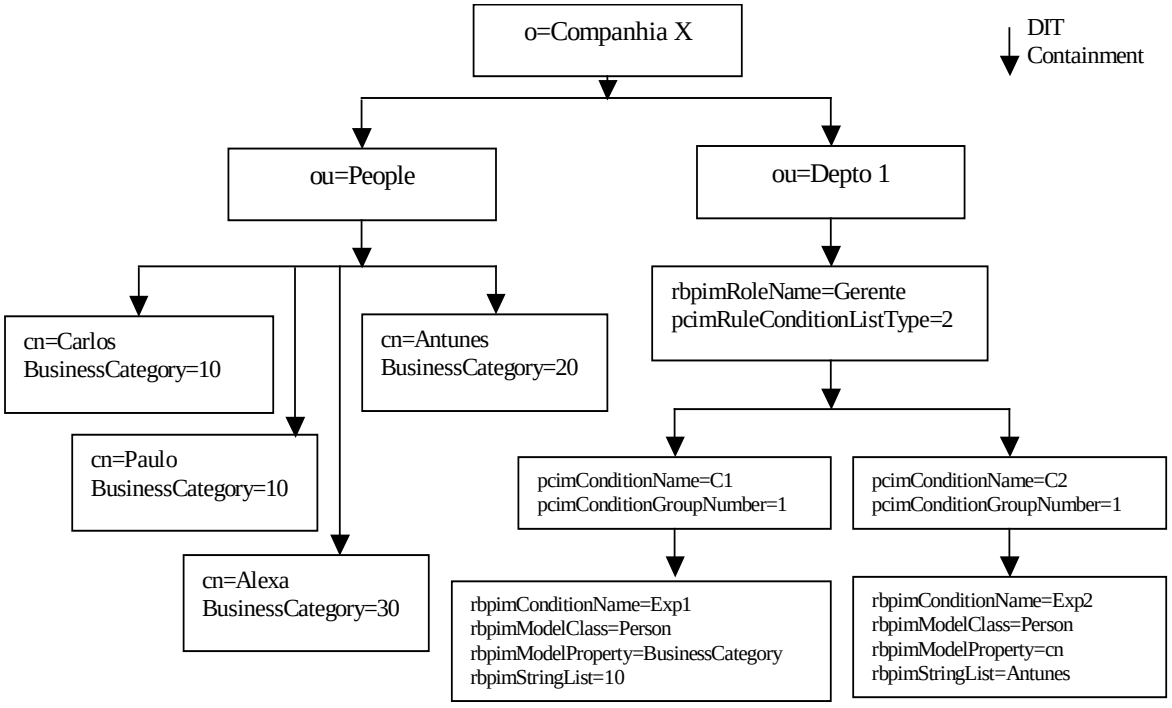

Figura 6.12 – Exemplo da associação entre papéis e usuário

d) Passo 6:

Consulta 5: Obter os intervalos de tempos associados com o objeto de período de ativação

A Consulta 2 retornou através do atributo *pcimRuleValidityPeriodList*, para cada papel, uma lista de referências *DNs* para objetos de períodos de ativação (*objectclass=pcimRuleValidityAssociation+pcimTPCAuxClass*). Representando cada uma destas referências *DN* por <*PeríodoAtivaçãoDN*>, tem-se a consulta LDAP mostrada abaixo. O escopo *base* indica que apenas o objeto referenciado pelo DN será considerado.

Base_dn:	<PeríodoAtivaçãoDN>
Atributos:	
Escopo:	base
Filtro:	

e) Passo 7:

Consulta 6: Obter objetos referentes aos papéis herdados

Representando uma referência *DN* de um papel herdado (contida no atributo *rbpimInheritedRoles)* por <*rbpimRole_DN*>, tem-se a consulta LDAP mostrada abaixo. O escopo *base* indica que apenas o objeto referenciado pelo DN será considerado.

Base_dn:	<rbpimRole_DN>
Atributos:	
Escopo:	base
Filtro:	

f) Passo 8:

Este passo utiliza a mesma consulta LDAP do *Passo 6*, ou seja, a Consulta 5. A única diferença é que as informações sobre os papéis foram obtidas no *Passo 7*.

g) Passo 10:

Consulta 7: Obter os conjuntos de relacionamentos de separação estática de tarefas

Representando a referência *DN* do papel por <*rbpimRole_DN*>, tem-se a consulta LDAP mostrada abaixo. Os objetos que indicam conflitos por *SSD* estão dentro do folder <*ou_usuário*>. O filtro seleciona apenas objetos que representam os conflitos por SSD nos quais o papel <*rbpimRole_DN*> é participante.

Base_dn:	ou=<ou_usuário>, o=<organização>
Atributos:	
Escopo:	base
Filtro:	(&(objectClass=rbpimSSD)(rbpimRoleSet= <rbpimRole_DN>))

6.5.3.2 Algoritmo do PDP para RBPEP_SelectedRoles (Fase 2 da Criação da Sessão)

A chamada *RBPEP_SelectedRoles* ativa uma sessão com o conjunto de papéis definidos pelo argumento *SelectedRoleSet[]*. Esta ativação será concluída com sucesso apenas se todos os papéis em *SelectedRoleSet[]* estiverem presentes na Base de Estados para a dada sessão e se todos eles estiverem livres de conflitos por DSD.

Passo 1:	Se já existe uma sessão ativa indicada pelo SessionID na Base de Estados com *Status = Fase 1* ir para o *Passo* 2. Caso contrário, retornar um objeto COPS <Error> através da mensagem DEC.
Passo 2:	*Determinar* **R**=$\{R_i\}_{i=1,..,n1}$ como sendo a lista de referências a papéis (objetos *rbpimRole*) associados à sessão indicada pelo SessionID na Base de Estados.
Passo 3:	Se *SelectedRoleSet[]* $\not\subset$ **R** então enviar uma mensagem DEC ao RBPEP indicando que a ativação foi negada. Caso contrário, ir para o *Passo 4*.
Passo 4:	*Determinar* **DSD**=$\{DSD_p\}_{p=1,...,n2}$ como sendo a lista de objetos *rbpimDSD* associados aos papéis contidos em **R**.
Passo 5:	Se *SelectedRoleSet[]* viola os conflitos indicados em **DSD** então enviar uma mensagem DEC ao RBPEP indicando que a ativação foi negada. Caso contrário, ir para o *Passo 6*.
Passo 6:	Atualizar a Base de Estados armazenando *SelectedRoleSet[]* como a lista de papéis ativos do usuário na sessão. Fazer *Status = Fase 2*. Então, enviar uma mensagem DEC ao RBPEP com *Result = true* encapsulado em um objeto COPS <*Decision*>.

a) Passo 4:

Consulta 8: Obter os conjuntos de relacionamentos de separação dinâmica de tarefas

Representando a referência *DN* do papel por <*rbpimRole_DN*>, tem-se a consulta LDAP mostrada abaixo.

Base_dn:	ou=<ou_usuário>, o=<organização>
Atributos:	
Escopo:	one
Filtro:	(&(objectClass=rbpimDSD)(rbpimRoleSet= <rbpimRole_DN>))

6.5.3.3 Algoritmo do PDP para RBPEP_CheckAccess

A informação sobre o(s) recurso(s) cujo acesso está sendo requisitado está contida no *array ObjectInfo[]*, o qual é suprido pela aplicação no momento da chamada. Conforme sintaxe definida na subseção 6.4.5, este *array* contém informações relativas ao acesso descritas na forma de variáveis explícitas e implícitas. Da mesma forma, os objetos de permissão possuem objetos *PolicyConditions* construídos com variáveis explícitas e implícitas.

Uma questão fundamental a ser considerada neste algoritmo é obtenção do conjunto de recursos indicado pelos objetos de permissão associados aos papéis assumidos pelo usuário na sessão, o qual será chamado de δ, e dos conjuntos requeridos pelo usuário na chamada, o qual será chamado de Φ. Uma resposta positiva a esta verificação de acesso depende de (1) $\Phi \subseteq \delta$ e (2) da operação requerida na chamada estar autorizada a todos os objetos pertencentes a Φ. Note que estes conjuntos podem ser constituídos por um grande número de recursos, tornando processo de verificação custoso e complexo. A estratégia adotada por este trabalho para a especificação deste algoritmo é baseada na filtragem das permissões associadas aos papéis assumidos pelo usuário na sessão, o que permite a redução do número de recursos presentes em δ. Esta estratégia é formada por 3 fases principais:

1ª. Fase: *Filtragem dos objetos de permissão que atendem às restrições indicadas pelos objetos PolicyConditions especificados com variáveis implícitas.* Nesta fase são obtidas as expressões lógicas especificadas pelas variáveis implícitas dos objetos de permissão associados aos papéis assumidos pelo usuário na sessão. Em seguida, é verificado se as informações relativas às variáveis implícitas contidas no *array ObjectInfo[ ]* atendem as restrições impostas pelas variáveis implícitas dos objetos de permissão. Caso esta verificação seja negativa, o objeto de permissão é descartado da lista de permissões associadas ao papel, quando o usuário não será autorizado a acessar os recursos indicados por este objeto de permissão.

2ª. Fase: *Filtragem dos objetos de permissão que possuem objetos PolicyActions cuja lista de operações contenha a operação requerida pelo usuário na chamada*. Caso a permissão não tenha a operação requerida pelo usuário, o objeto de permissão é descartado da lista de permissões associadas ao papel.

3ª. Fase: *Obtenção dos conjuntos de recursos*. Nesta etapa, são obtidas as expressões lógicas especificadas pelas variáveis explícitas dos objetos de permissão associados aos papéis assumidos pelo usuário na sessão. Agora, após a filtragem da lista de permissões realizada nas duas fases anteriores, o procedimento de autorização de acesso consiste em verificar se $\Phi \subseteq \delta$. Esta verificação ainda pode ser otimizada. Considere que θ seja o conjunto de objetos resultante da interseção dos dois primeiros conjuntos, ou seja, $\Phi \cap \delta$. Neste caso, a autorização de acesso seria concedida ao usuário caso $\Phi \subseteq \theta$. Do ponto de vista

prático, os filtros utilizados nas consultas LDAP para obtenção de θ são especificados com as condições indicadas pelas variáveis explícitas associadas às permissões mais os pares variável/valor das variáveis explícitas passados no *array ObjectInfo[ ]*. O algoritmo é mostrado a seguir.

Passo 1: *Verificar* se já existe uma sessão ativa, indicada por SessionID, na Base de Estados com *status* Fase 2. Se existe, ir para o Passo 2, caso contrário, retornar um objeto COPS <Error> através da mensagem DEC.

Passo 2: *Determinar* $\mathbf{R}=\{R_i\}_{i=1,..,n1}$ como sendo a lista de papéis do usuário ativos na sessão.

Passo 3: *Determinar* $\mathbf{R'}=\{R_j\}_{j\leq n1}$ como sendo o subconjunto de papéis de R_i que possuem objetos de períodos de ativação (classe *pcimRuleValidityAssociation)* compatíveis com a data e hora correntes.

Passo 4: *Determinar* $\mathbf{P}_j=\{P_{jk}\}_{k=1,..,n2}$ como sendo a lista de objetos de permissão (*rbpimPermission*) associados com o papel $R_j \in \mathbf{R'}$

Passo 5: *Determinar* $\mathbf{C}_{jk}=\{C_{jkl}\}_{l=1,..,n3}$, como sendo a lista de objetos de condição (classe *pcimRuleConditionAssociation*) associados ao objeto de permissão $P_{jk} \in \mathbf{P}_j$.

Passo 6: *Determinar* $\mathbf{P}_j'=\{P_{jm}\}_{m=1,..,n4}$ como sendo o subconjunto de permissões $\mathbf{P}_j$ que inclui somente os objetos $P_{jk} \in \mathbf{P}_j$ os quais as variáveis implícitas $\in \mathbf{C}_{jk}$ satisfaçam as condições indicadas pelas variáveis implícitas no *array ObjectInfo[]* passado na chamada *RBPEP_CheckAccess*.

Passo 7: *Determinar* $\mathbf{O}_{jm}=\{O_{jmn}\}_{n=1,..,n5}$ como sendo a lista de objetos de operações (classe *rbpimAssignerOperation*) associadas ao objeto de permissão $P_{jm} \in \mathbf{P}_j'$.

Passo 8: *Determinar* $\mathbf{P}_j'' =\{P_{jo}\}_{o=1,..,n6}$ como um subconjunto de $\mathbf{P}_j'$ que inclui apenas os objetos de permissão $P_{jm} \in \mathbf{P}_j'$ onde o argumento *Operation* passado na chamada $\in \mathbf{O}_{jm}$.

Passo 9: *Determinar* $\mathbf{C}_{jo}=\{C_{jop}\}'_{p=1,..,n7}$ como sendo a lista de objetos de condições (classe *pcimRuleConditionAssociation*) associadas com o objeto de permissão $P_{jo} \in \mathbf{P}_j''$.

Passo 10 : *Determinar* "e_{jo}" como sendo a expressão formada pelas variáveis explícitas das condições $\mathbf{C}_{jo}$, unidas na forma CNF ou DNF, conforme o atributo *pcimConditionListType* do objeto P_{jo} (classe *rbpimPermission*).

Passo 11 : *Determinar* "e_j" como sendo a expressão lógica formada concatenando-se as expressões "e_{jo}" com o operador "OU" lógico, para o=1,...,n_6.

Passo 12 : Determinar "e" como sendo a expressão lógica formada concatenando-se as expressões "e_j" com operador "OU" lógico, para j=1,...,n_2.

Passo 13 : *Determinar* "e*" como sendo a expressão lógica formada concatenado-se as condições de variáveis explícitas, passadas pelo argumento *ObjectInfo[]*, como o operador "E" lógico.

Passo 14 : *Determinar* $\theta = \{CIM_q\}_{q=1,..,n9}$, como sendo a lista de objetos CIM determinada pela consulta LDAP associada à expressão lógica (e "E" e*).

Passo 15 : *Determinar* $\Phi = \{CIM_r\}_{r=1,..,n10}$ como sendo a lista de objetos CIM determinada pela consulta LDAP associada à expressão lógica e*.

Passo 16 : Enviar mensagem DEC ao RBPEP com *result* = true se $\Phi \subseteq \theta$, caso contrário, enviar *result*=false

a) Passo 1:

Para que o PDP possa realizar a verificação de acesso a um recurso, é preciso que as duas fases da abertura de sessão tenham sido completadas com sucesso, o que é indicado por uma entrada com *Status = 2* na *Base de Estados* para a sessão do usuário. Caso esta entrada não exista ou não esteja com *Status = 2,* o PDP deve retornar uma mensagem DEC ao RBPEP contendo um objeto *error* indicando que ele não mantém uma sessão com estado para receber chamadas *CheckAccess*.

b) Passo 2:

Neste passo, deve ser construída a lista R_i contendo os papéis selecionados como ativos pelo usuário para a sessão indicada por SessionID. Estas informações foram inseridas previamente na *Base Estados* através da chamada *RBPEP_SelectedRoles*.

c) Passo 3:

Um procedimento importante que precisa ser efetuado é a checagem dos períodos de ativação dos papéis ativos na lista R_i , ou seja, verificar se eles ainda se encontram em seus períodos autorizados. Este procedimento já foi efetuado no algoritmo para a chamada *RBPEP_CreateSession*, mas, no entanto, ele deve ser repetido para cada chamada *RBPEP_CheckAccess*, pois, com o decorrer do tempo, um papel pode sair de sua faixa de ativação permitida e, portanto, deve ser descartado da lista R_i. A seguir, são apresentadas as consultas LDAP relacionadas.

Consulta 9: Obter as referências *DNs* dos objetos de período de ativação e permissões associados ao papel

Representando o nome do papel por <*rbpimRoleName*>, tem-se a consulta LDAP mostrada abaixo. Adicionalmente, esta consulta retornará as referências *DNs* a objetos *pcimRuleActionAssociation* através do atributo *pcimRuleActionList,* informação que será usada logo a seguir na Consulta 11.

Base_dn:	rbpimRoleName =<rbpimRoleName>, ou=<ou_usuário>, o=<organização>
Atributos:	pcimRuleValidityPeriodList, pcimRuleActionList
Escopo:	base
Filtro:	

Consulta 10: Obter os intervalos de tempos associados com o objeto de período de ativação

Representando cada a referência *DN* por <*PeríodoAtivaçãoDN*> retornada em *pcimRuleValidityPeriodList* na Consulta 9, tem-se a consulta LDAP mostrada abaixo.

Base_dn:	<PeríodoAtivaçãoDN>
Atributos:	
Escopo:	base
Filtro:	

d) Passo 4:

Consulta 11: Obter a referência DN da permissão associada ao papel

Representando cada referência *DN* por <*PermissãoDN*> retornada em *pcimRuleActionList* na Consulta 9, tem-se a consulta LDAP mostrada abaixo.

Base_dn:	<PermissãoDN>
Atributos:	rbpimPermissionDN
Escopo:	base
Filtro:	

Consulta 12: Obter as referências DNs dos objetos de condição e ação associados à permissão

Representando a referência *DN* do objeto de permissão como <*rbpimPermissionDN*> que foi retornada na Consulta 11, tem-se a consulta LDAP mostrada abaixo. O atributo *pcimRuleConditionListType* indica como os objetos de condição serão combinados para a construção da expressão que define os recursos associados enquanto o atributo *pcimRuleConditionList* retorna as referências *DNs* propriamente ditas. O atributo

pcimRuleActionList retorna as referências *DNs* aos objetos de ação que definem quais as operações permitidas aos recursos associados ao objeto de permissão.

Base_dn:	<rbpimPermissionDN>
Atributos:	pcimRuleConditionListType, pcimRuleConditionList, pcimRuleActionList
Escopo:	base
Filtro:	

e) Passo 5:

Consulta 13: Obter os atributos do objeto de condição e o par variável/valor

Representando a referência *DN* do objeto de condição por <*CondiçãoDN*>, a qual foi retornada na Consulta 12 pelo atributo *pcimRuleConditionList*, tem-se a consulta LDAP mostrada abaixo. Note que, como foi usado o escopo *sub*, o objeto de condição *pcimRuleConditonAssociation* e seu objeto associado por *DIT containment*, *rbpimConditionAssociation*, que define o par variável/valor, serão retornados.

Base_dn:	<CondiçãoDN>
Atributos:	
Escopo:	sub
Filtro:	

f) Passo 6:

A Consulta 13 pode ter retornando objetos de condição com expressões definidas por variáveis implícitas. Assim, caso elas existam, é preciso verificar se a informação referente ao usuário suprida na chamada da API (variáveis implícitas contidas no *array ObjectInfo[]*) atendem a estas restrições. Por exemplo, se existem variáveis implícitas que indicam que o acesso é permitido apenas para uma rede 192.168.1.0/24 o objeto de permissão deve ser descartado da lista de permissões caso não seja suprido o IP de origem do usuário na chamada ou, para um IP suprido, ele não esteja na faixa indicada.

g) Passos 7 e 8:

Consulta 14: Obter a lista de operações associadas à permissão filtrando pela operação solicitada na chamada da API

Representando a referência *DN* do objeto de ação por <Ação*DN*>, a qual foi retornada na Consulta 12 pelo atributo *pcimRuleActionList*, e a operação requerida pelo usuário na chamada da API por <*Operation*>, tem-se a consulta LDAP mostrada abaixo.

Base_dn:	<AçãoDN>
Atributos:	
Escopo:	base
Filtro:	(rbpimOperationList=<Operation>)

g) Passo 14:

Consulta 15: Obter a lista de recursos associados à permissão e selecionados no *array ObjectInfo[ ]*

O aspecto mais importante relacionado com esta consulta é a construção do filtro que será utilizado. Através da Consulta 13, foi possível obter os pares variável/valor associados aos objetos de condição, os quais serão utilizados na construção do filtro. Nesta consulta, apenas serão utilizadas as variáveis explícitas, as quais definem os recursos associados à permissão. Como mencionado anteriormente, objetos de condição construídos com variáveis explícitas informam a classe e o atributo do objeto que representam os recursos no contexto do CIM, respectivamente, *rbpimModelClass* e *rbpimProperty*. Juntamente com estes dois atributos, um terceiro atributo, adicionado ao objeto de condição em função do *attachment* de uma das subclasses de *rbpimPolicyValue* (por exemplo, o atributo *rbpimStringList* da subclasse *rbpimPolicyStringValue*), especifica qual valor será utilizado na comparação. Da mesma forma, os pares variável/valor passados no *array ObjectInfo[ ]* são adicionados ao filtro. Assim, representando o valor do atributo da subclasse de *rbpimPolicyValue* da condição por <*valor_atributo*> e sendo *rbpimModelClassObjInfo*, *rbpimPropertyObjInfo* e <*valor_atributo1*> a representação, respectivamente, do nome da classe, do nome da propriedade e do valor associado passados na chamada no *array ObjectInfo[ ]*, a expressão do filtro será construída com a seguinte estrutura:

<Expressão do Filtro>

(&(objectclass=<rbpimModelClass>)(<rbpimModelProperty>=<valor_atributo>)(objectclass=<rbpim ModelClassObjInfo>)(<rbpimPropertyObjInfo>=<valor_atributo1>))

A estrutura do filtro LDAP mostrado acima apresentou uma expressão lógica simples. Vale notar que, se existirem outras condições, elas serão agrupadas a esta expressão segundo

o esquema DNF/CNF, definido pelo atributo *pcimRuleConditionListType* do objeto de permissão, e segundo o número para agrupamento do objeto de condição, definido pelo atributo *pcimConditionGroupNumber*. Ainda, caso o atributo *pcimConditionNegated* do objeto de condição seja *TRUE*, a expressão no filtro deverá ser negada. Da mesma forma, caso existam outras variáveis explícitas no *array ObjectInfo[]*, elas serão adicionadas à expressão *and* do filtro acima. Admitindo que os objetos que representam os recursos se encontrem dentro do folder da própria organização, tem-se a seguinte consulta LDAP:

Base_dn:	o=<organização>
Atributos:	
Escopo:	sub
Filtro:	<Expressão do Filtro>

g) Passo 15:

Consulta 16: Obter a lista de recursos indicados pelas variáveis explícitas contidas *array ObjectInfo[ ]*

Representando por *rbpimModelClassObjInfo*, *rbpimPropertyObjInfo* e <*valor_atributo1*> a representação, respectivamente, do nome da classe, do nome da propriedade e do valor associado passados na chamada no *array ObjectInfo[ ]*, a expressão do filtro será construída com a seguinte estrutura:

<Expressão do Filtro>

(&(objectclass=<rbpimModelClassObjInfo>)(<rbpimPropertyObjInfo>=<valor_atributo1>))

Vale notar que, se existirem outras variáveis explícitas no *array ObjectInfo[]*, elas serão adicionadas à expressão *and* do filtro acima. Admitindo que os objetos que representam os recursos se encontrem dentro do folder da própria organização, tem-se a seguinte consulta LDAP:

Base_dn:	o=<organização>
Atributos:	
Escopo:	Sub
Filtro:	<Expressão do Filtro>

6.5.4 Algoritmo do PDP para a mensagem RPT

Neste trabalho, o PDP efetiva a decisão de política enviada ao RBPEP após o recebimento desta mensagem. Por exemplo, a criação de uma sessão e a seleção de papéis só são efetivadas quando o PDP recebe uma mensagem RPT associada com a mensagem DEC enviada anteriormente.

Se existe sessão ativa *então:*
    Atualizar a entrada da Base de Estados referente a SessionID para indicar que a última decisão de política enviada pelo PDP foi recebida pelo RBPEP.
Fim.

6.5.5 Algoritmo do PDP para a mensagem DRQ

Esta mensagem é enviada pelo RBPEP ao PDP para solicitar o fechamento de uma sessão RBPIM.

Verificar se existe uma sessão ativa na Base de Estados indicada pelo SessionID.
Se existe sessão ativa *então:*
    Remover a entrada da Base de Estados referente à sessão indicada por SessionID.
Fim.

6.5.6 Algoritmo do PDP para a mensagem CC

Esta mensagem é enviada pelo RBPEP ao PDP para solicitar o fechamento do serviço de política RBPIM aberto por uma mensagem OPN. Após esta mensagem, para que o RBPEP retome a utilização do serviço, faz-se necessário que ele solicite novamente sua abertura através da edição de uma nova mensagem OPN.

Remover todas entradas da Base de Estados referentes às sessões abertas pelo RBPEP que editou a mensagem.

6.6. Conclusões do Capítulo

O presente capítulo apresentou uma proposta de implementação para o *framework* do RBPIM baseado nas definições dos Capítulos 4 e 5. Para esta implementação, buscou-se propor uma arquitetura tal que uma dada aplicação cliente pudesse consumir políticas RBAC providas por uma aplicação fornecedora de um serviço de controle de acesso baseado no RBPIM. Em suma, através deste serviço, a aplicação cliente (p.ex. um servidor WEB, um servidor de aplicativos, etc..) poderia utilizar políticas RBPIM para controlar o acesso aos seus recursos, como, por exemplo, o acesso a aplicativos, funcionalidades de sistemas de *software*, arquivos e diretórios. A estratégia adotada por este trabalho foi definir a arquitetura

desta implementação baseada na arquitetura definida em [RFC 2753]. Assim, a arquitetura proposta ficou centrada nas entidades denominadas RBPEP e PDP, onde a aplicação cliente utiliza as chamadas da API do RBPEP, que também foram definidas neste capítulo, para requerer o serviço de controle de acesso baseado no RBPIM provido pelo servidor de políticas PDP. Para realizar o serviço de políticas, o PDP se baseia em um repositório LDAP, o qual contém objetos que descrevem as regras de políticas RBPIM, aplicados no diretório segundo o esquema apresentado no Capítulo 5.

De maneira geral, o RBPEP e o PDP trocam mensagens que transportam a informação de política RBPIM. Para estabelecer um padrão na troca de mensagens entre o PEP e o PDP, [RFC 2748] apresenta o COPS, o qual foi adotado por este trabalho. O presente capítulo padronizou as mensagens COPS para dar suporte às trocas de mensagens entre o RBPEP e o PDP. Por fim, foram apresentados os algoritmos que implementam as regras de controle de acesso do RBPIM. A partir das definições contidas neste capítulo, o Capítulo 7 apresentará um estudo de caso e a implementação do protótipo, objetivando avaliar a proposta contida neste trabalho.

Capítulo 7

Estudo de Caso e Avaliação da Proposta

7.1. Introdução

Os benefícios do uso de políticas baseadas em papéis ficam ainda mais evidentes com sua aplicação em situações comumente encontradas no cotidiano das organizações. Com o objetivo de validar e de demonstrar como o modelo e o esquema de diretório do RBPIM podem ser utilizados para a definição e a aplicação de políticas de segurança em ambientes distribuídos, a seção 7.2 apresenta um estudo de caso envolvendo o controle de acesso baseado em papéis em uma aplicação bancária. A seção 7.3 apresenta o protótipo desenvolvido a partir da arquitetura proposta no Capítulo 6, o qual será utilizado para validar os algoritmos do PDP e as mensagens COPS definidas. A seção 7.4 apresenta os resultados decorrentes da utilização do protótipo para a implementação do estudo de caso realizado. Por fim, a seção 7.5 conclui o Capítulo.

7.2. Estudo de Caso

O presente estudo de caso foi construído tomando como referência rotinas observadas em bancos brasileiros. As informações apresentadas foram obtidas a partir de entrevistas realizadas com funcionários de alguns bancos. Por questões didáticas, este estudo de caso se referirá a um banco hipotético denominado Banco ABC.

7.2.1 Cenário

O Banco ABC utiliza um grande conjunto de aplicações para dar suporte à operação das atividades relacionadas ao seu negócio. O controle de acesso a aplicações é realizado de forma discricionária, onde cada aplicação mantém um arquivo próprio para o controle dos direitos de acesso dos usuários. Assim, para cada usuário, uma aplicação deve ter uma entrada neste arquivo de direitos de acesso com as operações que ele pode realizar durante sua ativação. Fica evidente que o gerenciamento deste processo é sujeito a erro e custoso, pois os

direitos de acesso são administrados individualmente, com sua complexidade crescendo na mesma proporção do crescimento do número de usuários e de aplicações do sistema. De forma simplificada, duas aplicações e algumas das principais operações associadas são mostradas na Tabela 7.1. Os usuários do sistema e suas categorias funcionais estão mostrados na Tabela 7.2. Cada categoria funcional está associada com uma ocupação, conforme mostra a Tabela 7.3.

Aplicação	Operações
Gerenciador Financeiro	Agendar TED - Agendar DOC Autorizar TED - Autorizar DOC Auditar Transações – Efetuar Pagamentos
Gerenciador de Clientes	Abrir ContaCorrente – Conceder Limite Auditar Transações

Tabela 7.1 – Banco ABC: Aplicações e Operações

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

Tabela 7.2 – Banco ABC: Usuários do sistema e suas posições

Categoria Funcional	Ocupação
A1	Atendente
A2	Caixa
B1	Supervisor
C1	Auditor

Tabela 7.3 – Banco ABC: Categorias funcionais e ocupações

Segundo as regras de operação do banco, os funcionários da posição *Atendente* podem apenas agendar transações TED (Transferência Eletrônica Disponível) e DOC (Documento de Crédito) no Gerenciador Financeiro e abrir contas no Gerenciador de Clientes; os funcionários de nível *Supervisor* podem conceder limite a uma conta corrente desde que ela não tenha sido aberta por eles e ainda autorizar DOCs e TEDs desde que eles não tenham sido agendados por eles próprios; os funcionários da posição *Caixa* podem realizar as operações conferidas a um *Atendente* e ainda efetuar pagamentos; os funcionários de nível *Auditor* nunca poderão realizar outra operação senão aquelas relacionadas com os procedimentos de

auditoria das transações efetuadas. Todas as operações devem ser realizadas no horário de expediente bancário, que vai das 10:00hs às 16:00hs. de segunda-feira à sexta-feira. Por questões de segurança, apenas os acessos provenientes da rede interna (192.168.10.0/24) poderão executar as operações de auditoria.

No cenário descrito anteriormente, o administrador do sistema deve incluir, individualmente para cada usuário, todas as restrições impostas pelas regras do banco, o que é, de fato, uma atividade custosa e sujeita a erro. Um outro problema observado é que a responsabilidade pela realização das tarefas de controle de acesso fica a cargo da própria aplicação financeira, procedimento que, obviamente, não faz parte da atividade principal para a qual ela foi concebida.

7.2.2. O Mapeamento da Política de Controle Acesso através do RBPIM

Uma alternativa a esta forma de controle de acesso realizada no Banco ABC é a utilização de uma aplicação externa, integrada com os sistemas legados do banco, para a realização das tarefas de controle de acesso relacionadas às aplicações financeiras. Neste novo cenário, as diversas aplicações do Banco ABC se restringem a executar apenas sua atividade fim e utilizam os serviços providos por esta aplicação externa para o estabelecimento das políticas de controle de acesso.

Seguindo a arquitetura proposta no Capítulo 6 para a implementação desta aplicação, as informações de política do banco, as quais envolvem aplicações, operações, usuários e as regras de acesso, passam a ser administradas de forma centralizada, sendo aplicadas em serviços de diretórios LDAP segundo os modelos do CIM e o modelo e esquema do RBPIM propostos nos Capítulos 4 e 5. Esta centralização não implica que toda a informação esteja reunida apenas em um único local. Dependendo da complexidade do ambiente, as facilidades de distribuição dos serviços de diretórios e seus mecanismos de pesquisa entre servidores podem ser utilizados. Com eles, requisições de dados podem ser repassadas entre servidores. Por exemplo, cada agência do Banco ABC poderia ter um servidor de diretório e ser responsável pelo gerenciamento da informação de política local. Com as facilidades citadas acima, a centralização lógica permanece, nada mudando com a adoção desta arquitetura distribuída. Para maiores informações sobre estes mecanismos consulte [NETSCAPE 99].

7.2.2.1. Hierarquia de Papéis

Um dos principais benefícios alcançados com a utilização de políticas baseadas no RBAC é o fácil mapeamento de papéis no contexto das funções de negócio ocupadas pelas pessoas nas organizações e do relacionamento hierárquico entre elas. Um papel pode ser visto como uma função de negócio dentro da organização o qual tem alguma autoridade, competência e privilégios associados. A análise das ocupações dos funcionários e de suas atribuições no banco leva à hierarquia de papéis mostradas na Figura 7.1. Cabe a observação de que normalmente não se tem uma relação direta entre a ocupação do indivíduo dentro da organização e o papel RBAC correspondente na hierarquia. Este processo envolve uma análise mais refinada dos privilégios associados e dos níveis de responsabilidades das ocupações.

Na hierarquia de papéis proposta, foi inserido um papel de nível inferior denominado *Funcionário*. Este papel tem associado um conjunto de permissões de acesso básicas, comum a todo funcionário do banco. Para o cenário proposto neste estudo caso, estes privilégios não são relevantes e, portanto, não serão apresentados. Cada papel possui um conjunto de permissões próprias mais aquelas herdadas dos papéis de níveis inferiores na hierarquia.

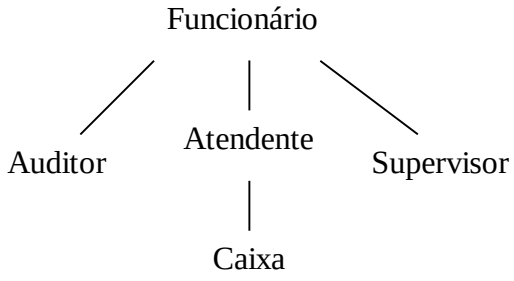

Figura 7.1 – Hierarquia de Papéis do Banco ABC

A hierarquia denota que as permissões associadas com o papel *Funcionário* são herdadas pelo papel *Atendente*. Aquelas associadas a *Atendente* são herdadas por *Caixa*. Os papéis *Auditor* e *Supervisor* herdam as permissões do papel *Funcionário*. A Tabela 7.4 apresenta os direitos de acesso atribuídos a cada papel da hierarquia. Note que, no caso do papel *Caixa*, as operações herdadas de *Atendente* não estão sendo mostradas.

Papel	Gerenciador Financeiro	Gerenciador de Clientes
Atendente	Agendar TED Agendar DOC	Abrir ContaCorrente
Supervisor	Autorizar TED Autorizar DOC	Conceder Limite
Caixa	Efetuar Pagamentos	
Auditor	Auditar Transações	Auditar Transações

Tabela 7.4 – Banco ABC: Papéis e direitos de acesso

7.2.2.2. Separação de tarefas entre papéis

A separação de tarefas entre papéis é especificada pelas regras de negócio decorrentes da política da organização. Alguns usuários na organização executam atividades similares e, portanto, possuem direitos de acesso em comum. A separação de tarefas indica que, para um grupo de transações em particular, nenhum papel poderá executar todas as transações dentro deste grupo. Por exemplo, no Banco ABC, (1) um funcionário de nível *Auditor* jamais poderá conceder limite a uma conta corrente; (2) um funcionário, por atribuição de sua ocupação, poderia, eventualmente, assumir o papel *Supervisor* e o papel *Atendente*. Neste caso, ele poderia abrir uma conta corrente e conceder limite a ela própria. As situações (1) e (2) podem ilustrar, respectivamente, relacionamentos de separação estática e dinâmica de tarefas. No caso (1), em nenhuma hipótese um funcionário que assume o papel de *Auditor* poderá assumir em qualquer sessão outro papel além do papel *Funcionário*. Já no caso (2), um funcionário nunca poderá, durante uma mesma sessão, assumir simultaneamente os papéis *Supervisor* e *Atendente*, mesmo tendo ambos em sua lista de papéis autorizados.

Apesar desta impossibilidade de assumir papéis conflitantes em uma mesma sessão, isto não evita, por exemplo, que o usuário mantenha duas sessões com papéis mutuamente exclusivos em cada uma delas. Neste caso, dependendo do tipo de operação a ser executada, a aplicação pode usar o número de sessões ativas do usuário para autorizar ou não a abertura de uma outra sessão. Este valor é retornado à aplicação pela chamada *RBPEP_CreateSession* da API do RBPEP. As Tabelas 7.5 e 7.6 apresentam os relacionamentos de separação estática (SSD) e dinâmica (DSD) de tarefas.

Nome	Papéis	Cardinalidade
SSD 01	Auditor, Atendente	2
SSD 02	Auditor, Supervisor	2
SSD 03	Auditor, Caixa	2

Tabela 7.5 – Banco ABC: Relacionamentos de SSD entre papéis

Nome	Papéis	Cardinalidade
DSD 01	Supervisor, Atendente	2

Tabela 7.6 – Banco ABC: Relacionamentos de DSD entre papéis

Nas Tabelas 7.5 e 7.6, as cardinalidades determinam a quantidade necessária de papéis selecionados do conjunto para violar a regra. Por exemplo, caso algum funcionário ative os papéis *Supervisor* e *Atendente* em uma sessão, ocorre violação por DSD, devendo o PDP negar esta ativação papéis.

Uma questão importante a ser considerada é forma com que a proposta de implementação do RBPIM, contida no Capítulo 6, realiza a seleção de papéis durante as duas fases que compõem a abertura de sessão. Na primeira, uma lista de papéis livre de conflitos por SSD é enviada à aplicação. Na segunda, após o usuário ter escolhido os papéis dentre a lista recebida na primeira fase, o PDP verifica possíveis conflitos por DSD na lista de papéis selecionados, recusando o conjunto, caso pelo menos um conflito tenha ocorrido. Na criação da lista de papéis da fase 1, caso tenha havido conflito por SSD, é preciso que o PDP decida qual dos papéis desta lista de conflitos ele irá incluir no conjunto que será enviado à aplicação. Neste caso, o critério utilizado é baseado na prioridade do papel, indicada pelo atributo *pcimRulePriority*, contido nos objetos que representam os papéis no diretório. Assim, papéis com níveis de prioridade superiores são selecionados em relação aos de menores níveis, cabendo ao administrador estabelecer os valores que atendam as necessidades da política do banco. A Tabela 7.7 apresenta os valores deste atributo para cada um dos papéis deste estudo de caso.

Papel	Prioridade
Atendente	1
Caixa	2
Supervisor	3
Auditor	4

Tabela 7.7 – Banco ABC: Prioridade entre papéis

Para ilustrar, considere o caso do funcionário *Matias*. Segundo suas categorias funcionais mostradas na Tabela 7.2 e os correspondentes papéis apresentados na Tabela 7.3,

ela poderá assumir os papéis *Auditor* e *Supervisor*, mas, no entanto, devido ao relacionamento SSD 02 mostrado na Tabela 7.5 estes papéis são mutuamente exclusivos. Em função das prioridades especificadas na Tabela 7.7, o PDP selecionará o papel *Auditor* e removerá o papel *Supervisor* da lista de papéis elegíveis, desta forma, permitindo que este funcionário apenas assuma os papéis *Auditor* e *Funcionário*. Uma outra situação semelhante ocorre com o funcionário *Pedro*. Este funcionário pode ativar em uma mesma sessão os papéis *Supervisor* e *Atendente*, pois ambos fazem parte da lista de papéis elegíveis do usuário. Face ao conflito por DSD mostrado na Tabela 7.6, caso o usuário proceda com esta ativação, o PDP retornará uma mensagem de erro, não permitindo que tal procedimento seja efetuado.

7.2.2.3. A DIT para o caso do BANCO ABC

A Figura 7.2 apresenta a DIT do diretório utilizada neste estudo de caso. Nesta Figura, estão sendo mostrados os principais *folders* do diretório com seus atributos *RDN*. Os objetos que representam os funcionários são inseridos no *folder People* enquanto os que representam os aplicativos instalados no banco são inseridos no *folder Aplicativos*. As informações de política RBPIM estão agrupadas no folder *Agencia_01*, o que significa que os papéis, as permissões e os relacionamentos de separação de tarefas relacionados aos funcionários e aplicativos do banco foram definidos segundo as regras de controle de acesso determinadas pela Agência 01. As informações neste folder seguem a estrutura hierárquica mostrada na Figura 6.11 do Capítulo 6.

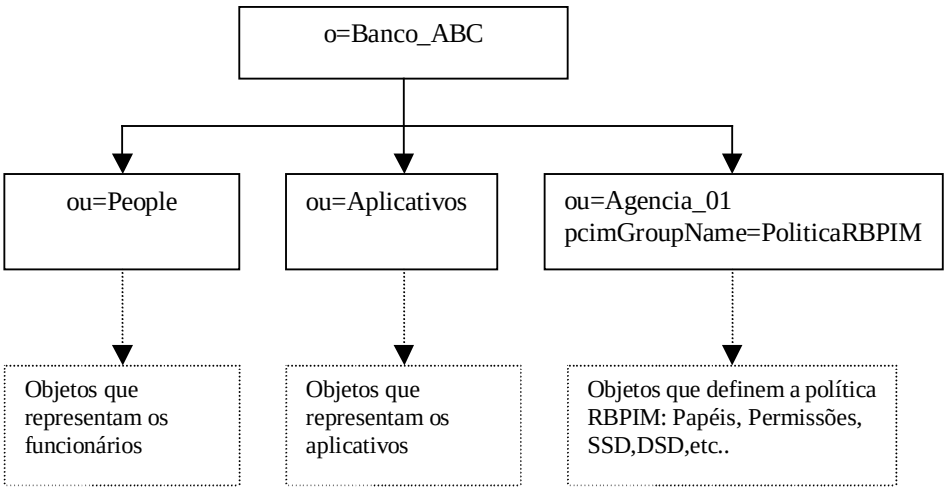

Figura 7.2 – A DIT do diretório para o caso do Banco ABC

7.2.2.4 Objetos do Diretório para o caso do BANCO ABC

Segundo a estratégia do RBPIM, as informações relativas ao mapeamento da política baseada em papéis são representadas pelo esquema de diretório apresentado no Capítulo 5. Por sua vez, as informações das entidades às quais se deseja aplicar esta política seguem os esquemas do CIM.

A Figura 7.3 mostra, esquematicamente, um papel e seus principais objetos associados. Seguindo a notação desta figura, (2) e (3) representam as condições que definem os usuários associados a (1), (4) indica a referência DN que associa um papel a uma permissão e (5) define um período permitido para ativação do papel. Para uma permissão (6), (7) e (8) representam as condições que definem os recursos que estão associados ao objeto de permissão, enquanto (9) especifica as operações autorizadas para os recursos indicados por (7) e (8). Vale observar que um papel pode ter várias condições, permissões e períodos de ativação associados. Da mesma forma, uma permissão pode ter várias condições e operações associadas. Para ilustrar o mapeamento do caso do Banco ABC , a Tabela 7.8 apresenta as principais classes e os respectivos RDNs dos objetos instanciados no diretório para este estudo de caso. A Tabela 7.9 apresenta os principais objetos relacionados ao papel *Auditor*. Nesta tabela, foi utilizada a notação apresentada na Figura 7.3.

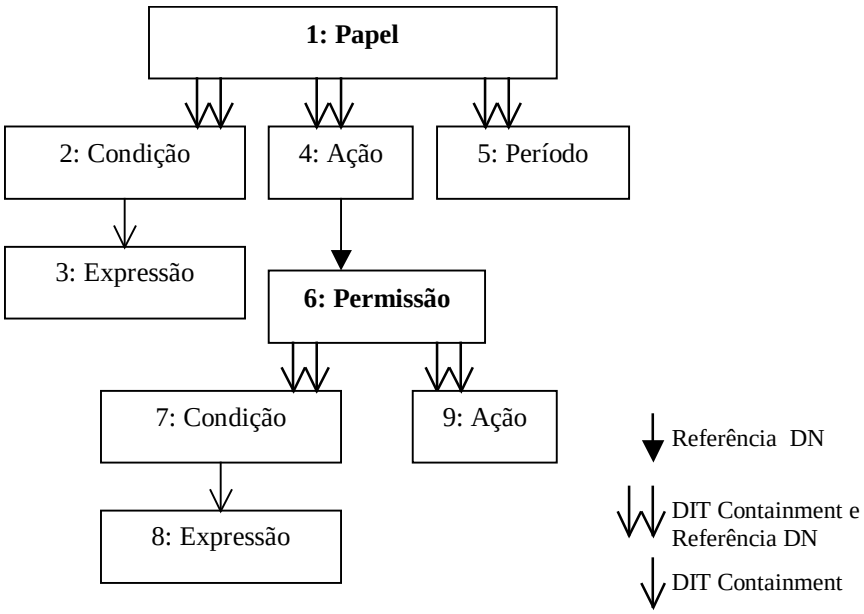

Figura 7.3: Um Papel e seus principais objetos

Classe	RDN dos Objetos	Esquema	Significado
rbpimRole	rbpimRoleName=Funcionario rbpimRoleName=Atendente rbpimRoleName=Supervisor rbpimRoleName=Caixa rbpimRoleName=Auditor	RBPIM	Objetos que representam Papéis
rbpimPermission	rbpimPermissionName=GF1 rbpimPermissionName=GF2 rbpimPermissionName=GF3 rbpimPermissionName=GC1 rbpimPermissionName=GC2 rbpimPermissionName=AUD	RBPIM	Objetos que representam Permissões do GF-Gerenciador Financeiro, go GC-Gerenciador de Cliente e de AUD-Auditoria
inetOrgPerson	cn=Carlos, cn=Ana, cn=Joana, cn=Rubens, cn=Marcos, cn=Ailton, cn=Matias, cn=Maria, cn=Silvia,cn=Vivian, cn=Pedro, cn=Carla,cn=Alex	CIM_User	Objetos que representam os usuários das aplicações financeiras.
applicationSystem	dlmName=GerFinanceiro dlmName=GerCliente	CIM_Application	Objetos que representam as aplicações financeiras.
rbpimSSD	rbpimSSDname=SSD01 rbpimSSDname=SSD02 rbpimSSDname=SSD03	RBPIM	Objetos que representam conjuntos de papéis com conflitos de SSD.
rbpimDSD	rbpimDSDname=DSD01	RBPIM	Objetos que representam conjuntos de papéis com conflitos de DSD.

Tabela 7.8 – Banco ABC: Principais Classes e Objetos

As informações contidas na Tabela 7.9 indicam que todos os funcionários cujo atributo *BusinessCategory* for igual a *C1* terão *Auditor* na sua lista de papéis elegíveis. Este atributo mantém o valor da categoria funcional de cada funcionário, conforme valores indicados na Tabela 7.2. Note que, com apenas uma única regra (2)(3), foi possível associar um conjunto de funcionários ao papel (associação *UA* do RBAC). O papel *Auditor* possui o objeto de permissão *AUD* (associação *PA* do RBAC), conforme indicado pelo objeto (4). O objeto (5) especifica que o papel *Auditor* poderá apenas ser ativado de segunda à sexta, no horário das 10hs às 16hs. O objeto de permissão *AUD* (6) possui uma lista de condições do tipo CNF, o que denota, juntamente com os objetos (7)(8), a expressão *(RecursosAUD1 OR RecursosAUD2) AND RestricoesAUD1*. Assim, aplicando-se a consulta LDAP com a expressão do filtro definida por *(objectclass=dlm1ApplicationSystem and dlmName=GerFinanceiro) or (objectclass= dlm1ApplicationSystem and dlmName=GerCliente)* os recursos associados com a permissão *AUD* são as aplicações *GerFinanceiro* e *GerCliente*. Ainda, *RestricoesAUD1* indica que o PDP deverá verificar se o

IP da máquina que está executando as aplicações está na faixa indicada por 192.168.10.0/24. Por fim, o objeto (9) indica que apenas a operação *Auditar_Transacoes* está autorizada para as aplicações *GerFinanceiro* e *GerCliente*. Para uma consulta completa a todos os objetos instanciados para este estudo de caso, consulte o arquivo LDIF cuja listagem encontra-se no Anexo C.

Objeto	Principais Atributos	Classe

(1)	rbpimRoleName=Auditor, pcimRuleEnabled=1
	pcimRulePriority=4,
	pcimRuleConditionListType=1 // DNF
	rbpimInheritedRoles= (rbpimRoleName:Funcionario)
		(1)-rbpimRole

(2)(3)	pcimConditionName=UsuariosAuditor1 pcimConditionGroupNumber=1 rbpimModelClass=inetorgperson rbpimModelProperty=BusinessCategory rbpimStringList=C1	(2)-pcimRuleConditionAssociation (3)-rbpimPolicyExplicitVariable rbpimPolicyStringValue
(4)	pcimActionName=PermissaoAuditor1 rbpimPermissionDN=(rbpimPermissionName:AUD)	(4)-pcimRuleActionAssociation
(5)	pcimValidityConditionName=Periodo1 pcimTPCDayOfWeekMask=01111100 pcimTPCTimeOfDayMask=T100000/T160000	(5)-pcimRuleValidityAssociation
(6)	rbpimPermissionName=AUD pcimRuleConditionListType=2 // CNF	(6)-rbpimPermission
(7)(8)	pcimConditionName=RecursosAUD1 pcimConditionGroupNumber=1 rbpimModelClass=dlm1ApplicationSystem rbpimModelProperty=dlmName rbpimStringList=GerFinanceiro	(7)-pcimRuleConditionAssociation (8)-rbpimPolicyExplicitVariable rbpimPolicyStringValue
(7)(8)	pcimConditionName= RecursosAUD 2 pcimConditionGroupNumber=1 rbpimModelClass= dlm1ApplicationSystem rbpimModelProperty=dlmName rbpimStringList=GerCliente	(7)-pcimRuleConditionAssociation (8)-rbpimPolicyExplicitVariable rbpimPolicyStringValue
(7)(8)	pcimConditionName=RestricoesAUD1 pcimConditionGroupNumber=2 rbpimPolicyIPv4AddrValue=192.168.10.0/24	(7)-pcimRuleConditionAssociation (8)-rbpimPolicySourceIPv4Var rbpimPolicyIpv4AddrValue
(9)	pcimActionName=OperacoesAUD1 rbpimOperationList=Auditar_Transacoes	(9)-pcimRuleActionAssociation

Tabela 7.9 – Banco ABC: O Papel Auditor e seus principais objetos

7.3. Implementação do Protótipo

O protótipo apresentado nesta seção tem por objetivo validar o modelo e o esquema do RBPIM, bem como a arquitetura, algoritmos e mensagens COPS especificados para a sua implementação.

7.3.1 Estrutura do Protótipo

A estrutura do protótipo foi baseada na arquitetura proposta no Capítulo 6, conforme mostra a Figura 7.4.

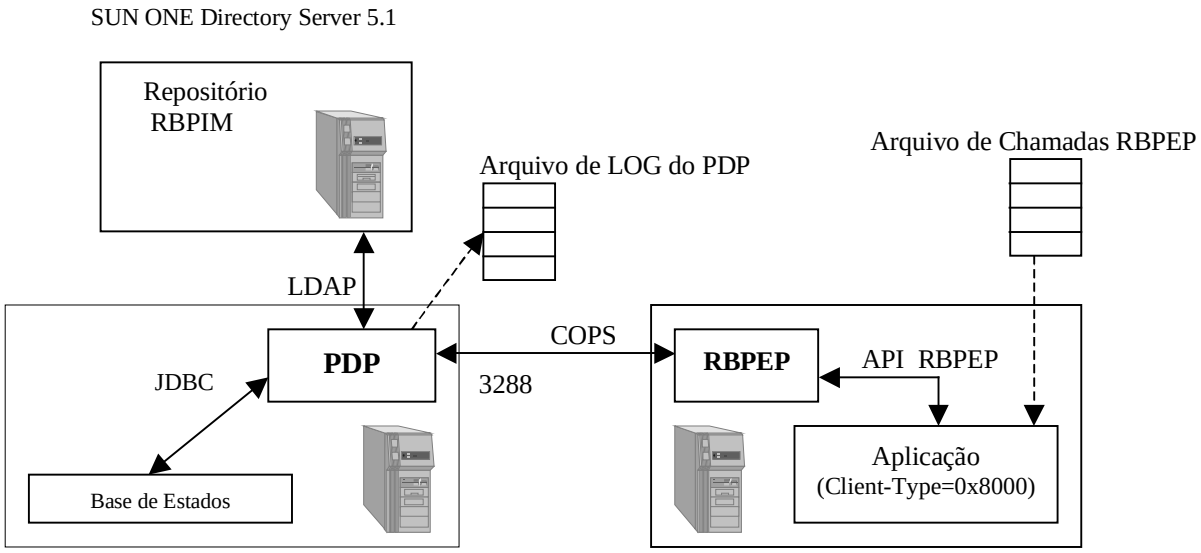

Figura 7.4 – Arquitetura do Protótipo

O protótipo desenvolvido não possui interface para interação humana. No entanto, objetivando facilitar a validação da arquitetura proposta, foram gerados diversos arquivos contendo uma seqüência de chamadas da API do RBPEP. Cada um destes arquivos simula um cenário de operação de uma aplicação consumidora de política RBPIM, neste caso, a aplicação bancária onde os funcionários do Banco ABC realizam suas operações financeiras.

As aplicações PDP, RBPEP e o módulo que representa a aplicação financeira foram desenvolvidos na plataforma *JAVA 2 Standard Edition v.1.4.0*.

Quando a aplicação financeira é lançada, ela ativa seu RBPEP. A partir deste momento, as chamadas da API contidas no arquivo de entrada correspondente à aplicação passam a ser executadas de forma seqüencial. Cada aplicação pode solicitar a abertura de diversas sessões, sejam elas de usuários já ativos em outras sessões ou não. As informações sobre as sessões abertas são mantidas na Base de Estados. Neste protótipo, a Base de Estados foi implementada de duas formas. Para a execução do PDP no ambiente *Windows* foi utilizado o *Microsoft Access* enquanto no ambiente *Linux* foi utilizado o banco de dados *MySql*. Em ambos os casos a Base de Estados é acessada pelo PDP via *JDBC-Java Database Connectivity-*, uma API para conexão de programas escritos em *JAVA* aos mais populares bancos de dados. O serviço de diretório LDAP utilizado como repositório da informação de

política do RBPIM foi o *Sun ONE Directory Server 5.1*. Na aplicação *JAVA* do PDP foi utilizado o pacote *Netscape Directory SDK 4.0 for Java*, um kit de desenvolvimento para escrever aplicações *LDAP*.

7.3.2 Cenário de Avaliação

No cenário de avaliação 20 (vinte) aplicações passam a requerer o serviço de política do RBPIM. Nesta situação, segundo o *framework* proposto pela RFC 2748, cada RBPEP mantém uma conexão TCP independente com o PDP, conforme ilustra a Figura 7.5.

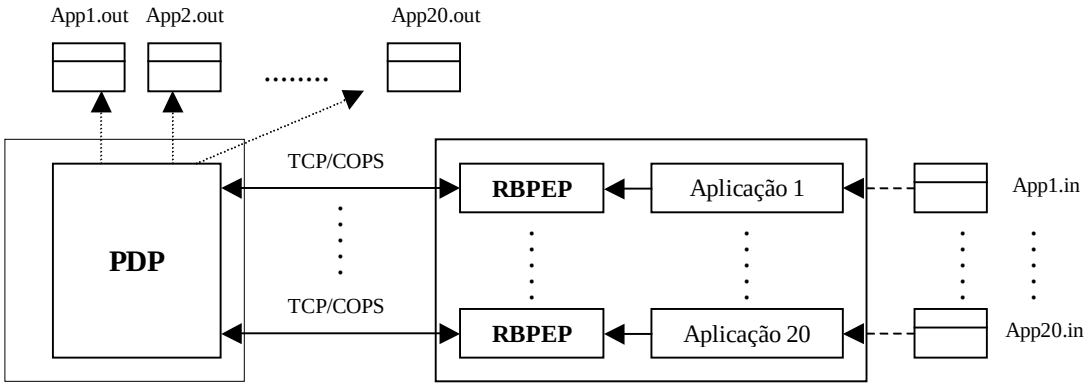

Figura 7.5 – Cenário de Avaliação com 20 aplicações

Os arquivos que contêm as chamadas RBPEP de cada aplicação foram chamados de *App1.in, App2.in ,...., App20.in*. Os resultados das decisões de política do PDP em função das chamadas efetuadas por cada aplicação foram armazenados em arquivos de *log*, os quais foram denominados *App1.out, App2.out, ..., e App20.out*. Consulte o Anexo D para obter a listagem dos arquivos de entrada e saída produzidos neste estudo de caso. O código *JAVA* das aplicações desenvolvidas encontra-se, de forma parcial, no Anexo E.

7.4. Apresentação dos Resultados

A partir do cenário apresentado na seção anterior, procurou-se avaliar a arquitetura sob duas perspectivas: 1) Perspectiva Funcional e 2) Perspectiva do Desempenho. No primeiro caso, foi verificada a consistência entre as chamadas da API do RBPEP editadas e as mensagens COPS correspondentes, juntamente com a decisão do PDP esperada baseada nos objetos de políticas contidos no repositório LDAP e na informação sobre as sessões de usuários ativas contidas na Base de Estados. No segundo caso, procurou-se estimar o desempenho do sistema, considerando o tempo médio que o PDP levou para produzir a

decisão de política a partir do momento do recebimento da mensagem do RBPEP, focando, desta forma, a avaliação nos tempos médios relacionados com a execução dos algoritmos e com a interação com o servidor LDAP e a Base de Estados. Neste caso, não foram considerados os tempos relacionados ao tráfego das mensagens nas conexões TCP/COPS.

7.4.1. Avaliação Funcional

A Tabela 7.10 apresenta os arquivos *App1.in* e *App1.out*, respectivamente, o arquivo de entrada proveniente da *Aplicação 1* e o arquivo de *log* gerado pelo PDP com as mensagens e as decisões de políticas correspondentes. Estão sendo apresentadas apenas as cinco primeiras chamadas.

Inicialmente, a aplicação solicita a abertura do serviço de política RBPIM através da chamada *RBPEP_OPEN()*, o que faz com que o RBPEP envie uma mensagem *OPN* ao PDP passando seu identificador *(PEPID) app1*. Como a solicitação foi aceita, o PDP emitiu uma mensagem *CAT* ao RBPEP, indicando, também, que não será utilizado o suporte *Keep Alive* do COPS na conexão (objeto *Keep Alive* com valor nulo).

Na linha 2 da tabela, a aplicação solicita a abertura de uma sessão para o usuário *Maria*. Como resultado, o RBPEP gera o *SessionID* da sessão (app1_1) e edita uma mensagem REQ ao PDP indicando a abertura de sessão na fase 1. O PDP responde com uma mensagem DEC informando que a abertura de sessão fase 1 foi concluída com sucesso e que o usuário *Maria* já possui outras quatro sessões abertas e terá, para a sessão corrente, os papéis *Atendente* e *Caixa* compondo a sua lista de papéis elegíveis.

Na linha 3, a aplicação solicita a ativação dos papéis *Caixa* e *Supervisor*. Isto faz com que o RBPEP edite a mensagem REQ correspondente ao PDP. Note que os papéis selecionados não estão incluídos na lista de papéis elegíveis recebidos na chamada 2, o que faz com que o PDP negue esta ativação, notificando o RBPEP com a mensagem DEC de erro mostrada. Em uma segunda tentativa, mostrada na linha 4, a aplicação agora solicita a ativação dos papéis *Atendente* e *Caixa*. Neste caso, o PDP aceita a solicitação e notifica isto ao RBPEP através da mensagem DEC, dando por concluída a criação de sessão, completada com o encerramento da fase 2 de seleção de papéis.

Por fim, a linha 5 apresenta uma chamada de verificação de acesso. Neste caso, a aplicação precisa verificar se o usuário da sessão indicada por *app1_1* (*Maria*) pode executar a operação *AbrirConta* no aplicativo *Gerenciador de Clientes*. Pela mensagem DEC enviada pelo PDP ao RBPEP, o acesso foi autorizado.

	App1.in	App1.out
1	RBPEP_OPEN()	RBPEP->PDP: |OPN |PEP Iden.: (PepID=app1)| PDP->RBPEP: |CAT |Keep-Ali.|
2	RBPEP_CreateSession(in: Maria)	RBPEP->PDP: |REQ |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de recursos, MTYPE=Criacao de Sessao Fase 1) |Client SI: (Maria)| PDP->RBPEP: |DEC |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos, MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=4) |Decision: (Client SI.=Atendente,Caixa)| RBPEP->PDP: |RPT |Handle: (SessionID=app1_1) |Repo.Type: (Sucesso)|
3	RBPEP_SelectedRoles(in: app1_1, Caixa, Supervisor)	RBPEP->PDP: |REQ |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos, MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Supervisor)| PDP->RBPEP: |DEC |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos, MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)| RBPEP->PDP: |RPT |Handle: (SessionID=app1_1) |Repo.Type: (Sucesso)|
4	RBPEP_SelectedRoles(in: app1_1, Caixa, Atendente)	RBPEP->PDP: |REQ |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos, MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Atendente_)| PDP->RBPEP: |DEC |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos, MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)| RBPEP->PDP: |RPT |Handle: (SessionID=app1_1) |Repo.Type: (Sucesso)|
5	RBPEP_CheckAccess(in: app1_1, AbrirConta, dlm1ApplicationSystem. dlmName =GerCliente)	RBPEP->PDP: |REQ |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos, MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem. dlmName=GerCliente)| PDP->RBPEP: |DEC |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos, MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)| RBPEP->PDP: |RPT |Handle: (SessionID=app1_1) |Repo.Type: (Sucesso)|

Tabela 7.10 – Arquivos App1.in e App1.out

A avaliação funcional exemplificada nesta subseção foi repetida para todos os cenários de utilização contidos nos arquivos de *App1.in* até *App20.in*. Os resultados observados foram positivos, com as respostas do PDP estando corretas em relação às regras de política RBPIM estabelecidas para este estudo de caso.

7.4.2. Avaliação quanto ao Desempenho

Com intuito de realizar uma avaliação do desempenho mais próxima da realidade, a partir do cenário mostrado na Figura 7.5, onde vinte aplicações simultaneamente requerem o serviço de política RBPIM, foram realizados 6 baterias de testes, tendo-se introduzido uma faixa de atraso gerada aleatoriamente entre as chamadas dos arquivos de entrada (*.in*), conforme sumarizado na Tabela 7.11. Estas 6 baterias de testes foram repetidas duas vezes. Na primeira vez a aplicação do PDP foi executada em uma máquina *Pentium IV 1.5 Ghz* 256 *Mb RAM* com sistema operacional *Windows 2000 Server*. Na segunda vez a aplicação do PDP foi executada em uma máquina *Dual Pentium III 1.5Ghz* com *2 Gb RAM* com sistema operacional *Linux RedHat 7.3. As* 20 aplicações e seus RBPEPs foram executados simultaneamente em uma máquina *Pentium IV 1.5 Ghz* com 256 *Mbytes* de *RAM*. Uma máquina *Pentium III 350 Mhz 128 Mb RAM* foi utilizada para a execução do serviço de diretório LDAP.

N_o Bateria de Teste	**Faixa de Atraso (segundos)**	**Taxa Média de requisições por segundo**
1	5 a 10	2,7
2	4 a 8	3,3
3	3 a 6	4,4
4	2 a 4	6,7
5	1 a 2	13,3
6	0 a 1	40,0

Tabela 7.11 – Baterias de testes para avaliação do desempenho

Como pode ser visto nesta tabela, os atrasos introduzidos permitem verificar o tempo de resposta do PDP às requisições de política RBPIM quando ele está submetido aos mais diferentes níveis de estresse. A bateria de teste número 1 realiza a avaliação no cenário de carga mais leve para o PDP, pois utiliza uma faixa de atraso de 5 a 10 segundos entre as chamadas, o que, em média, faz com que o PDP receba em torno de 2,7 requisições por segundo. A bateria de teste número 6, por sua vez, submete o PDP ao máximo nível de carga, pois foram introduzidos atrasos da ordem de zero a um segundo entre as chamadas das aplicações. Cabe ressaltar que, apesar das faixas de atraso terem sido especificadas em

segundos, os valores aleatórios gerados em cada uma delas tiveram sua granularidade em milisegundos.

As Tabelas de 7.12, 7.13 e 7.14 apresentam os tempos médios e máximos colhidos para, respectivamente, as três principais chamadas da API do RBPEP, ou seja, para as duas fases que compõem a abertura de sessão e para a verificação de acesso. Estas médias foram calculadas a partir dos tempos observados em cada um dos vinte arquivos de *log* (*.out*) gerados em cada bateria, para o caso do PDP rodando no ambiente *Windows*[14].

N_o Bateria de Teste	Tempo Médio	Desvio Padrão	Tempo Máximo	Tempo Médio (chamadas mal sucedidas)
1	71,10	41,81	281,00	0,00
2	67,88	41,85	297,00	8,35
3	70,27	44,51	312,00	11,45
4	98,05	55,70	312,00	23,48
5	145,89	116,74	547,00	21,47
6	542,29	333,82	1093,00	47,31

Tabela 7.12 – Tempos obtidos na chamada *RBPEP_CreateSession* (milisegundos)

N_o Bateria de Teste	Tempo Médio	Desvio Padrão	Tempo Máximo	Tempo Médio (chamadas mal sucedidas)
1	22,70	10,80	47,00	0,00
2	21,75	10,53	47,00	18,00
3	21,92	9,21	47,00	16,00
4	26,62	15,65	78,00	27,42
5	41,84	39,73	234,00	24,98
6	114,87	76,88	375,00	32,42

Tabela 7.13 – Tempos obtidos na chamada *RBPEP_SelectedRoles* (milisegundos)

Para uma aferição mais precisa, no cálculo dos tempos médios foram considerados apenas as chamadas nas quais o PDP conseguiu executar o procedimento por completo. Por exemplo, caso a aplicação solicite a abertura de uma sessão de um usuário inválido, o PDP

[14] Os tempos obtidos nos ambientes Windows e Linux foram semelhantes. Desta forma, optou-se por mostrar os tempos de apenas um deles.

aborta o processo de abertura logo no início, resultando, obviamente, em um tempo de resposta bastante inferior em relação as demais chamadas. Para evidenciar este fato, também são apresentados estes tempos médios, os quais foram denominados nestas tabelas de chamadas mal sucessidas.

N_o Bateria de Teste	Tempo Médio	Desvio Padrão	Tempo Máximo	Tempo Médio (chamadas mal sucedidas)
1	28,89	12,61	94,00	----
2	32,77	21,10	141,00	----
3	33,12	22,66	141,00	----
4	40,65	27,10	141,00	----
5	72,23	90,31	578,00	----
6	247,41	232,35	1141,00	----

Tabela 7.14 – Tempos obtidos na chamada *RBPEP_CheckAccess* (milisegundos)

A Figura 7.6 sumariza os tempos médios apresentados nas tabelas 7.12, 7.13 e 7.14. Como pode ser observado, nas seis baterias de testes realizadas, os maiores tempos de resposta do PDP estão relacionados com a chamada *RBPEP_CreateSession*. Este resultado era esperado, uma vez que os procedimentos relacionados com esta chamada envolvem muito esforço do PDP, entre os quais a obtenção da informação de papéis herdados, a verificação e a eliminação de papéis conflitantes por *SSD* e a validação do período de ativação de cada papel. Por outro lado, também deve ser considerado o fato que neste Estudo de Caso foram utilizados poucos objetos de permissão, o que contribuiu para que os tempos relativos à chamada *RBPEP_CheckAccess* se apresentassem inferiores aos da chamada *RBPEP_CreateSession.*

De maneira geral, os tempos de resposta do PDP se apresentaram satisfatórios. Note que, nas baterias de 1 a 5, onde faixas de atraso entre as chamadas variaram de 1 a 10 segundos, os tempos de respostas obtidos foram praticamente iguais, sempre abaixo dos 200 ms. Esta é uma informação importante, pois indica que, para vinte aplicações utilizando o serviço de política RBPIM simultaneamente, onde, em função das várias faixas de atrasos, as taxas de chegadas de requisições ao PDP também variariam bastante, foram obtidas respostas semelhantes. Isto denota um desempenho bastante adequado do PDP para taxas de chegadas de requisições comumente observadas em situações reais.

A bateria de teste 6 foi utilizada para ser observado uma eventual saturação do servidor. A faixa de atraso empregada de 0 a 1 segundo leva a situações extremas, não comumente observadas em cenários reais do ponto de vista de interação entre um cliente e uma aplicação de controle de acesso. No entanto, ela permitiu avaliar casos em que a taxa de chegada de requisições é alta, tipicamente observadas em situações em que o servidor deve atender a muitos clientes simultaneamente.

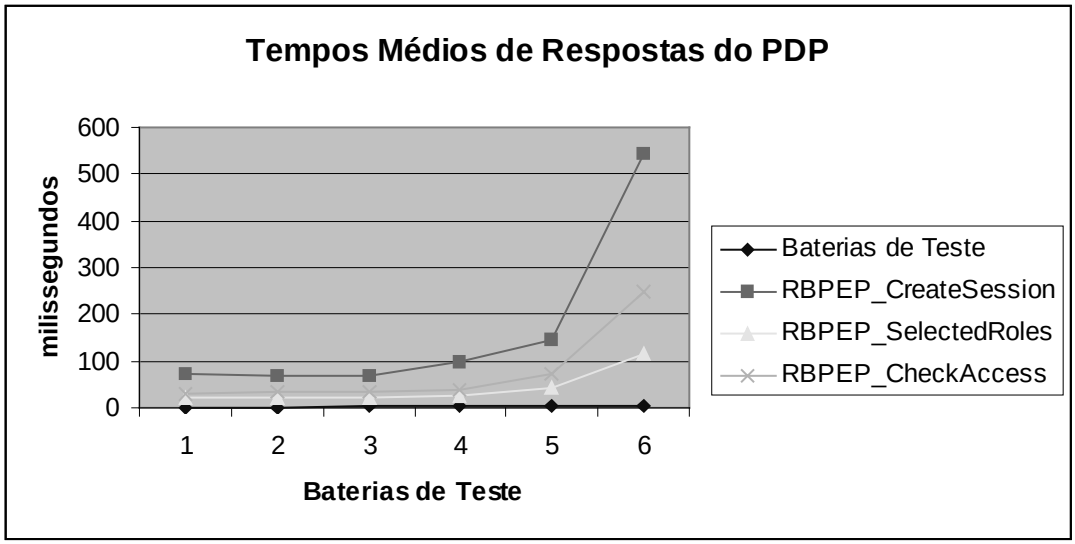

Figura 7.6 – Tempos Médios obtidos para os cenários do Estudo de Caso

Depois destes testes iniciais, o número de objetos RBPIM foi aumentado. Cada objeto RBPIM afeta de modo diferente o tempo de resposta das chamadas da API do RBPEP. Por causa da flexibilidade introduzida pela associação UA na abordagem adotada neste trabalho, o número de papéis afeta de forma significativa a chamada *RBPEP_CreateSession*. Aumentar o número de papéis de cinco para vinte praticamente dobra o tempo médio de resposta. Por outro lado, o efeito de aumentar o número de objetos de SSD não é importante. O tempo de resposta das outras chamadas não é afetado, pois os papéis ativados pelo usuário estão armazenados na Base de Estados para as subsequentes chamadas. A chamada *RBPEP_SelectedRoles* praticamente não é afetada pelo número de objetos de DSD. Além disso, ela não é afetada pelos outros objetos RBPIM. Pela análise dos algoritmos apresentados no Capítulo 6, pode-se supor que o número de objetos de permissão associados com os papéis deveria afetar a chamada *RBPEP_CheckAccess*. Entretanto, os testes mostraram que o aumento do número médio de permissões por papel de dois a dez não tem efeito significante no tempo de resposta. A justificativa para este resultado é que os passos 14 e 15 do algoritmo

da chamada *RBPEP_CheckAccess* são resolvidos com uma única consulta LDAP. Esta consulta cria um filtro complexo combinando as condições de todos os objetos de permissão. Similarmente, em todas as chamadas, o aumento do número de condições associadas com o papel ou permissão não teve efeito significante, por causa do fato de que todas as condições DNF e CNF também são transformadas em uma única consulta LDAP.

7.5. Conclusões do Capítulo

O presente Capítulo realizou um estudo de caso envolvendo cenários tipicamente observados em instituições financeiras. Este estudo de caso mostrou como o RBPIM poderia ser utilizado para se estabelecer uma política de controle de acesso baseada em papéis para controlar o acesso às operações financeiras e às operações relacionadas a clientes disponibilizadas pelas aplicações bancárias existentes no caso do banco hipotético analisado.

Com intuito de validar a arquitetura para implementação, os algoritmos, a API e as mensagens COPS definidas, foi desenvolvido um protótipo em linguagem JAVA seguindo uma arquitetura distribuída, a qual é centrada em duas entidades principais, o PDP e o PEP. Para realização desta avaliação foram utilizados vários arquivos de entrada, cada um contendo chamadas da API do RBPEP para simular diversos cenários de utilização. Uma análise dos arquivos de saída gerados pelo PDP mostrou que as decisões de política se apresentaram de acordo com a política de controle de acesso estabelecida para o caso do banco estudado. Além desta avaliação das decisões de políticas, a *performance* do sistema também foi avaliada em diversos cenários de carga do PDP nos ambientes *Windows* e *Linux*. Os tempos aferidos foram semelhantes para ambas as plataformas e se apresentaram com valores adequados para os cenários mais realistas dos experimentos.

Capítulo 8

Conclusões e Trabalhos Futuros

Este trabalho apresentou o RBPIM, um modelo para a definição e o gerenciamento de políticas de controle de acesso baseado em papéis em ambientes distribuídos. Derivado do PCIM, o RBPIM segue as definições RBAC contidas em [SANDHU 00], fornecendo suporte ao *CORE RBAC* e ao estabelecimento da hierarquia e separação de tarefas entre papéis. A justificativa para a definição do RBPIM a partir da extensão do PCIM pode ser resumida em dois fatores: 1) O PCIM permite a representação de políticas de uma forma padrão, possibilitando que *softwares* de fabricantes diferentes possam interpretar um mesmo conjunto políticas. Este aspecto é importante, pois há diversas situações onde um mesmo conjunto de políticas RBAC deve estar disponível para aplicações heterogêneas em um ambiente distribuído. Esta facilidade pode ser alcançada com a opção pelo *framework* do PDP/PEP, estratégia que foi adotada por este trabalho; 2) O *framework* do RBAC normalmente requer o acesso a informações sobre usuários, serviços, aplicações, etc. O CIM fornece modelos que permitem o armazenamento destas informações de forma padronizada em repositórios. Implementar o RBAC no PCIM permite, assim, utilizar a informação já existente em repositórios do CIM, possibilitando que seja mantida uma única fonte de informação sobre as entidades envolvidas na política, sem que haja a necessidade, portanto, de criar novos objetos de forma redundante para representar as mesmas entidades no contexto das políticas do RBPIM.

O mapeamento das associações *UA (Usuários-Papéis)* e *PA (Permissões-Papéis)* do RBAC fornecida pelo RBPIM dá um grau adicional de flexibilidade à criação de políticas RBAC. Na estratégia adotada, a associação de usuários a papéis e de recursos a permissões é implementada através de expressões booleanas, o que permite, por exemplo, que em uma única regra vários usuários e recursos sejam associados, respectivamente, a um papel e a uma permissão em vez do mapeamento direto entre usuários e papéis e entre recursos e permissões. Seguindo as definições do PCIM, o RBPIM também permite que sejam

associados períodos de ativação a papéis, um aspecto importante na descrição de políticas de controle de acesso organizacionais.

Uma das contribuições importantes deste trabalho foi a apresentação de um roteiro completo da utilização do *framework* do PCIM. A estratégia de extensão do PCIM foi mostrada no Capítulo 4, com a apresentação do próprio RBPIM. O Capítulo 5 apresentou um esquema de diretório que pode ser utilizado para o armazenamento da informação de política RBPIM em serviços de diretório LDAP, estratégia recomendada pelas especificações do DMTF. Por fim, o Capítulo 6 apresentou uma arquitetura para a implementação do RBPIM, a qual foi baseada nos padrões estabelecidos pelo *framework* de políticas, envolvendo o PDP, o PEP e o protocolo COPS.

O resultado dos experimentos realizados com a implementação do cenário contido no estudo de caso apresentado no Capítulo 7 mostrou que a adoção do modelo *Outsourcing* do COPS para a arquitetura do PDP/PEP é adequada para aplicações que necessitam de decisões tomadas como resposta a eventos de usuário. Este é o caso de uma típica interação entre uma aplicação cliente consumidora de política RBPIM e o servidor de política, cenário ao qual este trabalho foi direcionado. Porém, para o estabelecimento de políticas que demandem uma interação mais intensa entre o PDP e PEP (por exemplo, políticas de *QoS*) é possível que o modelo *Outsourcing* não seja adequado, requerendo uma migração deste ao modelo *Provisioning* ou a um modelo híbrido envolvendo os dois. Um trabalho futuro nesta direção fará um estudo sobre a viabilidade da adoção dessas alternativas.

O modelo e a arquitetura do RBPIM propostos neste trabalho se apresentam como uma opção viável para a aplicação de políticas RBAC em ambientes distribuídos. A partir desta proposta, alguns pontos importantes podem ser colocados para o prosseguimento deste trabalho, os quais serão relacionados nos parágrafos que seguem.

O PCIMe propõe algumas modificações às especificações do PCIM. Este trabalho utilizou apenas uma destas novas proposições, a hierarquia de classes para o estabelecimento de expressões booleanas com a semântica <*Variable*> *MATCH* <*Value*>. A viabilidade da incorporação das outras alterações, como o novo esquema para o encadeamento de regras de políticas, deve ser analisada.

As especificações do RBAC contidas em [SANDHU 00] definem uma série de funções necessárias para a criação e a manutenção dos componentes do modelo. No entanto, a API do RBPEP e os algoritmos do PDP apresentados no Capítulo 6 abordaram as funções de

suporte à operação do sistema durante a interação de um usuário, como a criação e o fechamento de uma sessão RBPIM e a verificação de acesso, as quais, sem dúvida, são as principais funções. Neste contexto, para atender de forma completa às especificações do modelo de referência do NIST, recomenda-se para um trabalho futuro a incorporação das funções de revisão e as funções administrativas ao *framework* do RBPIM.

O QPIM – *QoS Policy Information Model* – é uma extensão do PCIM para a descrição de políticas de qualidade de serviço [STR 01]. Uma estratégia viável que pode ser adotada como prosseguimento deste trabalho de pesquisa é a concepção de um novo modelo, a partir da extensão do RBPIM e do QPIM, para o estabelecimento de políticas de controle de acesso baseados em papéis no contexto das políticas de qualidade de serviço.

Este trabalho não discutiu os problemas que poderiam acontecer no caso do PDP, eventualmente, ter que interromper o serviço ou no caso de ele não poder atender ao serviço de política relacionado a um dado tipo de cliente requerido pelo PEP. Para estas situações, o *framework* do COPS define mensagens que podem ser utilizadas para o redirecionamento de chamadas de PEPs, permitindo que estas solicitações sejam encaminhadas para outros PDPs. Outra questão não abordada neste trabalho é o caso de uma falha do PDP. Trabalhos futuros avaliarão soluções para a introdução da redundância no serviço do PDP.

Por fim, foi deixado a cargo de um trabalho futuro a discussão sobre mecanismos que possam garantir conexões seguras entre o PDP e RBPEP e, ainda, a incorporação ao *framework* do RBPIM do suporte dado pelas mensagens *Keep-Alive* e de integridade do COPS.

Referências Bibliográficas

[BARTZ 01] Bartz, L.S.; *CADS-2 Information Model, not published, Internal Revenue Service, 2001.*

[BARTZ 97] Bartz, L.S; *LDAP Schema for Role Based Access Control, Internet Draft, Internal Revenue Service, October* 1997.

[BOOCH 00] Booch, Gary; Jacobson, Ivar;Rumbaugh, James. UML-Guia do Usuário. Ed. Campus. Rio de Janeiro, 2000.

[BS 01] Bearden, Mark; Sachin, Garg; *Integrating Goal Specification in Policy-Based Management, Workshop on Policies for Distributed Systems ans Networks, Bristol,U.K., January, 2001*.

[CISCO 99] Cisco Systems Inc.; *Policy-Based Management: Bridging the Gap;* 1999.

[DMTF 00] *DMTF - Distributed Management Task Force, Inc. Common Information Model (CIM) Specification Version 2.4, June 2000*.

[DMTF 02] *LDAP Schema for the CIM 2.4 Core Information Model v1.0. DMTF, May 2002.*

[DMTF 99] *DMTF - Distributed Management Task Force, Inc. Common Information Model (CIM) Specification Version 2.2, June 1999*.

[FB 97] Ferraiolo, D.F.; Barkley, John; *Specifying and Managing Role-Based Access Control within a Corporate Intranet; NIST, 1997;*

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

[FK 92] Ferraiolo, D.; Kuhn, R.; *Role-Based Access Control. In Proc. of the NIST-NSA Nat. (USA) Comp. Security Conf., pp 554-563, 1992.*

[FOW 01] Fraga, Joni da Silva; Obelheiro, Rafael Rodrigues; Westphall, Carla Merkle; Controle de Acesso Baseado em Papéis para o Modelo Corba de Segurança; 19º.

Simpósio Brasileiro de Redes de Computadores, Maio 2001.

[GUTZMANN 01] Gutzmann, Kurt; *Access Control and Session Management in the HTTP Environment; IEEE Internet Computing, Jan-Fev 2001.*

[MES 01] Moore, B; Ellenson, E; Strassner, J.; *Policy Core LDAP Schema, Policy Framework Working Group,Internet Draft,November 2001.*

[MOORE 02] MOORE, B.; Policy Core Information Model Extensions; Policy Framework Working Group; INTERNET-DRAFT, May 2002.

[NETSCAPE 99] Netscape Corporation*; Netscape Directory Server Deployment Guide, 1999.*

[NVL 97] Novell. *NDS White Paper*, URL: http://novell.com/products/nds/wpnds.html, 1997.

[OMG 99] OMG, *Security Service Specification, v1.7. OMG Doc. 99-12-02,Dec. 1999.*

[ON 94] Nyanchama, M. and Osborn, S.; *Access rights administration in role-based security systems. In J. Biskup, M. Morgenstern, and C. E. Landwehr, editors, Database Security, VIII: Status and Prospects, pages 37-56. North-Holland, 1994.*

[RFC 1510] Kohl, J., Newman. C.. *The Kerberos Authentication Service (v5). Request For Comments 1510, Setember 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.; *A LDAP URL Format. Request For Comments 1959, June 1996.*

[RFC 2251] Wahl, M.; Howes, T.; Killie, S.; *Lightweight Directory Access Protocol (v3), Request for Comments 2251, December*, 1997.

[RFC 2252] Wahl, M.; Coulbeck, A.; Howes, T. ; Kille, S.*; Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions. Request for Comments 2252, December 1997.*

[RFC 2748] Boyle, J.; Cohen, R.; *The COPS (Common Open Policy Service) Protocol ,*

RFC 2748, January 2000.

[RFC 2753] Yavatkar, R., Pendarakis, D. and R. Guerin; *A Framework for Policy-based Admission Control, RFC 2753, January 2000*.

[RFC 3060] MOORE, B. et al.; *Policy Core Information Model, IETF RFC 3060, February 2001*.

[RFC 3084] *COPS Usage for Policy Provisioning (COPS-PR) Network Working Group Request for Comments: 3084, March 2001*

[RFC 3198] WESTERINEN, A. et al.; *Terminology for Policy-Based Management, Request for Comments 3198, Policy Framework Working Group, November 2001*.

[SANDHU 00] Sandhu, R.S. et al.; *A Proposed Standard for Role-Based Access Control, Nist, December 2000.*

[SC 96] Sandhu, R.S.; Coyne,E.J.; *Role-Based Access Control Models, Computer vol.29, no.2, Feb.1996, pp. 38-47.*

[SDS 01] Sun MicroSystems, INC.; *iPlanet Directory Server 5.1 Installation Guide , December 2001*.

[SFK 00] Sandhu, R. S.; Ferraiolo D. F.; Kuhn, D. R.; *The Nist Model for Role Based Access Control: Towards a Unified Standard; In Proc. 5th ACM Workshop on RBAC, pages 47-63, 2000.*

[SL 98] D.A. Solomon, *The Windows NT Kernel Architecture,* IEEE COMPUTER Vol. 31, No. 10: OCTOBER 1998, pp. 40-47

[SS 94] Sandhu R. S., Samarati P. S., *Access Control: Principles and Practice, IEEE Communications, vol. 32, pp. 40-48, Sept. 1994.*

[STR 99] Strassner, John. *Policy-Based Networking Standards Report. Policy Workshop. HP-Laboratories. Bristol U.K., November 1999*.

[STR 01] Strassner J. et al; *Policy QoS Information Model, IETF internet-draft, November 2001*.

[SUN 01] Sun MicroSystems, INC.; *RBAC in the Solaris Operating Environment -*

White Paper; Sun MicroSystems, April 2001.

[WALKER 02] Walker, Jesse; Kulkarni, Amol; *COPS Over TLS; Internet Draft, June 2002*.

[WFW 00] Westphall, C. M.; Fraga, J. S.; Wangham, M. S.; PoliCap – Um Serviço de Política para o Modelo CORBA de Segurança. In Anais do 18º SBRC, pages 355-370, maio 2000.

[X500 88] *The Directory: Overview of Concepts, Models and Service. CCITT Recommendation X.500, 1988.*

[YU 98] Yu, Xinzhong. *Directory Enabled Networks. Helsinki University of Technology. December 1998*

ANEXO A - ESQUEMA DE DIRETÓRIO DO RBPIM

Um esquema de diretório contém as definições formais das classes a partir das quais podem ser criados objetos (entradas) no diretório. O esquema também contém a definição de cada atributo que pode ser utilizado nas definições das classes. Este anexo apresenta o esquema de diretório proposto no Capítulo 5. Este esquema de diretório foi aplicado ao servidor LDAP utilizado na implementação do protótipo, o *SUN One Directory Server*. A listagem apresentada a seguir representa o conteúdo do arquivo *99user.ldif*, no qual o servidor LDAP da *SUN* mantém a informação resultante da extensão do esquema realizada pelo usuário.

1.1. Classes

Cada linha da tabela abaixo apresenta uma classe do esquema. Para cada classe tem-se, nesta ordem, o identificador da classe (*OID*), o nome da classe, uma descrição breve da classe, a classe imediatamente superior na hierarquia, o tipo da classe (*ABSTRACT*, *STRUCTURAL* e *AUXILIARY*), a lista de atributos (tipos: *MAY*-Opcionais e *MUST*-Obrigatórios) e a origem da classe. Por exemplo, a classe *rbpimRole* abaixo possui o identificador *rbpimRole-oid*, é derivada da classe *pcimRule* e é do tipo *STRUCTURAL*, não possuindo atributos obrigatórios, apenas os atributos opcionais *rbpimRoleName* e *rbpimInheritedRoles*.

objectClasses: (dlm1ManagedElement-oid NAME 'dlm1ManagedElement' SUP top ABSTRACT MAY (dlmCaption $ dlmDescription) X-ORIGIN 'user defined')
objectClasses: (pcimPolicy-oid NAME 'pcimPolicy' DESC 'Classe base da hierarquia de politica' SUP dlm1ManagedElement ABSTRACT MAY cn X-ORIGIN 'user defined'
objectClasses: (pcimGroup-oid NAME 'pcimGroup' DESC 'Container para agrupamento de objetos de papeis, de permissoes e de separacao de tarefas' SUP pcimPolicy ABSTRACT MAY pcimGroupName X-ORIGIN 'user defined')
objectClasses: (pcimGroupInstance-oid NAME 'pcimGroupInstance' DESC 'Classe estrutural para agrupamento' SUP pcimGroup STRUCTURAL X-ORIGIN 'user defined')
objectClasses: (pcimGroupAuxClass-oid NAME 'pcimGroupAuxClass' DESC 'Classe Auxiliar para agrupamento' SUP pcimGroup AUXILIARY X-ORIGIN 'user defined')
objectClasses: (pcimRule-oid NAME 'pcimRule' DESC 'Classe abstrata para repre sentacao da semantica if-condition-then-action' SUP pcimPolicy ABSTRACT MAY (pcimRuleEnabled $ pcimRuleConditionListType $ pcimRuleConditionList $ pcim RuleActionList $ pcimRuleValidityPeriodList $ pcimRulePriority) X-ORIGIN 'user defined')
objectClasses: (rbpimRole-oid NAME 'rbpimRole' DESC 'Classe estrutural que representa um papel RBPIM' SUP pcimRule STRUCTURAL MAY (rbpimRoleName $ rbpimInheritedRoles) X-ORIGIN 'user defined')
objectClasses: (rbpimPermission-oid NAME 'rbpimPermission' DESC 'Classe estrutural que representa uma permissao RBPIM' SUP pcimRule STRUCTURAL MAY rbpimP ermissionName X-ORIGIN 'user defined')
objectClasses: (rbpimDSD-oid NAME 'rbpimDSD' DESC 'Classe estrutural que representa um relacionamento de DSD entre papeis' SUP pcimPolicy STRUCTURAL MAY (rbpimDSDName $ rbpimRoleSet $ rbpimCardinality) X-ORIGIN 'user defined')
objectClasses: (rbpimSSD-oid NAME 'rbpimSSD' DESC 'Classe estrutural que representa um relacionamento

de SSD entre papeis' SUP pcimPolicy STRUCTURAL MAY (rbpimSSDName $ rbpimRoleSet $ rbpimCardinality) X-ORIGIN 'user defined')
objectClasses:(pcimRuleConditionAssociation-oid NAME 'pcimRuleConditionAssociation' DESC 'Classe estrutural que representa a associacao entre papeis e condicoes e entre permissoes e condicoes' SUP pcimPolicy STRUCTURAL MAY (pcimConditionName $ pcimConditionGroupNumber $ pcimConditionNegated) X-ORIGIN 'user defined')
objectClasses: (pcimRuleActionAssociation-oid NAME 'pcimRuleActionAssociation ' DESC 'Classe estrutural que representa a associacao entre papeis e permissoes e entre permissoes e operacoes' SUP pcimPolicy STRUCTURAL MAY pcimAction Name X-ORIGIN 'user defined')
objectClasses: (pcimRuleValidityAssociation-oid NAME 'pcimRuleValidityAssociation' DESC 'Classe estrutural que representa a associacao entre papeis e seus periodos de ativacao' SUP pcimPolicy STRUCTURAL MAY pcimValidityConditionName X-ORIGIN 'user defined')
objectClasses: (rbpimConditionAssociation-oid NAME 'rbpimConditionAssociation ' DESC 'Classe estrutural que representa a associacao entre uma condicao e o par variavel/valor relacionados' SUP pcimPolicy STRUCTURAL MAY rbpimConditionName X-ORIGIN 'user defined')
objectClasses: (pcimGroupContainmentAuxClass-oid NAME 'pcimGroupContainmentAuxClass' SUP top STRUCTURAL MAY pcimGroupsAuxContainedSet X-ORIGIN 'user defined')
objectClasses: (rbpimSSDContainmentAuxClass-oid NAME 'rbpimSSDContainmentAuxClass' DESC 'Representa a agregacao entre um objeto container de grupo e objetos SSD contidos' SUP top AUXILIARY MAY rbpimSSDAuxContainedSet X-ORIGIN 'user defined')
objectClasses: (rbpimSSDContainmentAuxClass-oid NAME 'rbpimSSDContainmentAuxClass' DESC 'Representa a agregacao entre um objeto container de grupo e objetos SSD contidos' SUP top AUXILIARY MAY rbpimSSDAuxContainedSet X-ORIGIN 'user defined')
objectClasses: (rbpimDSDContainmentAuxClass-oid NAME 'rbpimDSDContainmentAuxClass' SUP top STRUCTURAL MAY rbpimDSDAuxContainedSet X-ORIGIN 'user defined')
objectClasses: (pcimConditionAuxClass-oid NAME 'pcimConditionAuxClass' DESC 'Representa uma condicao a ser avaliada' SUP top AUXILIARY X-ORIGIN 'user defined')
objectClasses: (pcimTPCAuxClass-oid NAME 'pcimTPCAuxClass' DESC 'Permite a especificacao de periodos de tempo para habilitar ou desabilitar um papel RBPIM' SUP pcimConditionAuxClass AUXILIARY MAY (pcimTPCTime $ pcimTPCMonthOfYearMask $ pcimTPCDayOfMonthMask $ pcimTPCDayOfWeekMask $ pcimTPCTimeOfDayMask $ pcimTPCLocalOrUtcTime) X-ORIGIN 'user defined')
objectClasses: (pcimActionAuxClass-oid NAME 'pcimActionAuxClass' DESC 'Representa uma acao a ser executada' SUP top AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimSimplePolicyConditionAuxClass-oid NAME 'rbpimSimplePolicyConditionAuxClass' DESC 'Representa uma expressao com a semantica <variavel> MATCH <valor>' SUP pcimConditionAuxClass AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimAssignerPermissionAuxClass-oid NAME 'rbpimAssignerPermissionAuxClass' DESC 'Representa uma referencia DN um objeto de permissao RBPIM' SUP pcimActionAuxClass AUXILIARY MAY rbpimPermissionDN X-ORIGIN 'user defined')
objectClasses: (rbpimAssignerOperationAuxClass-oid NAME 'rbpimAssignerOperationAuxClass' DESC 'Representa as operacoes associadas a um objeto rbpimPermission' SUP pcimActionAuxClass AUXILIARY MAY rbpimOperationList X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyVariable-oid NAME 'rbpimPolicyVariable' DESC 'Representa uma variavel a ser utilizada em uma condicao' SUP top AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyValue-oid NAME 'rbpimPolicyValue' DESC 'Representa um valor a ser avaliado em uma condicao' SUP top AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyExplicitVariable-oid NAME 'rbpimPolicyExplicitVariable' DESC 'Representa uma variavel cuja semantica e' SUP rbpimPolicyVariable AUXILIARY MAY (rbpimModelClass $ rbpimModelProperty) X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyImplicitVariable-oid NAME 'rbpimPolicyImplicitVariable' DESC 'Variavel cuja semantica nao e' SUP rbpimPolicyVariable AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyDestinationPortVar-oid NAME 'rbpimPolicyDestinationPortVar' DESC 'Valor inteiro na faixa 0..65535 que representa uma porta TCP/UDP' SUP rbpimPolicyImplicitVariable AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyIPProtocolVar-oid NAME 'rbpimPolicyIPProtocolVar' DESC 'Valor inteiro que representa o protocolo de nível superior contido no pacote' SUP rbpimPolicyImplicitVariable AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicySourceMACVar-oid NAME 'rbpimPolicySourceMACVar' DESC 'Valor que

representa um endereco MAC' SUP rbpimPolicyImplicitVariable AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyDestinationMACVar-oid NAME 'rbpimPolicyDestination MACVar' DESC 'Valor que representa um endereco MAC' SUP rbpimPolicyImplicitVariable AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicySourceIPv4Var-oid NAME 'rbpimPolicySourceIPv4Var' DESC 'Valor que representa um endereco IPv4' SUP rbpimPolicyImplicitVariable AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyDestIPv4Var-oid NAME 'rbpimPolicyDestIPv4Var' DESC 'Valor que representa um endereco IPv4' SUP rbpimPolicyImplicitVariable AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicySourceIPv6Var-oid NAME 'rbpimPolicySourceIPv6Var' DESC 'Valor que representa um endereco IPv6' SUP rbpimPolicyImplicitVariable AUXILIARY X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyIPv4AddrValue-oid NAME 'rbpimPolicyIPv4AddrValue' DESC 'Usada para manter uma lista de enderecos IPv4 e hostnames' SUP rbpimPolicyValue AUXILIARY MAY rbpimIPv4AddrList X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyIPv6AddrValue-oid NAME 'rbpimPolicyIPv6AddrValue' DESC 'Valor que representa conjuntos de enderecos IPv6' SUP rbpimPolicyValue AUXILIARY MAY rbpimIPv6AddrList X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyStringValue-oid NAME 'rbpimPolicyStringValue' DESC 'Valor que representa um conjunto de strings' SUP rbpimPolicyValue AUXILIARY MAY rbpimStringList X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyIntegerValue-oid NAME 'rbpimPolicyIntegerValue' DESC 'Representa um conjunto de inteiros' SUP rbpimPolicyValue AUXILIARY MAY rbpimIntegerList X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyMACAddrValue-oid NAME 'rbpimPolicyMACAddrValue' DESC 'Representa um conjunto de enderecos MAC' SUP rbpimPolicyValue AUXILIARY MAY rbpimMACAddrList X-ORIGIN 'user defined')
objectClasses: (rbpimPolicyBooleanValue-oid NAME 'rbpimPolicyBooleanValue' DESC 'Representa um conjunto de booleans' SUP rbpimPolicyValue AUXILIARY MAY rbpimBooleanList X-ORIGIN 'user defined')
objectClasses: (dlm1ManagedSystemElement-oid NAME 'dlm1ManagedSystemElement' DESC 'Classe base da hierarquia System Element' SUP dlm1ManagedElement ABSTRACT MAY (dlmInstallDate $ dlmName $ dlmStatus) X-ORIGIN 'user defined')
objectClasses: (dlm1LogicalElement-oid NAME 'dlm1LogicalElement' DESC 'Classe base da hierarquia de classes que representam todos componentes abstratos de um sistema' SUP dlm1ManagedSystemElement ABSTRACT X-ORIGIN 'user defined'
objectClasses: (dlm1System-oid NAME 'dlm1System' DESC 'Representa a agreGacao de um conjunto de ManagedSystemElements' SUP dlm1LogicalElement ABSTRACT X- ORIGIN 'user defined')
objectClasses: (dlm1ApplicationSystem-oid NAME 'dlm1ApplicationSystem' DESC 'Usada para representar uma aplicacao ou sistema de software que suporte uma funcao de negocio em particular.' SUP dlm1System STRUCTURAL X-ORIGIN 'user defined')

1.2. Atributos

Cada linha da tabela abaixo apresenta um atributo do esquema. Da mesma forma que para as classes, os dois primeiros valores definem o identificador e o nome, seguidos por uma breve descrição do atributo. Cada atributo possui uma sintaxe na sua definição. A sintaxe define como os valores do atributo são comparados durante uma operação de busca. [RFC 2252] define um conjunto de números de sintaxes de atributos padrão. Por exemplo, 1.3.6.1.4.1.1466.115.121.1.15 e 1.3.6.1.4.1.1466.115.121.1.12 representam, respectivamente, um String e um DN (*Distinguished name*). Cada atributo pode ter a capacidade de armazenar um único valor (*SINGLE-VALUE*) ou mais de um (*MULTI-VALUE*). Para a tabela abaixo, a não informação de nenhum destes dois valores, indica um atributo *MULTI-VALUE*.

attributeTypes: (pcimGroupName-oid NAME 'pcimGroupName' DESC ‘pcimGroupName' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')

attributeTypes: (rbpimConditionName-oid NAME 'rbpimConditionName' DESC 'Nome da expressao utilizada na condicao' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (dlmDescription-oid NAME 'dlmDescription' DESC 'dlmdescription' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
AttributeTypes: (rbpimPermissionName-oid NAME 'rbpimPermissionName' DESC 'Nome do objeto de permissao' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
AttributeTypes: (rbpimIntegerList-oid NAME 'rbpimIntegerList' DESC 'Lista de inteiros utilizados na condicao' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'user defined')
AttributeTypes: (pcimConditionGroupNumber-oid NAME ‘pcimConditionGroupNumber' DESC 'Valor utilizado para o agrupamento de condicoes' SYNTAX 1.3.6.1.4.1.166.115.121.1.27 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimValidityConditionName-oid NAME 'pcimValidityConditionName' DESC 'Nome do objeto que mantem periodos de ativacao para papeis' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimTPCMonthOfYearMask-oid NAME 'pcimTPCMonthOfYearMask' DESC 'String contendo os meses do ano possiveis para ativacao dos papeis' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (rbpimPermissionDN-oid NAME 'rbpimPermissionDN' DESC 'Referencia DN a um objeto de permissao associado ao papel' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimRuleEnabled-oid NAME 'pcimRuleEnabled' DESC 'Usado para habilitar ou desabilitar um papel' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimTPCTime-oid NAME 'pcimTPCTime' DESC 'String contendo um intervalo de tempo para ativacao dos papeis' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (rbpimRoleSet-oid NAME 'rbpimRoleSet' DESC 'Referncias DNs de objetos dos papeis pertencentes ao conjunto' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'user defined')
attributeTypes: (rbpimRoleName-oid NAME 'rbpimRoleName' DESC 'Papel RBPIM' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimConditionNegated-oid NAME 'pcimConditionNegated' DESC 'Utilizado para negar uma expressao' SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (dlmInstallDate-oid NAME 'dlmInstallDate' DESC 'Indica a data de instalacao do objeto' SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (rbpimSSDAuxContainedSet-oid NAME 'rbpimSSDAuxContainedSet' DESC 'Referencias DNs a objetos que representam conjuntos de papeis que possuem conflito por SSD' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'user defined')
attributeTypes: (rbpimBooleanList-oid NAME 'rbpimBooleanList' DESC 'Representa um conjunto de booleans' SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 X-ORIGIN 'user defined')
attributeTypes: (rbpimCardinality-oid NAME 'rbpimCardinality' DESC 'Cardinalidade do conjunto' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimTPCDayOfMonthMask-oid NAME 'pcimTPCDayOfMonthMask' DESC 'String contendo os dias do mes possiveis para ativacao dos papeis' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimActionName-oid NAME 'pcimActionName' DESC 'Nome da associacao entre papeis e permissoes e entre permissoes e operacoes' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (rbpimModelProperty-oid NAME 'rbpimModelProperty' DESC 'Nome do atributo do objeto que sera utilizado na condicao' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (rbpimInheritedRoles-oid NAME 'rbpimInheritedRoles' DESC 'Referencias DNs a outros objetos de papeis' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'user defined')
attributeTypes: (pcimRulesAuxContainedSet-oid NAME 'pcimRulesAuxContainedSet' DESC 'Referencias DNs a objetos que representam papeis e permissoes' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'user defined')
attributeTypes: (rbpimOperationList-oid NAME 'rbpimOperationList' DESC 'String contendo a lista operacoes associadas ao objeto' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'user defined')
attributeTypes: (rbpimDSDName-oid NAME 'rbpimDSDName' DESC 'Nome do conjunto de papeis que possuem DSD' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimRuleValidityPeriodList-oid NAME 'pcimRuleValidityPeriodList' DESC 'Referencias DNs

a objetos de periodos de ativacao de papeis' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'user defined')
attributeTypes: (pcimTPCLocalOrUtcTime-oid NAME 'pcimTPCLocalOrUtcTime' DESC 'Valor inteiro que indica qual o sistema de hora (Local ou UTC) utilizado' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (dlmCaption-oid NAME 'dlmCaption' DESC 'dlmcaption' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimRuleConditionListType-oid NAME 'pcimRuleConditionListType' DESC 'pcimRuleConditionListType' SYNTAX 1.3.6.1.4.1.1466.115.121.1.27 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimRuleConditionList-oid NAME 'pcimRuleConditionList' DESC 'pcimRuleConditionList' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'user defined')
attributeTypes: (rbpimSSDName-oid NAME 'rbpimSSDName' DESC 'Nome do conjunto de papeis que possuem SSD' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimConditionName-oid NAME 'pcimConditionName' DESC 'Nome do objeto de condicao' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (dlmName-oid NAME 'dlmName' DESC 'Indica o nome pelo qual o objeto e' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimGroupsAuxContainedSet-oid NAME 'pcimGroupsAuxContainedSet' DESC 'Referencias DNs de objetos pcimGroup que serao agregados' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'user defined')
attributeTypes: (rbpimDSDAuxContainedSet-oid NAME 'rbpimDSDAuxContainedSet' DESC 'Referencias DNs a objetos que representam conjuntos de papeis que possuem conflito por DSD' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'user defined')
attributeTypes: (rbpimMACAddrList-oid NAME 'rbpimMACAddrList' DESC 'Lista de strings contendo enderecos MAC utilizados na condicao' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'user defined')
attributeTypes: (rbpimIPv6AddrList-oid NAME 'rbpimIPv6AddrList' DESC 'Lista de enderecos IPv6' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'user defined')
attributeTypes: (pcimTPCDayOfWeekMask-oid NAME 'pcimTPCDayOfWeekMask' DESC 'String contendo os dias da semana possiveis para ativacao dos papeis' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (rbpimStringList-oid NAME 'rbpimStringList' DESC 'Lista de strings utilizados na condicao' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'user defined')
attributeTypes: (rbpimIPv4AddrList-oid NAME 'rbpimIPv4AddrList' DESC 'Lista de strings contendo enderecos IPv4 utilizados nas condicoes' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'user defined')
attributeTypes: (pcimTPCTimeOfDayMask-oid NAME 'pcimTPCTimeOfDayMask' DESC 'String contendo as horas do dia possiveis para ativacao dos papeis' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')
attributeTypes: (pcimRuleActionList-oid NAME 'pcimRuleActionList' DESC 'Referencias DNs a objetos de permissao e operacoes' SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 X-ORIGIN 'user defined')
attributeTypes: (dlmStatus-oid NAME 'dlmStatus' DESC 'Indica o status corrente do obejto' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'user defined')

ANEXO B - ESPECIFICAÇÕES COPS PARA O TIPO DE CLIENTE RBPIM

As recomendações contidas na RFC 2748 indicam que trabalhos subseqüentes, que se baseiam no protocolo COPS para a troca da informação de política, devem estabelecer padrões para os campos das mensagens do protocolo de forma específica a cada tipo de cliente. Este anexo resume os valores dos campos das mensagens COPS estabelecidos de forma específica ao tipo de cliente RBPIM. Trabalho futuros de implementação do RBPIM deverão seguir as especificações contidas neste anexo.

Local	Campo	Valor	Descrição
HEADER	CLIENT-TYPE	0x8000	Cliente de política RBPIM.
CONTEXT OBJECT	M-TYPE	0x01 0x02 0x03	Abertura de sessão Fase 1. Abertura de sessão Fase 2. Verificação de autorização de acesso.
DECISION OBJECT	COMMAND-CODE	0 1 2	Nenhuma decisão disponível. Requisição autorizada. Requisição negada.
REPORT-TYPE OBJECT	REPORT-TYPE	1 2	Sucesso. Falha.
ERROR OBJECT	ERROR SUB-CODE	0 100 101 102 103 104 105 106 107 108 109 110 111	Sucesso. Erro - Tipo de Cliente não suportado. Erro - RBPEP não autorizado. Erro - Serviço de política já aberto. Erro - Operação desconhecida. Erro - Campo RequestType inválido. Erro - Sessão já' aberta. Erro - Campo MessageType inválido. Erro - userID inválido . Fechamento do Serviço. Erro - Status da sessão inválido para esta OP. Erro - Seleção inválida de papéis. Erro - Papéis conflitantes para uma mesma sessão.
REASON OBJECT	REASON-CODE	0	Solicitação de encerramento de sessão.

ANEXO C – ARQUIVOS LDIF DO ESTUDO DE CASO DO BANCO ABC

O *LDAP Data Interchange Format* (LDIF) é um formato para definir entradas de diretórios no formato texto. Uma entrada *LDIF* especifica as seguintes informações:

- *O distinguished name (DN) que identifica a entrada*: O DN de uma entrada identifica a sua localização no diretório. Duas entradas nunca podem ter o mesmo DN.
- *As classes do esquema (object classes) utilizadas na entrada*: Uma classe define os atributos obrigatórios e permitidos para uma entrada.
- *Os atributos e seus valores correspondentes*: Os atributos e seus valores especificam a informação sobre uma entrada.

A listagem apresentada a seguir retrata o conteúdo do arquivo *BANCO_ABC.ldif*, o qual contém as entradas utilizadas no estudo de caso e na validação do protótipo apresentados no Capítulo 7.

```
dn: o=Banco_ABC, dc=com
objectclass: top
objectclass: organization
o: Banco_ABC

dn: ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: organizationalunit
ou: People

dn: cn= Carlos, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Carlos
sn: Machado
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: A1

dn: cn= Ana, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Ana
sn: Fernandes
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: A1

dn: cn= Joana, ou=People, o=Banco_ABC, dc=com
```

objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Joana
sn: Rodrigues
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: A1

dn: cn= Rubens, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Rubens
sn: Silva
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: A1

dn: cn= Marcos, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Marcos
sn: Melo
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: A1

dn: cn= Ailton, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Ailton
sn: Freire
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: A1

dn: cn= Matias, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Matias
sn: Oliveira
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: B1
businessCategory: C1

dn: cn= Maria, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Maria
sn: Oleg
ou: ou=Agencia_01, o=Banco_ABC, dc=com

businessCategory: A2

dn: cn= Silvia, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Silvia
sn: Karsten
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: A2

dn: cn= Vivian, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Vivian
sn: Fagundes
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: A2

dn: cn= Pedro, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Pedro
sn: Batista
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: B1
businessCategory: A1

dn: cn= Carla, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Carla
sn: Kiriale
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: C1

dn: cn= Alex, ou=People, o=Banco_ABC, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Alex
sn: Lopes
ou: ou=Agencia_01, o=Banco_ABC, dc=com
businessCategory: C1

dn: ou=Aplicativos, o=Banco_ABC, dc=com
objectclass: top
objectclass: organizationalunit
ou: Aplicativos

dn: dlmName=GerFinanceiro , ou=Aplicativos, o=Banco_ABC, dc=com

objectclass: top
objectclass: dlm1ManagedElement
objectclass: dlm1ManagedSystemElement
objectclass: dlm1LogicalElement
objectclass: dlm1System
objectclass: dlm1ApplicationSystem
dlmName: GerFinanceiro
dlmDescription: Aplicativo responsavel pela realizacao das operacoes financeiras do banco

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

dn: ou=Agencia_01, o=Banco_ABC, dc=com
objectclass: top
objectclass: organizationalunit
objectclass: pcimGroupAuxClass
ou: Agencia_01
pcimGroupName: PoliticaRBPIM

dn: rbpimRoleName=Funcionario, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimRole
rbpimRoleName: Funcionario
pcimRuleEnabled: 1

dn: rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimRole
rbpimRoleName: Atendente
pcimRuleEnabled: 1
pcimRuleConditionListType: 1
pcimRulePriority: 1
rbpimInheritedRoles: rbpimRoleName=Funcionario, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleConditionList: pcimConditionName=UsuariosAtendente, rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleValidityPeriodList: pcimValidityConditionName=Periodo1, rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=PermissaoAtendente1, rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=PermissaoAtendente2, rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName=UsuariosAtendente, rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top

objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: UsuariosAtendente
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName=UsuariosAtendente, rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: inetorgperson
rbpimModelProperty: BusinessCategory
rbpimStringList: A1

dn: pcimValidityConditionName=Periodo1, rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleValidityAssociation
objectClass: pcimTPCAuxClass
pcimValidityConditionName: Periodo1
pcimTPCDayOfWeekMask:01111100
pcimTPCTimeOfDayMask:T100000/T160000

dn: pcimActionName=PermissaoAtendente1, rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionName: PermissaoAtendente1
rbpimPermissionDN: rbpimPermissionName=GF1, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimActionName=PermissaoAtendente2, rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionName: PermissaoAtendente2
rbpimPermissionDN: rbpimPermissionName=GC1, ou=Agencia_01, o=Banco_ABC, dc=com

dn: rbpimRoleName=Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top

objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimRole
rbpimRoleName: Supervisor
pcimRuleEnabled: 1
pcimRuleConditionListType: 1
pcimRulePriority: 3
rbpimInheritedRoles: rbpimRoleName= Funcionario, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleConditionList: pcimConditionName=UsuariosSupervisor, rbpimRoleName= Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleValidityPeriodList: pcimValidityConditionName=Periodo1, rbpimRoleName= Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=PermissaoSupervisor1, rbpimRoleName= Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=PermissaoSupervisor2, rbpimRoleName= Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName=UsuariosSupervisor, rbpimRoleName= Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: UsuariosSupervisor
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName=UsuariosSupervisor, rbpimRoleName=Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: inetorgperson
rbpimModelProperty: BusinessCategory
rbpimStringList: B1

dn: pcimValidityConditionName=Periodo1, rbpimRoleName=Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleValidityAssociation
objectClass: pcimTPCAuxClass
pcimValidityConditionName: Periodo1
pcimTPCDayOfWeekMask:01111100
pcimTPCTimeOfDayMask:T100000/T160000

dn: pcimActionName=PermissaoSupervisor1, rbpimRoleName=Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top

objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionName: PermissaoSupervisor1
rbpimPermissionDN: rbpimPermissionName=GF2, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimActionName=PermissaoSupervisor2, rbpimRoleName=Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionName: PermissaoSupervisor2
rbpimPermissionDN: rbpimPermissionName=GC2, ou=Agencia_01, o=Banco_ABC, dc=com

dn: rbpimRoleName=Caixa, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimRole
rbpimRoleName: Caixa
pcimRuleEnabled: 1
pcimRuleConditionListType: 1
pcimRulePriority: 2
rbpimInheritedRoles: rbpimRoleName= Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleConditionList: pcimConditionName=UsuariosCaixa, rbpimRoleName= Caixa, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleValidityPeriodList: pcimValidityConditionName=Periodo1, rbpimRoleName= Caixa, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=PermissaoCaixa1, rbpimRoleName= Caixa, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName=UsuariosCaixa, rbpimRoleName= Caixa, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: UsuariosCaixa
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName=UsuariosCaixa, rbpimRoleName= Caixa, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1

rbpimModelClass: inetorgperson
rbpimModelProperty: BusinessCategory
rbpimStringList: A2

dn: pcimValidityConditionName=Periodo1, rbpimRoleName=Caixa, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleValidityAssociation
objectClass: pcimTPCAuxClass
pcimValidityConditionName: Periodo1
pcimTPCDayOfWeekMask:01111100
pcimTPCTimeOfDayMask:T100000/T160000

dn: pcimActionName=PermissaoCaixa1, rbpimRoleName=Caixa, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionName: PermissaoCaixa1
rbpimPermissionDN: rbpimPermissionName=GF3, ou=Agencia_01, o=Banco_ABC, dc=com

dn: rbpimRoleName=Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimRole
rbpimRoleName: Auditor
pcimRuleEnabled: 1
pcimRuleConditionListType: 1
pcimRulePriority: 4
rbpimInheritedRoles: rbpimRoleName= Funcionario, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleConditionList: pcimConditionName=UsuariosAuditor1, rbpimRoleName= Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleValidityPeriodList: pcimValidityConditionName=Periodo1, rbpimRoleName= Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=PermissaoAuditor1, rbpimRoleName= Auditor, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName=UsuariosAuditor1, rbpimRoleName= Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: UsuariosAuditor1
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName=UsuariosAuditor1, rbpimRoleName= Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy

objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: inetorgperson
rbpimModelProperty: BusinessCategory
rbpimStringList: C1

dn: pcimValidityConditionName=Periodo1, rbpimRoleName=Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleValidityAssociation
objectClass: pcimTPCAuxClass
pcimValidityConditionName: Periodo1
pcimTPCDayOfWeekMask: 01111100
pcimTPCTimeOfDayMask: T100000/T160000

dn: pcimActionName=PermissaoAuditor1, rbpimRoleName=Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerPermissionAuxClass
pcimActionName: PermissaoAuditor1
rbpimPermissionDN: rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com

dn: rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimPermission
pcimRuleConditionListType: 2
pcimRuleConditionList: pcimConditionName=RecursosAUD1, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleConditionList: pcimConditionName=RecursosAUD2, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleConditionList: pcimConditionName=RestricoesAUD1, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=OperacoesAUD1, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName= RecursosAUD1, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: RecursosAUD1
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName= RecursosAUD1, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: dlm1ApplicationSystem
rbpimModelProperty: dlmName
rbpimStringList: GerFinanceiro

dn: pcimConditionName= RecursosAUD2, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: RecursosAUD2
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName= RecursosAUD2, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: dlm1ApplicationSystem
rbpimModelProperty: dlmName
rbpimStringList: GerCliente

dn: pcimConditionName= RestricoesAUD1, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: RestricoesAUD1
pcimConditionGroupNumber: 2
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName=RestricoesAUD1, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement

objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyImplicitVariable
objectClass: rbpimPolicySourceIPv4Var
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyIPv4AddrValue
rbpimConditionName: Exp1
rbpimIPv4AddrList: 192.168.10.0/24

dn: pcimActionName=OperacoesAUD1, rbpimPermissionName=AUD, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionName: OperacoesAUD1
rbpimOperationList: Auditar_Transacoes

dn: rbpimPermissionName=GF1, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimPermission
pcimRuleConditionListType: 1
pcimRuleConditionList: pcimConditionName=RecursosGF1_1, rbpimPermissionName=GF1, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=OperacoesGF1_1, rbpimPermissionName=GF1, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName= RecursosGF1_1, rbpimPermissionName=GF1, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: RecursosGF1_1
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName= RecursosGF1_1, rbpimPermissionName=GF1, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: dlm1ApplicationSystem
rbpimModelProperty: dlmName
rbpimStringList: GerFinanceiro

dn: pcimActionName=OperacoesGF1_1, rbpimPermissionName=GF1, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionName: OperacoesGF1_1
rbpimOperationList: AgendarTED
rbpimOperationList: AgendarDOC

dn: rbpimPermissionName=GF2, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimPermission
pcimRuleConditionListType: 1
pcimRuleConditionList: pcimConditionName=RecursosGF2_1, rbpimPermissionName=GF2, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=OperacoesGF2_1, rbpimPermissionName=GF2, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName= RecursosGF2_1, rbpimPermissionName=GF2, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: RecursosGF2_1
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName= RecursosGF2_1, rbpimPermissionName=GF2, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: dlm1ApplicationSystem
rbpimModelProperty: dlmName
rbpimStringList: GerFinanceiro

dn: pcimActionName=OperacoesGF2_1, rbpimPermissionName=GF2, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass

objectClass: rbpimAssignerOperationAuxClass
pcimActionName: OperacoesGF2_1
rbpimOperationList: AutorizarDOC
rbpimOperationList: AutorizarTED

dn: rbpimPermissionName=GC1, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimPermission
pcimRuleConditionListType: 1
pcimRuleConditionList: pcimConditionName=RecursosGC1_1, rbpimPermissionName=GC1, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=OperacoesGC1_1, rbpimPermissionName=GC1, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName= RecursosGC1_1, rbpimPermissionName=GC1, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: RecursosGC1_1
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName= RecursosGC1_1, rbpimPermissionName=GC1, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: dlm1ApplicationSystem
rbpimModelProperty: dlmName
rbpimStringList: GerCliente

dn: pcimActionName=OperacoesGC1_1, rbpimPermissionName=GC1, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionName: OperacoesGC1_1
rbpimOperationList: AbrirConta

dn: rbpimPermissionName=GC2, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy

objectClass: pcimRule
objectClass: rbpimPermission
pcimRuleConditionListType: 1
pcimRuleConditionList: pcimConditionName=RecursosGC2_1, rbpimPermissionName=GC2, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=OperacoesGC2_1, rbpimPermissionName=GC2, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName= RecursosGC2_1, rbpimPermissionName=GC2, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: RecursosGC2_1
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName= RecursosGC2_1, rbpimPermissionName=GC2, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: dlm1ApplicationSystem
rbpimModelProperty: dlmName
rbpimStringList: GerCliente

dn: pcimActionName=OperacoesGC2_1, rbpimPermissionName=GC2, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionName: OperacoesGC2_1
rbpimOperationList: ConcederLimite

dn: rbpimPermissionName=GF3, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRule
objectClass: rbpimPermission
pcimRuleConditionListType: 1
pcimRuleConditionList: pcimConditionName=RecursosGF3_1, rbpimPermissionName=GF3, ou=Agencia_01, o=Banco_ABC, dc=com
pcimRuleActionList: pcimActionName=OperacoesGF3_1, rbpimPermissionName=GF3, ou=Agencia_01, o=Banco_ABC, dc=com

dn: pcimConditionName= RecursosGF3_1, rbpimPermissionName=GF3, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleConditionAssociation
objectClass: pcimConditionAuxClass
objectClass: rbpimSimplePolicyConditionAuxClass
pcimConditionName: RecursosGF3_1
pcimConditionGroupNumber: 1
pcimConditionNegated: FALSE

dn: rbpimConditionName=Exp1, pcimConditionName= RecursosGF3_1, rbpimPermissionName=GF3, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimConditionAssociation
objectClass: rbpimPolicyVariable
objectClass: rbpimPolicyExplicitVariable
objectClass: rbpimPolicyValue
objectClass: rbpimPolicyStringValue
rbpimConditionName: Exp1
rbpimModelClass: dlm1ApplicationSystem
rbpimModelProperty: dlmName
rbpimStringList: GerFinanceiro

dn: pcimActionName=OperacoesGF3_1, rbpimPermissionName=GF3, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: pcimRuleActionAssociation
objectClass: pcimActionAuxClass
objectClass: rbpimAssignerOperationAuxClass
pcimActionName: OperacoesGF3_1
rbpimOperationList: EfetuarPagamentos

dn: rbpimSSDname=SSD01, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimSSD
rbpimSSDname: SSD01
rbpimRoleSet: rbpimRoleName=Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
rbpimRoleSet: rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
rbpimCardinality: 2

dn: rbpimSSDname=SSD02, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimSSD
rbpimSSDname: SSD02
rbpimRoleSet: rbpimRoleName=Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
rbpimRoleSet: rbpimRoleName=Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
rbpimCardinality: 2

dn: rbpimSSDname=SSD03, ou=Agencia_01, o=Banco_ABC, dc=com

objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimSSD
rbpimSSDname: SSD03
rbpimRoleSet: rbpimRoleName=Auditor, ou=Agencia_01, o=Banco_ABC, dc=com
rbpimRoleSet: rbpimRoleName=Caixa, ou=Agencia_01, o=Banco_ABC, dc=com
rbpimCardinality: 2

dn: rbpimDSDname=DSD01, ou=Agencia_01, o=Banco_ABC, dc=com
objectClass: top
objectClass: dlm1ManagedElement
objectClass: pcimPolicy
objectClass: rbpimDSD
rbpimDSDname: DSD01
rbpimRoleSet: rbpimRoleName=Supervisor, ou=Agencia_01, o=Banco_ABC, dc=com
rbpimRoleSet: rbpimRoleName=Atendente, ou=Agencia_01, o=Banco_ABC, dc=com
rbpimCardinality: 2

ANEXO D – ARQUIVOS DE ENTRADA E SAÍDA DO ESTUDO DE CASO

Este anexo apresenta os vinte arquivos de entrada com as chamadas realizadas pelas aplicações ao PDP. Ainda, são apresentados os arquivos de saída gerados pelo PDP correspondentes aos dez primeiros arquivos de entradas. Os arquivos de saída correspondentes aos arquivos de entrada de onze a vinte podem ser obtidos a partir do endereço *http://www.ppgia.pucpr.br/~rcnabhen/rbpim*. Nos arquivos de entrada (*.in*) o primeiro número de cada linha representa a chamada da API do RBPEP solicitada pela aplicação. A seguinte convenção foi utilizada: *1-RBPEP_OPEN, 2-RBPEP_CreateSession, 3- RBPEP_SelectedRoles, 4-RBPEP_CloseSession, 5-RBPEP_CheckAccess* e *X-Fechamento de Serviço.* Os arquivos de saída *(.out)* apresentam a seqüência de mensagens COPS geradas em função das chamadas editadas nos arquivos de entrada. Na primeira coluna está a indicação da origem e o destino da mensagem. Na segunda, é apresentado o conteúdo completo da mensagem COPS, sendo mostrado o cabeçalho seguido pelos objetos específicos COPS transportados.

Como ilustração, considere a segunda chamada de verificação de acesso contida no arquivo de entrada *App1.in*, a qual está sendo mostrada em destaque. Nesta chamada, a aplicação está solicitando ao PDP uma decisão de política RBPIM, que, neste caso, é saber se o usuário ativo na sessão 2 (*app1_2*) pode realizar a operação *EfetuarPagamentos* no aplicativo *Gerenciador Financeiro*. Como resultado, a seqüência composta pelas três mensagens mostradas em destaque no arquivo *App1.out* é efetuada. Inicialmente, uma mensagem REQ é enviada do RBPEP ao PDP, sendo composta do cabeçalho seguido por quatro objetos específicos COPS, um *Handle* informando o *sessionID*, um *Context* indicando o tipo de requisição e dois *ClientSI* transportando, respectivamente, a operação e o objeto cujo acesso está sendo verificado (i.e., a aplicação *Gerenciador Financeiro*). Como resultado, o PDP edita uma mensagem DEC ao RBPEP, a qual transporta dois objetos específicos COPS *Handle* e *Context* iguais aos da mensagem REQ correspondente, seguido de um objeto de decisão negando o acesso. Por fim, após o recebimento da mensagem DEC, o RBPEP edita uma mensagem RPT ao PDP, indicando que recebeu a decisão de política.

App1.in

1
2,Maria
3,app1_1,Caixa,Supervisor
3,app1_1,Caixa,Atendente
5,app1_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
2,Maria
3,app1_2,Atendente
5,app1_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app1_1
5,app1_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app1_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app1_2,EfetuarEmprestimo,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app1_2
X

App1.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app1)|
PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	REQ |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Maria)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=4) |Decision: (Client SI.=Atendente,Caixa,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app1_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app1_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app1_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app1_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app1_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Maria)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=7) |Decision: (Client SI.=Atendente,Caixa,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app1_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app1_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	**|REQ |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|**
PDP->RBPEP:	**|DEC |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|**
RBPEP->PDP:	**|RPT |Handle: (SessionID=app1_2) |Repo.Type: (Sucesso)|**
RBPEP->PDP:	|DRQ |Handle: (SessionID=app1_1) |Reason c.|

RBPEP->PDP:	|REQ |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app1_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app1_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarEmprestimo) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app1_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app1_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app1_2) |Reason c.|
RBPEP->PDP:	|CC |Error: (Fechamento do Servico.)|

App2.in

```
1
2,Matias
3,app2_1,Caixa
3,app2_1,Auditor,Supervisor
3,app2_1,Auditor,Funcionario
5,app2_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app2_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente
5,app2_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.168.100.15
5,app2_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.168.10.15
```

App2.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app2)|
PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Matias)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=0) |Decision: (Client SI.=Funcionario,Auditor,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app2_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app2_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Auditor,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app2_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Auditor,Funcionario)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app2_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso)

	|Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app2_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app2_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente) |Client SI: (rbpimPolicySourceIPv4Var=192.168.100.15)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app2_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente) |Client SI: (rbpimPolicySourceIPv4Var=192.168.10.15)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app2_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|

App3.in

```
1
2,Pedro
2,Luiz
3,app3_1,Caixa,Supervisor
3,app3_1,Supervisor,Atendente
3,app3_1,Atendente
5,app3_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app3_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app3_1,AgendarDOC,dlm1ApplicationSystem.dlmName=GerFinanceiro
2,Carlos
5,app3_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app3_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app3_1,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app3_1
```

App3.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app3)|
PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Pedro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=1) |Decision: (Client SI.=Funcionario,Atendente,Supervisor,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Luiz)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Error: (Erro - userID invalido!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|

RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Supervisor,Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Papeis conflitantes para uma mesma sessao!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_3) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Carlos)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_3) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=2) |Decision: (Client SI.=Funcionario,Atendente,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_3) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app3_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app3_1) |Repo.Type: (Sucesso)|

App4.in

```
1
2,Carla
3,app4_1,Supervisor
3,app4_1,Caixa
3,app4_1,Funcionario
5,app4_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
2,Alex
3,app4_2,Auditor
5,app4_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app4_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app4_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app4_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app4_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app4_1
5,app4_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app4_2
X
```

App4.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app4)|
PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Carla)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=1) |Decision: (Client SI.=Funcionario,Auditor,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Funcionario)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Alex)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=3) |Decision: (Client SI.=Funcionario,Auditor,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Auditor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_2) |Repo.Type: (Sucesso)|

RBPEP->PDP:	|REQ |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app4_1) |Reason c.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app4_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app4_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app4_2) |Reason c.|

App5.in

```
1
2,Maria
3,app5_1,Caixa,Supervisor
3,app5_1,Caixa,Atendente
5,app5_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
2,Maria
3,app5_2,Atendente
5,app5_2,EfetuarPagamentos,1,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app5_1
5,app5_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app5_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app5_2,EfetuarEmprestimo,1,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app5_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app5_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app5_2,EfetuarEmprestimo,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app5_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app5_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app5_2,EfetuarEmprestimo,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app5_2
X
```

App5.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app5)|

PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Maria)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=3) |Decision: (Client SI.=Atendente,Caixa,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Maria)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=6) |Decision: (Client SI.=Atendente,Caixa,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app5_1) |Reason c.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarEmprestimo) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso)

	|Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarEmprestimo) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarEmprestimo) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app5_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app5_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app5_2) |Reason c.|

App6.in

```
1
2,Matias
3,app6_1,Caixa
3,app6_1,Auditor,Supervisor
3,app6_1,Auditor,Supervisor
3,app6_1,Auditor,Supervisor
3,app6_1,Auditor,Funcionario
5,app6_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app6_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente
5,app6_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.1
68.100.15
5,app6_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.1
68.10.15
5,app6_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente
5,app6_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.1
68.100.15
5,app6_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.1
68.10.15
5,app6_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente
5,app6_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.1
68.100.15
5,app6_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.1
68.10.15
```

App6.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app6)|
PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Matias)|

PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=3) |Decision: (Client SI.=Funcionario,Auditor,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Auditor,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Auditor,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Auditor,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Auditor,Funcionario)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente) |Client SI: (rbpimPolicySourceIPv4Var=192.168.100.15)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente) |Client SI: (rbpimPolicySourceIPv4Var=192.168.10.15)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente) |Client SI: (rbpimPolicySourceIPv4Var=192.168.100.15)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso)

	|Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente) |Client SI: (rbpimPolicySourceIPv4Var=192.168.10.15)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente) |Client SI: (rbpimPolicySourceIPv4Var=192.168.100.15)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app6_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (Auditar_Transacoes) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente) |Client SI: (rbpimPolicySourceIPv4Var=192.168.10.15)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app6_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|

App7.in

```
1
2,Maria
3,app7_1,Caixa,Supervisor
3,app7_1,Caixa,Atendente
5,app7_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
2,Maria
3,app7_2,Atendente
5,app7_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app7_1
5,app7_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app7_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app7_2,EfetuarEmprestimo,1,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app7_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app7_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app7_2,EfetuarEmprestimo,1,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app7_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app7_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app7_2,EfetuarEmprestimo,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app7_2
X
```

App7.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app7)|
PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Maria)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=0) |Decision: (Client SI.=Atendente,Caixa,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Maria)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=5) |Decision: (Client SI.=Atendente,Caixa,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app7_1) |Reason c.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarEmprestimo) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|

RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarEmprestimo) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarEmprestimo) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app7_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app7_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app7_2) |Reason c.|
RBPEP->PDP:	|CC |Error: (Fechamento do Servico.)|

App8.in

```
1
2,Pedro
2,Luiz
3,app8_1,Caixa,Supervisor
3,app8_1,Supervisor,Atendente
3,app8_1,Atendente
5,app8_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app8_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app8_1,AgendarDOC,dlm1ApplicationSystem.dlmName=GerFinanceiro
2,Carlos
2,Carlos
5,app8_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app8_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app8_1,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app8_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app8_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app8_1,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app8_1
```

App8.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app8)|
PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Pedro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=3) |Decision: (Client SI.=Funcionario,Atendente,Supervisor,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Luiz)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Error: (Erro - userID invalido!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Supervisor,Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Papeis conflitantes para uma mesma sessao!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_3) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Carlos)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_3) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=0) |Decision: (Client SI.=Funcionario,Atendente,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_3) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_4) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Carlos)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_4) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=3) |Decision: (Client SI.=Funcionario,Atendente,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_4) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso)

	|Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app8_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app8_1) |Repo.Type: (Sucesso)|

App9.in

```
1
2,Carla
3,app9_1,Supervisor
3,app9_1,Caixa
3,app9_1,Funcionario
5,app9_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
2,Alex
3,app9_2,Auditor
5,app9_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app9_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app9_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app9_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app9_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app9_1
5,app9_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app9_2
X
```

App9.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app9)|
PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Carla)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=0) |Decision: (Client SI.=Funcionario,Auditor,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Funcionario)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Alex)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=0) |Decision: (Client SI.=Funcionario,Auditor,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Auditor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_1) |Repo.Type: (Sucesso)|

RBPEP->PDP:	|REQ |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app9_1) |Reason c.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app9_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app9_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app9_2) |Reason c.|

App10.in

```
1
2,Maria
2,Pedro
2,Aroldo
2,Vivian
3,app10_1,Caixa,Supervisor
3,app10_1,Caixa,Atendente
5,app10_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
2,Maria
3,app10_2,Atendente
5,app10_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app10_1
5,app10_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
2,Silvia
5,app10_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app10_2,EfetuarEmprestimo,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app10_2
X
```

App10.out

RBPEP->PDP:	|OPN |PEP Iden.: (PepID=app10)|
PDP->RBPEP:	|CAT |Keep-Ali.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Maria)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=2) |Decision: (Client SI.=Atendente,Caixa,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Pedro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=5) |Decision: (Client SI.=Funcionario,Atendente,Supervisor,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_3) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Aroldo)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_3) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Error: (Erro - userID invalido!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_3) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_4) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Vivian)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_4) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=0) |Decision: (Client SI.=Atendente,Caixa,)|

RBPEP->PDP:	|RPT |Handle: (SessionID=app10_4) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Supervisor)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Error: (Erro - Selecao invalida de papeis!)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Caixa,Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AbrirConta) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_1) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_1) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_5) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Maria)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_5) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=7) |Decision: (Client SI.=Atendente,Caixa,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_5) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Client SI: (Atendente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 2) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarPagamentos) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app10_1) |Reason c.|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarTED) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Aceita)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_6) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Client SI: (Silvia)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_6) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Criacao de Sessao Fase 1) |Decision: (Requisicao Aceita) |Decision: (Client SI.=0) |Decision: (Client SI.=Atendente,Caixa,)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_6) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (AgendarDOC) |Client SI: (dlm1ApplicationSystem.dlmName=GerCliente)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|REQ |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Client SI: (EfetuarEmprestimo) |Client SI: (dlm1ApplicationSystem.dlmName=GerFinanceiro)|
PDP->RBPEP:	|DEC |Handle: (SessionID=app10_2) |Context: (RTYPE=Aloc.de Recursos,MTYPE=Verificacao de Acesso) |Decision: (Requisicao Negada)|
RBPEP->PDP:	|RPT |Handle: (SessionID=app10_2) |Repo.Type: (Sucesso)|
RBPEP->PDP:	|DRQ |Handle: (SessionID=app10_2) |Reason c.|

App11.in

```
1
2,Matias
3,app11_1,Caixa
3,app11_1,Auditor,Supervisor
```

```
3,app11_1,Auditor,Supervisor
3,app11_1,Auditor,Supervisor
3,app11_1,Auditor,Funcionario
3,app11_1,Auditor,Supervisor
5,app11_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app11_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente
3,app11_1,Auditor,Supervisor
5,app11_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.
168.100.15
3,app11_1,Auditor,Supervisor
5,app11_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.
168.10.15
```

App12.in

```
1
2,Pedro
2,Luiz
3,app12_1,Caixa,Supervisor
3,app12_1,Supervisor,Atendente
3,app12_1,Atendente
5,app12_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app12_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app12_1,AgendarDOC,dlm1ApplicationSystem.dlmName=GerFinanceiro
2,Carlos
5,app12_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app12_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app12_1,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app12_1
```

App13.in

```
1
2,Carla
3,app13_1,Supervisor
3,app13_1,Caixa
3,app13_1,Funcionario
5,app13_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
2,Alex
3,app13_2,Auditor
5,app13_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app13_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app13_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app13_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app13_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app13_1
5,app13_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app13_2
X
```

App14.in

```
1
2,Maria
3,app14_1,Caixa,Supervisor
3,app14_1,Caixa,Atendente
5,app14_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
```

```
5,app14_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app14_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app14_2,EfetuarEmprestimo,dlm1ApplicationSystem.dlmName=GerFinanceiro
2,Maria

3,app14_2,Atendente
5,app14_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app14_1
5,app14_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app14_2,AgendarDOC,dlm1ApplicationSystem.dlmName=GerCliente
5,app14_2,EfetuarEmprestimo,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app14_2
X
```

App15.in

```
1
2,Matias
3,app15_1,Caixa
3,app15_1,Auditor,Supervisor
3,app15_1,Auditor,Funcionario
3,app15_1,Caixa
3,app15_1,Auditor,Supervisor
3,app15_1,Auditor,Funcionario
5,app15_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app15_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente
5,app15_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.
168.100.15
5,app15_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.
168.10.15
5,app15_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente
5,app15_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.
168.100.15
5,app15_1,Auditar_Transacoes,dlm1ApplicationSystem.dlmName=GerCliente,rbpimPolicySourceIPv4Var=192.
168.10.15
```

App16.in

```
1
2,Pedro
2,Luiz
3,app16_1,Caixa,Supervisor
3,app16_1,Supervisor,Atendente
3,app16_1,Atendente
5,app16_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app16_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app16_1,AgendarDOC,dlm1ApplicationSystem.dlmName=GerFinanceiro
2,Carlos
5,app16_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app16_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app16_1,AgendarDOC,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app16_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app16_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app16_1,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app16_1
```

App17.in

```
1
2,Carla
3,app17_1,Supervisor
```

```
3,app17_1,Caixa
3,app17_1,Funcionario
5,app17_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
2,Alex
3,app17_2,Auditor
5,app17_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app17_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app17_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app17_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app17_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app17_1
5,app17_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app17_2
X
```

App18.in

```
1
2,Pedro
2,Luiz
3,app18_1,Caixa,Supervisor
3,app18_1,Supervisor,Atendente
3,app18_1,Atendente
5,app18_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app18_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app18_1,AgendarDOC,dlm1ApplicationSystem.dlmName=GerFinanceiro
2,Carlos
5,app18_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app18_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app18_1,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app18_1
```

App19.in

```
1
2,Carla
3,app19_1,Supervisor
3,app19_1,Caixa
3,app19_1,Funcionario
5,app19_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
2,Alex
3,app19_2,Auditor
5,app19_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app19_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app19_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app19_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app19_2,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app19_1
5,app19_2,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app19_2
X
```

App20.in

```
1
2,Pedro
2,Luiz
3,app20_1,Caixa,Supervisor
3,app20_1,Supervisor,Atendente
3,app20_1,Atendente
5,app20_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
```

```
5,app20_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app20_1,AgendarDOC,dlm1ApplicationSystem.dlmName=GerFinanceiro
2,Carlos
5,app20_1,AbrirConta,dlm1ApplicationSystem.dlmName=GerCliente
5,app20_1,EfetuarPagamentos,dlm1ApplicationSystem.dlmName=GerFinanceiro
5,app20_1,AgendarTED,dlm1ApplicationSystem.dlmName=GerFinanceiro
4,app20_1
```

ANEXO E – CÓDIGO FONTE DOS PRINCIPAIS MÓDULOS DO PROTÓTIPO DESENVOLVIDO

O protótipo desenvolvido neste trabalho é composto dos arquivos apresentados na tabela a seguir:

Arquivo	Descrição
Pdp.java	Implementa a funcionalidade do Pdp na arquitetura do RBPIM.
Rbpep.java	Implementa a funcionalidade do Rbpep na arquitetura do RBPIM.
MESSAGE_STRUCUTURE.java	Fornece o suporte à criação e manipulação das mensagens COPS.
COPS_Protocol.java	Realiza a leitura e interpretação das mensagens COPS.
RbpimRepository.java	Classes que atuam como interface ao repositório de política LDAP. Contém todas as consultas LDAP definidas para os algoritmos do Capítulo 6.
RbpimStateBase.java	Classes que atuam como interface à base de estados.
Util.java	Classes de suporte à conversão de valores e montagem das mensagens COPS.
Applications.Java	Representa uma aplicação na arquitetura do RBPIM.

Este anexo apresenta apenas a listagem dos dois principais módulos, os módulos do *PDP* e do *RBPEP*. O demais arquivos podem ser obtidos a partir do endereço *http://www.ppgia.pucpr.br/~rcnabhen/rbpim.*

1.1. Arquivo Pdp.java

```
/*
Nome da classe: Pdp
Descricao: Representa o PDP na arquitetura definida para implementacao do RBPIM
*/

import java.net.*;
import java.io.*;
import java.util.Vector;
import java.util.StringTokenizer;
import java.util.Enumeration;
import java.util.Hashtable;


public class Pdp {

	public static void main(String[] args) throws IOException {
		activeRbpeps = new Vector();
		 ServerSocket serverSocket = null;
		int clients = 0;
		 boolean listening= true;

		if ( args.length != 3 ) {
```

```
                System.out.println( "Uso: java Pdp " +
                  "<LDAP Server> <Port Number> <Password Directory Manager>");
                System.exit(1);
            }

            rbpimStateBase = new RbpimStateBase();
            rbpimRepository = new RbpimRepository (args[0],Integer.parseInt(args[1]),"cn=Directory
            Manager",args[2]);

            try {
                serverSocket = new ServerSocket(3288);
            }
            catch (IOException e) {
                System.err.println("Um erro ocorreu ao ativar o PDP na porta 3288.");
                System.exit(-1);
            }

            System.out.println("RBPIM PDP - ativo.");


            while (listening) {
                clients++;
                new PdpThread(serverSocket.accept(),clients).start();

              }

            serverSocket.close();
        }


        public synchronized static void addActiveRbpep(String pepId)          {
            activeRbpeps.addElement(pepId);
        }


        public synchronized static void removeActiveRbpep(String pepId)     {
            activeRbpeps.remove(pepId);
        }


        public static boolean containsAuthRbpep(String pepId) {

            for(int i=0; i<20; i++)
                if(authorizedRbpeps[i].trim().equalsIgnoreCase(pepId.trim()))
                return true;

            return false;

        }

        public synchronized static boolean containsActiveRbpep(String pepId)
        {
            return activeRbpeps.contains(pepId);
        }

        private static String[] authorizedRbpeps = {"app1","app2","app3","app4","app5",
                                                    "app6","app7","app8","app9","app10",
                                                      "app11","app12","app13","app14","app15",
                                                     "app16","app17","app18","app19","app20"};
        private static Vector activeRbpeps;
        protected static RbpimStateBase rbpimStateBase;
        protected static RbpimRepository rbpimRepository;



}
```

```
/*
Nome da classe: PdpThread
Descricao: Representa uma Thread para tratamento das conexoes provenientes dos RBPEPs.
*/

class PdpThread extends Thread {


	private static final byte DEC = (byte)2;
	private static final byte CAT = (byte)7;
	private static final byte CC  = (byte)8;

	private static final byte REQ = (byte)1;
	private static final byte RPT = (byte)3;
	private static final byte DRQ = (byte)4;
	private static final byte OPN = (byte)6;

	private static final short RBPEPCLIENT  = (short)0x8000;

	private String pepId;
	private MESSAGE_STRUCTURE message;

	private Socket socket = null;

	private File pdpOUT=null;
	private String filename;
	private BufferedWriter fileOut;

	public PdpThread(Socket socket, int clients) {
		super("PdpThread");
		System.out.println("Numero de RBPEP conectados: " + clients);
		this.socket = socket;
	}

	public void run() {
		MESSAGE_STRUCTURE message_in=null;
		MESSAGE_STRUCTURE message_out=null;

		try {
			socket.setTcpNoDelay(true);
			InputStream in = socket.getInputStream();
			OutputStream out = socket.getOutputStream();

			while(true){
				message_in=COPS_Protocol.readMessage(in);
				System.out.println("RBPEP -> PDP: " + message_in.display());
				if(pdpOUT!=null)
					print(0l,1,message_in.display());

				long T1=System.currentTimeMillis();
				message_out=processInMessage(message_in,message_out);

				if(message_out!=null) {
					long T2=System.currentTimeMillis();
					print(T2-T1,2,message_out.display());
					System.out.println("PDP -> RBPEP: " + message_out.display());
					message_out.writeBytes(out);
					out.flush();

			}
		}
		catch (Exception e) {
			e.printStackTrace();
		}
```

```
	}


	public MESSAGE_STRUCTURE processInMessage(MESSAGE_STRUCTURE m_in,
                                                  MESSAGE_STRUCTURE m_out)
	{
			String errorCodeSubcode = null;
			switch(m_in.getopCode())
			{
				case OPN: return readOPN(m_in);
				case REQ: return readREQ(m_in);
				case RPT: readRPT(m_in,m_out); break;
				case DRQ: readDRQ(m_in);  break;
				case CC:  readCC(m_in);
			}

			return null;

	}



	public synchronized MESSAGE_STRUCTURE createCAT(String pepId) {
		Pdp.addActiveRbpep(pepId);
		this.pepId=pepId;
		MESSAGE_STRUCTURE m_out=null;
		m_out=new MESSAGE_STRUCTURE((byte)0x10,(byte)CAT,(short)RBPEPCLIENT);
		m_out.add_cops_Object(new COPS_SObject((byte)10,(byte)1,Util.int2bytes(0)));
		return m_out;
	}

	public MESSAGE_STRUCTURE createCC(short errorCode, short subCode) {
		MESSAGE_STRUCTURE m_out=null;
		m_out=new MESSAGE_STRUCTURE((byte)0x10,(byte)CC,(short)RBPEPCLIENT);
		m_out.add_cops_Object(new COPS_SObject((byte)8,(byte)1,
                                Util.short2bytes(errorCode),Util.short2bytes(subCode)));

		return m_out;
	}

// Mensagem DEC para resposta a um pedido de abertura de sessao Fase 1
	public  MESSAGE_STRUCTURE createDECsession1(String sessionID, String userID) {
		MESSAGE_STRUCTURE m_out=null;

		// Verificar se ja' existe sessao aberta para sessionID
		if(Pdp.rbpimStateBase.checkSessionID(sessionID))
			return createDECerror((short)16,(short)105,sessionID,(short)2,(short)1);

		String ou_user = Pdp.rbpimRepository.Consulta1(userID);

		if(ou_user==null)
			return createDECerror((short)16,(short)107,sessionID,(short)2,(short)1);

		LDAP_objs roles = Pdp.rbpimRepository.Consulta2(ou_user);

		int i=0;

		// Loop papeis
		while(i<roles.size()) {
			LDAP_obj role=roles.getObj(i);
			if(!role.getInherited()) {
				int conditionListType=1; // DNF Default
				if(!role.contains("pcimRuleConditionListType")) {
                                        i++;
                                        continue;
```

```
                }

                conditionListType= integer.parseInt(role.getAttribute
                        ("pcimRuleConditionListType").getValue(0));

                String filter = Util.makeFilter(role,conditionListType);

                // Verifica se o usuario pertence a lista de CNs do papel e
                // habilita/desabilita o papel na lista em funcao desta verificacao
                role.setStatus(Pdp.rbpimRepository.Consulta4(filter,userID));

                if(!role.getStatus()) {
                        i++;
                        continue;
                }
        }

        // Verificar periodos de ativacao
         if(role.contains("pcimRuleValidityPeriodList")) {
                LDAP_attr validityPeriodDN = role.getAttribute("pcimRuleValidityPeriodList");

                int j=0;
                boolean authPeriod=false;
                while(j<validityPeriodDN.size()){
                        LDAP_obj validityPeriod =
                        Pdp.rbpimRepository.Consulta5(validityPeriodDN.getValue(j));


                        if(Util.checkPeriod(validityPeriod)) {
                                authPeriod=true;
                                break;
                        }
                        j++;
                }
                role.setStatus(authPeriod);

                if(!role.getStatus()) {
                        i++;
                        continue;
                }
        }


        // Obter papeis herdados
        if(role.contains("rbpimInheritedRoles")) {
                LDAP_attr inheritedRolesDN = role.getAttribute("rbpimInheritedRoles");

                int j=0;
                while(j<inheritedRolesDN.size()){
                        LDAP_obj inheritedRole =
                        Pdp.rbpimRepository.Consulta6(inheritedRolesDN.getValue(j));
                        inheritedRole.setStatus(true);
                        inheritedRole.setInherited(true);

                        if(!roles.checkObj(inheritedRole.getDN())) {
                                roles.addObj(inheritedRole);
                        }
                        else{
                                roles.getObj(inheritedRole.getDN()).setStatus(true);
                        }
                        j++;
                }

        }
        i++;
}
```

```
i=0;
// Loop para eliminacao dos conflitos por SSD entre os papeis
// contidos na lista de papeis.

while(i<roles.size()) {
	LDAP_obj role=roles.getObj(i);
	if(!role.getStatus()){
		i++;
		continue;
	}

	LDAP_objs ssd = Pdp.rbpimRepository.Consulta7(ou_user,role.getDN());

	// Loop de objetos de SSD
	int j=0;

	while(j<ssd.size()) {
		LDAP_obj s=ssd.getObj(j);


		if(!s.contains("rbpimRoleSet") || !s.contains("rbpimCardinality")) {
			j++;
			continue;
		}

		Util.checkSeparationOfDuty(0,s.getAttribute("rbpimRoleSet"),roles,Integer.parseInt(
		s.getAttribute("rbpimCardinality").getValue(0)));

		j++;
	}

	i++;
}

int numberOfRoles=0;
i=0;

// Loop para inserir os papeis elegiveis do usuario na Base de Estados

while(i<roles.size()) {
	LDAP_obj role=roles.getObj(i);
	if(!role.getStatus()){
			i++;
			continue;
	}

	numberOfRoles++;
	synchronized(this) {
		Pdp.rbpimStateBase.insertRole(sessionID,role.getDN(),pepId,role.getDescription());
	}
	i++;
}

//  gerar DEC message
m_out=new MESSAGE_STRUCTURE((byte)0x11,(byte)DEC,(short)RBPEPCLIENT);

// Handle object(Sessao) C_NUM=1  C_TYPE=1  Contents=SessionID
m_out.add_cops_Object(new COPS_SObject((byte)1,(byte)1,sessionID.getBytes()));

// Context object C_NUM=2  C_TYPE=1  Contents=R-TYPE+M-TYPE
m_out.add_cops_Object(new
COPS_SObject((byte)2,(byte)1,Util.short2bytes((short)2),Util.short2bytes((short)1)));

int numOfSessions=Pdp.rbpimStateBase.getNumOfSessions(userID);

if(numberOfRoles>0) {
```

```
                        // Inserir sessao ativa do usuario na Base de Estado
                        synchronized(this) {
                                Pdp.rbpimStateBase.insertSession(sessionID,userID,0,pepId);
                        }

        // Decision object C_NUM=6  C_TYPE=1  Contents=(COMMAND_CODE=1)(FLAGS=0) --> Requisao aceita
                        m_out.add_cops_Object(new
                        COPS_SObject((byte)6,(byte)1,Util.short2bytes((short)1),Util.short2bytes((short)0)));

        // Decision object: Client Specific Data C_NUM=6  C_TYPE=4  Contents=(NumberOfSessions))
                        m_out.add_cops_Object(new COPS_SObject((byte)6,(byte)4,Util.int2bytes(numOfSessions)));

        // Decision object: Client Specific Data C_NUM=6  C_TYPE=4  Contents=(Roles))
                        m_out.add_cops_Object(new
                        COPS_SObject((byte)6,(byte)4,Pdp.rbpimStateBase.getRoles(sessionID).getBytes()));

                        }

                else  {
// Decision object C_NUM=6  C_TYPE=1  Contents=(COMMAND_CODE=0)(FLAGS=0) --> Nenhuma decisao disponivel
                        m_out.add_cops_Object(new
                        COPS_SObject((byte)6,(byte)1,Util.short2bytes((short)0),Util.short2bytes((short)0)));

// Decision object: Client Specific Data C_NUM=6  C_TYPE=4  Contents=(NumberOfSessions))
                        m_out.add_cops_Object(new COPS_SObject((byte)6,(byte)4,Util.int2bytes(numOfSessions)));

                }

                return m_out;

        }


// Mensagem DEC para resposta a um pedido de abertura de sessao Fase 2
   public  MESSAGE_STRUCTURE createDECsession2(String sessionID, String selectedRoles) {

                MESSAGE_STRUCTURE m_out=null;

                // Verificar se ja' existe sessao aberta para sessionID para fase 1
                if(!Pdp.rbpimStateBase.checkSessionID(sessionID,1))
                        return createDECerror((short)16,(short)109,sessionID,(short)2,(short)2);

                StringTokenizer Tokens=new StringTokenizer(selectedRoles,",");
                String roleDN="";
                String userID=Pdp.rbpimStateBase.getUserID(sessionID);
                String ou_user=Pdp.rbpimRepository.Consulta1(userID);

                LDAP_objs roles=new LDAP_objs();

                while (Tokens.hasMoreTokens()) {
                        roleDN=Pdp.rbpimStateBase.getRole(sessionID,Tokens.nextToken());
                        if(roleDN==null)
                                return createDECerror((short)16,(short)110,sessionID,(short)2,(short)2);

                           roles.addObj(Pdp.rbpimRepository.Consulta8(roleDN));

                }
                int i=0;

                // Loop para verificacao dos conflitos por DSD entre os papeis
                // contidos na lista de papeis.

                while(i<roles.size()) {
                        LDAP_obj role=roles.getObj(i);
                        LDAP_objs dsd = Pdp.rbpimRepository.Consulta9(ou_user,role.getDN());
```

```
                        // Loop de objetos de DSD
                        int j=0;
                        while(j<dsd.size()) {
                                LDAP_obj s=dsd.getObj(j);
                                if(!s.contains("rbpimRoleSet") || !s.contains("rbpimCardinality")) {
                                        j++;
                                        continue;
                                }

                                // Mudar este trecho no algorimo
                                if(Util.checkSeparationOfDuty(1,s.getAttribute("rbpimRoleSet"),roles,
                                        Integer.parseInt(s.getAttribute("rbpimCardinality").getValue(0)))) {
                                        return createDECerror((short)16,(short)111,sessionID,(short)2,(short)2);

                                  }
                                j++;
                        }

                        i++;
                }

                // Loop para atualizar os papeis selecionados pelo usuario na Base de Estados

                i=0;

                while(i<roles.size()) {
                        LDAP_obj role=roles.getObj(i);
                        synchronized(this) {
                                Pdp.rbpimStateBase.setRoleSelected(sessionID,role.getDN());
                        }
                        i++;
                }
                //  gerar DEC message
                m_out=new MESSAGE_STRUCTURE((byte)0x11,(byte)DEC,(short)RBPEPCLIENT);
                // Handle object(Sessao) C_NUM=1  C_TYPE=1  Contents=SessionID
                m_out.add_cops_Object(new COPS_SObject((byte)1,(byte)1,sessionID.getBytes()));
                // Context object C_NUM=2  C_TYPE=1  Contents=R-TYPE+M-TYPE
                m_out.add_cops_Object(new
                COPS_SObject((byte)2,(byte)1,Util.short2bytes((short)2),Util.short2bytes((short)2)));
        // Decision object C_NUM=6  C_TYPE=1  Contents=(COMMAND_CODE=1)(FLAGS=0) --> Requisao aceita
                m_out.add_cops_Object(new
                COPS_SObject((byte)6,(byte)1,Util.short2bytes((short)1),Util.short2bytes((short)0)));
                return m_out;

        }


// Mensagem DEC para resposta a um pedido de verificacao de acesso
public  MESSAGE_STRUCTURE createDECcheckAccess(String sessionID, String operation, String expExplicit, String
                expImplicit)  {

                MESSAGE_STRUCTURE m_out=null;

                // Verificar se ja' existe sessao aberta para sessionID para fase 2
                if(!Pdp.rbpimStateBase.checkSessionID(sessionID,2))
                        return createDECerror((short)16,(short)109,sessionID,(short)2,(short)2);

                LDAP_objs roles=new LDAP_objs();
                roles=Pdp.rbpimStateBase.getSelectedRoles(sessionID);
                int i=0;

                Vector filters=new Vector();
                String filter;

                // Loop papeis
                while(i<roles.size()) {
```

```
LDAP_obj role=roles.getObj(i);
// Verificar periodos de ativacao
 if(role.contains("pcimRuleValidityPeriodList")) {
        LDAP_attr validityPeriodDN = role.getAttribute("pcimRuleValidityPeriodList");
        int j=0;
        boolean authPeriod=false;
        while(j<validityPeriodDN.size()){
                LDAP_obj validityPeriod =
                Pdp.rbpimRepository.Consulta5(validityPeriodDN.getValue(j));

                if(Util.checkPeriod(validityPeriod)) {
                        authPeriod=true;
                        break;
                }
                j++;
        }
        role.setStatus(authPeriod);
        if(!role.getStatus()) {
                i++;
                continue;
        }
}

// Obter lista de objetos de permissao associados ao papel
if(role.contains("pcimRuleActionList")) {
        LDAP_attr permissionRef = role.getAttribute("pcimRuleActionList");
        int j=0;
        while(j<permissionRef.size()){
                String permissionDN =
                Pdp.rbpimRepository.Consulta10(permissionRef.getValue(j));
                LDAP_obj permission = Pdp.rbpimRepository.Consulta11(permissionDN)
                if(!permission.contains("pcimRuleConditionListType")) {
                        j++;
                        continue;
                }

                int conditionListType= Integer.parseInt(
                permission.getAttribute("pcimRuleConditionListType").getValue(0));

                conditionListType= Integer.parseInt(permission.getAttribute
                ("pcimRuleConditionListType").getValue(0));

                if(!Util.checkRestrictions(permission,conditionListType,expImplicit)){
                        j++;
                        continue;
                }

                if(permission.contains("pcimRuleActionList")) {
                        String operationRef = permission.getAttribute
                        ("pcimRuleActionList").getValue(0);


                        if(!Pdp.rbpimRepository.Consulta12(operationRef,operation)) {
                                j++;
                                continue;
                        }
                        else {
                                filter = Util.makeFilter(permission,conditionListType);
                                filters.addElement(filter);
                        }
                }

                j++;
        }
}
```

```
                    i++;
                }

                boolean check=true;

                if(filters.size()>0) {
                    // Obter lista de recursos 1
                    LDAP_objs resources1 = Pdp.rbpimRepository.Consulta13(filters,expExplicit);
                    // Obter lista de recursos 2
                    LDAP_objs resources2 = Pdp.rbpimRepository.Consulta14(expExplicit);
                    if(resources1.size()>0 && resources2.size()>0 && resources2.size()<=resources1.size())
                        for(i=0; i<resources2.size(); i++) {
                            if(!resources1.checkObj(resources2.getObj(i).getDN())) {
                                check=false;
                                break;
                            }
                        }
                    }
                    else
                        check=false;
                }
                else
                    check=false;

                //  gerar DEC message
                m_out=new MESSAGE_STRUCTURE((byte)0x11,(byte)DEC,(short)RBPEPCLIENT);

                // Handle object(Sessao) C_NUM=1  C_TYPE=1  Contents=SessionID
                m_out.add_cops_Object(new COPS_SObject((byte)1,(byte)1,sessionID.getBytes()));

                // Context object C_NUM=2  C_TYPE=1  Contents=R-TYPE+M-TYPE  -- M-TYPE=3 checkaccess
                m_out.add_cops_Object(new
                COPS_SObject((byte)2,(byte)1,Util.short2bytes((short)2),Util.short2bytes((short)3)));

                if(check) {
            // Decision object C_NUM=6  C_TYPE=1  Contents=(COMMAND_CODE=1)(FLAGS=0) --> Requisao aceita
                    m_out.add_cops_Object(new
                    COPS_SObject((byte)6,(byte)1,Util.short2bytes((short)1),Util.short2bytes((short)0)));

                }
                else  {
    // Decision object C_NUM=6  C_TYPE=1  Contents=(COMMAND_CODE=2)(FLAGS=0)  --> Requisicao negada
                    m_out.add_cops_Object(new
                    COPS_SObject((byte)6,(byte)1,Util.short2bytes((short)2),Util.short2bytes((short)0)));
                }
                return m_out;
            }

                    // DEC ERROR message
            public MESSAGE_STRUCTURE createDECerror(short errorCode, short subCode, String s1, short RTYPE, short
                    MTYPE) {

                    MESSAGE_STRUCTURE m_out=null;
                    byte versionFlags=0x11;

                    m_out=new MESSAGE_STRUCTURE(versionFlags,(byte)DEC,(short)RBPEPCLIENT);

                    // Handle object(Sessao) C_NUM=1  C_TYPE=1  Contents=SessionID
                    m_out.add_cops_Object(new COPS_SObject((byte)1,(byte)1,s1.getBytes()));

                     // Context object C_NUM=2  C_TYPE=1  Contents=R-TYPE+M-TYPE
                    m_out.add_cops_Object(new
                    COPS_SObject((byte)2,(byte)1,Util.short2bytes(RTYPE),Util.short2bytes(MTYPE)));

                    // ERROR object C_NUM=8  C_TYPE=1  Contents=ErrorCodeSubcode
                    m_out.add_cops_Object(new
```

```
		COPS_SObject((byte)8,(byte)1,Util.short2bytes(errorCode),Util.short2bytes(subCode)));

		return m_out;
}



public MESSAGE_STRUCTURE readOPN(MESSAGE_STRUCTURE m_in)
{
		COPS_SObject copsObj = m_in.readCopsObj(0);
		String pepId=new String(copsObj.getContents());

		filename=pepId+".out";
		pdpOUT = new File(filename);
		try {
			boolean f=false;
			if(!pdpOUT.exists())
				f=true;
			pdpOUT.createNewFile();
			fileOut = new BufferedWriter(new FileWriter(pdpOUT.getPath(),true));
			if(f)
				print(0,0,">> Arquivo de LOG do PDP para RBPEP: " + pepId + "<<");
			print(0,1,m_in.display());

		}
		catch(IOException e)
			{
			System.out.println(e);
		}


		if(m_in.getclientType()!=RBPEPCLIENT) {
			return createCC((short)16,(short)100);  // RBPEP Nao suportado
		}

		if(!Pdp.containsAuthRbpep(pepId)) {
			return createCC((short)16,(short)101);  // RBPEP Nao autorizado
		}

		if(Pdp.containsActiveRbpep(pepId)) {
			return createCC((short)16,(short)102);  // Servico ja aberto
		}

		return createCAT(pepId);
}

public void readCC(MESSAGE_STRUCTURE m_in)
{

		COPS_SObject copsObj = m_in.readCopsObj(0);
		byte [ ] err = new byte[2];
		byte [ ] sub = new byte[2];
		System.arraycopy(copsObj.getContents(),0,err,0,2);
		System.arraycopy(copsObj.getContents(),2,sub,0,2);
		short ERRORCODE   =  Util.readShort(err);
		short ERRORSUBCODE=  Util.readShort(err);

		synchronized(this) {
			Pdp.rbpimStateBase.closeService(pepId);
		}

		Pdp.removeActiveRbpep(pepId);
}
```

```
public MESSAGE_STRUCTURE readREQ(MESSAGE_STRUCTURE m_in)
{
	String sessionID=new String(m_in.readCopsObj(0).getContents());

	byte [ ] rt = new byte[2];
	byte [ ] mt = new byte[2];

	System.arraycopy(m_in.readCopsObj(1).getContents(),0,rt,0,2);
	System.arraycopy(m_in.readCopsObj(1).getContents(),2,mt,0,2);

	short RTYPE= Util.readShort(rt);
	short MTYPE= Util.readShort(mt);

	if(RTYPE!=2)	{
		return createDECerror((short)16,(short)104,sessionID,RTYPE,MTYPE);
	}

	switch(MTYPE)
		{
		case (short)1: String userCN = new String(m_in.readCopsObj(2).getContents());
				return createDECsession1(sessionID, userCN);

		case (short)2:  String selectedRoles = new String(m_in.readCopsObj(2).getContents());
				return createDECsession2(sessionID, selectedRoles);

		case (short)3:  String operation = new String(m_in.readCopsObj(2).getContents());
				String expExplicit = new String(m_in.readCopsObj(3).getContents());
				String expImplicit=null;
				if(m_in.getSize()>4)
				    expImplicit = new String(m_in.readCopsObj(4).getContents());
				    return createDECcheckAccess(sessionID, operation, expExplicit,
				    expImplicit);

	}
	return createDECerror((short)16,(short)106,sessionID,RTYPE,MTYPE);
}


public void readDRQ(MESSAGE_STRUCTURE m_in) {
	COPS_SObject copsObj = m_in.readCopsObj(0);
	String sessionID=new String(copsObj.getContents());

	synchronized(this) {
		Pdp.rbpimStateBase.closeSession(sessionID);
	}
}


public void readRPT(MESSAGE_STRUCTURE m_in, MESSAGE_STRUCTURE m_out)  {
	if(m_out.readCopsObj(2).getC_NUM()!=(byte)8) {
		COPS_SObject copsObj = m_in.readCopsObj(0);
		String sessionID=new String(copsObj.getContents());
		copsObj = m_in.readCopsObj(1);
		byte [ ] rpt = new byte[2];
		System.arraycopy(copsObj.getContents(),0,rpt,0,2);
		short REPORTTYPE= Util.readShort(rpt);

		if(REPORTTYPE==1 && !Pdp.rbpimStateBase.checkSessionID(sessionID,2)) {
			synchronized(this) {
				Pdp.rbpimStateBase.setSessionStatus(sessionID);
			}

		}
	}
}
```

```
	public void print(long t, int i, String s) {
		String value = Long.toString(t).trim();
		int len=value.length();
		for(int j=0;j<8-len;j++) {
			value+=" ";
		}

		value+="%";

		if(i==1)
			s=value+"RBPEP->PDP:"+s;
		else if(i==2)
			s=value+"PDP->RBPEP:"+s;

		try{
			fileOut.write(s);
			fileOut.newLine();
			fileOut.flush();
		}
		catch(IOException e) {
			System.out.println(e);
		}

	}

}
```

1.2. Arquivo Rbpep.Java

```
/*
Nome da classe: Rbpep
Descricao: Representa o Rbpep (Pep para clientes RBPIM) na arquitetura
definida para implementacao do RBPIM.  */

import java.io.*;
import java.net.*;
import java.util.Vector;

public class Rbpep  {
	private String pepId=null;
	private Socket socket=null;
	private Vector   sessionIds;
	private int sequenceSession;
	private static final byte REQ = (byte)1;
	private static final byte RPT = (byte)3;
	private static final byte DRQ = (byte)4;
	private static final byte OPN = (byte)6;
	private static final byte DEC = (byte)2;
	private static final byte CAT = (byte)7;
	private static final byte CC  = (byte)8;
	private static final short RBPEPCLIENT  =  (short) 0x8000;

	Rbpep(String Pdp, String pepId) {
		this.pepId = pepId;
		sequenceSession=0;
		sessionIds=new Vector();

		try {
			socket = new Socket(Pdp, 3288);
			socket.setTcpNoDelay(true);
		}
		catch (UnknownHostException e) {
			System.err.println("Host " + Pdp + " nao encontrado!");
```

```
                System.exit(1);
        }
        catch (IOException e) {
                System.err.println("ERRO: nao foi possivel conectar a " + Pdp + ":3288!");
                System.exit(1);
        }
}

public String createSessionId()   {
        sequenceSession++;
        String newSessionId = pepId+"_"+String.valueOf(sequenceSession);
        sessionIds.addElement(newSessionId);
        return newSessionId;
}


public void closeSession(String sessionID)    {
        sessionIds.remove(sessionID);
}

char callID='0';

public Vector processCall(char callID, Vector parameters) {
        MESSAGE_STRUCTURE message_out=null;
        MESSAGE_STRUCTURE message_in=null;

        switch(callID) {
                case '1':   message_out = createOPN(); // RBPEP_Open
                                break;

                                                // RBPEP_CreateSession (Fase 1)
                case '2':   message_out = createREQ1((String)parameters.get(0));
                            break;
                                                // RBPEP_CreateSession (Fase 2)
                case '3':   message_out = createREQ2(parameters);
                            break;
                                                // RBPEP_CloseSession
                case '4':   message_out = createDRQ((String)parameters.get(0));
                            break;
                                                  // RBPEP_CheckAccess
                case '5':   message_out = createREQ3(parameters);
                            break;
                                // Solicitar fechamento do servico de politica
                case 'X':   message_out = createCC();
        }

        Vector result=new Vector();

        if(message_out!=null) {
                try {
                        System.out.println("RBPEP -> PDP: " + message_out.display());
                        // enviar mensagem PDP
                        OutputStream out = socket.getOutputStream();
                        message_out.writeBytes(out);
                        out.flush();
                        // receber mensagem do PDP
                         if(message_out.getopCode()!=CC && message_out.getopCode()!=DRQ) {
                                InputStream in = socket.getInputStream();

                                message_in=COPS_Protocol.readMessage(in);
                        System.out.println("PDP -> RBPEP: " + message_in.display());
                                result = processInMessage(message_in);
                                if(message_in.getopCode()==DEC) {
                                message_out =
                                createRPT(message_in.readCopsObj(0).getContents());
                                message_out.writeBytes(out);
```

```
                                        out.flush();
                                        }
                                }
                        }
                         catch (Exception e) {e.printStackTrace(); }
        }
        return result;
}


// Open client message
public MESSAGE_STRUCTURE createOPN() {
        MESSAGE_STRUCTURE message;
        message=new MESSAGE_STRUCTURE((byte)0x10,OPN,RBPEPCLIENT);
        // PEP ID object C_NUM=11  C_TYPE=1  Contents=PEPID
        message.add_cops_Object(new COPS_SObject((byte)11,(byte)1,pepId.getBytes()));
return message;
}


// Client close message
public MESSAGE_STRUCTURE createCC() {
        MESSAGE_STRUCTURE message;
        message=new MESSAGE_STRUCTURE((byte)0x10,CC,RBPEPCLIENT);
        // ERROR object C_NUM=8  C_TYPE=1  Contents=ErrorCodeSubcode
        message.add_cops_Object(new COPS_SObject((byte)8,(byte)1,
                                        Util.short2bytes((short)16),Util.int2bytes((short)108)));
        return message;
}

// Delete Request State
public MESSAGE_STRUCTURE createDRQ(String sessionID) {
        MESSAGE_STRUCTURE message;
        message=new MESSAGE_STRUCTURE((byte)0x10,DRQ,RBPEPCLIENT);
        // Handle object(Sessao) C_NUM=1  C_TYPE=1  Contents=SessionID
        message.add_cops_Object(new COPS_SObject((byte)1,(byte)1,sessionID.getBytes()));
// Reason object C_NUM=5  C_TYPE=1  Contents=ReasonCode+ReasonSubCode
message.add_cops_Object(new COPS_SObject((byte)5,(byte)1,
                                        Util.short2bytes((short)1),Util.short2bytes((short)0)));
        this.closeSession(sessionID);
        return message;
}

// Request  message         //Create Session  (Fase 1)
public MESSAGE_STRUCTURE createREQ1(String userCN) {
        MESSAGE_STRUCTURE message;
        message=new MESSAGE_STRUCTURE((byte)0x10,REQ,RBPEPCLIENT);
        String newSessionID = createSessionId();
        // Handle object(Sessao) C_NUM=1  C_TYPE=1  Contents=newSessionID
        message.add_cops_Object(new COPS_SObject((byte)1,(byte)1,newSessionID.getBytes()));


        // Context object C_NUM=2  C_TYPE=1  Contents=R-TYPE+M-TYPE  (2)Resource
        // allocation (1) Create Session Fase 1
        message.add_cops_Object(new COPS_SObject((byte)2,(byte)1,
        Util.short2bytes((short)2),Util.short2bytes((short)1)));
        // ClientSI object C_NUM=9  C_TYPE=2  Contents=user cn
        message.add_cops_Object(new COPS_SObject((byte)9,(byte)2,userCN.getBytes()));

        return message;
}

public MESSAGE_STRUCTURE createREQ2(Vector parameters) {

        MESSAGE_STRUCTURE message;
        message=new MESSAGE_STRUCTURE((byte)0x10,REQ,RBPEPCLIENT);
```

```
		String sessionID = (String)parameters.get(0);
		// Handle object(Sessao) C_NUM=1  C_TYPE=1  Contents=newSessionID
		message.add_cops_Object(new COPS_SObject((byte)1,(byte)1,sessionID.getBytes()));

		// Context object C_NUM=2  C_TYPE=1  Contents=R-TYPE+M-TYPE  (2)Resource
		// allocation (2)Create Session Fase 2
		message.add_cops_Object(new COPS_SObject((byte)2,(byte)1,
		Util.short2bytes((short)2),Util.short2bytes((short)2)));                                                    int
i=1;
		String list="";
		while(true) {
			list+=(String)parameters.get(i);
			i++;
			if(i<parameters.size())
				list+=",";
			else
				break;
		}

		// ClientSI object C_NUM=9  C_TYPE=2  Contents=lista de papeis selecionados
		message.add_cops_Object(new COPS_SObject((byte)9,(byte)2,list.getBytes()));

		return message;
}

public MESSAGE_STRUCTURE createREQ3(Vector parameters) {
		MESSAGE_STRUCTURE message;
		message=new MESSAGE_STRUCTURE((byte)0x10,REQ,RBPEPCLIENT);
		String sessionID = (String)parameters.get(0);
		String operation = (String)parameters.get(1);

		// Handle object(Sessao) C_NUM=1  C_TYPE=1  Contents=SessionID
		message.add_cops_Object(new COPS_SObject((byte)1,(byte)1,sessionID.getBytes()));

		// Context object C_NUM=2  C_TYPE=1  Contents=R-TYPE+M-TYPE  (2)Resource
		// allocation (3)Check Access
		message.add_cops_Object(new COPS_SObject((byte)2,(byte)1,
		Util.short2bytes((short)2),Util.short2bytes((short)3)));

		// ClientSI object C_NUM=9  C_TYPE=2  Contents= operacao
		message.add_cops_Object(new COPS_SObject((byte)9,(byte)2,operation.getBytes()));


		int quant = Integer.parseInt(((String)parameters.get(2)));
		int i=0;
		String Exp="";
		while(true) {
			Exp+=(String)parameters.get(i+3);
			i++;
			if(i<quant)
				Exp+=",";
			else
				break;
		}

		// ClientSI object C_NUM=9  C_TYPE=1  Contents=lista com exps. definidas pelas var.
		// explicitas
		message.add_cops_Object(new COPS_SObject((byte)9,(byte)1,Exp.getBytes()));

		Exp="";
		quant = Integer.parseInt(((String)parameters.get(i+3)));
		int j=0;
		while(true && quant>0) {
			Exp+=(String)parameters.get(i+i+3);
			j++;
			if(j<quant)
```

```
                        Exp+=",";
                else
                        break;
        }

        if(j>0) {
                // ClientSI object C_NUM=9  C_TYPE=1  Contents=lista com exps. definidas pelas
                // var. implicitas
                message.add_cops_Object(new COPS_SObject((byte)9,(byte)1,Exp.getBytes()));
}
        return message;
}

// Report  message
public MESSAGE_STRUCTURE createRPT(byte [ ]  sessionID) {
        MESSAGE_STRUCTURE message;
        message=new MESSAGE_STRUCTURE((byte)0x11,RPT,RBPEPCLIENT);
        // Handle object(Sessao) C_NUM=1  C_TYPE=1  Contents=sessionID
        message.add_cops_Object(new COPS_SObject((byte)1,(byte)1,sessionID));
// Report type object C_NUM=12  C_TYPE=1  Contents=1 (Sucesso)
        message.add_cops_Object(new COPS_SObject((byte)12,(byte)1,
                                Util.short2bytes((short)1),Util.short2bytes((short)0)));
        return message;
}

public Vector processInMessage(MESSAGE_STRUCTURE m_in)  {
        Vector v=new Vector();
        switch(m_in.getopCode())     {
                        case DEC: v=readDEC(m_in);break;
                        case CAT: v=readCAT(m_in);break;
                        case  CC: v=readCC(m_in);
                }
        return v;
}

public Vector readCAT(MESSAGE_STRUCTURE m_in) {
        Vector v=new Vector();
        COPS_SObject copsObj = m_in.readCopsObj(0);
        byte [ ] e1 = new byte[4];
        System.arraycopy(copsObj.getContents(),0,e1,0,4);
        v.addElement(Util.returnMessages(0));
        return v;
}

public Vector readDEC(MESSAGE_STRUCTURE m_in) {
        Vector v=new Vector();
        COPS_SObject copsObj = m_in.readCopsObj(0);
        String sessionID=new String(copsObj.getContents());
        copsObj = m_in.readCopsObj(1);
        byte [ ] rt = new byte[2];
        byte [ ] mt = new byte[2];
        System.arraycopy(copsObj.getContents(),0,rt,0,2);
        System.arraycopy(copsObj.getContents(),2,mt,0,2);
        short RTYPE= Util.readShort(rt);
        short MTYPE= Util.readShort(mt);
        byte [ ] cmd = new byte[2];
        copsObj = m_in.readCopsObj(2);
        System.arraycopy(copsObj.getContents(),0,cmd,0,2);
        short COMMANDCODE = Util.readShort(cmd);
        v.addElement("Sessao: "+ sessionID);
        switch(MTYPE) {
                case 1:  // Resposta ao Create Session fase 1
                        if(COMMANDCODE==1)  {
                                v.addElement("Sessao Fase 1 com sucesso!");
                                copsObj = m_in.readCopsObj(3);
                                byte [ ] nos = new byte[4];
```

```
                                        System.arraycopy(copsObj.getContents(),0,nos,0,4);
                                        v.addElement("Num sessoes abertas: " + Util.readInt(nos));
                                        copsObj = m_in.readCopsObj(4);
                                        String rolesDN=new String(copsObj.getContents());
                                        v.addElement("Papeis: " + rolesDN);
                                }
                                else
                                        v.addElement("Nenhuma informacao disponivel!");
                                break;
                        case 2:  // Resposta ao Create Session fase 2
                                if(COMMANDCODE==1)
                                        v.addElement("Sessao Fase 2 com sucesso!");
                                else
                                         v.addElement("Erro na selecao de papeis!");
                        break;
                        case 3:  // Resposta ao Checkaccess
                                if(COMMANDCODE==1)
                                        v.addElement("Acesso autorizado!");
                                else
                                        v.addElement("Acesso negado!");
                }

                return v;
        }


        public Vector readCC(MESSAGE_STRUCTURE m_in) {
        Vector v=new Vector();
                COPS_SObject copsObj = m_in.readCopsObj(0);
                byte [ ] e1 = new byte[2];
                byte [ ] e2 = new byte[2];
                System.arraycopy(copsObj.getContents(),0,e1,0,2);
                System.arraycopy(copsObj.getContents(),2,e2,0,2);
                v.addElement(Util.returnMessages(Util.bytes2short(e2)));
                return v;
        }
}
```